Introduction to Bayesian Networks

Introduction to Bayesian Networks

Timo Koski,
Institutionen för matematik,
Kungliga Tekniska Högskolan,
10044 STOCKHOLM, Sweden

John M. Noble,
Faculty of Mathematics, Informatics and Mechanics,
University of Warsaw,
ul. Banacha 2,
02-097 WARSZAWA, Poland

Contents

Introduction

The models that were later to be called Bayesian networks were introduced into artificial intelligence by J. Pearl in (1982) [103], a seminal article in the literature of that field. A *Bayesian network* is simply a factorisation of a probability distribution and a corresponding *directed acyclic graph* (henceforth written DAG), where the edges of the DAG correspond to direct associations between variables in the factorisation.

The first Bayesian networks were connected with *causal* models, where the order of the variables in the factorisation represented cause to effect, and the directed arrows in the DAG represented direct causes. This is still one of the major uses of Bayesian networks. The leaf nodes of the network are the observables. From observations of leaf variables, inference is made for hidden variables via Bayes rule, hence the term Bayesian network. The terminology, therefore, derives from the probabilistic use of Bayes rule; the statistical use of Bayes rule whereby uncertainty in the parameter value is expressed in terms of a prior distribution which is then updated to a posterior distribution when data is considered, is not in view here.

Graphical directed separation statements for the DAG imply corresponding conditional independence statements for the probability distribution. For large and complex systems, graphical separation algorithms provide a convenient and efficient method to establish probabilistic conditional independence statements.

The description 'Bayesian networks' now covers a large field of problems and techniques of data analysis and probabilistic reasoning, where data is collected on a large number of variables and the aim is to factorise the distribution, represent it graphically and exploit the graphical representation. Perhaps the earliest work that explicitly uses directed graphs to represent possible dependencies among random variables is that by S. Wright (1921) [146], developed by the same author in 1934 [147].

Bayesian networks represent a small part of the wider field of *graphical models*. A Bayesian network is a probability distribution factorised along a DAG. In many examples this is not the most efficient model for representing the independence structure and there is a wider field of graphical models.

Situations where Bayesian networks provide the natural tools for analysis are, for example: computing the overall reliability of a system given the reliability of the individual components and how they interact, system security where Bayesian networks are used as a tool for assessing intrusion evidence and whether a network is under attack, forensic analysis. Further applications are, for example: finding the most likely message that was sent across a noisy channel, restoring a noisy image, mapping genes onto a chromosome. One of the leading applications of techniques from the area is to establish-

ing genome pathways. Given DNA hybridisation arrays, which simultaneously measure the expression levels for thousands of genes, a major challenge in computational biology is to uncover, from such measurements, gene/protein interactions and key biological features of cellular systems. This is discussed, for example, by Nir Friedman et. al. in [46] (2000).

DAGs have proved useful in a large number of situations where the graph is constructed along causal principles; parent variables are considered to be *direct causes*. One field where causal networks have proved particularly effective has been epidemiological research, where DAGs have provided a framework for the problem of multiple confounding factors in genetic epidemiology, as discussed by Greenland, Pearl and Robins (1999) [56]. Bayesian networks offer an alternative to 'naïve Bayes' models of supervised classification in machine learning, which enable more of the structure to be exploited. One of the first examples of this was the Chow-Liu tree (1968) [28].

Chapter 1

Conditional Independence and Graphical Models

A *graphical model* for a probability distribution over several variables is, quite simply, a graph, where the random variables correspond to the node set of the graph and each graphical separation statement implies the corresponding conditional independence statement for the random variables. The opposite (that conditional independence implies graphical separation) in general does not hold. In a system with a large numbers of variables, the task of determining graphical separation statements is, in general, computationally far less demanding than the task of determining conditional independence, hence the motivation for graphical models and applying graph theoretic results.

A *Bayesian network* is the representation of a probability distribution on a directed acyclic graph (DAG). In this setting, the most useful notion of separation is *D-separation* (short for *directed* separation), which is defined later. If a probability distribution factorises along a DAG, then D-separation statements in the DAG imply the corresponding conditional independence statements (although the reverse implication is, in general, false).

In many problems, for example gene expression data where there are thousands of variables, it may not be either possible or desirable to obtain a complete description of the dependence structure. The aim for such problems is to learn a DAG which encodes the most important features of the dependence structure. In classification problems, a complete description of the dependence structure is usually unnecessary; algorithms only locate the key features of the dependence structure to ensure accurate classification.

1.1 Notational preliminaries: Graphical and Probabilistic

Random Variables Let $X = (X_1, \ldots, X_d)'$ denote a vector of random variables. The random variables under consideration are of two types, discrete with finite state space and continuous. Let $\mathcal{X}_j$ denote the state space of variable j. If X_j is continuous, the state space is $\mathbb{R}$. If X_j has a finite state space with k_j elements, say $(x_j^{(1)}, \ldots, x_j^{(k_j)})$, then $\mathcal{X}_j = \{0, \ldots, k_j - 1\}$ denotes the indexing set. The state space of the random vector X is the product space $\mathcal{X} = \times_{j=1}^{d} \mathcal{X}_j$.

In the subject of Bayesian Networks, there are usually three situations in view: *multinomial*, *Gaussian* and *Conditional Gaussian*. $\mathbb{P}_{X_1,\ldots,X_d}$ will be used to denote the probability distribution over $X_1,\ldots,X_d$. That is, for the multinomial case, this is simply the probability function; the quantity $\mathbb{P}_{X_1,\ldots,X_d}(x_1,\ldots,x_d)$ is simply the probability of obtaining a configuration with indices $(x_1,\ldots,x_d) \in \mathcal{X}$. When X is a multivariate Gaussian random vector, $\mathbb{P}_{X_1,\ldots,X_d}$ refers to the probability density function. When X is conditional Gaussian, the discrete variables are listed with lower index than the continuous, so that $X = (X_1,\ldots,X_a,X_{a+1},\ldots,X_d)$ where there are a discrete variables and $d-a$ continuous variables. Here $\mathbb{P}_{X_1,\ldots,X_a}$ is the probability *function* for the discrete variables, while for each configuration $(x_1,\ldots,x_a)$ of the discrete variables, $\mathbb{P}_{X_{a+1},\ldots,X_d|X_1,\ldots,X_a}(.|x_1,\ldots,x_a)$ is a multivariate Gaussian probability distribution over $\mathbb{R}^{d-a}$ for the variables $a+1,\ldots,d$.

If $X = (X_1,\ldots,X_d)'$ and $A \subset \{1,\ldots,d\}$ where $A = (a(1),\ldots,a(m))$ then $X_A := (X_{a(1)},\ldots,X_{a(m)})'$.

In this treatment, the discussion will be presented for *discrete* random variables, unless explicitly stated otherwise.

Notations and Definitions for Graphs

Definition 1.1 (Graph, Simple Graph). *A* graph $\mathcal{G} = (V,E)$ *consists of a finite set of nodes* V *and an edge set* E*, where each edge is contained in* $V \times V$*. The edge set therefore consists of ordered pairs of nodes, which we denote* (α,β) *or* $\alpha \to \beta$*.*

Let $V = \{1,\ldots,d\}$*. A graph* $\mathcal{G} = (V,E)$ *is said to be* simple *if* E *does not contain any edges of the form* (α,α) *(that is a loop from the node to itself) and any edge* $(\alpha,\beta) \in E$ *that appears in* E *does so exactly once. That is, multiple edges are not permitted.*

For any two distinct nodes α *and* $\beta \in V$*, the ordered pair* $(\alpha,\beta) \in E$ *if and only if there is a directed edge from* α *to* β*. An* undirected *edge will be denoted* $\langle\alpha,\beta\rangle$*. In terms of directed edges,*

$$\langle\alpha,\beta\rangle \in E \Leftrightarrow (\alpha,\beta) \in E \qquad \textit{and} \qquad (\alpha,\beta) \in E.$$

For a simple graph that may contain both directed and undirected edges, the edge set E *may be decomposed as* $E = D \cup U$*, where* $D \cap U = \varnothing$*, the empty set. The sets* U *and* D *are defined by*

$$\langle\alpha,\beta\rangle \in U \Leftrightarrow (\alpha,\beta) \in E \quad \textit{and} \quad (\beta,\alpha) \in E.$$

$$(\alpha,\beta) \in D \Leftrightarrow (\alpha,\beta) \in E \quad \textit{and} \quad (\beta,\alpha) \notin E.$$

If $(\alpha,\beta) \in D$*, we may also denote this by* $\alpha \to \beta \in D$*. If* $\langle\alpha,\beta\rangle \in U$*, we may also denote this by* $\alpha - \beta \in U$*. If either* $(\alpha,\beta) \in D$ *or* $(\beta,\alpha) \in D$ *or* $\langle\alpha,\beta \in U$*, but we do not specify which, we may denote this by* $\alpha \sim \beta$*. For the definitions of 'path', 'trail' and 'cycle', an undirected edge will be considered as a single edge.*

All the graphs considered in this treatment will be simple graphs and the term 'graph' will be used to mean 'simple graph'. If $(\alpha,\beta) \in D$, this is denoted by an arrow going *from* α *to* β. If $\langle\alpha,\beta\rangle \in U$, this is denoted by an undirected edge between the two variables α and β.

Definition 1.2 (Parent, Child, Directed and Undirected Neighbour, Family). *Consider a graph* $\mathcal{G} = (V, E)$*, where* $V = \{1, \ldots, d\}$ *and let* $E = D \cup U$*, where* D *is the set of directed edges and* U *the set of undirected edges. Let* $\alpha, \beta \in V$*. If* $(\alpha, \beta) \in D$*, then* β *is referred to as a* child *of* α *and* α *as a* parent *of* β*.*

For any node $\alpha \subseteq V$*, the set of* parents *is defined as*

$$Pa(\alpha) = \{\beta \in V \mid (\beta, \alpha) \in D\} \tag{1.1}$$

and the set of children *is defined as*

$$Ch(\alpha) = \{\beta \in V \mid (\alpha, \beta) \in D\}. \tag{1.2}$$

For any subset $A \subseteq V$*, the set of parents of* A *is defined as*

$$Pa(A) = \cup_{\alpha \in A}\{\beta \in V \backslash A \mid (\beta, \alpha) \in D\}. \tag{1.3}$$

The set of directed neighbours *of a node* α *is defined as*

$$N_{(d)}(\alpha) = Pa(\alpha) \cup Ch(\alpha)$$

and the set of undirected neighbours *of* α *as*

$$N_{(u)}(\alpha) = \{\beta \in V \mid \langle \alpha, \beta \rangle \in U\}. \tag{1.4}$$

For any subset $A \subseteq V$*, the set of* undirected neighbours *of* A *is defined as*

$$N_{(u)}(A) = \cup_{\alpha \in A}\{\beta \in V \backslash A \mid \langle \alpha, \beta \rangle \in U\}. \tag{1.5}$$

For a node α*, the set of neighbours* $N(\alpha)$ *is defined as*

$$N(\alpha) = N_{(u)}(\alpha) \cup N_{(d)}(\alpha).$$

The family *of a node* β *is the set containing the node* β *together with its parents and undirected neighbours. It is denoted:*

$$F(\beta) = \{\beta\} \cup Pa(\beta) \cup N_{(u)}(\beta) = \{family\ of\ \beta\}.$$

When $\mathcal{G}$ *is undirected, this reduces to* $F(\beta) = \{\beta\} \cup N(\beta)$*.*

The notation $\alpha \sim \beta$ will be used to denote that $\alpha \in N(\beta)$; namely, that α and β are neighbours. Note that $\alpha \in N(\beta) \implies \beta \in N(\alpha)$.

In this text, a directed edge (α, β) is indicated by a pointed arrow from α to β; that is, from the parent to the child.

Definition 1.3 (Directed, Undirected Graph). *If* all *edges of a graph are undirected, then the graph $\mathcal{G}$ is said to be* undirected. *If* all *edges are directed, then the graph is said to be* directed. *The* undirected *version of a graph $\mathcal{G}$, denoted by $\tilde{\mathcal{G}}$, is obtained by replacing the directed edges of $\mathcal{G}$ by undirected edges.*

Definition 1.4 (Trail). *Let $\mathcal{G} = (V, E)$ be a graph, where $E = D \cup U$; $D \cap U = \varnothing$, D denotes the directed edges and U the undirected edges. A* trail *τ between two nodes $\alpha \in V$ and $\beta \in V$ is a collection of nodes $\tau = (\tau_1, \ldots, \tau_m)$, where $\tau_i \in V$ for each $i = 1, \ldots, m$, $\tau_1 = \alpha$ and $\tau_m = \beta$ and such that for each $i = 1, \ldots, m-1$, $\tau_i \sim \tau_{i+1}$. That is, for each $i = 1, \ldots, m-1$, either $(\tau_i, \tau_{i+1}) \in D$ or $(\tau_{i+1}, \tau_i) \in D$ or $\langle \tau_i, \tau_{i+1} \rangle \in U$.*

Definition 1.5 (Sub-graph, Induced Sub-graph). *Let $A \subseteq V$ and $E_A \subseteq E \cap A \times A$. Then $\mathcal{F} = (A, E_A)$ is a* sub graph *of $\mathcal{G}$.*

If $A \subset V$ and $E_A = E \cap A \times A$, then $\mathcal{G}_A = (A, E_A)$ is the sub-graph induced by *A.*

Note that in general it is possible for a sub-graph to contain the same nodes, but fewer edges, but the sub-graph *induced* by the same node set will have the same edges.

Definition 1.6 (Connected Graph, Connected Component). *A graph is said to be* connected *if between any two nodes $\alpha_j \in V$ and $\alpha_k \in V$ there is a trail. A* connected component *of a graph $\mathcal{G} = (V, E)$ is an induced sub-graph $\mathcal{G}_A$ such that $\mathcal{G}_A$ is connected and such that if $A \neq V$, then for any two nodes $(\alpha, \beta) \in V \times V$ such that $\alpha \in A$ and $\beta \in V \backslash A$, there is no trail between α and β.*

Definition 1.7 (Path, Directed Path). *Let $\mathcal{G} = (V, E)$ denote a simple graph, where $E = D \cup U$. That is, $D \cap U = \varnothing$, D denotes the directed edges and U denotes the undirected edges. A* path *of length m from a node α to a node β is a sequence of distinct nodes $(\tau_0, \ldots, \tau_m)$ such that $\tau_0 = \alpha$ and $\tau_m = \beta$ such that $(\tau_{i-1}, \tau_i) \in E$ for each $i = 1, \ldots, m$. That is, for each $i = 1, \ldots, m$, either $(\tau_{i-1}, \tau_i) \in D$, or $\langle \tau_{i-1}, \tau_i \rangle \in U$.*

The path is a directed path *if $(\tau_{i-1}, \tau_i) \in D$ for each $i = 1, \ldots, m$. That is, there are no undirected edges along the directed path.*

It follows that a *trail* in $\mathcal{G}$ is a sequence of nodes that form a path in the undirected version $\tilde{\mathcal{G}}$.

Unlike a trail, a directed path $(\tau_0, \ldots, \tau_m)$ requires that the directed edge $(\tau_i, \tau_{i+1}) \in D$ for all $i = 0, \ldots, m-1$.

Definition 1.8 (Descendant, Ancestor). *Let $\mathcal{G} = (V, E)$ be a graph. A node α is a* descendant *of a node β if and only if there is a directed path from β to α. A node γ is an* ancestor *of a node α if and only if there is a directed path from γ to α.*

Let $E = U \cup D$, where U denotes the undirected edges and D denotes the directed edges. The set of descendants *$D(\alpha)$ of a node α is defined as*

$$D(\alpha) = \{\beta \in V \mid \exists \tau = (\tau_0, \ldots, \tau_k) : \tau_0 = \alpha,\ \tau_k = \beta,\ (\tau_j, \tau_{j+1}) \in D,\ j = 0, 1, \ldots, k\}. \tag{1.6}$$

That is, nodes β such that there is a directed path from α to β.

The set of ancestors *$A(\alpha)$ of a node α is defined as*

$$A(\alpha) = \{\beta \in V \mid \exists \tau = (\tau_0, \ldots, \tau_k) : \tau_0 = \beta,\ \tau_k = \alpha,\ (\tau_j, \tau_{j+1}) \in D,\ j = 0, 1, \ldots, k\}. \tag{1.7}$$

That is, nodes β such that there is a directed path from β to α.

In both cases, the paths are directed; they consist of directed edges only; they do not contain undirected edges.

Definition 1.9 (Cycle). *Let $\mathcal{G} = (V, E)$ be a graph. An m-cycle in $\mathcal{G}$ is a sequence of distinct nodes*

$$\tau_0, \ldots, \tau_{m-1}$$

such that $\tau_0, \ldots, \tau_{m-1}, \tau_0$ is a path (Definition 1.7).

Definition 1.10 (Directed Acyclic Graph (DAG)). *A* graph *$\mathcal{G} = (V, E)$ is said to be a* directed acyclic graph *if each edge is* directed *(that is, $\mathcal{G}$ is a simple graph such that for each pair $(\alpha, \beta) \in V \times V$, $(\alpha, \beta) \in E \implies (\beta, \alpha) \notin E$) and for any node $\alpha \in V$ there does not exist any set of distinct nodes $\tau_1, \ldots, \tau_m$ such that $\alpha \neq \tau_i$ for all $i = 1, \ldots, m$ and $(\alpha, \tau_1, \ldots, \tau_m, \alpha)$ forms a directed path. That is, there are no m-cycles in $\mathcal{G}$ for any $m \geq 1$.*

Definition 1.11 (Tree, Leaf). *A* tree *is a graph $\mathcal{G} = (V, E)$ that is* connected *and such that for any node $\alpha \in V$, there is no trail between α and α and for any two nodes α and β in V with $\alpha \neq \beta$, there is a unique trail. A* leaf *of a tree is a node that is connected to exactly one other node.*

1.2 Conditional Independence and Factorisation

Definition 1.12 (Independence). *Two random vectors X and Y are independent if their joint probability distribution factorises as*

$$\mathbb{P}_{X,Y} = \mathbb{P}_X \mathbb{P}_Y.$$

X and Y are conditionally independent given a random vector Z if

$$\mathbb{P}_{X,Y,Z} = \mathbb{P}_{X|Z} \mathbb{P}_{Y|Z} \mathbb{P}_Z.$$

This is written $X \perp Y | Z$.

The following characterisations of conditional independence follow from the definition.

Theorem 1.13. *The following are all equivalent to $X \perp Y | Z$: using $\mathcal{X}_X$, $\mathcal{X}_Y$ and $\mathcal{X}_Z$ to denote the state spaces of X, Y and Z respectively:*

1. *For all $(x, y, z) \in \mathcal{X}_Z \times \mathcal{X}_Y \times \mathcal{X}_Z$ such that $\mathbb{P}_{Y|Z}(y|z) > 0$ and $\mathbb{P}_Z(z) > 0$,*

$$\mathbb{P}_{X|Y,Z}(x|y, z) = \mathbb{P}_{X|Z}(x|z).$$

2. *There exists a function* $a : \mathcal{X}_X \times \mathcal{X}_Z \to [0,1]$ *such that for all* $(x,y,z) \in \mathcal{X}_X \times \mathcal{X}_Y \times \mathcal{X}_Z$ *satisfying* $\mathbb{P}_{Y,Z}(y,z) > 0$,

$$\mathbb{P}_{X|Y,Z}(x|y,z) = a(x,z)$$

3. *There exist functions* $a : \mathcal{X}_X \times \mathcal{X}_Z \to \mathbf{R}_+$ *and* $b : \mathcal{X}_Y \times \mathcal{X}_Z \to \mathbf{R}_+$ *such that for all* $(x,y,z) \in \mathcal{X}_X \times \mathcal{X}_Y \times \mathcal{X}_Z$ *satisfying* $\mathbb{P}_{\underline{Z}}(z) > 0$,

$$\mathbb{P}_{X,Y|Z}(x,y|z) = a(x,z)b(y,z)$$

4. *For all* $(x,y,z) \in \mathcal{X}_X \times \mathcal{X}_Y \times \mathcal{X}_Z$ *such that* $\mathbb{P}_Z(z) > 0$,

$$\mathbb{P}_{X,Y,Z}(x,y,z) = \frac{\mathbb{P}_{X,Z}(x,z)\mathbb{P}_{Y,Z}(y,z)}{\mathbb{P}_Z(z)}.$$

5. *There exist functions* $a : \mathcal{X}_X \times \mathcal{X}_Z \to \mathbf{R}_+$ *and* $b : \mathcal{X}_Y \times \mathcal{X}_Z \to \mathbf{R}_+$ *such that*

$$\mathbb{P}_{X,Y,Z}(x,y,z) = a(x,z)b(y,z).$$

Proof of Theorem 1.13 The proof is trivial and is therefore omitted. □

Recall that for any collection of events $A_1, \ldots, A_n$,

$$\mathbb{P}(A_1 \cap \ldots \cap A_n) = \mathbb{P}(A_1)\mathbb{P}(A_2|A_1)\ldots\mathbb{P}(A_n|A_1 \cap \ldots \cap A_{n-1}).$$

Clearly, any probability distribution $\mathbb{P}_{X_1,\ldots,X_d}$ over $\mathcal{X}$ may be factorised as

$$\mathbb{P}_{X_1,\ldots,X_d} = \mathbb{P}_{X_{\sigma(1)}} \prod_{j=2}^{d} \mathbb{P}_{X_{\sigma(j)}|X_{\sigma(1)},\ldots,X_{\sigma(j-1)}}$$

for any permutation σ of $1,\ldots,d$. Let $\mathrm{Pa}^{(\sigma)}(j) \subset \{\sigma(1),\ldots,\sigma(j-1)\}$ satisfy

- $$X_{\sigma(j)} \perp \{X_{\sigma(1)},\ldots,X_{\sigma(j-1)}\} \backslash X_{\mathrm{Pa}^{(\sigma)}(j)} | X_{\mathrm{Pa}^{(\sigma)}(j)}$$

- $$X_{\sigma(j)} \not\perp \{X_{\sigma(1)},\ldots,X_{\sigma(j-1)}\} \backslash X_{\Theta(j)} | X_{\Theta(j)}$$

  for any strict subset $\Theta_j \subset \mathrm{Pa}_j^{(\sigma)}$.

Then, by the first characterisation of conditional independence and setting $\mathbb{P}_{X_{\sigma(j)}|X_{\mathrm{Pa}^{(\sigma)}(j)}} = \mathbb{P}_{X_{\sigma(j)}}$ when $\mathrm{Pa}_{\sigma(j)} = \varnothing$ the empty set,

$$\mathbb{P}_{X_1,\ldots,X_d} = \prod_{j=1}^{d} \mathbb{P}_{X_{\sigma(j)}|X_{\mathrm{Pa}^{(\sigma)}(j)}}.$$

Definition 1.14 (Factorisation, Bayesian Network)**.** *A* factorisation *of a probability distribution is a decomposition*

$$\mathbb{P}_{X_1,\ldots,X_d} = \prod_{j=1}^{d} \mathbb{P}_{X_{\sigma(j)}|X_{\Xi^{(\sigma)}(j)}} \tag{1.8}$$

such that for each $j \in \{1,\ldots,d\}$, $\Xi^{(\sigma)}(j) \subseteq \{\sigma(1),\ldots,\sigma(j-1)\}$.
A Bayesian Network *is a factorisation of a probability distribution*

$$\mathbb{P}_{X_1,\ldots,X_d} = \prod_{j=1}^{d} \mathbb{P}_{X_{\sigma(j)}|X_{Pa^{(\sigma)}(j)}} \tag{1.9}$$

such that

1. $Pa_1^{(\sigma)} = \varnothing$ *(the empty set)*
2. $Pa_j^{(\sigma)} \subseteq \{\sigma(1),\ldots,\sigma(j-1)\}$
3. $X_{\sigma(j)} \perp \{X_{\sigma(1)},\ldots,X_{\sigma(j-1)}\} \backslash X_{Pa^{(\sigma)}(j)} | X_{Pa^{(\sigma)}(j)}$
4. *For any strict subset* $\Theta(j) \subset Pa^{(\sigma)}(j)$ *of* $Pa^{(\sigma)}(j)$,
$$X_{\sigma(j)} \not\perp \{X_{\sigma(1)},\ldots,X_{\sigma(j-1)}\} \backslash X_{\Theta(j)} | X_{\Theta(j)}.$$

Unless otherwise stated, it will be assumed that the variables are labelled in such a way that $\sigma = I$, the identity.

For $\mathrm{Pa}_j = \{l_{j,1},\ldots,l_{j,m_j}\}$, the state space of $X_{\mathrm{Pa}(j)}$ is $\mathcal{X}_{l_{j,1}} \times \ldots \times \mathcal{X}_{l_{j,m_j}}$. For discrete variables, there are $q_j = \prod_{a=1}^{m_j} k_{l_{j,a}}$ configurations. These may be labelled $(\pi_j^{(l)})_{l=1}^{q_j}$ and the parameters required for the probability distribution $\mathbb{P}_{X_1,\ldots,X_d}$ are

$$\theta_{jil} = \mathbb{P}_{X_j|X_{\mathrm{Pa}(j)}}(i|\pi_j^{(l)}) \quad j = 1,\ldots,d \quad i = 0,\ldots,k_j-1, \quad l = 1,\ldots,q_j.$$

The factorisation of Equation (1.8) Definition 1.14 may be represented by a *Directed Acyclic Graph*. For example, if the probability distribution over X, Y, Z, W satisfies

$$\mathbb{P}_{X,Y,Z,W} = \mathbb{P}_X \mathbb{P}_{Y|X} \mathbb{P}_{Z|X} \mathbb{P}_{W|Y,Z},$$

the factorisation may be represented by the graph in Figure 1.1.

1.2.1 Directed Acyclic Graphs and Probability Distributions

Now consider a random vector $X = (X_1,\ldots,X_d)$.

Definition 1.15 (Factorisation along a Directed Acyclic Graph)**.** *A decomposition of a probability distribution over a random vector* $X = \{X_1,\ldots,X_d\}$ *which satisfies Equation* (1.8) *with respect to an ordering* σ *is said to factorise according to a directed acyclic graph* (V, D) *if* $V = \{1,\ldots,d\}$*, the indexing set for the variables, is the node set of the graph and for each* $j = 1,\ldots,d$, $\Xi^{(\sigma)}(j)$ *is the parent set for node* $\sigma(j)$*. The factorisation corresponds to a Bayesian network if* $\Xi^{(\sigma)}(j) = Pa^{(\sigma)}(j)$ *where* $X_{\sigma(j)} \not\perp \{X_{\sigma(1)},\ldots,X_{\sigma(j-1)}\} \backslash X_{\Theta(j)} | X_{\Theta(j)}$ *for any strict subset* $\Theta(j) \subset Pa^{(\sigma)}(j)$.

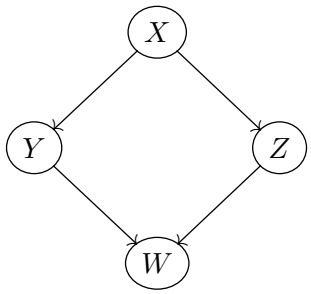

Figure 1.1: DAG representing the factorisation of a probability distribution

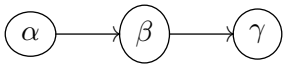

Figure 1.2: A Chain Connection

1.2.2 Connections in a Directed Acyclic Graph and Conditional Independence

Definition 1.16 (Instantiated)**.** *When the state of variable is known, the variable is said to be* instantiated.

Within a directed acyclic graph, there are three basic ways in which two nodes α, γ such that $(\alpha, \gamma) \notin D$ and $(\gamma, \alpha) \notin D$ can be connected via a third node. They are the *chain*, *fork* and *collider* connections respectively.

Chain Connections A chain connection between nodes α and γ is a connection via a node β such that the graph contains directed edges $\alpha \to \beta$ and $\beta \to \gamma$, but no edge between α and γ.

Consider a probability distribution over $(X_\alpha, X_\beta, X_\gamma)$ factorised according to the graph in Figure 1.2, as $\mathbb{P}_{X_\alpha}\mathbb{P}_{X_\beta|X_\alpha}\mathbb{P}_{X_\gamma|X_\beta}$.
Clearly, $X_\alpha \not\perp X_\gamma$ in general;

$$\mathbb{P}_{X_\alpha,X_\gamma}(x_1, x_3) = \mathbb{P}_{X_\alpha}(x_1) \sum_{x_2 \in \mathcal{X}_2} \mathbb{P}_{X_\beta|X_\alpha}(x_2|x_1)\mathbb{P}_{X_\gamma|X_\beta}(x_3|x_2)$$

and, without further assumptions, this cannot be expressed in product form.

Conditioned on the instantiation $X_\beta = x_2$,

$$\begin{aligned}\mathbb{P}_{X_\alpha,X_\gamma|X_\beta}(.,.|x_2) &= \frac{\mathbb{P}_{X_\alpha,X_\beta,X_\gamma}(.,x_2,.)}{\mathbb{P}_{X_\beta}(x_2)} = \frac{\mathbb{P}_{X_\alpha}(.)\mathbb{P}_{X_\beta|X_\alpha}(x_2|.)\mathbb{P}_{X_\gamma|X_\beta}(.|x_2)}{\mathbb{P}_{X_\beta}(x_2)} \\ &= \left(\frac{\mathbb{P}_{X_\alpha}(.)\mathbb{P}_{X_\beta|X_\alpha}(x_2|.)}{\mathbb{P}_{X_\beta}(x_2)}\right)\left(\mathbb{P}_{X_\gamma|X_\beta}(.|x_2)\right) = \left(\mathbb{P}_{X_\alpha|X_\beta}(.|x_2)\right)\left(\mathbb{P}_{X_\gamma|X_\beta}(.|x_2)\right)\end{aligned}$$

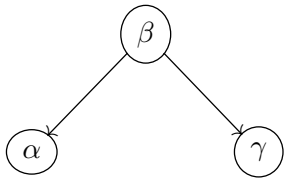

Figure 1.3: A Fork Connection

where Bayes rule has been used and so, following characterisation 3 of conditional independence from Theorem 1.13, $X_\alpha \perp X_\gamma | X_\beta$.

Fork Connections A *fork* connection between two nodes X_α and X_γ is a situation where there is no edge between X_α and X_γ, but there is a node X_β such that the graph contains directed edges $X_\beta \mapsto X_\alpha$ and $X_\beta \mapsto X_\gamma$. It is illustrated in Figure 1.3.

A distribution over the variables $(X_\alpha, X_\beta, X_\gamma)$ that factorises according to the DAG in Figure 1.3 has factorisation

$$\mathbb{P}_{X_\alpha, X_\beta, X_\gamma} = \mathbb{P}_{X_\beta} \mathbb{P}_{X_\alpha | X_\beta} \mathbb{P}_{X_\gamma | X_\beta}.$$

It is clear that $X_\alpha \not\perp X_\gamma$ in general;

$$\mathbb{P}_{X_\alpha, X_\gamma}(x_1, x_3) = \sum_{x_2 \in \mathcal{X}_2} \mathbb{P}_{X_\beta}(x_2) \mathbb{P}_{X_\alpha | X_\beta}(x_1 | x_2) \mathbb{P}_{X_\gamma | X_\beta}(x_3 | x_2)$$

and, without further assumptions, this cannot be expressed in product form. Conditioned on X_β, though:

$$\mathbb{P}_{X_\alpha, X_\gamma | X_\beta} = \frac{\mathbb{P}_{X_\alpha, X_\gamma, X_\beta}}{\mathbb{P}_{X_\beta}} = \frac{\mathbb{P}_{X_\beta} \mathbb{P}_{X_\alpha | X_\beta} \mathbb{P}_{X_\gamma | X_\beta}}{\mathbb{P}_{X_\beta}} = \mathbb{P}_{X_\alpha | X_\beta} \mathbb{P}_{X_\gamma | X_\beta}.$$

It follows that $X_\alpha \perp X_\gamma | X_\beta$ following characterisation 3) from the characterisations of conditional independence listed in the statement of Theorem 1.13.

Collider Connections A *collider connection* between two nodes α and γ is a connection such that the graph does not contain an edge between α and γ, but there is a node β such that the graph contains directed edges $\alpha \mapsto \beta$ and $\gamma \mapsto \beta$. A collider connection is illustrated in Figure 1.4.

The factorisation of the distribution $\mathbb{P}_{X_\alpha, X_\beta, X_\gamma}$ corresponding to the DAG for the collider is

$$\mathbb{P}_{X_\alpha, X_\beta, X_\gamma} = \mathbb{P}_{X_\alpha} \mathbb{P}_{X_\gamma} \mathbb{P}_{X_\beta | X_\alpha, X_\gamma}.$$

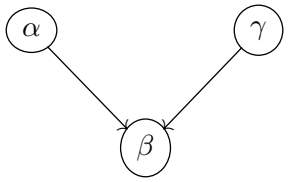

Figure 1.4: A Collider Connection

In general, $X_\alpha \not\perp X_\gamma | X_\beta$. But for each $(x, z) \in \mathcal{X}_\alpha \times \mathcal{X}_\gamma$,

$$\begin{aligned}
\mathbb{P}_{X_\alpha, X_\gamma}(x, z) &= \sum_{y \in \mathcal{X}_\beta} \mathbb{P}_{X_\alpha}(x)\mathbb{P}_{X_\gamma}(z)\mathbb{P}_{X_\beta | X_\alpha, X_\gamma}(y|x, z) \\
&= \mathbb{P}_{X_\alpha}(x)\mathbb{P}_{X_\gamma}(z) \sum_{y \in \mathcal{X}_\beta} \mathbb{P}_{X_\alpha | X_\beta, X_\gamma}(y|x, z) \\
&= \mathbb{P}_{X_\alpha}(x)\mathbb{P}_{X_\gamma}(z).
\end{aligned}$$

so that $X_\alpha \perp X_\gamma$.

A Causal Interpretation So far, the discussion has considered sets of random variables where, based on the ordering of the variables, the parent set of a variable is a subset of those of a lower order. The representation of a probability distribution by factorising along a Directed Acyclic Graph may be particularly useful if there are cause to effect relations between the variables, the ancestors being the cause and the descendants the effect. For a causal model, the connections have the following interpretations:

Fork Connection: Common cause For the fork connection, illustrated by Figure 1.2, X_β may be a *cause* that influences both X_α and X_γ which are *effects*. The variables are only related through X_β. The situation is illustrated by the following example, taken from a cartoon by Albert Engström; 'during a convivial discussion at the bar one evening, about the unhygienic nature of galoshes, one of the participants pipes up, "you have a very good point there. Every time I wake up wearing my galoshes, I have a sore head."

Let X_α denote the state of the feet and X_γ the state of the head. These two variables are related; $X_\alpha \not\perp X_\beta$. But there is a common cause; X_2, which denotes the activities of the previous evening. Once it is known that he has spent a convivial evening drinking, the state of the feet gives no further information about the state of the head; $X_\alpha \perp X_\gamma | X_\beta$.

Chain Connection This may similarly be understood as cause to effect. X_α influences X_β, which in turn influences X_γ, but there is no direct causal relationship between the values taken by X_α and those taken by X_γ. If X_β is unknown, then $X_\alpha \not\perp X_\gamma$, but once the state of X_β is established, X_α and X_γ give no further information about each other; $X_\alpha \perp X_\gamma | X_\beta$.

Collider Connection For the collider connection, X_α and X_β are unrelated; $X_\alpha \perp X_\gamma$. But they both influence X_β. For example, consider a burglar alarm (X_β) that is activated if a burglary takes place, but can also be activated if there is a minor earth tremor.

One day, somebody calls you while you are at work to say that your burglar alarm is activated. You get into the car to go home. But on the way home, you hear on the radio that there has been an earth tremor in the area. As a result, you return to work.

Once X_β is instantiated, the information that there has been an earth tremor influences the likelihood that a burglary has taken place; $X_\alpha \not\perp X_\gamma | X_\beta$.

This is known as *explaining away*.

Attention is now turned to trails within a DAG, and characterisation of those along which information can pass.

Definition 1.17 (S-Active Trail). *Let $\mathcal{G} = (V, D)$ be a directed acyclic graph. Let $S \subset V$ and let $\alpha, \beta \in V \backslash S$. A trail τ between the two variables α and β is said to be* S-active *if*

1. *Every collider node in τ is in S, or has a descendant (Definition 1.8) in S.*
2. *Every other node is outside S.*

Definition 1.18 (Blocked Trail). *A trail between α and β that is not S-active is said to be* blocked by S.

The following definition is basic; it will be seen that if a probability distribution factorises along a DAG $\mathcal{G}$ and two nodes α and β are D-separated by S, then $X_\alpha \perp X_\beta | X_S$.

Definition 1.19 (D-separation). *Let $\mathcal{G} = (V, D)$ be a directed acyclic graph, where $V = \{1, \ldots, d\}$. Let $S \subset V$. Two distinct nodes α and β not in S are* D-separated *by S if all trails between α and β are blocked by S.*

Let A and B denote two sets *of nodes. If every trail from any node in A to any node in B is blocked by S, then the sets A and B are said to be D-separated by S. This is written*

$$A \perp\!\!\!\perp B \parallel_{\mathcal{G}} S. \tag{1.10}$$

The terminology D-separation is short for *directed* separation. The insertion of the letter 'D' points out that this is not the standard use of the term 'separation' found in graph theory.

Definition 1.20 (D-connected). *If two nodes α and β are not D-separated, they are said to be D-connected.*

Notation The notation $\alpha \not\perp\!\!\!\perp \beta \parallel_{\mathcal{G}} S$ denotes that α and β are D-connected by S in the DAG $\mathcal{G}$. Here α and β may refer to individual nodes or sets of nodes.

Example 1.21.

Consider the chain connection $\alpha \mapsto \beta \mapsto \gamma$ in the DAG in Figure 1.2 and the fork connection of Figure 1.3. For the chain connection of Figure 1.2, the D-separation statements are: $\alpha \perp\!\!\!\perp \gamma \| _{\mathcal{G}} \beta$ while $\alpha \not\perp\!\!\!\perp \gamma \| _{\mathcal{G}} \varnothing$ ($\varnothing$ denotes the empty set). For the DAG in Figure 1.3, $\alpha \perp\!\!\!\perp \gamma \| _{\mathcal{G}} \beta$ while $\alpha \not\perp\!\!\!\perp \gamma \| _{\mathcal{G}} \varnothing$. These correspond to the conditional independence statements derived for probability distributions that factorise along these graphs. For Figure 1.4, $\alpha \perp\!\!\!\perp \gamma \| _{\mathcal{G}} \varnothing$ while $\alpha \not\perp\!\!\!\perp \gamma \| _{\mathcal{G}} \beta$. Again, these statements correspond to the conditional independence statements that may be derived from the fact that a distribution factorises along the DAG of Figure 1.4. □

Let $\mathrm{MB}(\alpha)$ denote the set of nodes which are either parents of α or children of α or a node which shares a common child with α. Then α is D-separated from the rest of the network by $\mathrm{MB}(\alpha)$. This set of nodes is known as the *Markov blanket* of the node α.

Definition 1.22 (Markov Blanket)**.** *The* Markov blanket *of a node α in a DAG $\mathcal{G} = (V, D)$, denote $MB(\alpha)$, is the set consisting of the parents of α, the children of α and the nodes sharing a common child with α.*

1.2.3 Bayes Ball

The *Bayes ball* provides a convenient method for deciding whether or not two nodes are D-separated by a set S in a DAG $\mathcal{G} = (V, D)$. Variables are D-connected by a set S if the Bayes ball can be passed between them employing the following rule. The nodes which are *not* in S are depicted as unshaded; nodes in S as shaded.

Definition 1.23 (Instantiated Nodes)**.** *Let $\mathcal{G} = (V, D)$ be a directed acyclic graph. When considering statements $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} S$ and $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} S$, the nodes in S are referred to as* instantiated.

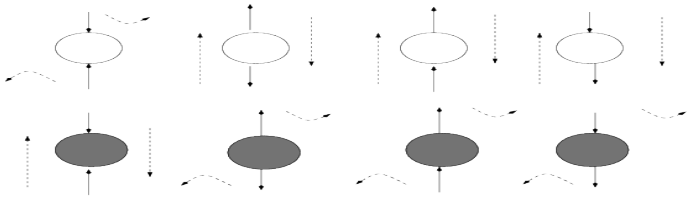

Figure 1.5: Bayes Ball

Consider the three types of connection in a DAG; chain, collider and fork.

- For the *chain* connection illustrated in Figure 1.2, the Bayes ball algorithm indicates that *if* node β is instantiated, *then* the ball does not move from α to γ through β. The communication in the trail is *blocked*. If the node is *not* instantiated, then communication is possible.
- For the *fork* connection illustrated in Figure 1.3, the algorithm states that *if* node β is instantiated, then again communication between α and γ is *blocked*. If the node is *not* instantiated, then communication is possible.
- For the *collider* connection illustrated in Figure 1.4, the Bayes ball algorithm states that the ball *does* move from α to γ if node α or any of its descendants is instantiated. If β or a descendant is instantiated, this opens communication between the parents. If neither β nor any of its descendants are instantiated, then there is no communication.

For a collider node β, instantiating any of the descendants of β *also* opens communication. If node β is *not* instantiated, *and* none of its descendants are instantiated, then there is no communication.

A DAG $\mathcal{G} = (V, D)$ satisfies the following important property:

Theorem 1.24. *A DAG $\mathcal{G} = (V, D)$ contains an edge between two nodes $\alpha, \beta \in V$ if and only if $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} S$ for any $S \subseteq V \backslash \{\alpha, \beta\}$.*

Proof The proof of this is straightforward and left as an exercise (Exercise 6 page 22). □

1.3 D-Separation and Conditional Independence

The following key result shows that if a probability distribution factorises along a given DAG $\mathcal{G}$, then every D-separation statement for the DAG implies the corresponding conditional independence statement for the distribution.

Theorem 1.25 (D-Separation Implies Conditional Independence)**.** *Let $\mathcal{G} = (V, D)$ be a directed acyclic graph and let $\mathbb{P}$ be a probability distribution that factorises along $\mathcal{G}$. Then for any three disjoint subsets $A, B, S \subset V$, it holds that $X_A \perp X_B | X_S$ (X_A and X_B are independent given X_S) if $A \perp\!\!\!\perp B \| _{\mathcal{G}} A$ (A and B are D-separated by S).*

Proof of Theorem 1.25 Let $X = (X_1, \ldots, X_d)$ be a random vector. Let $V = \{1, \ldots, d\}$ denote the set of nodes of a Directed acyclic graph $\mathcal{G}_(V, D)$ and suppose that $\mathbb{P}_X$ factorises along $\mathcal{G}$. Let $A \subset V$, $B \subset V$ and $S \subset V$ be three disjoint sets of nodes. Suppose that $A \perp\!\!\!\perp B \| _{\mathcal{G}} S$. Let A, B and S denote also the random vectors X_A, X_B and X_S respectively and let $\mathcal{X}_A$, $\mathcal{X}_B$ and $\mathcal{X}_S$ denote their respective state spaces. It is required to show that for all $a \in \mathcal{X}_A$, $b \in \mathcal{X}_B$ and $s \in \mathcal{X}_S$,

$$\mathbb{P}_{A,B|S}(a, b|s) = \mathbb{P}_{A|S}(\underline{a}|\underline{s}) \mathbb{P}_{B|S}(\underline{b}|\underline{s}).$$

Let $R = V \backslash (A \cup B \cup S)$. Let

$$E_1 = \{\alpha \in V | \text{there is an } S\text{-active trail from } A \text{ to } \alpha\}$$

$$E_2 = \{\alpha \in V | \text{there is an } S\text{-active trail from } B \text{ to } \alpha\}$$

$$R_1 = R \cap E_1 \cap E_2, \qquad R_2 = R \cap E_1 \cap E_2^c, \qquad R_3 = R \cap E_2 \cap E_1^c, \qquad R_4 = R \cap (R_1^c \cup R_2^c \cup R_3^c).$$

From Characterisation 5 of Theorem 1.13, it is required to show that there are two functions F and G such that

$$\mathbb{P}_{A,B,S}(\underline{a}, \underline{b}, \underline{s}) = F(\underline{a}, \underline{s})G(\underline{b}, \underline{s}).$$

Let $\mathbb{P}(X_j|\text{Pa}_j)$ denote the conditional probability function of X_j given the parent variables $X_{\text{Pa}(j)}$. Then

$$\begin{aligned}\mathbb{P}(X_1, \ldots, X_d) = &\prod_{j\in A} \mathbb{P}(X_j|\text{Pa}_j) \prod_{j\in B} \mathbb{P}(X_j|\text{Pa}_j) \prod_{j\in S} \mathbb{P}(X_j|\text{Pa}_j) \\ &\times \prod_{j\in R_1} \mathbb{P}(X_j|\text{Pa}_j) \prod_{j\in R_2} \mathbb{P}(X_j|\text{Pa}_j) \prod_{j\in R_3} \mathbb{P}(X_j|\text{Pa}_j) \prod_{j\in R_4} \mathbb{P}(X_j|\text{Pa}_j).\end{aligned}$$

Since all the nodes of R are uninstantiated and there is no S-active trail from A to B, it follows that any node $\alpha \in R_1$ is either a *collider* which is not in S, nor does it have any descendants (Definition 1.8) in S *or* the descendant of such a collider. Furthermore, any descendant of a variable in R_1 is also in R_1. Therefore, marginalising over the variables in R_1 does not involve the parent variables of A, B or S, nor does it involve the variables in R_2 or R_3 or their ancestors since $\sum_{\mathcal{X}_j} \mathbb{P}(X_j|\text{Pa}_j) = 1$.

There is no S-active trail from a variable in R_4 to any variable in A or B. It follows that parents of variables in R_4 are either in R_4 or in S (if the parent is not in S and there is an S-active trail between the parent and a variable in either A or B, then there is an S-active trail from the variable itself; the link between variable, its parent and the next variable on the trail is either an uninstantiated fork or uninstantiated chain connection).

Now, using $\varnothing$ to denote the empty set, let $S_2 = \{\alpha \in S \mid \text{Pa}(\alpha) \cap R_2 \neq \varnothing\}$ (there is an S-active trail from A to a parent of $\alpha \notin S$ but not from B to a parent of $\alpha \notin S$), $S_3 = \{\alpha \in S \mid \text{Pa}(\alpha) \cap R_3 \neq \varnothing\}$ (there is an S-active trail from B to a parent of $\alpha \notin S$ but not from A to a parent of $\alpha \notin S$) and $S_4 = S \cap S_2^c \cap S_3^c$ (nodes $\alpha \in S$ such that there is no S-active trail either from A to a parent of $\alpha \notin S$ or from B to a parent of $\alpha \notin S$). Then $S_2 \cap S_3 = \varnothing$, the empty set, otherwise there would be a collider node in S that would result in an active trail from A to B.

It is also clear that $\text{Pa}(S_4) \subseteq S \cup R_4$, where $\text{Pa}(S_4)$ denotes the parent variables of the variables in S_4; that is, $\text{Pa}(S_4) = \{Y|(Y, X) \in E, \quad X \in S_4\}$. The sets S_2, S_3, S_4 are disjoint. It follows that

$$\begin{aligned}\mathbb{P}(A,B,S) &= \sum_{\mathcal{X}_{R_2}}\sum_{\mathcal{X}_{R_3}}\sum_{\mathcal{X}_{R_4}}\sum_{\mathcal{X}_{R_1}}\left(\prod_{j\in R_1}\mathbb{P}(X_j|\mathrm{Pa}_j)\prod_{j\in R_4}\mathbb{P}(X_j|\mathrm{Pa}_j)\prod_{j\in S_4}\mathbb{P}(X_j|\mathrm{Pa}_j)\right)\\ &\times\left(\prod_{j\in A}\mathbb{P}(X_j|\mathrm{Pa}_j)\prod_{j\in S_2}\mathbb{P}(X_j|\mathrm{Pa}_j)\prod_{j\in R_2}\mathbb{P}(X_j|\mathrm{Pa}_j)\right)\\ &\times\left(\prod_{j\in B}\mathbb{P}(X_j|\mathrm{Pa}_j)\prod_{j\in S_3}\mathbb{P}(X_j|\mathrm{Pa}_j)\prod_{j\in R_3}\mathbb{P}(X_j|\mathrm{Pa}_j)\right).\end{aligned}$$

The sums are taken from right to left, starting with $\sum_{\mathcal{X}_{R_1}}$. None of the variables in R_1 are in $A\cup B\cup S$ or the parent sets of variables in A, B, S_2, S_3, R_2 or R_3. The parents of variables in R_4 are either in R_4 or S, the parents of variables in S_4 are either in S_4 or R_4. The parents of variables in R_3 in $R_3\cup B$. The parents of variables in R_2 are in $R_2\cup A$. It follows that $\mathbb{P}(A,B,S)$ has a factorisation

$$\mathbb{P}(A,B,S)=\psi_1(S)\psi_2(A,S)\psi_3(B,S)$$

where the constructions of ψ_1, ψ_2 and ψ_3 are clear from the context. This factorisation clearly satisfies the required criteria. It follows that D-separation implies conditional independence. □

Of course, the converse is not true in general; D-separation is a convenient way of locating some of the independence structure of a distribution. It does not, in general, locate the entire independence structure.

1.4 The Locally Directed Markov Property

This section introduces the *local directed Markov condition*, a necessary and sufficient condition so that a probability function $\mathbb{P}$ over a set of variables V can be factorised along a graph $\mathcal{G}$.

Definition 1.26 (Local Directed Markov Condition, Locally $\mathcal{G}$ - Markovian)**.** *Let $X=(X_1,\ldots,X_d)$ be a random vector. A probability function $\mathbb{P}$ over X satisfies the* local directed Markov condition *with respect to a DAG $\mathcal{G}=(V,D)$ with node set $V=\{1,\ldots,d\}$ or, equivalently, is said to be* locally $\mathcal{G}$-Markovian *if and only if there is an ordering of the variables σ such that $Pa^{(\sigma)}(j)\in\{\sigma(1),\ldots,\sigma(j-1)\}$ for each $j\in\{1,\ldots,d\}$ and such that $X_{\sigma(j)}$ is conditionally independent, given $X_{Pa^{(\sigma)}(j)}$ of $X_{V\setminus(V^{(\sigma)}(j)\cup Pa^{(\sigma)})(j)}$, where $V^{(\sigma)}(j)$ is the set of all* descendants *of $\sigma(j)$ in $\mathcal{G}$. That is,*

$$V^{(\sigma)}(j)=\{\beta\in V|\text{there is a directed path from }\sigma(j)\text{ to }\beta\} \tag{1.11}$$

and:

$$X_{\sigma(j)}\perp X_{V\setminus(V^{(\sigma)}(j)\cup Pa^{(\sigma)}(j)\cup\{\sigma(j)\})}|X_{Pa^{(\sigma)}(j)}.$$

Proposition 1.27. *Let $\mathbb{P}$ be a probability distribution over a random vector $X=(X_1,\ldots,X_d)$. Then $\mathbb{P}$ satisfies the l.d.m.p. with respect to a graph $\mathcal{G}=(V,D)$ if and only if there is an ordering of the variables σ such that $\mathbb{P}$ factorises along $\mathcal{G}$.*

Proof Firstly, assume that $\mathbb{P}$ is locally $\mathcal{G}$-Markovian and assume that the variables are ordered in such a way that for each $j \in \{1,\ldots,d\}$, $X_j \perp X_{V\setminus(V(j)\cup \mathrm{Pa}(j))}|X_{\mathrm{Pa}(j)}$ where $V(j)$ is defined in Equation (1.11). Let $\pi_j(x_1,\ldots,x_{j-1})$ denote the instantiation of $X_{\mathrm{Pa}(j)}$ when X is instantiated as $(x_1,\ldots,x_d)$. By Characterisation 1) of Theorem 1.13, for all $j = 1,\ldots,d$ and any π_j such that $\mathbb{P}_{X_{\mathrm{Pa}(j)}}(\pi_j) > 0$,

$$\mathbb{P}_{X_j|X_1,\ldots,X_{j-1}}(x_j|x_1,\ldots,x_{j-1}) = \mathbb{P}_{X_j|X_{\mathrm{Pa}(j)}}(x_j|\pi_j)$$

with $\mathbb{P}_{X_j|X_1,\ldots X_{j-1}}(x_j|x_1,\ldots,x_{j-1}) = \mathbb{P}_{X_j}(x_j)$ if $\mathrm{Pa}_j = \varnothing$. It follows directly that

$$\mathbb{P}_{X_1,\ldots,X_d} = \prod_{j=1}^{d} \mathbb{P}_{X_j|X_{\mathrm{Pa}(j)}}$$

and hence, by definition, that $\mathbb{P}$ factorises along $\mathcal{G}$.

Secondly, suppose that $\mathbb{P}$ factorises along a directed acyclic graph $\mathcal{G} = (V, D)$. Then it is clear (for example by using the Bayes ball algorithm) that

$$X_j \perp\!\!\!\perp X_{V\setminus(V(j)\cup \mathrm{Pa}(j))} \| _{\mathcal{G}} X_{\mathrm{Pa}(j)}$$

where $V(j)$ is the set of variables defined by Equation (1.11). If $\mathrm{Pa}(j)$ is instantiated, then any trail from j to $V\setminus(V_j \cup \mathrm{Pa}_j \cup \{X_j\})$ has to pass through a node in Pa_j, which will be either a chain or fork connection. It follows from Theorem 1.25 that

$$X_j \perp X_{V\setminus(V(j)\cup \mathrm{Pa}(j)\cup\{j\})}|X_{\mathrm{Pa}(j)},$$

from which it follows that $\mathbb{P}$ is locally $\mathcal{G}$-Markovian. □

Once a probability distribution has been factorised according to a Bayesian Network, the next task is to use it to answer *queries*.

Definition 1.28 (Query)**.** *A* query *in probabilistic inference is simply a conditional probability distribution, over the variables of interest (the* query *variables) conditioned on information received.*

Discussion of the main algorithms for answering queries is the subject of chapters 7 and 8.

1.5 Quick Medical Reference - Decision Theoretic: An Example

In *classification* problems, the aim is to infer the value of a *class* variable, given the values of the observables. It is often unrealistic to hope to obtain a full profile of the probability distribution; the aim is rather to exploit enough of the structure to obtain a good classifier.

We now consider the *noisy logic gate*, which we express as a Bayesian Network and give the QMR - DT (Quick Medical Reference - Decision Theoretic) data base of diseases and symptoms as an example.

A disease *may* result in a symptom, but this is not certain. The noisy logic gate approximation may be used to construct a classifier; given the symptoms exhibited, the problem is to diagnose the illnesses.

1.5.1 Propositional Logic and Noisy Logic Gates

In logic, the 'Or' disjunction of two propositions p and q is denoted by $p \vee q$ and is defined by the truth table

p	q	$p \vee q$
1	1	1
1	0	1
0	1	1
0	0	0

while the 'And' disjunction of two propositions p and q, denoted by $p \wedge q$ is defined by the truth table

p	q	$p \wedge q$
1	1	1
1	0	0
0	1	0
0	0	0

Here 1 = the proposition is true, 0 = the proposition is false.

Now consider the situation where p and q are *independent* causes of some effect, but that p and q only cause the effect with some probability less than 1.

1.5.2 QMR - DT Data Base

The QMR - DT database is a large scale probabilistic data base that is intended to be used as a diagnostic aid in the domain of internal medicine. It stores information on approximately 600 diseases and approximately 4000 symptoms. The quantities $\mathbb{P}_D$ (the joint probability function for the selection of diseases that a randomly chosen individual may have) and $\mathbb{P}_{S|D}$ (the joint probability function that the victim exhibits a selection of symptoms given a particular selection of diseases) are estimated from the data bank. Consider the example of *diseases* and *symptoms*. Let q_{0j} denote the probability that symptom j is present in the absence of any disease and q_{ij} the probability that disease i induces symptom j. Using $S = (S_1, \ldots, S_n)$ to denote *symptoms* and $D = (D_1, \ldots, D_m)$ to denote *diseases*, an instantiation s of S will be a n-vector where each entry is either 0 to denote that the symptom is absent, or 1 to denote that it is present. Similarly, an instantiation d of D is an m-vector of 1's and 0's where 1 corresponds to presence and 0 to absence of the corresponding disease.

Under the modelling assumption, the probability that symptom j is absent, given a vector of diseases d is

$$\mathbb{P}_{S_j|D}(0|d) = (1 - q_{0j}) \prod_i (1 - q_{ij})^{d_i}.$$

Another simplifying assumption is that an individual contracts different diseases independently of each other. Under this assumption,

$$\mathbb{P}_D = \prod_{i=1}^{m} \mathbb{P}_{D_i}.$$

For the problem of *classification*, that is diagnosing diseases given a list of symptoms, these two modelling assumptions come under the umbrella *independence of competing risks*. This is a simplification, but nevertheless, can produce an effective classifier.

Noisy 'or' as a causal network Consider the DAG given in Figure 1.6 where $B = A_1 \vee A_2 \vee \ldots \vee A_n$. This is the logical 'or' and there is no noise. The noise then enters, as in the DAG given in Figure 1.7, by considering that if any of the variables A_i, $i = 1, \ldots, n$ is present, then B is present *unless something has inhibited it*, the inhibitors on each variable acting independently of each other.

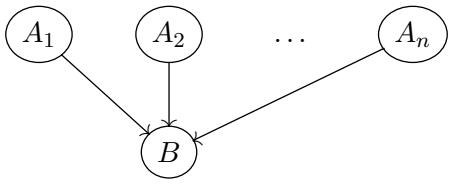

Figure 1.6: A Logical 'Or' Gate

Noisy 'or': inhibitors Consider the DAG in figure 1.7, where q_i denotes the probability that the impact of A_i is inhibited.

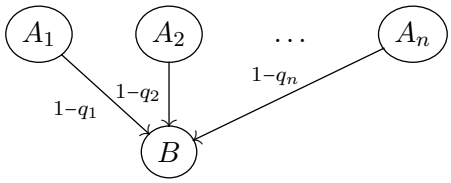

Figure 1.7: Noisy 'Or' Junction

All variables are binary, and take value 1 if the cause, or effect, is present and 0 otherwise. In other words, $\mathbb{P}_{B|A_i}(0|1) = q_i$. The assumption from the DAG is that all the inhibitors are independent. This implies that

$$\mathbb{P}_{B|A_1,\ldots,A_n}(0|a_1, \ldots, a_n) = \prod_{j \in Y} q_j,$$

where $Y = \{j \in \{1, \ldots, n\} | a_j = 1\}$. This may be described by a *noisy 'or' gate*.

Noisy 'or' Gate The noisy 'or' can be modelled directly, introducing the variables B_i $i = 1, \ldots, n$, where B_i takes the value 1 if the cause A_i is on and it is not inhibited and 0 otherwise. The corresponding DAG is given in figure 1.8

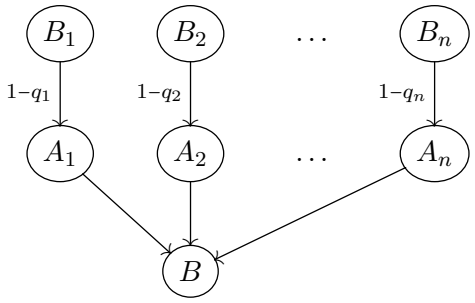

Figure 1.8: Noisy 'Or' Gate

where

$$\mathbb{P}_{B|B_1,\ldots,B_n}(1|b_1,\ldots,b_n) = b_1 \vee \ldots \vee b_n.$$

The $B_1, \ldots, B_n$ are introduced as mutually independent inhibitors, and

$$\mathbb{P}_{B_i|A_i}(0|1) = q_i,$$

giving the result given above.

Notes The models that were later to be called Bayesian networks were introduced into artificial intelligence by J. Pearl, in the article [103] (from 1982). Within the Artificial Intelligence literature, this is a seminal article. Perhaps the earliest work that uses directed graphs to represent possible dependencies among random variables is that by S. Wright (1921) [146]. An early article that considered the notion of a factorisation of a probability distribution along a directed acyclic graph representing causal dependencies is that by H. Kiiveri, T.P. Speed and J.B. Carlin (1984) [74], where a Markov property for Bayesian networks was defined. This was developed by J. Pearl in [106] (from 1990). D-separation, and the extent to which it characterises independence is discussed by J.Pearl and T. Verma in [112] and by J.Pearl, D. Geiger and T. Verma in [111]. The Bayes ball is taken from R.D. Schachter [122]. The results for identifying independence in Bayesian networks are taken from D. Geiger, T. Verma and J. Pearl [51].

1.6 Exercises

1. Let (X, Y, W, Z) be disjoint sets of random variables, each with a finite state space. Prove that the following logical relations hold:

   (a) **decomposition** Prove that if $X \perp Y \cup W | Z$ then $X \perp Y | Z$ and $X \perp W | Z$.

   (b) **contraction** Prove that if $X \perp Y | Z$ and $X \perp W | Y \cup Z$ then $X \perp W \cup Y | Z$.

   (c) **weak union** Prove that if $X \perp Y \cup Z | W$ then $X \perp Y | Z \cup W$.

   (d) **intersection** Prove that if $X \perp Y | W \cup Z$ and $X \perp W | Y \cup Z$ then $X \perp W \cup Y | Z$.

2. Let (X, Y, W, Z) be four sets of nodes in a DAG $\mathcal{G} = (V, D)$. Prove the following;

   (a) **decomposition** Prove that if $X \perp\!\!\!\perp Y \cup W \| _{\mathcal{G}} Z$ then $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z$ and $X \perp\!\!\!\perp W \| _{\mathcal{G}} Z$.

   (b) **contraction** Prove that if $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z$ and $X \perp\!\!\!\perp W \| _{\mathcal{G}} Y \cup Z$ then $X \perp\!\!\!\perp W \cup Y \| _{\mathcal{G}} Z$.

   (c) **weak union** Prove that if $X \perp\!\!\!\perp Y \cup Z \| _{\mathcal{G}} W$ then $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z \cup W$.

   (d) **intersection** Prove that if $X \perp\!\!\!\perp Y \| _{\mathcal{G}} W \cup Z$ and $X \perp\!\!\!\perp W \| _{\mathcal{G}} Y \cup Z$ then $X \perp\!\!\!\perp W \cup Y \| _{\mathcal{G}} Z$.

3. Let $\mathcal{X}$ denote the state space for (X, Y, W, Z) and assume that $\mathbb{P}_{X,Y,W,Z}(x, y, w, z) > 0$ for each $(x, y, w, z) \in \mathcal{X}$. Does it hold in general that if $X \perp Y \mid Z \cup W$ and $X \perp W \mid Z \cup Y$, then $X \perp Z \mid Y \cup W$? Either prove the result or illustrate why it is false.

4. Let $V = A \cup B \cup S$ where A, B and S are disjoint subsets and suppose that $A \perp B | S$. Prove that for any $\alpha \in A$ and $\gamma \in B$,

$$\alpha \perp \gamma | (A \cup S) \backslash \{\alpha, \gamma\} \Leftrightarrow \alpha \perp \gamma | (A \cup B \cup S) \backslash \{\alpha, \gamma\}.$$

5. Let A be a variable in a DAG. Prove that if all the variables in the *Markov blanket* of A are instantiated, then A is d-separated from the remaining uninstantiated variables.

6. Prove Theorem 1.24.

7. Let $\mathcal{G} = (V, D)$ denote a directed acyclic graph. Let $X \subseteq V$, $Y \subseteq V$ and $Z \subseteq V$ denote sets of nodes and let $\alpha, \beta, \gamma, \delta \in V \backslash X \cup Y \cup Z$ denote individual nodes.

   (a) Prove that if $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z$ and $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z \cup \{\gamma\}$ then either $X \perp\!\!\!\perp \{\gamma\} \| _{\mathcal{G}} Z$ or $Y \perp\!\!\!\perp \{\gamma\} \| _{\mathcal{G}} Z$

   (b) Prove that if $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \{\gamma, \delta\}$ and $\gamma \perp\!\!\!\perp \delta \| _{\mathcal{G}} \{\alpha, \beta\}$ then either $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \{\gamma\}$ or $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \{\delta\}$.

8. The notation $\underline{X}_A$ is used to denote the random (row) vector of all variables in set A. Let $V = \{X_1, \ldots, X_d\}$ be the d variables of a Bayesian network and assume that $\underline{X}_{V \backslash \{X_i\}} = \underline{w}$. That is, all the variables except X_i are instantiated. Assume that X_i is a binary variable, taking values 0 or 1. The *odds* of an event A given B is defined as:

$$O_{\mathbb{P}}(A|B) = \frac{\mathbb{P}(A|B)}{\mathbb{P}(A^c|B)}$$

where A^c denotes the complement of A. Consider the odds

$$O_{\mathbb{P}}\left(\{X_i = 1\} \mid \{\underline{X}_{V \setminus \{X_i\}} = \underline{w}\}\right),$$

and show that this depends only on the variables in the Markov blanket (Definition 1.22) of X_i.

1.7 Answers

1. (a) $X \perp Y \cup W|Z$ means $\mathbb{P}_{W,X,Y,Z}(w,x,y,z) = \mathbb{P}_{X|Z}(x|z)\mathbb{P}_{W,Y|Z}(w,y|z)\mathbb{P}_Z(z)$ Summing over W gives $\mathbb{P}_{X,Y,Z}(x,y,z) = \mathbb{P}_{X|Z}(x|z)\mathbb{P}_{Y|Z}(y|z)\mathbb{P}_Z(z)$; equivalent to $X \perp Y|Z$.
   Similarly, summing over Y gives $\mathbb{P}_{W,X,Z}(w,x,z) = \mathbb{P}_{X|Z}(x|z)\mathbb{P}_{W|Z}(w|z)\mathbb{P}_Z(z)$, equivalent to $X \perp W|Z$.

   (b) $X \perp Y|Z$ implies $\mathbb{P}_{X,Y,Z}(x,y,z) = \mathbb{P}_{X|Z}(x|z)\mathbb{P}_{Y|Z}(y|z)\mathbb{P}_Z(z)$ and $X \perp W|Y \cup Z$ implies $\mathbb{P}_{W,X,Y,Z}(w,x,y,z) = \mathbb{P}_{X|Y,Z}(x|y,z)\mathbb{P}_{W|Y,Z}(w|y,z)\mathbb{P}_{Y,Z}(y,z)$. The first statement implies that for (x,y,z) such that $\mathbb{P}_{X,Y,Z}(x,y,z) > 0$, $\mathbb{P}_{X|Y,Z} = \mathbb{P}_{X|Z}$, so, using $\mathbb{P}_{Y,Z} = \mathbb{P}_{Y|Z}p_Z$, it follows that

$$\begin{aligned}\mathbb{P}_{W,X,Y,Z}(w,x,y,z) &= \mathbb{P}_{X|Z}(x|z)\mathbb{P}_{W|Y,Z}(w|y,z)\mathbb{P}_{Y|Z}(y|z)\mathbb{P}_Z(z)\\ &= \mathbb{P}_{X|Z}(x|z)\mathbb{P}_{W,Y|Z}(w,y|z)\mathbb{P}_Z(z),\end{aligned}$$

   so that $X \perp W \cup Y|Z$.

   (c)
$$\mathbb{P}_{XYZW} = \frac{\mathbb{P}_{XW}\mathbb{P}_{WYZ}}{\mathbb{P}_W} = a_{XW}b_{YZW}$$
   where $a_{XW} = \frac{\mathbb{P}_{XW}}{\mathbb{P}_W}$ and $b_{YZW} = \mathbb{P}_{WYZ}$ so that $X \perp Y|Z \cup W$ from the characterisations of independence.

   (d)
$$\mathbb{P}_{X,Y,W,Z} = \frac{\mathbb{P}_{XWZ}\mathbb{P}_{YWZ}}{\mathbb{P}_{WZ}} = \frac{\mathbb{P}_{XYZ}\mathbb{P}_{WYZ}}{\mathbb{P}_{YZ}}$$
$$\frac{\mathbb{P}_{XWZ}}{\mathbb{P}_{WZ}} = \frac{\mathbb{P}_{XYZ}}{\mathbb{P}_{YZ}}$$
   so that
$$\mathbb{P}_{X|WZ} = \mathbb{P}_{X|YZ} = \mathbb{P}_{X|Z}$$
   giving
$$\mathbb{P}_{X,Y,W,Z} = \frac{\mathbb{P}_{XZ}\mathbb{P}_{YWZ}}{\mathbb{P}_Z}$$
   and hence
$$X \perp Y \cup W|Z.$$

2. (a) This is clear from the definition: Z blocks all trails between X and Y and all trails between X and W.

   (b) Consider $\alpha \in X$ and $\beta \in W$. Any trail $\alpha \leftrightarrow \beta$ has either an instantiated fork or chain node in $Y \cup Z$ or an uninstantiated collider that is not in $Y \cup Z$, neither any of its descendants. It follows that such an uninstantiated collider is not in Z, neither are any of its descendants. If it has an instantiated fork or chain node in Y, then the trail from α to the instantiated fork or chain in Y is blocked by Z since $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z$. Hence $X \perp\!\!\!\perp W \cup Y \| _{\mathcal{G}} Z$.

(c) Let $\alpha \in X$ and $\beta \in Y$. Any trail is blocked by W. That is, it has either a fork or chain node in W or a collider node that is not in W, neither are any of its descendants.

If it is blocked by a chain or fork in W, then the trail is also blocked by $Z \cup W$. Consider the first collider on the trail, proceeding from α, not in W, with no descendants in W, that is either in Z or has a descendant in Z. Then the trail between α and the node in Z is blocked by W since $X \perp\!\!\!\perp Y \cup Z \| _{\mathcal{G}} W$. Since neither the collider nor any of its descendants are in W, it follows that the trail between α and the collider node is blocked by W, from which it follows that it has a chain or fork in W, from which it follows that $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z \cup W$.

(d) Let $\alpha \in X$ and $\beta \in Y$. Any trail between them with no other nodes in X or Y is blocked by $W \cup Z$. That is, it has either a fork or chain node in $W \cup Z$ or a collider not in $W \cup Z$ with no descendants in $W \cup Z$. Such a collider is therefore not in Z and has no descendants in Z.

Assume that the trail blocked by $W \cup Z$ is not blocked by Z. Let γ be the first fork or chain node along the trail that is in W. This trail is Z active, but is blocked by $Y \cup Z$. It therefore contains a fork or chain node in Y, contradicting the assertion that α was the only node in X and β the only node in Y on the trail.

3. The result stated is false. Counterexample: any distribution that factorises as

$$\mathbb{P}_Z \mathbb{P}_{X|Z} \mathbb{P}_{W|Z} \mathbb{P}_{Y|W,Z}$$

clearly satisfies $X \perp Y | Z \cup W$ and $X \perp W | Y \cup Z$, but there are distributions with such a factorisation that do not satisfy $X \perp Z | Y \cup W$.

4. Since $A \perp B | S$, it follows from the weak union result in Exercise 1 that $\alpha \perp B | A \cup S \backslash \{\alpha\}$. This, together with the condition $\alpha \perp \gamma | A \cup S \backslash \{\alpha, \gamma\}$ imply (using $X = \{\alpha\}$, $W = B$, $Z = A \cup S \backslash \{\alpha, \gamma\}$, $Y = \{\gamma\}$ in the **contraction** statement Exercise 1) that

$$\alpha \perp \gamma | A \cup B \cup S \backslash \{\alpha, \gamma\}.$$

as required.

Now suppose that $\alpha \perp \gamma | (A \cup B \cup S) \backslash \{\alpha, \gamma\}$. Since $A \perp B | S$, it follows that $\alpha \perp B | A \cup S \backslash \{\alpha\}$. This, together with the condition, give (using $X = \{\alpha\}$, $Y = B$, $W = \{\gamma\}$ and $Z = A \cup S \backslash \{\alpha, \gamma\}$ in the **intersection** statement of Exercise 1) that $\alpha \perp B \cup \{\gamma\} | A \cup S \backslash \{\alpha, \gamma\}$ as required.

5. Recall definition of *Markov blanket*; parents of A, children of A and any variables sharing a child with A. Consider the 'Bayes Ball' algorithm, started at A. The ball cannot travel through an instantiated chain or fork connection, nor can it travel through a collider, where none of the descendants are instantiated. Otherwise, it can travel through a node along the graph.

Therefore: if all variables in the Markov blanket are instantiated, Bayes ball cannot pass through any of the parents (by definition, the connection is necessarily chain or fork). It cannot pass through a child to any offspring of the child (the connection necessarily chain). If it passes

through an instantiated child to another parent of the instantiated child, it cannot pass further: connection at the point of the instantiated parent of the instantiated child is either chain or fork.

6. Firstly, clearly if there is an edge between α and β, then $\alpha - \beta$ is an S-active trail for any $S \subseteq V \backslash \{\alpha, \beta\}$. If there is no edge between α and β, then there are two cases. Firstly, if $\alpha \notin MB(\beta)$ (where MB denotes Markov blanket), then $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} MB(\beta)$. If $\alpha \in MB(\beta)$, but there is no edge between α and β, then α and β are parents of a common child. Let C denote the set of variables that are common children of α and β. Let

$$V_C = C \cup \{\delta \quad | \quad \text{there is a directed path} \quad \gamma \to \delta \quad \text{some} \quad \gamma \in C\}.$$

Let $S = V \backslash (\{\alpha, \beta\} \cup V_C)$, then $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} S$. Then any trail between α and β through a common child is blocked by virtue of an uninstantiated collider where none of the descendants are instantiated. Any trail with a common ancestor is blocked by virtue of an instantiated fork. On any trail where α is an ancestor of β or β an ancestor of α, there is an instantiated chain connection.

7. (a) All trails between X and Y contain either a fork or chain node in Z, or collider not in Z with no descendants in Z. When Z and γ are instantiated, there is no trail between X and Y where all the colliders are either instantiated or have an instantiated descendant and all chain and fork connections are uninstantiated.

   Suppose that $X \not\perp\!\!\!\perp \{\gamma\} \| _{\mathcal{G}} Z$ and $Y \not\perp\!\!\!\perp \{\gamma\} \| _{\mathcal{G}} Z$. Then for any $x \in X$ and any $y \in Y$ there is a Z-active trail between x and γ and a Z-active trail between y and γ. Consider the trail between x and y formed by joining the two. If γ is a chain or fork node, then the trail is active when γ is not instantiated, contradicting $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z$.

   (b) Assume the result is not true and that $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} \{\gamma\}$ and $\alpha \not\perp\!\!\!\perp \beta | \{\delta\}$ and that $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \{\gamma, \delta\}$. Then there is a $\{\gamma\}$-active trail between α and β with δ as a fork or chain node. Assume that there is a collider node on the trail with γ as a descendant, then there is a collider ρ and a trail $\delta \to \rho_1 \to \ldots \to \rho_n \to \rho$ and hence a directed path from δ to γ that does not contain α or β contradicting $\gamma \perp\!\!\!\perp \delta \| _{\mathcal{G}} \{\alpha, \beta\}$. It follows that there is a trail between α and β containing δ with only fork and chain connections. Similarly, there is a trail between α and β containing δ with only forks and chains. Then there is a trail between δ and γ containing α with at most one collider α and another trail between δ and γ containing β with at most one collider $\{\beta\}$. If $\delta \perp\!\!\!\perp \gamma | \{\alpha, \beta\}$ then neither of these are colliders and hence there is a cycle, hence a contradiction.

8. This is a direct consequence of the definition. Let $\underline{x} = (x_1, \ldots, x_d)$ and $\underline{y} = (y_1, \ldots, y_d)$ where $y_j = x_j = w_j$ for $j \neq i$, $x_i = 1$, $y_i = 0$. Let $\pi_j(\underline{x})$ denote the parent configuration for variable j when $\underline{X} = \underline{x}$. Then, since

$$O_{\mathbb{P}}(\{X_i = 1\} | \{\underline{X}_{V \backslash \{X_i\}} = \underline{w}\}) = \frac{\mathbb{P}(\{X_i = 1\}, \{\underline{X}_{V \backslash \{X_i\}} = \underline{w}\})}{\mathbb{P}(\{X_i = 0\}, \{\underline{X}_{V \backslash \{X_i\}} = \underline{w}\})}$$

$$
\begin{aligned}
O_{\mathbb{P}}(\{X_i = 1\}|\underline{X}_{V\setminus\{X_i\}} = \underline{w}\}) &= \frac{\prod_{j=1}^{d} \mathbb{P}_{X_j|\mathrm{Pa}_j}(x_j|\pi_j(\underline{x}))}{\prod_{j=1}^{d} \mathbb{P}_{X_j|\mathrm{Pa}_j}(y_j|\pi_j(\underline{y}))} \\
&= \frac{\mathbb{P}_{X_i|\mathrm{Pa}_i}(1|\pi_i(\underline{x})) \prod_{j|X_i\in\mathrm{Pa}_j} \mathbb{P}_{X_j|\mathrm{Pa}_j}(x_j|\pi_j(\underline{x}))}{\mathbb{P}_{X_i|\mathrm{Pa}_i}(0|\pi_i(\underline{y})) \prod_{j|X_i\in\mathrm{Pa}_j} \mathbb{P}_{X_j|\mathrm{Pa}_j}(y_j|\pi_j(\underline{y}))}
\end{aligned}
$$

and, from the definition, this only involves the Markov blanket of X_i; $\mathbb{P}_{X_i|\mathrm{Pa}_i}$ involves X_i and the parents of X_i, the other conditional probabilities involve the children of X_i and their parents.

Chapter 2

Markov models and Markov equivalence

2.1 I-maps and Markov equivalence

If a probability distribution factorises according to a directed acyclic graph, then any D-separation statement in the graph implies the corresponding conditional independence statement for the distribution. If each D-separation statement for a DAG $\mathcal{G}$ implies the corresponding conditional independence statement for a probability distribution $\mathbb{P}$, then $\mathbb{P}$ is said to belong to the *Markov model* of $\mathcal{G}$.

Definition 2.1 (Markov Model)**.** *Let $V = \{1, \ldots, d\}$ and let $\mathcal{G} = (V, D)$ be a directed acyclic graph with node set V and directed edge set D. Let $\mathcal{V}$ denote the entire set of subsets of V. The* Markov Model *$\mathcal{M}_{\mathcal{G}}$ of $\mathcal{G} = (V, D)$ is the set of triples $(A, B, S) \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$, A, B, S disjoint, such that the D-separation statement $A \perp\!\!\!\perp B \| _{\mathcal{G}} S$ holds in the DAG. That is,*

$$\mathcal{M}_{\mathcal{G}} = \{(A, B, S) \in \mathcal{V} \times \mathcal{V} \times \mathcal{V} \quad | \quad A, B, S \quad \textit{disjoint} \quad A \perp\!\!\!\perp B \| _{\mathcal{G}} S\}. \tag{2.1}$$

Let $\mathbb{P}$ be a probability distribution of a random vector $\underline{X} = (X_1, \ldots, X_d)$, whose components are indexed by V. Let $\mathcal{I}(\mathbb{P})$ denote the entire set of conditional independence statements associated with $\mathbb{P}$;

$$\mathcal{I}(\mathbb{P}) = \{(A, B, S) \in \mathcal{V} \times \mathcal{V} \times \mathcal{V} \quad | \quad A, B, S \quad \textit{disjoint} \qquad X_A \perp X_B | X_S\} \tag{2.2}$$

where, for any set $C \subseteq V$, X_C denotes the sub-vector of random variables indexed by C. The convention is that if $S = \varnothing$ (the empty set) then $X_A \perp X_B | X_S$ means $X_A \perp X_B$. A distribution $\mathbb{P}$ is said to belong to the Markov Model of $\mathcal{G}$, written $\mathbb{P} \in \mathcal{M}_{\mathcal{G}}$, if and only if $\mathcal{M}_{\mathcal{G}} \subseteq \mathcal{I}(\mathbb{P})$. The Markov model *is the set of conditional independence relations satisfied by all distributions that are locally $\mathcal{G}$-Markovian (Definition 1.26).*

If a distribution $\mathbb{P}$ factorises along a DAG $\mathcal{G} = (V, D)$, then the collection of triples $\mathcal{M}_{\mathcal{G}}$ defined in Equation (2.1) Definition 2.1 represents the entire set of conditional independence statements that it is possible to infer from the DAG. Clearly, this collection does not, in general, represent the *complete* set of conditional independence statements that hold for $\mathbb{P}$. In fact, the probability distributions modelling real world situations, corresponding to data sets, very rarely factorise along a DAG whose D-separation

statements encode the entire set of independence statments. When it does hold, the DAG is known as a *perfect I-map*.

Definition 2.2 (Perfect I-Map, Faithful)**.** *Let $\mathcal{G} = (V, D)$ be a DAG with node set $V = \{1, \ldots, d\}$ which indexes a random vector $X = (X_1, \ldots, X_d)$. Let $\mathcal{V}$ denote the set of all subsets of V. The DAG $\mathcal{G}$ is known as a* perfect I-map *for a probability function $\mathbb{P}$ over X if and only if $\mathcal{I}(\mathbb{P}) = \mathcal{M}_{\mathcal{G}}$. A DAG $\mathcal{G}$ such that $\mathcal{I}(\mathbb{P}) = \mathcal{M}_{\mathcal{G}}$ is said to be* faithful *to $\mathbb{P}$.*

A DAG $\mathcal{G} = (V, D)$ is consistent with $\mathcal{I}(\mathbb{P})$ defined by Equation (2.2) if and only if $\mathcal{G}$ is faithful to $\mathbb{P}$, if and only if $\mathcal{G}$ is a perfect I-map of $\mathbb{P}$.

A DAG that is faithful to a probability distribution $\mathbb{P}$, satisfies the following important property:

Theorem 2.3. *Let $\mathcal{G} = (V, D)$ be faithful to a probability distribution $\mathbb{P}$. Then the edge set D contains an edge between two nodes α and β if and only if $X_\alpha \not\perp X_\beta | X_S$ for any $S \subseteq V \backslash \{\alpha, \beta\}$.*

Proof This is a straightforward consequence of Theorem 1.24, that the edge set D contains an edge between α and β if and only if $\alpha \not\perp \beta \| _{\mathcal{G}} S$ for any $S \subseteq V \backslash \{\alpha, \beta\}$. □

A set of variables $(X_1, \ldots, X_d)$, may be ordered in $d!$ ways. Each permutation σ of $1, \ldots, d$ gives an ordering $(X_{\sigma(1)}, \ldots, X_{\sigma(d)})$ and for each permutation, the distribution may be factorised according to a Bayesian network, represented by the corresponding directed acyclic graph.

The input for the construction of the directed acyclic graph consists of a list of d conditional independence statements, one for each variable, all of the form $X_{\sigma(j)} \perp \Sigma_j^\sigma | \mathrm{Pa}_j^\sigma$, where

$$\Sigma_j^\sigma = \{X_{\sigma(1)}, \ldots, X_{\sigma(j-1)}\} \backslash \mathrm{Pa}^\sigma(j).$$

This is the set of σ-predecessors of $X_{\sigma(j)}$, without $\mathrm{Pa}^\sigma(j)$.

For a given collection of variables $V = \{X_1, \ldots, X_d\}$, there may be several *different* DAGs, each with exactly the *same* Markov model $\mathcal{M}$. Two DAGs which determine exactly the same Markov model are said to be I-equivalent.

Definition 2.4 (I-sub-map, I-map, I-equivalence, Markov Equivalence)**.** *Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two DAGs with the same node set. The DAG $\mathcal{G}_1$ is said to be an* I-sub-map *of $\mathcal{G}_2$ if $\mathcal{M}_{\mathcal{G}_1} \subseteq \mathcal{M}_{\mathcal{G}_2}$. They are said to be* I-equivalent *if $\mathcal{M}_{\mathcal{G}_1} = \mathcal{M}_{\mathcal{G}_2}$. I-equivalence is also known as* Markov equivalence.

Example 2.5.

In the following example on three variables, all three factorisations give the same independence structure. Consider a probability distribution $\mathbb{P}_{X_1, X_2, X_3}$ with factorisation

$$\mathbb{P}_{X_1, X_2, X_3} = \mathbb{P}_{X_1} \mathbb{P}_{X_2 | X_1} \mathbb{P}_{X_3 | X_2}.$$

It follows that

$$\mathbb{P}_{X_1,X_2,X_3} = \mathbb{P}_{X_2}\mathbb{P}_{X_1|X_2}\mathbb{P}_{X_3|X_1,X_2} = \mathbb{P}_{X_2}\mathbb{P}_{X_1|X_2}\mathbb{P}_{X_3|X_2},$$

using $X_1 \perp X_3|X_2$. Also,

$$\mathbb{P}_{X_1,X_2,X_3} = \mathbb{P}_{X_3}\mathbb{P}_{X_2|X_3}\mathbb{P}_{X_1|X_2,X_3} = \mathbb{P}_{X_3}\mathbb{P}_{X_2|X_3}\mathbb{P}_{X_1|X_2},$$

since $X_1 \perp X_3|X_2$. For the first and last of these, X_2 is a chain node, while in the second of these X_2 is a fork node. The conditional independence structure associated with chains and forks is the same. The three corresponding DAGs are given in Figure 2.1.

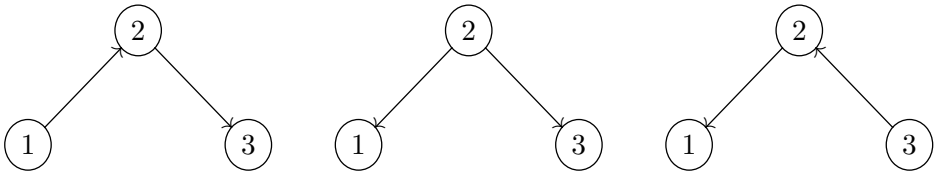

Figure 2.1: Three DAGs, each with the same D-separation structure

□

In general, the factorisations resulting from different orderings of the variables will not necessarily give I-equivalent maps. This is illustrated by the following example on four variables.

Example 2.6.

Consider a probability distribution over four variables, which may be factorised as

$$\mathbb{P}_{X_1,X_2,X_3,X_4} = \mathbb{P}_{X_1}\mathbb{P}_{X_2}\mathbb{P}_{X_3|X_1,X_2}\mathbb{P}_{X_4|X_3}.$$

The DAG associated with the factorisation is the one on the left in Figure 2.2. Assume that the DAG on the left in Figure 2.2 and $\mathbb{P}_{X_1,X_2,X_3,X_4}$ are faithful to each other. The factorisation obtained using the ordering (X_1, X_4, X_3, X_2) is:

$$\mathbb{P}_{X_1,X_2,X_3,X_4} = \mathbb{P}_{X_1}\mathbb{P}_{X_4|X_1}\mathbb{P}_{X_3|X_1,X_4}\mathbb{P}_{X_2|X_1,X_3,X_4} = \mathbb{P}_{X_1}\mathbb{P}_{X_4|X_1}\mathbb{P}_{X_3|X_1,X_4}\mathbb{P}_{X_2|X_1,X_3}.$$

This is the factorisation corresponding to a Bayesian network since

- $X_1 \not\perp X_4$,
- $X_3 \not\perp \{X_1, X_4\}\backslash\Theta|\Theta$ for any $\Theta \subseteq \{X_1, X_4\}$ and
- $X_2 \perp X_4|\{X_1, X_3\}$, but $X_2 \not\perp \{X_1, X_3, X_4\}\backslash\Theta|\Theta$ for any strict subset $\Theta \subset \{X_1, X_3\}$.

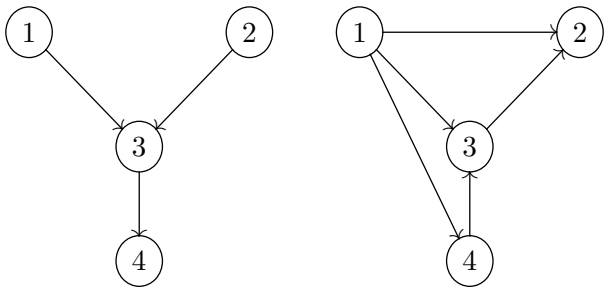

Figure 2.2: DAGs with different D-separation properties, corresponding to different factorisations of the same distribution

The corresponding DAG (the graph on the right of Figure 2.2) gives less information on conditional independence; $X_4 \not\perp\!\!\!\perp X_1 \| _{\mathcal{G}_2} X_3$, using $\mathcal{G}_2$ to denote the DAG on the right in Figure 2.2. The two corresponding DAGs are shown in Figure 2.2. The graph on the right is a strict I-sub-map of the graph on the left. □

This example illustrates that while D-separated variables are conditionally independent conditioned on the separating set, it does not hold that conditionally independent variables are necessarily D-separated.

2.1.1 Properties of Conditional Expectation and D-Separation

For a probability distribution $\mathbb{P}$ over a set of variables, the collection of conditional independence statements $\mathcal{I}(\mathbb{P})$ satisfies the following: Let (X, Y, W, Z) be disjoint sets of random variables, then the following relations hold:

1. **decomposition** If $X \perp Y \cup W | Z$ then $X \perp Y | Z$ and $X \perp W | Z$.
2. **contraction** If $X \perp Y | Z$ and $X \perp W | Y \cup Z$ then $X \perp W \cup Y | Z$.
3. **weak union** If $X \perp Y \cup Z | W$ then $X \perp Y | Z \cup W$.
4. **intersection** If $X \perp Y | W \cup Z$ and $X \perp W | Y \cup Z$ then $X \perp W \cup Y | Z$.

These relations are discussed by Pearl in [105] (1988). The proofs of these are quite straightforward and have been left as exercises (Exercise 1 page 22).

The Markov model $\mathcal{M}_{\mathcal{G}}$ for a DAG $\mathcal{G}$ also satisfies the following: let (X, Y, W, Z) be four sets of nodes in a DAG $\mathcal{G} = (V, D)$, then the following relations hold:

1. **decomposition** If $X \perp\!\!\!\perp Y \cup W \| _{\mathcal{G}} Z$ then $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z$ and $X \perp\!\!\!\perp W \| _{\mathcal{G}} Z$.
2. **contraction** If $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z$ and $X \perp\!\!\!\perp W \| _{\mathcal{G}} Y \cup Z$ then $X \perp\!\!\!\perp W \cup Y \| _{\mathcal{G}} Z$.
3. **weak union** If $X \perp\!\!\!\perp Y \cup Z \| _{\mathcal{G}} W$ then $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z \cup W$.

4. **intersection** If $X \perp\!\!\!\perp Y \| _{\mathcal{G}} W \cup Z$ and $X \perp\!\!\!\perp W \| _{\mathcal{G}} Y \cup Z$ then $X \perp\!\!\!\perp W \cup Y \| _{\mathcal{G}} Z$.

These have been left as exercises (Exercise 2 page 22). A collection of triples $(X_i, Y_i, S_i)_{i \in \mathcal{I}}$, where $\mathcal{I}$ denotes the indexing set and each $(X_i, Y_i, S_i) \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$, X_i, Y_i, S_i mutually disjoint which satisfies these four conditions has come to be known as a *graphoid*. These statements do not axiomatise conditional independence; if a given set of triples satisfies these four conditions, there does not necessarily exist a probability distribution $\mathbb{P}$ for which the set of conditional independence statements is $\mathcal{I}(\mathbb{P})$. Conditional independence cannot be axiomatised; this was proved by Studený [130].

A collection of D-separation statements for a DAG also satisfies the *composition* property;

5 **composition** If both $X \perp\!\!\!\perp Y \| _{\mathcal{G}} S$ and $X \perp\!\!\!\perp Z \| _{\mathcal{G}} S$ hold, then $X \perp\!\!\!\perp Y \cup Z \| _{\mathcal{G}} S$.

A graphoid that also satisfies *composition* is known as a *compositional graphoid*. The Markov model of a DAG $\mathcal{M}_{\mathcal{G}}$ is always a compositional graphoid; the collection of independence statements $\mathcal{I}(\mathbb{P})$ is not necessarily a compositional graphoid.

A Markov model $\mathcal{M}_{\mathcal{G}}$ also satisfies the following two properties, which are not necessarily satisfied by a collection of conditional independence statements $\mathcal{I}(\mathbb{P})$.

- Let $V = A \cup B \cup S$ where A, B and S are disjoint subsets and suppose that $A \perp\!\!\!\perp B \| _{\mathcal{G}} S$. Then for any $\alpha \in A$ and $\gamma \in A \cup S$,

$$\alpha \perp\!\!\!\perp \gamma \| _{\mathcal{G}} (A \cup S) \backslash \{\alpha, \gamma\} \Leftrightarrow \alpha \perp\!\!\!\perp \gamma \| _{\mathcal{G}} (A \cup B \cup S) \backslash \{\alpha, \gamma\}.$$

- Let $\mathcal{G} = (V, D)$ denote a directed acyclic graph. Let $X \subseteq V$, $Y \subseteq V$ and $Z \subseteq V$ denote sets of nodes and let $\alpha, \beta, \gamma, \delta \in V \backslash X \cup Y \cup Z$ denote individual nodes.
  - If $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z$ and $X \perp\!\!\!\perp Y \| _{\mathcal{G}} Z \cup \{\gamma\}$ then either $X \perp\!\!\!\perp \{\gamma\} \| _{\mathcal{G}} Z$ or $Y \perp\!\!\!\perp \{\gamma\} \| _{\mathcal{G}} Z$
  - If $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \{\gamma, \delta\}$ and $\gamma \perp\!\!\!\perp \delta \| _{\mathcal{G}} \{\alpha, \beta\}$ then either $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \{\gamma\}$ or $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \{\delta\}$.

The proofs of these statements are left as exercises. They are included here simply as illustration of the additional structure that is required for a Markov model $\mathcal{M}_{\mathcal{G}}$ over and above the set of independence statements $\mathcal{I}(\mathbb{P})$ for a probability distribution that factorises over $\mathcal{G}$.

The following basic example illustrates a situation where *composition* does not hold for the probability distribution and where there is no faithful DAG.

Example 2.7 (Tossing Three Coins)**.**

Let Y_1, Y_2, Y_3 be three independent identically distributed binary variables, with probability function $\mathbb{P}_Y(0) = \mathbb{P}_Y(1) = \frac{1}{2}$. Let

$$X_1 = \begin{cases} 1 & Y_2 = Y_3 \\ 0 & Y_2 \neq Y_3 \end{cases}$$

$$X_2 = \begin{cases} 1 & Y_1 = Y_3 \\ 0 & Y_1 \neq Y_3 \end{cases}$$

$$X_3 = \begin{cases} 1 & Y_1 = Y_2 \\ 0 & Y_1 \neq Y_2 \end{cases}$$

Then X_1, X_2, X_3 provide the classic example of three random variables that are pairwise independent, but not jointly independent.

$$\mathbb{P}_{X_1,X_2,X_3}(1,1,1) = \mathbb{P}(Y_1 = Y_2 = Y_3) = \mathbb{P}_{Y_1,Y_2,Y_3}(1,1,1) + \mathbb{P}_{Y_1,Y_2,Y_3}(0,0,0) = \frac{1}{4}$$

$$\mathbb{P}_{X_1,X_2,X_3}(1,1,0) = \mathbb{P}_{X_1,X_2,X_3}(1,0,1) = \mathbb{P}_{X_1,X_2,X_3}(0,1,1) = \mathbb{P}(Y_2 = Y_3 = Y_1, Y_1 \neq Y_2) = 0$$

$$\begin{aligned} \mathbb{P}_{X_1,X_2,X_3}(1,0,0) &= \mathbb{P}_{X_1,X_2,X_3}(0,1,0) = \mathbb{P}_{X_1,X_2,X_3}(0,0,1) = \mathbb{P}(Y_2 = Y_3, Y_1 \neq Y_3, Y_1 \neq Y_2) \\ &= \mathbb{P}(Y_1 = 1, Y_2 = Y_3 = 0) + \mathbb{P}(Y_1 = 0, Y_2 = Y_3 = 1) = \frac{1}{4} \end{aligned}$$

$$\mathbb{P}_{X_1,X_2,X_3}(0,0,0) = 0$$

It follows that

$$\mathbb{P}_{X_1,X_2}(1,1) = \mathbb{P}_{X_1,X_2}(1,0) = \mathbb{P}_{X_1,X_2}(0,1) = \mathbb{P}_{X_1,X_2}(0,0) = \frac{1}{4}$$

so that $\mathbb{P}_{X_1}(1) = \mathbb{P}_{X_1}(0) = \frac{1}{2}$ and in all cases

$$\mathbb{P}_{X_1,X_2} = \mathbb{P}_{X_1}\mathbb{P}_{X_2}.$$

But

$$\frac{1}{4} = \mathbb{P}_{X_1,X_2,X_3}(1,1,1) \neq \mathbb{P}_{X_1}(1)\mathbb{P}_{X_2}(1)\mathbb{P}_{X_3}(1) = \frac{1}{8}.$$

Since $X_1 \perp X_2$ but $X_3 \not\perp \{X_1, X_2\}$, $X_3 \not\perp X_1|X_2$ and $X_3 \not\perp X_2|X_1$, it follows that the factorisation obtained for the distribution $\mathbb{P}_{X_1,X_2,X_3}$ is

$$\mathbb{P}_{X_1,X_2,X_3} = \mathbb{P}_{X_1}\mathbb{P}_{X_2}\mathbb{P}_{X_3|X_1,X_2}.$$

In the corresponding DAG, $X_1 \not\perp X_2 \| _{\mathcal{G}} \varnothing$, $X_1 \not\perp X_3 \| _{\mathcal{G}} \varnothing$ and $X_2 \not\perp X_3 \| _{\mathcal{G}} \varnothing$, even though the independence statements $X_1 \perp X_3$ and $X_2 \perp X_3$ hold.

By considering other orderings of the variables, the other possible factorisations are

$$\mathbb{P}_{X_1,X_2,X_3} = \mathbb{P}_{X_1}\mathbb{P}_{X_3}\mathbb{P}_{X_2|X_1,X_3} = \mathbb{P}_{X_2}\mathbb{P}_{X_3}\mathbb{P}_{X_1|X_2,X_3}$$

and in none of the cases do the D-separation statements of the DAG corresponding to the Bayesian Network represent all the conditional independence statements of the distribution.

The type of situation described here, where the distribution does not satisfy a composition property, can be summarised as follows: it is the situation where X_1 tells you nothing about X_3 and X_2 tells you nothing about X_3, but X_1 and X_2 taken together tell you everything about X_3. This is the principle on which any good detective novel is based, as Edward Nelson puts it in his book 'Radically Elementary Probability Theory' [100].

The argument shows that in experimental design situations where there are interaction effects, but no main effects (e.g. each chemical taken separately does not produce an effect, but the interaction between two chemicals causes an effect), composition will not hold and there will not exist a faithful DAG.

There is a whole industry of structure learning algorithms, based on the principle of Theorem 2.3, which deleted an edge as soon as a conditioning set is found such that $X \perp Y|S$. These algorithms are elegant, cost-effective, efficient, and return accurate results *if* the underlying distribution has a faithful graph. They are discussed in chapter 16. Their draw-back is that they produce wildly inaccurate results when there does not exist an underlying faithful graph. Note that if such a structure learning algorithm were applied to the three-coin example above, where $X_1 \perp X_2$, $X_1 \perp X_3$ and $X_2 \perp X_3$, such an algorithm would remove all the edges based on the results of conditioning on $S = \varnothing$ and return the empty graph. The model delivered by the algorithm would then be the independence model, which represents a disastrous failure.

Example 2.8 (Hidden Variables)**.**

In many causal situations, the set of variables X may be split into observable variables Z and unobservable variables, U. Usually, the observable variables are descendants of the unobservable; observations are made on the observable and, from these observations, inferences made about the unobservable. For example, the variables of Z could represent *symptoms*, while those of U could represent the diseases that cause the symptoms.

Even if there is a faithful DAG corresponding to the full set of variables $X = (U, V)$, there is often no faithful DAG corresponding to the observable variables Z. For example, consider a probability distribution over the 5 variables $\{U, Z_1, Z_2, Z_3, Z_4\}$ which factorises according to

$$\mathbb{P}_{U,Z_1,Z_2,Z_3,Z_4} = \mathbb{P}_U \mathbb{P}_{Z_3} \mathbb{P}_{Z_4|Z_3} \mathbb{P}_{Z_1|U,Z_3} \mathbb{P}_{Z_2|U,Z_4}$$

and suppose the corresponding graph given by Figure 2.3 is faithful. In this example, there is no faithful DAG for the distribution over (Z_1, Z_2, Z_3, Z_4); the set of D-separation statements for any DAG along which the distribution can be factorised will be a strict subset of the set of conditional independence statements. Two examples of factorisations are:

$$\mathbb{P}_{Z_1,Z_2,Z_3,Z_4} = \mathbb{P}_{Z_1} \mathbb{P}_{Z_2|Z_1} \mathbb{P}_{Z_3|Z_1,Z_2} \mathbb{P}_{Z_4|Z_1,Z_2,Z_3}$$

$$\mathbb{P}_{Z_1,Z_3,Z_4,Z_2} = \mathbb{P}_{Z_1} \mathbb{P}_{Z_3|Z_1} \mathbb{P}_{Z_4|Z_3} \mathbb{P}_{Z_2|Z_1,Z_3,Z_4}$$

When all 24 permutations are considered, either an edge $Z_2 \sim Z_3$ or an edge $Z_1 \sim Z_4$ will be present, even though $Z_2 \perp Z_3|Z_4$ and $Z_1 \perp Z_4|Z_3$. None of the DAGs corresponding to the Bayesian Networks of all 24 possible orderings of the variables will represent all the CI statements.

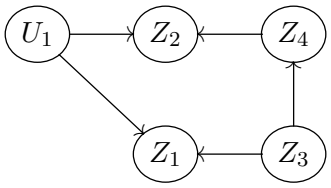

Figure 2.3: Faithful DAG, U_1 hidden

2.2 Characterisation of Markov Equivalence

When trying to fit a graphical model to data, all the DAGs in a Markov equivalence class will fit the data equally well and efficient algorithms for finding the structure will therefore only examine the different equivalence classes, rather than all the different possible DAGs.

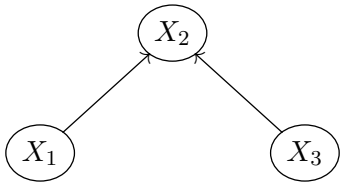

Figure 2.4: Not Markov Equivalent to the Graphs in Figure 2.1

Figure 2.1 shows three directed acyclic graphs, each with the same D-separation statements; $X_1 \perp\!\!\!\perp X_3 \| _{\mathcal{G}} X_2$ and the DAG does not admit any other D-separation statements. The D-separation statements of the DAG in Figure 2.4 are different from those for the DAGs in Figure 2.1.

The key result in this section characterising Markov equivalence is Theorem 2.11, which states that the two features of a directed acyclic graph which are necessary and sufficient for determining its Markov structure are its *immoralities* and its *skeleton*. These are defined below.

Definition 2.9 (Immorality)**.** *Let $\mathcal{G} = (V, E)$ be a graph. Let $E = D \cup U$, where D is the set of directed edges, U is the set of undirected edges and $D \cap U = \varnothing$. An* immorality *in a graph is a triple (α, β, γ) such that $(\alpha, \beta) \in D$ and $(\gamma, \beta) \in D$, but $(\alpha, \gamma) \notin D$, $(\gamma, \alpha) \notin D$ and $\langle \alpha, \gamma \rangle \notin U$.*

Definition 2.10 (Skeleton)**.** *The* skeleton *of a graph $\mathcal{G} = (V, E)$ is the graph obtained by making the graph* undirected*. That is, the skeleton of $\mathcal{G}$ is the graph $\widetilde{\mathcal{G}} = (V, \widetilde{E})$ where $\langle \alpha, \beta \rangle \in \widetilde{E} \Leftrightarrow (\alpha, \beta) \in D$ or $(\beta, \alpha) \in D$ or $\langle \alpha, \beta \rangle \in U$.*

The characterisation is given by the following theorem.

Theorem 2.11. *Two DAGs are Markov equivalent if and only if they have the same* skeleton *and the same* immoralities*.*

The key to establishing Theorem 2.11 will be to consider the active trails (Definition 1.17) in the graph. The following two definitions are also required.

Definition 2.12 (S-active node)**.** *Let $\mathcal{G} = (V, E)$ be a Directed Acyclic Graph and let $S \subset V$. Recall the definition of a trail (Definition 1.4) and the definition of an active trail (Definition 1.17). A node $\alpha \in V$ is said to be* S-active *if either $\alpha \in S$ or there is a directed path* from *the node α* to *a node $\beta \in S$.*

Definition 2.13 (Minimal S- active trail)**.** *Let $\mathcal{G} = (V, E)$ be a Directed Acyclic Graph and let $S \subset V$. An S-active trail τ in $\mathcal{G}$ between two nodes α and β is said to be a* minimal S- active trail *if it satisfies the following two properties:*

1. *if k is the number of nodes in the trail, the first node is α and the kth node is β, then there does not exist an S-active trail between α and β with fewer than k nodes and*
2. *there does not exist a* different *S-active trail ρ between α and β with exactly k nodes such that for all $1 < j < k$ either $\rho_j = \tau_j$ or ρ_j is a descendant of τ_j.*

The proof of Theorem 2.11 follows directly from Lemma 2.14.

Lemma 2.14. *Let $\mathcal{G}_1 = (V, D_1)$ and $\mathcal{G}_2 = (V, D_2)$ be two directed acyclic graphs with the same skeletons and the same immoralities. Then for all $S \subset V$, a trail is a minimal S-active trail in $\mathcal{G}_1$ if and only if it is a minimal S-active in $\mathcal{G}_2$.*

Proof of Lemma 2.14 Recall the notation from Definition 1.2; $\alpha \sim \beta$ denotes that two nodes $(\alpha, \beta) \in V \times V$ are neighbours. For a directed graph $\mathcal{G} = (V, D)$, that is either $(\alpha, \beta) \in D$ or $(\beta, \alpha) \in D$. Since $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same skeletons, any trail τ in $\mathcal{G}_1$ is also a trail in $\mathcal{G}_2$. Let $S \subset V$. Assume that τ is a minimal S-active trail in $\mathcal{G}_1$. It is now proved, by induction on the number of collider nodes along the path, that τ is also an S-active trail in $\mathcal{G}_2$. By definition, a single node will be considered an S-active trail, for any $S \subset V$. The proof is in three parts: Let τ be a *minimal* S-active trail in $\mathcal{G}_1$. Then

1. If τ contains no colliders in $\mathcal{G}_1$, then it is S-active in $\mathcal{G}_2$.
2. If τ contains at least one collider connection centred at node τ_j, then τ is S-active in $\mathcal{G}_2$ if and only if τ_j is S-active in $\mathcal{G}_2$.
3. If τ contains at least one collider centred at node τ_j, then τ_j is an S-active node in $\mathcal{G}_2$.

Part 1 If τ is an S-active trail in $\mathcal{G}_1$ and does not contain any collider connections in $\mathcal{G}_1$, then none of the nodes on τ are in S. This can be seen by considering the Bayes ball algorithm, which characterises d-separation. It follows that the trail is S-active in $\mathcal{G}_2$ if and only if it does not contain a collider connection in $\mathcal{G}_2$.

Let τ be a minimal S-active trail in $\mathcal{G}_1$ with k nodes and no collider connections in $\mathcal{G}_1$. Suppose that a node τ_i is a collider node in $\mathcal{G}_2$, so that τ_{i-1} and τ_{i+1} are parents of τ_i in $\mathcal{G}_2$. Then, so that no new immoralities are introduced, it follows that $\tau_{i-1} \sim \tau_{i+1}$. Since τ_i is either a chain or a fork in $\mathcal{G}_1$, it follows that in $\mathcal{G}_1$, the connections between nodes $\tau_{i-2}, \tau_{i-1}, \tau_i, \tau_{i+1}, \tau_{i+2}$ take one of the forms shown in Figure 2.5 when τ_i a chain node or those in Figure 2.6 when τ_i a fork node.

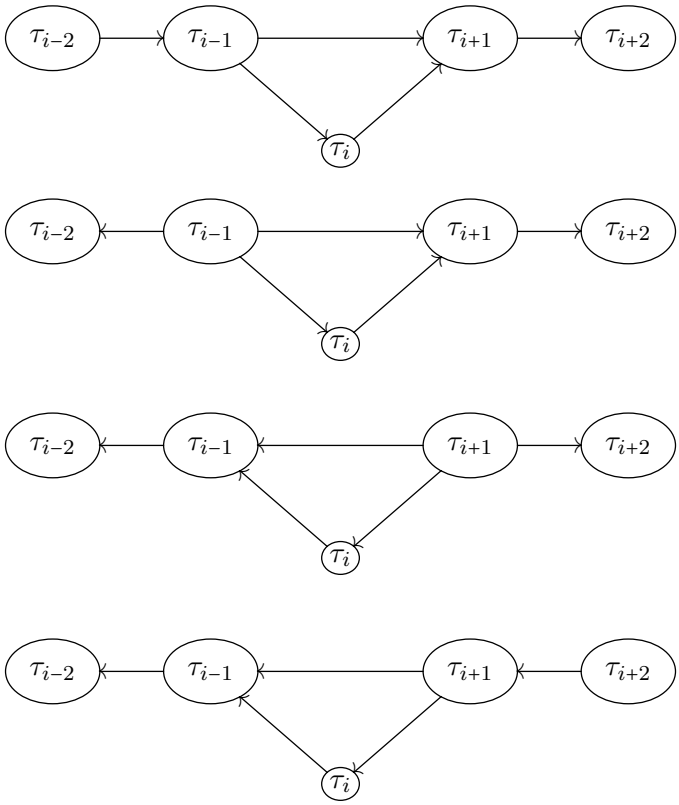

Figure 2.5: Possible connections between the nodes when τ_i is chain node

It is clear from the figure that the trail of length $k-1$ in $\mathcal{G}_1$, obtained by removing τ_i and using the direct link from τ_{i-1} to τ_{i+1} is also an S-*active* trail in $\mathcal{G}_1$, contradicting the assumption that τ was a minimal S-active trail. Hence τ_i is a chain node or a fork node in $\mathcal{G}_2$.

It follows that there are no collider connections along the trail τ taken in $\mathcal{G}_2$ and hence, since it does not contain any nodes that are in S, it is an S- active trail in $\mathcal{G}_2$.

Part 2 Assume that any minimal S-active trail in $\mathcal{G}_1$ containing n collider connections is also S-active in $\mathcal{G}_2$. This is true for $n = 0$ by part 1. Let τ be a trail with k nodes that is minimal S-active in $\mathcal{G}_1$ and with $n+1$ collider connections in $\mathcal{G}_1$. Consider one of the collider connections centred at τ_j, with parents τ_{j-1} and τ_{j+1}. Let $\widetilde{\tau}^{(0,j-1)} = (\tau_0, \tau_1, \ldots, \tau_{j-2}, \tau_{j-1})$ and let $\widetilde{\tau}^{(j+1,k)} = (\tau_{j+1}, \ldots, \tau_k)$. Both $\widetilde{\tau}^{(0,j-1)}$ and $\widetilde{\tau}^{(j+1,k)}$ are minimal S-active in $\mathcal{G}_1$ and they both have a number of collider connections less than or equal to n. By the inductive hypothesis, they are therefore both S-active in $\mathcal{G}_2$.

Because the trail τ is minimal S-active in $\mathcal{G}_1$, it follows that $\tau_{j-1} \not\sim \tau_{j+1}$. This is because both τ_{j-1} and τ_{j+1} are S- active nodes in $\mathcal{G}_1$ (they have a common descendant in S to make the trail active), and neither is in S (neither is the centre of a collider along τ) it follows that if $\tau_{j-1} \sim \tau_{j+1}$, then the trail on $k-1$ nodes obtained by removing the node τ_j would be S-active in $\mathcal{G}_1$, for the following reason: any chain or fork $(\tau_{j-2}, \tau_{j-1}, \tau_{j+1})$ or $(\tau_{j-1}, \tau_{j+1}, \tau_{j+2})$ would be active because both τ_{j-1} and τ_{j+1} are

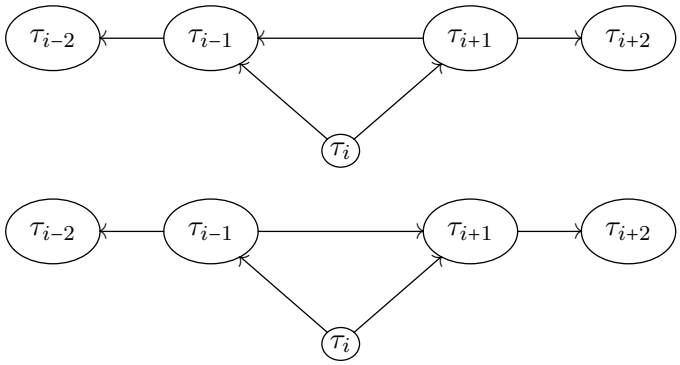

Figure 2.6: Possible connections between the nodes if τ_i is a fork node

uninstantiated. Any collider $(\tau_{j-2}, \tau_{j-1}, \tau_{j+1})$ or $(\tau_{j-1}, \tau_{j+1}, \tau_{j+2})$ would be active because both τ_{j-1} and τ_{j+1} have a descendant in S. It follows that $\tau_{j-1} \not\perp \tau_{j+1}$. This holds in both $\mathcal{G}_1$ and $\mathcal{G}_2$, since the skeletons are the same.
Since $\widetilde{\tau}^{(1,i-1)}$ and $\widetilde{\tau}^{(i+1,k)}$ are both active, and $\tau_{j-1} \to \tau_j \leftarrow \tau_{j+1}$ is a collider, the trail τ is active if and only if τ_j is an active node. That is, it is either in S or has a descendant in S.

Part 3 Let τ be a minimal S-active trail in $\mathcal{G}_1$ and let $\tau_j \in \tau$ be a collider node in $\mathcal{G}_1$. Since the trail τ is a minimal S-active trail in $\mathcal{G}_1$, it follows either that $\tau_j \in S$ or τ_j, considered in $\mathcal{G}_1$, has a descendant in S. That is, considered in $\mathcal{G}_1$, there is a directed *path* from τ_j to a node $w \in S$. Let ρ denote the shortest such path. If $\tau_j \in S$, then the length of the path is 0 and τ_j is also an S-active node in $\mathcal{G}_2$.

Assume there is a directed edge from τ_j to $w \in S$ in $\mathcal{G}_1$. If there are links from τ_{j-1} to w or τ_{j+1} to w, then these links are $\tau_{j-1} \to w$ or $\tau_{j+1} \to w$ respectively, otherwise the DAG would have cycles. If both are present, then the trail τ violates the second assumption of the minimality requirement. This is seen by considering the trail formed by taking w instead of τ_j in τ. It follows that either $\tau_{j-1} \not\perp w$ or $\tau_{j+1} \not\perp w$ or neither of the edges are present. Without loss of generality, assume $\tau_{j-1} \not\perp w$ (since the argument proceeds in the same way if $\tau_{j+1} \not\perp w$). The diagram in Figure 2.7 may be useful.

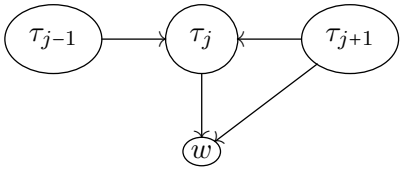

Figure 2.7: Illustration where τ_j is an uninstantiated collider node

Since w is not a parent of τ_j in $\mathcal{G}_1$, it cannot be a parent of τ_j in $\mathcal{G}_2$, since both graphs have the same immoralities and (τ_{j-1}, τ_j, w) is not an immorality in $\mathcal{G}_1$.

Furthermore, $\tau_{j-1} \not\perp \tau_{j+1}$ (since they are both uninstantiated, and, in $\mathcal{G}_1$ both have a common

descendant in S, so that if $\tau_{j-1} \sim \tau_{j+1}$ then the trail with τ_j removed would be active whether the connections at τ_{j-1} and τ_{j+1} are chain, fork or collider, contradicting the minimality assumption). Since both graphs have the same immoralities and $\tau_{j-1} \not\sim \tau_{j+1}$, it follows that $(\tau_{j-1}, \tau_j, \tau_{j+1})$ is an immorality in both $\mathcal{G}_1$ and $\mathcal{G}_2$ and hence that τ_{j-1} is a parent of τ_j in $\mathcal{G}_2$. Therefore, τ_j is a parent of w in $\mathcal{G}_2$ and is therefore w is an S- active node in $\mathcal{G}_2$.

Assume that for the shortest directed path ρ from τ_j to w in $\mathcal{G}_1$, the first l links have the same directed edges in $\mathcal{G}_2$. Now suppose that the shortest directed path is ρ, where $\tau_j = \rho_0, \ldots, \rho_{l+p} = w$ and consider the links $\rho_l \sim \rho_{l+1}$ and $\rho_{l+1} \sim \rho_{l+2}$. If $\rho_l \sim \rho_{l+2}$, then in $\mathcal{G}_1$, the directed edge $\rho_l \to \rho_{l+2}$ is present, otherwise there is a cycle. If the directed edge $\rho_l \to \rho_{l+2}$ is present in $\mathcal{G}_1$, then the path ρ is not minimal. Therefore, $\rho_l \not\sim \rho_{l+2}$. This holds in both $\mathcal{G}_1$ and $\mathcal{G}_2$, because both graphs have the same skeletons. By a similar argument, $\rho_{l-1} \not\sim \rho_{l+1}$. (there would be a cycle in $\mathcal{G}_1$ if (ρ_{l+1}, ρ_{l-1}) were present; ρ would not be minimal in $\mathcal{G}_1$ if (ρ_{l-1}, ρ_{l+1}) were present. Since the skeletons are the same, $\rho_{l-1} \not\sim \rho_{l+1}$ in either $\mathcal{G}_1$ or $\mathcal{G}_2$). Since $\rho_l \not\sim \rho_{l+2}$, it follows that ρ_l and ρ_{l+2} are not both parents of ρ_{l+1} in $\mathcal{G}_2$; otherwise $\mathcal{G}_2$ would contain an immorality not present in $\mathcal{G}_1$. Similarly, since $\rho_{l-1} \not\sim \rho_{l+1}$, the edge $\rho_l \to \rho_{l+1}$ is present in $\mathcal{G}_2$, otherwise $\mathcal{G}_2$ would have either the immorality $(\rho_{l-1}, \rho_l, \rho_{l+1})$, since the edge (ρ_{l-1}, ρ_l) is present in $\mathcal{G}_2$ by assumption. It follows that the directed edges (ρ_l, ρ_{l+1}) and (ρ_{l+1}, ρ_{l+2}) are both present in $\mathcal{G}_2$. By induction, therefore, the whole directed path ρ is also present in $\mathcal{G}_2$ and hence τ_j is an S- active in both $\mathcal{G}_1$ and $\mathcal{G}_2$. □

Proof of Theorem 2.11 This follows directly: let $\mathcal{G}_1$ and $\mathcal{G}_2$ denote two DAGs with the same skeleton and the same immoralities. For any set S and any two nodes α and β, it follows from the lemma, together with the definition of D- separation, that

$$\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}_1} S \Leftrightarrow \alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}_2} S; \tag{2.3}$$

if there is an S-active trail between the two variables in one of the graphs, then there is a minimal S-active trail in that graph and hence there is also a minimal S-active trail between the two variables in the other. If there is no S-active trail between the two variables in one of the graphs then there is no S active trail between the two variables in the other. By definition, two variables are D-separated by a set of variables S if and only if there is no S-active trail between the two variables. Two graphs are Markov equivalent, or I-equivalent (Definition 2.4), if and only if Equation (2.3) holds for all $(\alpha, \beta, S) \in V \times V \times \mathcal{V}$. □

2.2.1 Example 2.8 (Hidden Variables) Revisited

Consider the situation where there is a faithful DAG for the variable set $X = (U, Z)$ where U denotes the set of hidden variables and Z the set of observable variables. In situations where there is no faithful DAG for the variable set Z, it is possible using the principles of Theorem 2.3 and 5.5 to locate a set of hidden variables $\widetilde{U}$ such that a faithful graph can be constructed for $(\widetilde{U}, Z)$.

If the principle of Theorem 2.3 is applied to $Z = (Z_1, Z_2, Z_3, Z_4)$ in Example 2.8, then there is an edge $Z_1 \sim Z_2$, $Z_2 \sim Z_4$, $Z_3 \sim Z_4$, $Z_1 \sim Z_3$. The directions are yet to be specified. No other edges will be

present, since $Z_2 \perp Z_3|Z_4$ and $Z_1 \perp Z_4|Z_3$. Since $Z_1 \perp Z_4|Z_3$, it follows that $Z_1 - Z_3 - Z_4$ cannot be an immorality. Since $Z_3 \perp Z_2|Z_4$, it follows that $Z_3 - Z_4 - Z_2$ cannot be an immorality. Both $Z_1 - Z_2 - Z_4$ and $Z_2 - Z_1 - Z_3$ are required to be immoralities, which is not possible. Therefore, either $Z_1 - Z_2 - Z_4$ is not an immorality, in which case the model returned contains the false independence statement $Z_1 \perp Z_4|\{Z_2, Z_3\}$, or else $Z_2 - Z_1 - Z_3$ is not an immorality, in which case the model returned contains the false independence statement $Z_2 \perp Z_3|\{Z_1, Z_4\}$.

At the same time, the requirement that both $Z_1 - Z_2 - Z_4$ and $Z_2 - Z_1 - Z_3$ are immoralities can be resolved by adding in a hidden variable U to obtain Figure 2.3.

2.3 Markov Equivalence and the Essential Graph

Consider the DAG in Figure 2.8. Using the characterisation given by Theorem 2.11, the DAG in Figure 2.8 is equivalent to the DAGs in Figure 2.9.

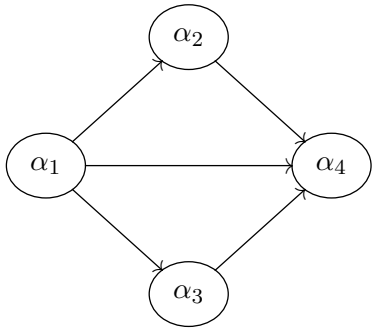

Figure 2.8: A DAG on four nodes

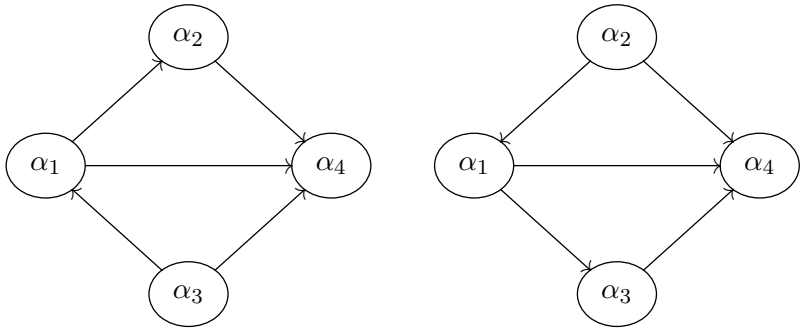

Figure 2.9: The equivalent DAGs

For the DAG in Figure 2.8, all the DAGs with the same skeleton can be enumerated, and it is clear that those in Figure 2.9 are the only two that satisfy the criteria. To find the DAGs equivalent to the one in Figure 2.8, the immorality $(\alpha_2, \alpha_4, \alpha_3)$ has to be preserved and no new immoralities

may be added. The directed edges (α_1, α_4), (α_2, α_4) and (α_3, α_4) are therefore *essential*; the directed edges (α_2, α_4) and (α_3, α_4) to form the immorality, the directed edge (α_1, α_4) because the connection $(\alpha_2, \alpha_1, \alpha_3)$ is either a fork or chain, forcing (α_1, α_4) to prevent a cycle. These three directed edges will be present in any equivalent DAG. The other three edges may be oriented in 2^3 different ways, but only 5 of these lead to DAGs (the other graphs contain cycles) and of these 5, only the three shown in Figures 2.8 and 2.9 have the same immoralities.

A useful starting point for locating all the DAGs that are Markov equivalent to a given DAG is to locate the *essential graph*, given in the following definition.

Definition 2.15 (Essential Graph)**.** *Let $\mathcal{G}$ be a Directed Acyclic Graph. The* essential graph *$\mathcal{G}^*$ associated with $\mathcal{G}$ is the graph with the same skeleton as $\mathcal{G}$, but where an edge is directed in $\mathcal{G}^*$ if and only if it occurs as a directed edge with the same orientation in* every *DAG that is Markov equivalent to $\mathcal{G}$. The* directed *edges of $\mathcal{G}^*$ are the* essential *edges of $\mathcal{G}$.*

The edges that are directed in an essential graph are the *compelled* edges.

Definition 2.16 (Compelled Edge)**.** *Let $\mathcal{G} = (V, E)$ be a chain graph, where $E = D \cup U$. A directed edge $(\alpha, \beta) \in D$ is said to be* compelled *if it occurs in at least one of the configurations in Figure 2.10.*

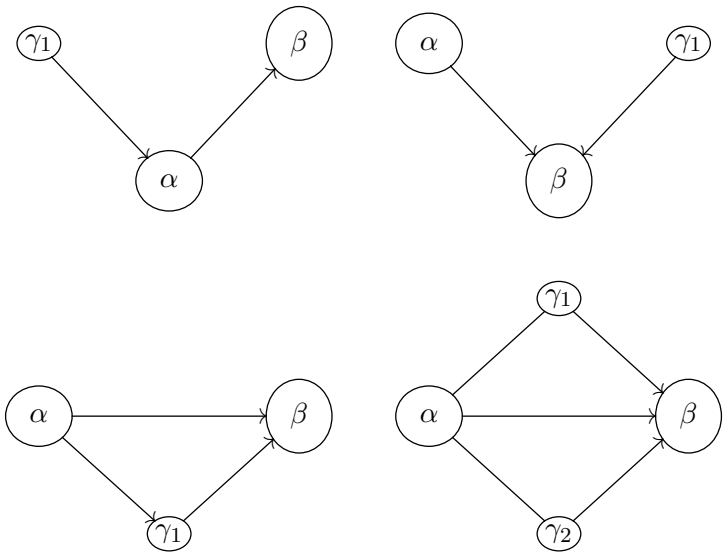

Figure 2.10: The directed edge (α, β) is compelled (Definition 2.16)

Lemma 2.17. *In an essential graph, the directed edges are the compelled edges; all other edges are undirected.*

Proof From the definition, the directed edges are those that necessarily have the same direction in every Markov equivalent DAG. The figure in the top right shows the immoralities; these are necessarily

directed. The direction is forced in the figure in the top left; otherwise the graph contains an additional immorality. The direction is forced in the structure on the bottom left; otherwise there is a cycle. The direction is forced in the structure in the bottom right; otherwise both (γ_1, α) and (γ_2, α) are forced to prevent cycles appearing and $(\gamma_1, \alpha, \gamma_2)$ is an additional immorality.

To show that these are the only compelled edges: Consider two nodes α and β which are neighbours. Firstly, suppose that α and β do not have any other common neighbours. If α does not have a neighbour γ such that there is a directed edge (γ, α), then the direction (α, β) is not forced; no additional immorality or cycle is created by either direction.

Now suppose that α and β have at least one common neighbour. Suppose that there are no neighbours γ such that both (α, γ), (γ, β) in the directed edge set, then it is not necessary to force the direction (α, β) to prevent a cycle.

Suppose, furthermore, that there is no pair of common neighbours γ_1 and γ_2, such that (γ_1, β) and (γ_2, β) are both in the directed edge set and $(\gamma_1, \alpha, \gamma_2)$ is not an immorality. Then it is not necessary to force the direction (α, β) to prevent either $(\gamma_1, \alpha, \gamma_2)$ becoming an immorality or else a cycle appearing. □

Notes The terminology *Markov model* corresponding to a Directed Acyclic Graph $\mathcal{G} = (V, E)$ was introduced into the literature and may be found in Andersson, Madigan, Perlman and Triggs [2].The results on Markov equivalence are taken from T. Verma and J. Pearl [140]. The rules for determining *compelled* edges were formulated by Meek in [93]. A rigorous treatment covering chain graphs is [129] (Studený). Exercise 7 page 352 is taken from [150], while Exercises 4 page 352 and 5 page 352 are taken from Chickering [24].

Chapter 3

Intervention Calculus

3.1 Causal Models and Bayesian Networks

In many applications, a Bayesian network is constructed as a *causal* model, where for each variable, its parent variables are considered to be *direct causes* that influence the value taken by the variable.

For example, an earth tremor or a burglary can *cause* the burglar alarm to go off and the arrows in the associated collider DAG represent *cause* to *effect* relations. It is self evident, but nevertheless has to be stated, that only *associations* can be inferred from an $n \times d$ data matrix $\mathbf{x}$ of instantiations; directions of cause to effect cannot be inferred from data alone. When conditional independence statements are learned from data, this can be interpreted as a Markov model and it may be possible to construct an efficient factorisation of the distribution using these conditional independence statements. Clearly, this factorisation cannot be understood as a *causal* model, unless there are other modelling assumptions.

For example, consider a model containing observable variables A, B, C, where there are hidden variables H_1, H_2 that are unknown to the experimenter. If the causal diagram representing the causal relations between these variables is given by the DAG on the left in Figure 3.1, then the learned DAG, along which the distribution of A, B, C can be factorised, is the DAG on the right of Figure 3.1.

This is the correct DAG, in that it preserves the D-connection properties between A, B, C, but the collider connection cannot be interpreted as A and B having a *causal* effect on C; they are *effects* of the latent common causes H_1 and H_2.

If a Bayesian network is to be interpreted as a causal model, then the possible directions of cause

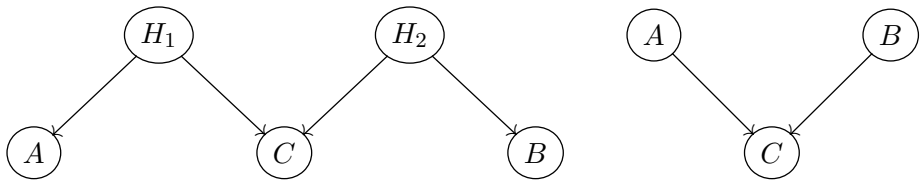

Figure 3.1: Hidden causes and the learned DAG

to effect must be part of the modelling assumptions before the data is analysed, determined by other considerations. The data analysis only determines which directed edges remain and which are removed. From data, one can determine whether or not there is an association between earth tremors and alarms triggered; it is not possible to determine from the data what causes what.

This is self evident, but surprisingly it turns out that it is necessary to state this. The purpose of the article by Freedman and Humphreys [43] (1999) was to point out the obvious fact that causality could not be inferred from data alone and was a necessary response to obvious errors in the literature; the term 'causal discovery' has been used in surprising ways, to describe learning a directed edge in a DAG. even after it had been established, with simple concrete and obvious examples, that the idea that such arrows learned from data alone actually represented causality, was ridiculous and long after publication of [43] illustrating that it was ridiculous. The article by Freedman and Humphreys is a good article; it is surprising that the literature had degenerated to such an extent that it was necessary for the authors to write it.

To define a *causal* network, an additional ingredient is needed; this is the concept of *intervention*, introduced by Judea Pearl in the seminal article [107] from 1995.

3.2 Conditioning by Observation and by Intervention

Let X and Y be two random variables and suppose that $X = x$ is *observed*. Then the conditional probability of $Y = y$ is defined as

$$\mathbb{P}_{Y|X}(y|x) = \frac{\mathbb{P}_{X,Y}(x,y)}{\mathbb{P}_X(x)}.$$

This formula describes the way that the probability distribution of the random variable Y changes after $X = x$ is observed. If, instead, the value $X = x$ is *forced* by the observer, irrespective of other considerations, the conditional probability statement is invalid.

If random variables are linked through a *causal* model, expressed by a directed acyclic graph, where parent variables have a causal effect on their children, some attempt can be made to compute the probability distribution over the remaining variables when the states of some variables are forced.

In a controlled experiment, a variable is *forced* to take a particular value, chosen at random, irrespective of the other variables in the network. In terms of the directed acyclic graph, the variable is instantiated with this value, the directed edges between the variable and its parents are removed (because the parents no longer have influence on the state of the variable) and all other conditional probabilities remain unaltered.

3.3 The Intervention Calculus for a Bayesian Network

Notations For any $A = (i_1, \ldots, i_m) \subset V$, let $X_A = (X_{i_1}, \ldots, X_{i_m})$ and, for $x = (x_1, \ldots, x_d) \in \mathcal{X}$, let $x_A = (x_{i_1}, \ldots, x_{i_m})$. The *set difference*, written $V \backslash A$, is defined as all the indices in V which are not included in A. A change of order of the variables will be employed for convenience, which should be clear from the context:

$$x = x_V = \times_{v=1}^{d} x_v = (\times_{v\in V\setminus A} x_v).(\times_{v\in A} x_v) = x_{V\setminus A}.x_A.$$

Let ϕ be a function defined on $\mathcal{X}$. Then the quantity $\sum_{V\setminus A}\phi$ is defined as

$$\left(\sum_{V\setminus A}\phi\right)(x_A) = \sum_{x_{V\setminus A}} \phi(x_{V\setminus A}.x_A).$$

Definition 3.1 (The Intervention Formula)**.** *The conditional probability of $X_{V\setminus A} = x_{V\setminus A}$, given that the variables X_A were forced to take the values x_A independently of all else, is written*

$$\mathbb{P}_{V\setminus A\|A}(x_{V\setminus A}|X_A \leftarrow x_A) \quad or \quad \mathbb{P}_{V\setminus A\|A}(x_V\|x_A)$$

and defined as

$$\mathbb{P}_{V\setminus A\|A}(x_{V\setminus A}|X_A \leftarrow x_A) = \mathbb{P}_{V\setminus A\|A}(x_{V\setminus A}\|x_A) = \prod_{v\in V\setminus A} \mathbb{P}_{v|Pa(v)}(x_v|x_{Pa(v)}). \tag{3.1}$$

Note that (3.1) is equivalent to:

$$\mathbb{P}_{V\setminus A\|A}(x_{V\setminus A}\|x_A) = \frac{\mathbb{P}_V(x_V)}{\prod_{v\in A}\mathbb{P}_{v|\mathrm{Pa}(v)}(x_v|x_{\mathrm{Pa}(v)})}. \tag{3.2}$$

The last expression of Equation (3.1) is in terms of the required factorisation; *instantiation of the variables indexed by the set A* and *elimination of those edges in D which lead from the parents of the nodes in$A$ to the nodes in $V\setminus A$*. The terminology 'local surgery' is used to describe such an elimination. A local surgery is performed and the conditional probabilities on the remaining edges are multiplied. This yields a factorisation along a *mutilated graph* where the *direct causes* of the manipulated variable are put out of effect.

The intervention formula (3.1) is obtained by wiping out those factors from the factorisation which correspond to the interventions. An explicit translation of intervention in terms of 'wiping out' equations was first proposed by Strotz and Wold [128] (1960).
The quantity $\mathbb{P}_{V\setminus A\|A}(.\|x_A)$ from Definition 3.1 defines a family of probability measures over $\mathcal{X}_{V\setminus A}$, which depends on the values x_A, which may be considered as *parameters*. These are the values forced on the variables indexed by A. This family includes original probability measure; if $A = \varnothing$, then $\mathbb{P}_{V\setminus A\|A}(.\|x_A) = \mathbb{P}_X(.)$. This family is known as the *intervention measure*. In addition, the expression on the right hand side of (3.1) is called the *intervention formula*.

Intervention An 'intervention' is an action taken to force a variable into a certain state, without reference to its own current state, or the states of any of the other variables. It may be thought of as choosing the values x_A^* for the variables X_A by using a random generator independent of the variables X.

Remark In the same style of notation, *conditioning by observation* is

$$\mathbb{P}_{X_{V\setminus A}|X_A}(x_{V\setminus A}|\text{see}(x_A)) = \mathbb{P}_{X_{V\setminus A}|X_A}(x_{V\setminus A}|x_A) \tag{3.3}$$

where, by the standard definition of conditional probability,

$$\mathbb{P}_{V\setminus A|A}(x_{V\setminus A}|x_A) = \frac{\mathbb{P}_V(x)}{\mathbb{P}_X(x_A)}. \tag{3.4}$$

Example 3.2.

Consider the DAG given in Figure 3.2, for 'X having causal effect on Y'.

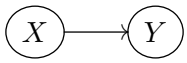

Figure 3.2: A DAG for X having causal effect on Y

The factorisation of $\mathbb{P}_{X,Y}$ along the DAG in Figure 3.2 is

$$\mathbb{P}_{X,Y}(x,y) = \mathbb{P}_{Y|X}(y|x)\mathbb{P}_X(x)$$

and the intervention formula gives

$$\mathbb{P}_{Y\|X}(y\|x) = \mathbb{P}_{Y|X}(y|x).$$

Since X is a parent of Y, the intervention to force $X = x$ produces exactly the same conditional probability distribution over Y as *observing* $X = x$. But if instead Y is forced, the intervention formula yields

$$\mathbb{P}_{X\|Y}(x\|y) = \mathbb{P}_X(x).$$

Clearly, $\mathbb{P}_{X\|Y}(x\|y) \neq \mathbb{P}_{X|Y}(x|y)$ as functions unless X and Y are independent. □

Example 3.3 (The DAG for a wet pavement)**.**

The 'wet pavement' example is a classic illustration, introduced by Judea Pearl. See, for example, page 15 of [109]. The DAG represents a causal model for a wet pavement and is given in Figure 3.3. The *season* A has four states; spring, summer autumn, winter. Rain B has two states; yes / no. Sprinkler C has two states; on / off. Wet pavement D has two states; yes / no. Slippery pavement E has two states; yes / no.

The joint probability distribution is factorised as

$$\mathbb{P}_{A,B,C,D,E} = \mathbb{P}_A\mathbb{P}_{B|A}\mathbb{P}_{C|A}\mathbb{P}_{D|B,C}\mathbb{P}_{E|D}.$$

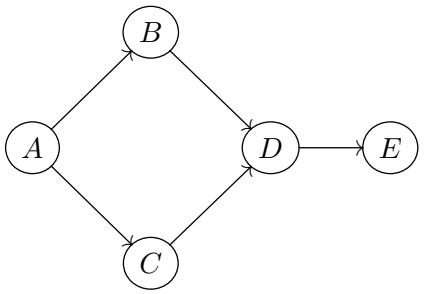

Figure 3.3: DAG for wet pavement, no intervention

Suppose, without reference to the values of any of the other variables and without reference to the current state of the sprinkler, 'sprinkler on' is now *enforced*. This could be, for example, regular maintenance work, which is carried out at regular intervals, irrespective of the season or other considerations. Then

$$\begin{aligned}\mathbb{P}_{A,B,D,E\|C}(.\|C \leftarrow 1) &= \frac{\mathbb{P}_{A,B,C,D,E}(.,.,1,.,.)}{\mathbb{P}_{C|A}(1|.)} \\ &= \mathbb{P}_A\mathbb{P}_{B|A}\mathbb{P}_{D|B,C}(.|.,1)\mathbb{P}_{E|D}.\end{aligned}$$

After *observing* that the sprinkler is on, it may be inferred that the season is dry and that it probably did not rain and so on. If 'sprinkler on' is enforced, without reference to the state of the system when the action is taken, then no such inference should be drawn in evaluating the effects of the intervention. The resulting DAG is given in Figure 3.4. It is the same as before, except that $C = 1$ is fixed and the edge between C and A disappears. The deletion of the factor $\mathbb{P}_{C|A}$ represents the understanding that whatever relationships existed between sprinklers and seasons prior to the action, found from

$$\mathbb{P}_{A,B,D,E|C}(.,.,.,.|1)$$

are no longer in effect when the state of the variable is *forced*, as in a controlled experiment, without reference to the state of the system.

After *observing* that the sprinkler is on, it may be inferred that the season is dry, that it probably did not rain and so on. No such inferences may be drawn in evaluating the effects of the intervention 'ensure that the sprinkler is on'. □

3.4 Causal Models

Having defined the family of intervention measures, the concept of *causal model* may now be defined.

Definition 3.4 (Causal Model)**.** *Let* $X = (X_1, \ldots, X_d)$ *be a random vector and let* $V = \{1, \ldots, d\}$ *denote the indexing set. A* causal model *consists of the following:*

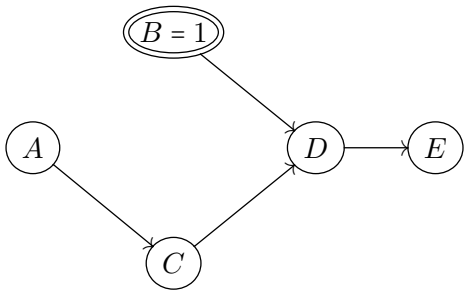

Figure 3.4: Sprinkler 'on' is forced

1. *A* Bayesian Network *for $\mathbb{P}_X$, that is, an ordering σ of the indices V, a factorisation of the probability distribution*

$$\mathbb{P}_V = \prod_{j=1}^{d} \mathbb{P}_{\sigma(j)|Pa^{(\sigma)}(j)} \tag{3.5}$$

where $Pa^{(\sigma)}(j) \subseteq \{\sigma(1), \ldots, \sigma(j-1)\}$ and is the smallest such subset such that (3.5) holds.

2. *The node set V consists of two types of nodes; V_I and V_N, where $V_I \cap V_N = \varnothing$ and $V_I \cup V_N = V$. The nodes V_I are the* interventional *nodes and V_I are the* non-interventional *nodes, where no intervention is possible. The intervention formula (3.1) holds for each subset $A \subseteq V_I$ of interventional nodes and each $x_A \in \mathcal{X}_A$.*

The arrows $\alpha \mapsto \beta$ of the DAG for either α or β (or both) in V_I are causal *arrows, indicating* direct cause *to* effect. *The remaining arrows are* non-causal*; a cause to effect relation between nodes α and β cannot be inferred from an arrow $\alpha \mapsto \beta$ if both $\alpha, \beta \in V_N$.*

In many cases, a model contains hidden variables, which cannot be observed. A special case of this is the *semi-Markov model*, where the hidden variables are common causes and where none of the hidden variables have observable anscestors.

Definition 3.5 (Semi-Markov Model)**.** *A* semi-Markov model *is a* causal model *for a random vector X with node set $V = \mathbf{V} \cup \mathbf{U}$, where $\mathbf{V}$ are the* observable *variables, $V_I \subset \mathbf{V}$ (intervention can be made on a subset of the observable nodes), $\mathbf{V}_N = \mathbf{V} \backslash V_I$ (observable nodes on which intervention cannot be made) and $V_N = \mathbf{V}_N \cup \mathbf{U}$.*

The nodes of V_I correspond to interventional variables, $\mathbf{U} \subset V_N$ are the hidden (latent) variables and $\mathbf{V}_N$ represent the observable variables on which no intervention can be made.

For a semi-Markov model*, the requirement is that the variables of $\mathbf{U}$ have no ancestors in $\mathbf{V}$.*

Notation Throughout, if a variable is named U, or U_i (for some index i), it may be assumed that the variable (or its index) belongs to $\mathbf{U}$.

3.4.1 Establishing a Causal Model via a Controlled Experiment

If sufficient data is available, a suitable Bayesian Network may be learned from the data. A *causal* model cannot be established from data alone. Additional information is needed, which is obtained through interventions on the interventional variables.

For example, the three graphs in Figure 3.5 are Markov equivalent; if the probability distribution factorises along one of these graphs, it also factorises along the others. The chains $\alpha \to \gamma \to \beta$ and $\alpha \leftarrow \gamma \leftarrow \beta$ and the fork $\alpha \leftarrow \gamma \to \beta$ are all Markov equivalent, with D-separation structure $\alpha \perp\!\!\!\perp \beta \| \gamma$. If any of these DAGs represents a *causal* network, then it is not possible to learn the causal network from the data alone.

Suppose that it is possible to intervene by controlling the variable X_γ, then if one of these graphs is the DAG for a *causal* network, it will be possible to establish which one through a controlled experiment. Figure 3.6 shows the associated structural model when the control $X_\gamma \leftarrow z$ has been applied, forcing X_γ to be independent of its ancestors. A controlled experiment, where the direct causal links between X_γ and its parent variables have been eliminated, will exhibit independence structure $X_\alpha \perp \{X_\beta, X_\gamma\}$ in the first case, $X_\alpha \perp X_\beta | X_\gamma$ in the second $\{X_\alpha, X_\gamma\} \perp X_\beta$ in the third. Once the associations $X_\alpha \perp X_\beta | X_\gamma$, $X_\alpha \not\perp X_\beta$, $X_\alpha \not\perp X_\gamma$, $X_\alpha \not\perp X_\gamma | X_\beta$, $X_\beta \not\perp X_\gamma$ and $X_\beta \not\perp X_\gamma | X_\alpha$ have been established, an additional controlled experiment, if it is possible to control the variable X_γ with interventions to force all possible values of X_γ, will determine which graph within the equivalence class is appropriate.

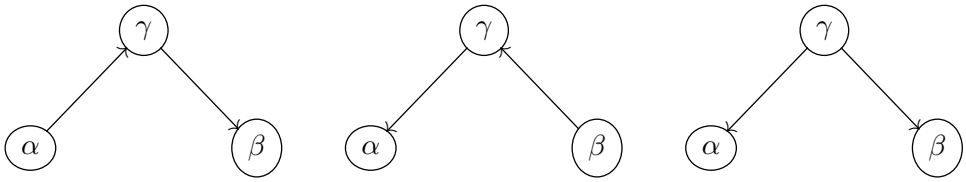

Figure 3.5: Three Markov Equivalent Graphs

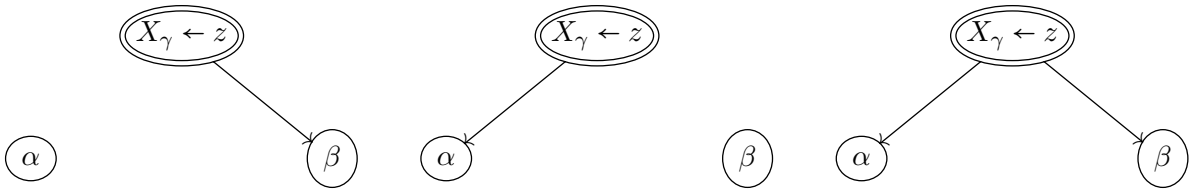

Figure 3.6: Graphs from Figure 3.5 with intervention $X_\gamma \leftarrow z$ applied

If it is possible to control variables, then it is possible to learn whether or not a collider represents independent causes with a common effect. If the DAG on the left hand side of Figure 3.1 represents a causal structure, then an experiment where variable A is controlled will establish that it is not a

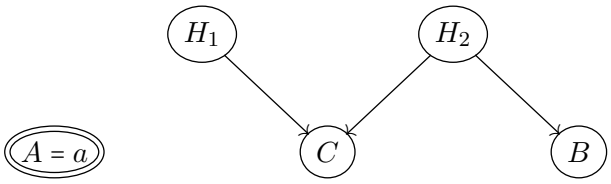

Figure 3.7: Hidden causes H_1 and H_2; intervention $A = a$

direct cause of C, since an intervention on A leaves it separated from the rest of the network, as in Figure 3.7.

3.5 Properties of Intervention Calculus

The following propositions summarise some basic properties of the intervention calculus.

Proposition 3.6. *If X_i has no parents, then for all $x \in \mathcal{X}$*

$$\mathbb{P}_{V\setminus\{i\}\|i}(x_{V\setminus\{i\}}\|x_i) = \mathbb{P}_{V\setminus\{i\}|i}(x_{V\setminus\{i\}}|x_i).$$

Proof This is straightforward from the definition;

$$\mathbb{P}_{V\setminus\{i\}\|i}(x_{V\setminus\{i\}}\|x_i) = \prod_{j\neq i}\mathbb{P}_{j|\mathrm{Pa}(j)}(x_j|\pi_j)$$

while if X_i does not have any parents

$$\mathbb{P}_{V\setminus\{i\}|i}(x_{V\setminus\{i\}}|x_i) = \frac{\prod_j \mathbb{P}_{j|\mathrm{Pa}(j)}(x_j|\pi_j)}{\mathbb{P}_i(x_i)} = \mathbb{P}_i(x_i)\frac{\prod_{j\neq i}\mathbb{P}_{j|\mathrm{Pa}(j)}(x_j|\pi_j)}{\mathbb{P}_i(x_i)} = \prod_{j\neq i}\mathbb{P}_{j|\mathrm{Pa}(j)}(x_j|\pi_j).$$

□

Proposition 3.7. *For each $j \in V$, let $Pa(j)$ denote the set of parents of node j and $\mathcal{X}_{Pa(j)}$ the state space of $X_{Pa(j)}$. For each $(x,\pi) \in \mathcal{X}_j \times \mathcal{X}_{Pa(j)}$,*

$$\mathbb{P}_{j\|Pa(j)}(x\|\pi) = \mathbb{P}_{j|Pa(j)}(x|\pi). \tag{3.6}$$

For all $j \in V$ and each $S \subseteq V$ such that $S \cap (\{j\} \cup \{Pa(j)\}) = \varnothing$, for each $(x_j, \pi_j, x_S) \in \mathcal{X}_j \times \mathcal{X}_{Pa(j)} \times \mathcal{X}_S$,

$$\mathbb{P}_{j\|Pa(j),S}(x_j\|\pi_j, x_S) = \mathbb{P}_{j|Pa(j)}(x_j|\pi_j). \tag{3.7}$$

Proof Equation (3.6) is established first. $\mathbb{P}_{j\|\mathrm{Pa}(j)}(.\|\pi_j)$ is a marginal distribution which depends on the enforced value $X_{\mathrm{Pa}(j)} \leftarrow \pi_j$. For all $(x_j, \pi_j) \in \mathcal{X}_j \times \mathcal{X}_{\mathrm{Pa}(j)}$,

$$\mathbb{P}_{j\|\mathrm{Pa}(j)}(x_j\|\pi_j) = \sum_{y_{V\setminus\mathrm{Pa}(j)}|y_j=x_j} \mathbb{P}_{V\setminus\mathrm{Pa}(j)\|\mathrm{Pa}(j)}(y_{V\setminus\mathrm{Pa}(j)}\|\pi_j).$$

An application of (3.1), the intervention formula, yields

$$\mathbb{P}_{j\|\mathrm{Pa}(j)}(x\|\pi) = \sum_{x_{V\setminus\mathrm{Pa}(j)}|x_j=x} \left(\prod_{v\in V\setminus\mathrm{Pa}(j)} \mathbb{P}_{v|\mathrm{Pa}(v)}(x_v|\underline{x}_{\mathrm{Pa}(v)}) \right)\Bigg|_{(x_j,\underline{x}_{\mathrm{Pa}(j)})=(x,\pi)}.$$

It follows, using

$$\sum_{x\in\mathcal{X}_v} \mathbb{P}_{v|\mathrm{Pa}(v)}(x|\pi_v) = 1$$

for any $\pi_v \in \mathcal{X}_{\mathrm{Pa}(v)}$ that

$$\mathbb{P}_{j\|\mathrm{Pa}(j)}(x\|\pi) = \mathbb{P}_{j|\mathrm{Pa}(j)}(x|\pi)$$

for all $(x,\pi) \in \mathcal{X}_j \times \mathcal{X}_{\mathrm{Pa}(j)}$ as required. The proof of Equation (3.7) is similar. □

Once all the *direct* causes of a variable X_j are controlled, no other interventions will affect the conditional probability distribution of X_j.

The following property is another straightforward consequence of the definition.

Proposition 3.8. *For any* $(x,\pi) \in \mathcal{X}_j \times \mathcal{X}_{Pa(j)}$,

$$\mathbb{P}_{Pa(j)\|j}(\pi\|x) = \mathbb{P}_{Pa(j)}(\pi).$$

Proof By marginalisation, followed by an application of the intervention formula (3.1), for each $(x,\pi) \in \mathcal{X}_j \times \mathcal{X}_{\mathrm{Pa}(j)}$,

$$\begin{aligned}
\mathbb{P}_{\mathrm{Pa}(j)\|j}(\pi\|x) &= \sum_{x_{V\setminus(\{j\}\cup\mathrm{Pa}(j))}} \mathbb{P}_{V\setminus(\{j\}\cup\mathrm{Pa}(j)),\mathrm{Pa}(j)\|j}(x_{V\setminus(\{j\}\cup\mathrm{Pa}(j))},\pi\|x) \\
&= \sum_{x_{V\setminus(\{j\}\cup\mathrm{Pa}(j))}} \frac{\mathbb{P}_V(x_V)}{\mathbb{P}_{X_j|\mathrm{Pa}(j)}(x|\pi)}\Bigg|_{(x_j,x_{\mathrm{Pa}(j)})=(x,\pi)} \\
&= \frac{\sum_{x_{V\setminus(\{j\}\cup\mathrm{Pa}(j))}} \mathbb{P}_V(x_V)}{\mathbb{P}_{j|\mathrm{Pa}(j)}(x|\pi)}\Bigg|_{(x_j,x_{\mathrm{Pa}(j)})=(x,\pi)} \\
&= \frac{\mathbb{P}_{j,\mathrm{Pa}(j)}(x,\pi)}{\mathbb{P}_{j|\mathrm{Pa}(j)}(x|\pi)} = \frac{\mathbb{P}_{\mathrm{Pa}(j)}(\pi)\mathbb{P}_{j|\mathrm{Pa}(j)}(x|\pi)}{\mathbb{P}_{j|\mathrm{Pa}(j)}(x|\pi)} \\
&= \mathbb{P}_{\mathrm{Pa}(j)}(\pi).
\end{aligned}$$

□

The probability measure after intervention is factorised along the mutilated graph. The following proposition determines the probabilities on the mutilated graph.

Proposition 3.9. *Let $A \subset V$. For $j \notin A$ and any $(y, x_{Pa(j)\setminus A}, x_A) \in \mathcal{X}_j \times \mathcal{X}_{Pa(j)} \times \mathcal{X}_A$,*

$$\mathbb{P}_{j|Pa(j)\setminus A\|A}(y|x_{Pa(j)\setminus A}\|x_A) = \mathbb{P}_{j|Pa(j)}(y|x_{Pa(j)}),$$

where the conditioning is taken in the sense of: first *the 'do' conditioning $X_A \leftarrow x_A$ is applied and* then *the set of variables $Pa(j)\setminus A$ is observed.*

Proof By definition of conditional probability,

$$\mathbb{P}_{j|\mathrm{Pa}(j)\setminus A\|A}(y|x_{\mathrm{Pa}(j)\setminus A}\|x_A) = \frac{\mathbb{P}_{j,\mathrm{Pa}(j)\setminus A\|A}(y, x_{\mathrm{Pa}(j)\setminus A}\|x_A)}{\mathbb{P}_{\mathrm{Pa}(j)\setminus A\|A}(x_{\mathrm{Pa}(j)\setminus A}\|x_A)}.$$

An application of the intervention formula to the numerator gives

$$\begin{aligned}\mathbb{P}_{j,\mathrm{Pa}(j)\setminus A\|A}(y, x_{\mathrm{Pa}(j)\setminus A}\|x_A) &= \sum_{\mathcal{X}_{V\setminus(\{j\}\cup \mathrm{Pa}(j)\cup A)}} \prod_{v\notin A} \mathbb{P}_{v|\mathrm{Pa}(v)}(x_v|x_{\mathrm{Pa}(v)}) \\ &= \mathbb{P}_{j|\mathrm{Pa}(j)}(y|x_{\mathrm{Pa}(j)}) \sum_{\mathcal{X}_{V\setminus(\{j\}\cup \mathrm{Pa}(j)\cup A)}} \prod_{v\notin A} \mathbb{P}_{v|\mathrm{Pa}(v)}(x_v|x_{\mathrm{Pa}(v)})\end{aligned}$$

where $x_j = y$ and to the denominator gives

$$\mathbb{P}_{\mathrm{Pa}(j)\setminus A\|A}(x_{\mathrm{Pa}(j)\setminus A}\|x_A) = \sum_{\mathcal{X}_j} \mathbb{P}_{j|\mathrm{Pa}(j)}(x_j|x_{\mathrm{Pa}(j)}) \sum_{\mathcal{X}_{V\setminus(\{j\}\cup \mathrm{Pa}(j)\cup A)}} \prod_{v\in \mathrm{Pa}(j)\setminus A} \mathbb{P}_{v|\mathrm{Pa}(v)}(x_v|x_{\mathrm{Pa}(v)}).$$

Summing from right to left, the variables in $V\setminus(A \cup \mathrm{Pa}(j) \cup \{j\})$ with j in their parent set have been summed over so that

$$\mathbb{P}_{\mathrm{Pa}(j)\setminus A\|A}(x_{\mathrm{Pa}(j)\setminus A}\|x_A) = \sum_{\mathcal{X}_{V\setminus(\{j\}\cup \mathrm{Pa}(j)\cup A)}} \prod_{v\in \mathrm{Pa}(j)\setminus A} \mathbb{P}_{v|\mathrm{Pa}(v)}(x_v|x_{\mathrm{Pa}(v)}),$$

giving

$$\mathbb{P}_{j|\mathrm{Pa}(j)\setminus A\|A}(y|x_{\mathrm{Pa}(j)\setminus A}\|x_A) = \mathbb{P}_{j|\mathrm{Pa}(j)}(y|x_{\mathrm{Pa}(j)})$$

as claimed. The proof is complete. □

Example 3.10 (Wet Pavement Revisited)**.**

Consider the conditional probability $\mathbb{P}_{D|B\|C}(.|.\|1)$. Here B and C are parents of D, $\mathrm{Pa}(D)\setminus B = C$. Plugging into the formula in the preceding proposition,

$$\mathbb{P}_{D|B\|C}(.|.\|1) = \mathbb{P}_{D|B,C}(.|., 1).$$

□

The right hand side may be thought of as a *pre-intervention probability*, which can be estimated from the data *before* the intervention $C \leftarrow 1$ is made. In this case, an estimate of the pre-intervention probability $\mathbb{P}_{D|B,C}(.|.,1)$ is also an estimate of the *post-intervention* probability $\mathbb{P}_{D|B\|C}(.|.\|1)$.

Transformations of Probability The following proposition is almost a direct consequence of the definition. It presents a simple rearrangement of the intervention formula in a special case.

Proposition 3.11.

$$\mathbb{P}_{V\setminus\{j\}\|j}(x_{V\setminus\{j\}}\|y) = \mathbb{P}_{V\setminus(\{j\}\cup Pa(j))|j,Pa(j)}(x_{V\setminus(\{j\}\cup Pa(j))}|y, x_{Pa(j)})\mathbb{P}_{Pa(j)}(x_{Pa(j)})$$

Proof An application of the definition gives

$$\mathbb{P}_{V\setminus\{j\}\|j}(x_{V\setminus\{j\}}\|y) = \prod_{v\in V\setminus\{j\}} \mathbb{P}_{v|\mathrm{Pa}(v)}(x_v|x_{\mathrm{Pa}(j)}).$$

One term has been removed in the product, namely, $\mathbb{P}_{j|\mathrm{Pa}(j)}(y|x_{\mathrm{Pa}(j)})$, so that (with $x_j = y$)

$$\begin{aligned}\prod_{v\in V\setminus\{j\}} \mathbb{P}_{v|\mathrm{Pa}(v)}(x_v|x_{\mathrm{Pa}(j)}) &= \frac{\mathbb{P}_V(x_V)}{\mathbb{P}_{j|\mathrm{Pa}(j)}(y|x_{\mathrm{Pa}(j)})} \\ &= \frac{\mathbb{P}_V(x)\mathbb{P}_{\mathrm{Pa}(j)}(x_{\mathrm{Pa}(j)})}{\mathbb{P}_{j,\mathrm{Pa}(j)}(y, x_{\mathrm{Pa}(j)})} \\ &= \mathbb{P}_{V\setminus(\{j\}\cup\mathrm{Pa}(j))|j,\mathrm{Pa}(j)}(x_{V\setminus(\{j\}\cup\mathrm{Pa}(j))}|y, x_{\mathrm{Pa}(j)})\mathbb{P}_{\mathrm{Pa}(j)}(x_{\mathrm{Pa}(j)})\end{aligned}$$

as required. □

Proposition 3.12 (Adjustment for Direct Causes)**.** *Let $\mathcal{G} = (V, D)$ be a DAG and let $B \subset V$ such that $(\{j\} \cup Pa(j)) \cap B = \varnothing$ (the empty set). Then for any $(x, y) \in \mathcal{X}_j \times \mathcal{X}_B$,*

$$\mathbb{P}_{B\|j}(y\|x) = \sum_{\mathcal{X}_{Pa(j)}} \mathbb{P}_{B|j,Pa(j)}(y|x, x_{Pa(j)})\mathbb{P}_{Pa(j)}(x_{Pa(j)}). \tag{3.8}$$

Proof of Proposition 3.12 Firstly, with $x_j = y$,

$$\mathbb{P}_{B\|j}(x_B\|y) = \sum_{\mathcal{X}_{V\setminus(B\cup\{j\})}} \mathbb{P}_{V\setminus\{j\}\|j}(x_{V\setminus\{j\}}\|y).$$

By Proposition 3.11, this may be written as

$$\mathbb{P}_{B\|j}(x_B\|y) = \sum_{\mathcal{X}_{V\setminus(B\cup\{j\})}} \mathbb{P}_{V\setminus(\{j\}\cup\mathrm{Pa}(j))|\{j\}\cup\mathrm{Pa}(j)}(x_{V\setminus(\{j\}\cup\mathrm{Pa}(j))}|y, x_{\mathrm{Pa}(j)})\mathbb{P}_{\mathrm{Pa}(j)}(x_{\mathrm{Pa}(j)}).$$

A marginalisation over $\mathcal{X}_{V\setminus(B\cup\{j\})}$ gives

$$\mathbb{P}_{B|j}(x_B|y) = \sum_{x_{\mathrm{Pa}(j)}} \mathbb{P}_{B|j,\mathrm{Pa}(j)}(x_B|y, x_{\mathrm{Pa}(j)})\mathbb{P}_{\mathrm{Pa}(j)}(x_{\mathrm{Pa}(j)})$$

as required. The proof is complete. □

3.6 Confounding, The 'Sure Thing' Principle and Simpson's Paradox

3.6.1 Confounding

Consider the DAG given in Figure 3.8. It corresponds to the factorisation:

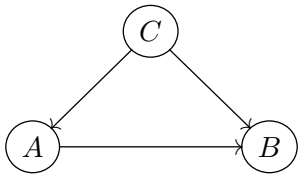

Figure 3.8: Illustration for Confounding

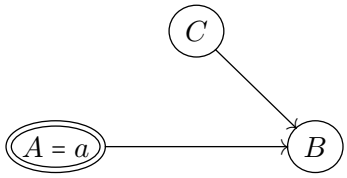

Figure 3.9: Intervention on A

$$\mathbb{P}_{A,B,C} = \mathbb{P}_{B|A,C}\mathbb{P}_{A|C}\mathbb{P}_C.$$

Consider the conditional probability of B, when A is controlled; $\mathbb{P}_{B\|A}(.\|a)$. The DAG illustrating the intervention is shown in Figure 3.9. Note that

$$\mathbb{P}_{B\|A}(.\|a) = \sum_{c\in\mathcal{X}_C} \mathbb{P}_{B,C\|A}(.,c\|a).$$

and that

$$\mathbb{P}_{B,C\|A}(.,.\|a) = \mathbb{P}_{B|C\|A}(.|.\|a)\mathbb{P}_{C\|A}(.\|a) = \mathbb{P}_{B|A,C}(.|a,.)\mathbb{P}_C,$$

where in the second term, the do-conditioning of $A \leftarrow a$ is applied first, and then C is observed. It follows that

$$\mathbb{P}_{B\|A}(.\|a) = \sum_{c\in\mathcal{X}_C} \mathbb{P}_{B|A,C}(.|a,c)\mathbb{P}_C(c).$$

This shows that to estimate $\mathbb{P}_{B\|A}(.\|a)$ from data alone (i.e. without controlling A), it is necessary to be able to estimate $\mathbb{P}_{B|A,C}$ and $\mathbb{P}_C$ from data. If C is *observable*, then the effect on the probability distribution of B of manipulating A may be estimated. But if C is a *hidden* random variable (sometimes the term *latent* is used) in the sense that no *direct* sample of the outcomes of C may be obtained, it will not be possible to estimate the probabilities used on the right hand side and hence it will not be possible to predict the effect on B of manipulating A. This is known as *confounding*.

3.6.2 Simpson's Paradox

Consider three binary variables, A, B and C. Simpson's paradox is the observation that there are situations where

$$\frac{\mathbb{P}_{B|C,A}(1|1,1)/\mathbb{P}_{B|C,A}(0|1,1)}{\mathbb{P}_{B|C,A}(1|1,0)/\mathbb{P}_{B|C,A}(0|1,0)} > 1 \qquad \text{and} \qquad \frac{\mathbb{P}_{B|C,A}(1|0,1)/\mathbb{P}_{B|C,A}(0|0,1)}{\mathbb{P}_{B|C,A}(1|0,0)/\mathbb{P}_{B|C,A}(0|0,0)} > 1,$$

but

$$\frac{\mathbb{P}_{B|A}(1|1)/\mathbb{P}_{B|A}(0|1)}{\mathbb{P}_{B|A}(1|0)/\mathbb{P}_{B|A}(0|0)} < 1.$$

For example let A denote 'treatment', B 'recovery' and C 'blood pressure'. Simpson's paradox states that even if the 'treatment' may improve the chances of recovery for those with high blood pressure and those with low blood pressure, it may nevertheless be bad for the population as a whole. It could be that although the treatment is comparatively good within the group where high blood pressure is observed after treatment and also comparatively good within the group where low blood pressure is observed after treatment, it may be bad for the population as a whole. This occurs if 'treatment' increases blood pressure and increased blood pressure reduces the chances of recovery.

This situation is illustrated by the DAG given in Figure 3.10, where A denotes treatment, B recovery and C blood pressure. Suppose that C is a hidden variable. Even if the 'treatment' variable A can be controlled, an intervention on A does not remove any arrows from the causal diagram; there is the possibility of a Simpson's paradox, even with a controlled experiment.

If A denotes 'treatment' and B 'recovery' and C denotes a *common cause* of both A and B, as in Figure 3.8, Simpson's paradox may be resolved if A can be controlled, because controlling A breaks the causal link between C and A. This is the *sure thing* principle, considered next, which states that if the treatment improves the chances of recovery for each level of the 'common cause' variable C, then it is good for the population as a whole.

3.6.3 The Sure Thing Principle

Consider again the situation of Figure 3.8. Suppose that A is *controlled*; values for the variable A are assigned at random, so the link $C \to A$ is broken and hence the effect on B of manipulating A is not

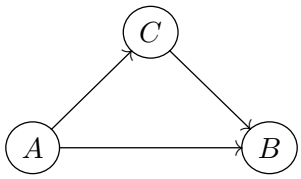

Figure 3.10: A=treatment / B=recovery / C=blood pressure

confounded by the effects of hidden variables. The following result is referred to as 'The Sure Thing Principle'. It states that when Figure 3.8 represents the causal structure and there is do-conditioning on A, then Simpson's paradox does not hold.

Proposition 3.13. *Consider three binary variables A, B, C with the network given in Figure 3.8. If*

$$\mathbb{P}_{B|C\|A}(1|1\|1) < \mathbb{P}_{B|C\|A}(1|1\|0)$$

and

$$\mathbb{P}_{B|C\|A}(1|0\|1) < \mathbb{P}_{B|C\|A}(1|0\|0)$$

then

$$\mathbb{P}_{B\|A}(1\|1) < \mathbb{P}_{B\|A}(1\|0).$$

The notation means: first A *is forced,* then C *is observed.*

Proof Firstly,

$$\mathbb{P}_{B\|A}(1\|1) = \mathbb{P}_{B|C\|A}(1|1\|1)\mathbb{P}_{C\|A}(1\|1) + \mathbb{P}_{B|C\|A}(1|0\|1)\mathbb{P}_{C\|A}(0\|1).$$

Since C is a parent of A,

$$\mathbb{P}_{C\|A}(.\|1) = \mathbb{P}_C(.).$$

It follows that

$$\mathbb{P}_{B\|A}(1\|1) = \sum_{x=0}^{1} \mathbb{P}_{B|C\|A}(1|x\|1)\mathbb{P}_{C\|A}(x\|1) = \sum_{x=0}^{1} \mathbb{P}_{B|C\|A}(1|x\|1)\mathbb{P}_C(x).$$

Similarly,

$$\mathbb{P}_{B\|A}(1\|0) = \sum_{x=0}^{1} \mathbb{P}_{B|C\|A}(1|x\|0)\mathbb{P}_C(x).$$

It now follows directly from the assumptions that

$$\mathbb{P}_{B\|A}(1\|1) < \mathbb{P}_{B\|A}(1\|0),$$

which is the stated result. □

3.7 Identifiability: Back-Door and Front-Door Criteria

In a wide variety of situations, the aim is to compute the effects of an intervention, when it is not possible to carry out a controlled experiment. The following example, quoted at the beginning of Pearl's seminal article [107] illustrates the issues involved.

Example 3.14.

Consider an experiment in which soil fumigants X are to be used to increase oat crop yields Y, by controlling the eelworm population, Z. These may also have direct effects, both beneficial and adverse, on yields, besides the control of eelworms. We would like to assess the total effects of the fumigants on yields when the study is complicated by several factors. First, controlled, randomised experiments are infeasible: farmers insist on deciding for themselves which plots are to be fumigated. Secondly, the farmers' choice of treatment depends on last year's eelworm population Z_0. This is an unknown quantity, but is strongly correlated with this year's population. This presents a classic case of confounding bias, which interferes with the assessment of the treatment effects, regardless of sample size. Fortunately, through laboratory analysis of soil samples, the eelworm populations before and after treatment can be determined. Furthermore, since fumigants are only active for a short period, they do not affect the growth of eelworms surviving the treatment; eelworm growth depends on the population of bird and other predators. This, in turn, is correlated with last year's eelworm population and hence with the treatment itself.

The situation may be represented by the causal diagram in Figure 3.11. The variables are:

- X fumigants,
- Y crop yields,
- Z_0 last year's eelworm population,
- Z_1 eelworm population before treatment,
- Z_2 eelworm population after treatment,
- Z_3 eelworm population at the end of the season,
- B population of birds and other predators.

In this example, the variables B and Z_0 are *hidden* variables.

The issue is whether *interventional* probabilities $\mathbb{P}_{Y\|X}(.\|X \leftarrow x)$ may be computed from information on the observables (Z_1, Z_2, Z_3, X, Y). When they can, they are said to be *identifiable*.

Definition 3.15 (Identifiable)**.** *The causal effect of X on Y is said to be* identifiable *if the quantity $\mathbb{P}_{Y\|X}$ can be computed uniquely from the probability distribution of the observable variables.*

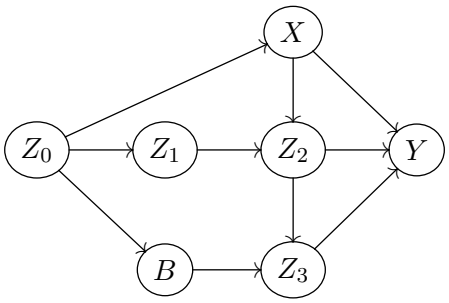

Figure 3.11: A causal diagram representing the effect of fumigants X on yields Y

In this section, two graphical conditions are described which ensure that causal effects can be estimated consistently from observational data. The first of these is named *back door criterion* and is equivalent to the *ignorability condition* of Rosenbaum and Rubin [119] (1983). The second of these is the *front-door criterion*. This involves covariates which are affected by the treatment (in this example Z_2 and Z_3).

3.7.1 Back Door Criterion

The back door criterion is defined as follows:

Definition 3.16 (Back Door Criterion)**.** *A set of nodes C satisfies the* back door criterion *relative to an ordered pair of nodes $(X, Y) \in V \times V$ if*

1. *no node of C is a descendant of X* **and**
2. *C blocks every trail (in the sense of D-separation) between X and Y which contains an edge pointing to X.*

If A and B are two disjoint subsets of nodes, C is said to satisfy the back door criterion relative to (A, B) if it satisfies the back door criterion relative to any pair $(X_i, X_j) \in A \times B$.

Example 3.17.

In Figure 3.11, the set $C = \{Z_0\}$ satisfies the back door criterion relative to (X, Y). The node Z_0 is unobservable. The set $C = \{Z_1, Z_2, Z_3\}$ does block all trails between X and Y with an arrow pointing into X, but clearly does not satisfy the back door criterion since both Z_2 and Z_3 are descendants of X. □

The name 'back door criterion' reflects the fact that the second condition requires that only trails with nodes pointing at X_i be blocked. The remaining trails can be seen as entering X_i through a back door.

Example 3.18.

Consider the back door criterion DAG, given in Figure 3.12. The sets of variables $C_1 = \{Z_3, Z_4\}$ and $C_2 = \{Z_4, Z_5\}$ satisfy the back door criterion relative to the ordered pair of nodes (X, Y), whereas $C_3 = \{Z_4\}$ does *not* satisfy the criterion relative to the ordered pair of nodes (X, Y); if Z_4 is instantiated, the Bayes ball may pass through the *collider* connection from Z_1 to Z_2.

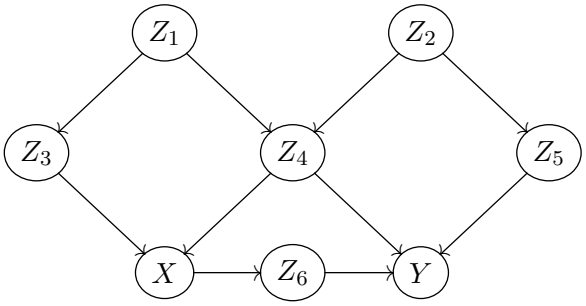

Figure 3.12: Back Door Criterion

Identifiability Consider a causal network and let $A \subseteq V \backslash \{X, Y\}$ be a subset of the variables which satisfies the back door criterion with respect to an ordered pair (X, Y). The aim is to show that the set of variables A plays a similar role to the variable C in the discussion on confounding.
The quantity $\mathbb{P}_{Y\|X}$ may be expressed as:

$$\mathbb{P}_{Y\|X} = \left(\mathbb{P}_{Y,A\|X}\right)^{\downarrow A} = \left(\mathbb{P}_{Y|A\|X}\mathbb{P}_{A\|X}\right)^{\downarrow A}$$

where the notation $\downarrow A$ means: marginalise over the variables in A.

The two terms in the sum may be expressed in terms of see-conditioning; $\mathbb{P}_{A\|X} = \mathbb{P}_A$ and $\mathbb{P}_{Y|A\|X} = \mathbb{P}_{Y|A,X}$. These may be seen as follows: firstly, since no variables of A are descendants of X, it follows that $\mathbb{P}_{A\|X} = \mathbb{P}_A$. This is seen as follows: the variables may be ordered as $V = (Y_1, \ldots, Y_n, X, Y_{n+1}, \ldots, Y_{n+m})$ where the ordering is chosen such that $\mathrm{Pa}(Y_j) \subseteq \{Y_1, \ldots, Y_{j-1}\}$ for $j \leq n$, $\mathrm{Pa}(X) \subseteq \{Y_1, \ldots, Y_n\}$ and $\mathrm{Pa}(Y_j) \subseteq \{Y_1, \ldots, Y_n, X, Y_{n+1}, \ldots, Y_{j-1}\}$ for $j \in \{m+1, \ldots, n+m\}$ and where $A \subseteq \{Y_1, \ldots, Y_n\}$. From the intervention formula,

$$\mathbb{P}_{V\backslash X\|X} = \prod_{j=1}^{m+n} \mathbb{P}_{Y_j|\mathrm{Pa}_j}$$

while

$$\mathbb{P}_V = \mathbb{P}_{X|\mathrm{Pa}(X)}(x|\pi_X) \prod_{j=1}^{m+n} \mathbb{P}_{Y_j|\mathrm{Pa}_j}.$$

Now, marginalise over variables $Y_{n+1}, \ldots, Y_{n+m}$ in both expressions, then marginalise over X in the second expression and finally marginalise over all remaining variables not in A. The same answer

obtains for both expressions, so that

$$\mathbb{P}_{A\|X} = \mathbb{P}_A.$$

Second, since A blocks all trails between Y and X that have an edge pointing towards X, it follows that $Y \perp\!\!\!\perp (\mathrm{Pa}(X)\backslash A)\|_{\mathcal{G}} A$. It follows, with notation that should be clear, using Proposition 3.12 that

$$\begin{aligned}
\mathbb{P}_{Y|A\|X} &= \left(\mathbb{P}_{Y|A,\mathrm{Pa}(X)\|X}\mathbb{P}_{\mathrm{Pa}(X)\backslash A\|X}\right)^{\downarrow \mathrm{Pa}(X)\backslash A} \\
&= \left(\mathbb{P}_{Y|A,\mathrm{Pa}(X),X}\mathbb{P}_{\mathrm{Pa}(X)\backslash A}\right)^{\downarrow \mathrm{Pa}(X)\backslash A} \\
&= \left(\mathbb{P}_{Y|A,X}\mathbb{P}_{\mathrm{Pa}(X)\backslash A}\right)^{\downarrow \mathrm{Pa}(X)\backslash A} \\
&= \mathbb{P}_{Y|A,X}.
\end{aligned}$$

To go from first to second line, do-conditioning on X does not alter the probabilities for ancestors of X, hence $\mathbb{P}_{\mathrm{Pa}(X)\backslash A\|X} = \mathbb{P}_{\mathrm{Pa}(X)\backslash A}$ Also, for conditional probabilities of Y, do-conditioning on ancestors of Y is the same as see-conditioning on ancestors of Y, hence $\mathbb{P}_{Y|A,\mathrm{Pa}(X)\|X} = \mathbb{P}_{Y|A,\mathrm{Pa}(X),X}$.

To go from second to third line: once X is known, $\mathrm{Pa}(X)$ gives no further information.

Hence, if $A \cap (X \cup Y) = \varnothing$, then (using $\mathbb{P}_{A\|X} = \mathbb{P}_A$ and summing over A) gives:

$$\mathbb{P}_{Y\|X} = \left(\mathbb{P}_{Y|A,X}\mathbb{P}_A\right)^{\downarrow A} \tag{3.9}$$

If a set of variables A satisfying the back door criterion with respect to (X,Y) can be chosen such that $\mathbb{P}_A$ and $\mathbb{P}_{Y|A,X}$ can be estimated from the observed data, then the distribution $\mathbb{P}_{Y\|X}$ can also be estimated from the observed data.

Lemma 3.19 (Identifiability)**.** *If a set of variables Z satisfies the back door criterion relative to (X,Y), then the causal effect of X to Y is given by the formula*

$$\mathbb{P}_{Y\|X} = \left(\mathbb{P}_{Y|X,Z}\mathbb{P}_Z\right)^{\downarrow Z} \tag{3.10}$$

and the intervention of X on Y is said to be identifiable.

Proof This follows directly from the definition of identifiable (Definition 3.15) and the analysis above. □

Formula (3.10) is named *adjustment for concomitants.* The word *identifiability* refers to the fact that the concomitants Z satisfying the back door criterion are observable and hence it is possible to compute, or *identify* the *intervention* probability $\mathbb{P}_{Y\|X}(y\|x)$ using the 'see' conditional probabilities $(\mathbb{P}_{X_j|\mathrm{Pa}_j})_{j=1}^d$.

3.7.2 Front Door Criterion

The *front door criterion* is defined as follows:

Definition 3.20 (Front Door Criterion)**.** *A set of variables Z satisfies the front door criterion relative to the ordered pair (X, Y) if:*

- *Z intercepts all directed paths from X to Y,*
- *there is no back-door path between X and Z,*
- *every back-door path between Z and Y is blocked by X.*

The situation is illustrated in Figure 3.13. The variable U is a hidden (latent) variable. The variable Z satisfies the front door criterion relative to (X, Y).

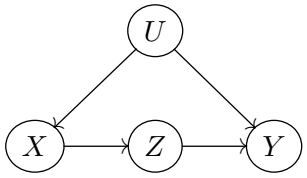

Figure 3.13: Front Door Criterion

The result is the following:

Theorem 3.21 (Front Door Criterion)**.** *Let Z satisfy the front door criterion relative to the ordered pair (X, Y). Then the causal effect on Y of an intervention on X is:*

$$\mathbb{P}_{Y\|X} = \left(\mathbb{P}_{Z|X}\mathbb{P}_{Y|Z}\right)^{\downarrow Z}.$$

This is self evident; note that $\mathbb{P}_{Y|Z} = \left(\mathbb{P}_{Y|Z,U}\mathbb{P}_U\right)^{\downarrow U}$. In other words, if the see-conditional $\mathbb{P}_{Z|X}$ and $\mathbb{P}_{Y|Z}$ are available, then the intervention $\mathbb{P}_{Y\|X}$ may be computed. □

3.7.3 Non-Indentifiability

There are various conditions for non-identifiability of $\mathbb{P}_{Y\|X}$. These include:

1. A *necessary* condition is that there is an unblockable back-door path between X and Y; that is, a path ending with an arrow pointing into X which cannot be blocked by observable non-descendants of X. This is not a sufficient condition, as Figure 3.13 illustrates. This shows a situation where there is a non-blockable back-door path, yet $\mathbb{P}_{Y\|X}$ is identifiable (front-door criterion).
2. A *sufficient* condition for identifiability of $\mathbb{P}_{Y\|X}$ is existence of a confounding path between X and any of its children on a path from X to Y; two examples are given in Figure 3.14.

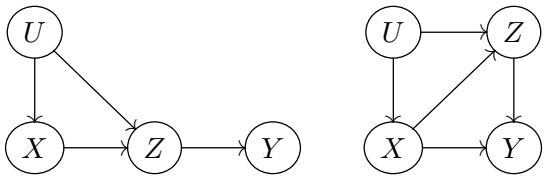

Figure 3.14: Sufficient condition for identifiability

3. Local identifiability is not a sufficient condition for global identifiability. In Figure 3.15, $\mathbb{P}_{Z_1\|X}$, $\mathbb{P}_{Z_2\|X}$, $\mathbb{P}_{Y\|Z_1}$, $\mathbb{P}_{Y\|Z_2}$ are all identifiable, but $\mathbb{P}_{Y\|X}$ is not.

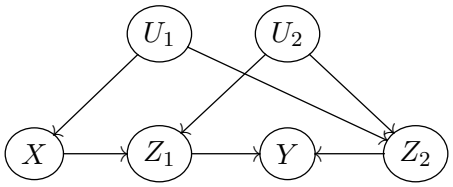

Figure 3.15: Sufficient condition for identifiability

3.8 Inference Rules for Intervention Calculus

Let X, Y and Z be arbitrary disjoint sets of nodes in a DAG $\mathcal{G} = (V, D)$. $\mathcal{G}_{\overline{X}}$ denotes the graph obtained from $\mathcal{G}$ by deleting all arrows $\{(\alpha, \beta) \in D : \beta \in X\}$, while $\mathcal{G}_{\underline{X}}$ denotes the graph obtained from $\mathcal{G}$ by deleting all arrows $\{(\alpha, \beta) : (\alpha, \beta) \in D, \alpha \in X\}$. The notation $\mathcal{G}_{\overline{X},\underline{Z}}$ denotes the graph obtained from $\mathcal{G}$ by deleting all arrows $\{(\alpha, \beta) \in D : \beta \in X \quad \text{or} \quad \alpha \in Z\}$. The following theorem states three rules that are valid for every interventional distribution compatible with $\mathcal{G}$.

Theorem 3.22 (Rules for Intervention Calculus)**.** *Let $\mathcal{G} = (V, D)$ be a DAG associated with a causal model and let $\mathbb{P}$ denote the probability distribution. For any disjoint subsets X, Y, Z, W of V, the following rules hold:*

1. *(insertion / deletion of observations)*

$$Y \perp\!\!\!\perp Z \| _{\mathcal{G}_{\overline{X}}} X \cup W \Rightarrow \mathbb{P}_{Y|Z,W\|X} = \mathbb{P}_{Y|W\|X} \tag{3.11}$$

2. *(action / observation exchange)*

$$Y \perp\!\!\!\perp Z \| _{\mathcal{G}_{\overline{X},\underline{Z}}} X \cup W \Rightarrow \mathbb{P}_{Y|W\|X,Z} = \mathbb{P}_{Y|Z,W\|X} \tag{3.12}$$

3. *(insertion / deletion of actions)*

$$Y \perp\!\!\!\perp Z \| _{\mathcal{G}_{\overline{X},\overline{Z(W)}}} X \cup W \Rightarrow \mathbb{P}_{Y|W\|X,Z} = \mathbb{P}_{Y|W\|X}. \tag{3.13}$$

where the set $Z(W)$ in the graph $\mathcal{G}_{\overline{X}}$ is the set of Z-nodes which are not ancestors of any W node in $\mathcal{G}_{\overline{X}}$.

Proof of Theorem 3.22

1. The interventional distribution $\mathbb{P}_{V\setminus X\|X}$ factorises along the graph $\mathcal{G}_{\overline{X}}$. Since the variables of X have no parents in $\mathcal{G}_{\overline{X}}$, do- and see- conditioning on X are equivalent for distributions that factorise along $\mathcal{G}_{\overline{X}}$. The separation statement implies that, for the mutilated graph (where X has been instantiated by intervention), Y is D-separated from Z by $X \cup W$. Equation (3.11) follows because a D-separation statement implies the corresponding conditional independence statement.

2. The interventional distribution $\mathbb{P}_{V\setminus X\|X}$ factorises along $\mathcal{G}_{\overline{X}}$. The D-separation statement of (3.12) implies, furthermore, that all $X \cup W$-active trails in $\mathcal{G}_{\overline{X}}$ between Y and Z have an arrow from a node in Z to one of its children, hence all *back-door* paths from Z to Y in $\mathcal{G}_{\overline{X}}$ are blocked by $X \cup W$. It follows that the operations setting $Z \leftarrow z$ or conditioning on $Z = z$ have the same effect on Y.

3. Assume that the D-separation condition holds, then all $W \cup X$ - active trails between Y and Z in $\mathcal{G}_{\overline{X}}$ have an edge $\gamma \mapsto \beta$ where $\beta \in Z(W)$ (since removing arrows into $Z(W)$ blocks the trail). For such a node β, none of its descendants are in W. Therefore,

$$\mathbb{P}_{Y|W\|X,Z(W)} = \mathbb{P}_{Y|W\|X}.$$

The D-separation statement implies that $Y \perp\!\!\!\perp Z\setminus Z(W)\|_{\mathcal{G}_{\overline{X}}} X \cup W$ from which the result follows.

□

Corollary 3.23. *Let $\mathbb{P}$ be a probability distribution which factorises according to a causal model (Definition 3.4). An intervention probability $q = \mathbb{P}_{Y\|X}(y\|x)$, where X and Y are disjoint subsets of V where $X \subseteq \mathbf{V}$ is* identifiable *if there is a finite sequence of transformations, each conforming to one of the inference rules in Theorem 3.22 which reduces q to a probability expression that only involves see conditioning.*

Proof Clear □

The converse of Corollary 3.23 is also true. This will now be dealt with. Firstly, Tian and Pearl [135](2002) developed systematic criteria for establishing the interventional statements that could be computed from see-conditioning statements. Huang and Valtorta [64] (2006) then established that these criteria could be obtained from the three rules of Theorem 3.22. The problem was also dealt with in Shpitser and Pearl [124] (2006); graphical criteria are discussed in Tian and Shpister [136] (2010).

Only the interventional probability distributions over the *observable* nodes are of interest; the following rather obvious lemma helps to simplify the problem.

Lemma 3.24. *If any of the three rules can be used on a model with graph $\mathcal{G}$, it can also be used on a model that is obtained by removing all hidden nodes $U \in \mathcal{U}$ that have no descendants among the observable nodes $\mathbf{V}$.*

Proof Clear. □

The following lemma establishes that only rules 2 and rules 3 need be considered for a completeness theorem.

Lemma 3.25. *Rule 1 follows from rule 2 and rule 3.*

Proof Since all D-separation statements *before* removal of an edge remain true after the edge is removed, the conditions for the application of rules 2 and 3 are satisfied if the condition for rule 1 is satisfied. The application of rule 1 can be replaced by the application of rule 2 followed by the application of rule 3.

In detail: suppose the D-separation statement of (3.11) holds. Then the D-separation statements of (3.12) and (3.13) both hold, so an application of rule 2 gives:

$$\mathbb{P}_{Y|W,Z\|X} = \mathbb{P}_{Y|W\|X,Z}$$

and an application of rule 3 gives:

$$\mathbb{P}_{Y|W\|X,Z} = \mathbb{P}_{Y|W\|X}$$

from which implies that:

$$\mathbb{P}_{Y|Z,W\|X} = \mathbb{P}_{Y|W\|X}.$$

This is the statement of Rule 1. □

At the heart of the systematic identification of interventional statements that may be expressed in terms of see-conditioning by Tian and Pearl [135] is the concept of c-components. All the c-factors are computable from the probability distribution over the observed variables.

Definition 3.26 (c-component)**.** *Recall that $V = \mathbf{V} \cup \mathbf{U}$, $\mathbf{V} \cap \mathbf{U} = \varnothing$, where $\mathbf{V}$ is the set of observable nodes and $\mathbf{U}$ is the set of hidden nodes. Two nodes $\alpha, \beta \in \mathbf{U}$ are related under the c-component relation (written $\alpha \sim_c \beta$) if and only if at least one of the following holds:*

1. *there is an edge between α and β*
2. *α and β are both parents of the same node $\gamma \in \mathbf{V}$*
3. *both α and β are in c-component relation with respect to another node $\delta \in \mathbf{U}$.*

A c-component of the node set V is either a set containing a single node from $\mathbf{V}$ if that node has no parents in $\mathbf{U}$ or it consists of all the $\mathbf{U}$ nodes which are c-component related to each other, together with all $\mathbf{V}$ nodes that have a parent in $\mathbf{U}$ which is a member of that c-component.

Let H denote all the nodes of a c-component and let $H' = H \cap \mathbf{V}$ (the observable nodes of a c-component). Then a c-factor is simply $\mathbb{P}_{H'}$, the probability distribution over the observable nodes of the c-component.

The relation $\sim_c$ on $\mathbf{U}$ is *reflexive*, *symmetric* and *transitive* and hence defines a partition of $\mathbf{U}$. Based on this relation, $\mathbf{U}$ can be divided into disjoint and mutually exclusive c-component related parts.

Now suppose that $\mathbb{P}$ factorises according to a semi-Markov model (Definition 3.5). Lemmas 3.27 and 3.28 form the basis of the characterisation of Tian and Pearl [135] of interventional statements that can be expressed by see-conditioning statements. The proofs given here are from [64], which demonstrate that they follow from Rules 2 and 3 of Theorem 3.22.

Lemma 3.27. *If $W \subseteq C \subseteq \mathbf{V}$ and W is an* ancestral set *in $\mathcal{G}_{C \cup (Pa(C) \cap \mathbf{U})}$, then*

$$(\mathbb{P}_{C \| \mathbf{V} \backslash C})^{\downarrow C \backslash W} = \mathbb{P}_{W \| \mathbf{V} \backslash W}.$$

Furthermore, this statement is a consequence of Rule 3 from Theorem 3.22.

A set $S \subseteq V$ in a graph $\mathcal{G}$ is called an *ancestral set* if for each $\alpha \in \mathcal{G}$, every ancestor of α is also in $\mathcal{G}$. The set $\mathrm{an}(\alpha)$ denotes the ancestors of the node α; $\alpha \notin \mathrm{an}(\alpha)$. A *topological ordering* of the nodes in a graph $\mathcal{G}$ is an ordering σ of the nodes such that for each node β and each node $\gamma \in \mathrm{an}(\beta)$, $\sigma(\gamma) < \sigma(\beta)$.

Proof Trivially,

$$(\mathbb{P}_{C \| \mathbf{V} \backslash C})^{\downarrow C \backslash W} = \mathbb{P}_{W \| \mathbf{V} \backslash C}.$$

It has to be shown that this is equal to $\mathbb{P}_{W \| \mathbf{V} \backslash W}$.

If W is an ancestral set in $\mathcal{G}_{C \cup (\mathrm{Pa}(C) \cap \mathbf{U})}$, it follows that none of the nodes of W have parents in $C \cup (\mathrm{Pa}(C) \cap \mathbf{U}) \backslash W$, although they may have parents in $\mathrm{Pa}(C) \cap \mathbf{V}$.

Since W is an ancestral set in $\mathcal{G}_{C \cup (\mathrm{Pa}(C) \cap \mathbf{U})}$, there is a topological ordering of the nodes in $\mathcal{G}_{C \cup (\mathrm{Pa}(C) \cap \mathbf{U})}$ that starts with all the nodes in W and continues with the other nodes.

The lemma may be proved by induction. If $W = V$, the lemma is trivially true. Otherwise, consider the first node, say α in the topological order just described that is in C, but not in W. By induction, it is necessary and sufficient to prove that if $W \subset V$, $\alpha \in V \backslash W$, $C = W \cup \{\alpha\}$ and W is an ancestral set in $\mathcal{G}_{C \cup (\mathrm{Pa}(C) \cap \mathbf{U})}$, then

$$(\mathbb{P}_{C \| \mathbf{V} \backslash C})^{\downarrow C \backslash W} = \mathbb{P}_{W \| \mathbf{V} \backslash W}.$$

Let the nodes of W be labelled: $W = \{1, \ldots, k\}$ and let $Y = \mathbf{V} \backslash (W \cup \{\alpha\})$. With obvious notation, the identity to be established may be rewritten as:

$$(\mathbb{P}_{W, \alpha \| Y})^{\downarrow C \backslash W} = \mathbb{P}_{W \| Y, \alpha}$$

Marginalising over the variable labelled α and using the fact that a probability distribution sums to 1 gives:

$$(\mathbb{P}_{W,\alpha\|Y})^{\downarrow\alpha} = (\mathbb{P}_{\alpha|W\|Y})^{\downarrow\alpha}\mathbb{P}_{W\|Y} = \mathbb{P}_{W\|Y}.$$

By construction,

$$W \perp\!\!\!\perp \{\alpha\}\|_{\mathcal{G}_{\overline{Y},\overline{\{\alpha\}}}} Y$$

This is because in graph $\mathcal{G}_{\overline{Y},\overline{\{\alpha\}}}$, if there is a Y-active trail between α and a node $i \in \{1,\ldots,k\}$, the path cannot include any nodes in Y, since Y nodes are instantiated fork nodes. Therefore, any Y-active trail in $\mathcal{G}_{\overline{Y},\overline{\{\alpha\}}}$ between α and i which does not contain any other nodes of W firstly, cannot have an arrow pointing into α, since the arrows between α and its parents have been removed. If the trail has no collider connections, then it is of the form $\alpha \mapsto \ldots \mapsto i$, which is a contradiction since the nodes of W are of a lower topological order than α. If it contains a collider which is either instantiated, or one of its descendants is instantiated, then the links to the parents of the instantiated nodes have been removed, hence such a connection does not exist. Therefore, such a trail does not exist.

Using rule 3,

$$\mathbb{P}_{W\|Y} = \mathbb{P}_{W\|\alpha,Y}$$

and the result follows. □

Lemma 3.28. *Let $H \subseteq \mathbf{V}$ and let $H'_1,\ldots,H'_n$ denote the c-components of the sub-graph $\mathcal{G}_{H\cup(Pa(H)\cap\mathbf{U})}$. Let $H_i = H'_i \cap H$. Then*

1. $$\mathbb{P}_{H\|\mathbf{V}\setminus H} = \prod_{i=1}^{n} \mathbb{P}_{H_i\|\mathbf{V}\setminus H_i}$$

2. *Each $\mathbb{P}_{H_i\|\mathbf{V}\setminus H_i}$ is computable from $\mathbb{P}_{H\|\mathbf{V}\setminus H}$ in the following way. Let k be the number of variables in H and let $\alpha_1 < \ldots < \alpha_k$ be a topological order of the variables of H in $\mathcal{G}_{H\cup(Pa(H)\cap\mathbf{U})}$. Let $H^{(j)} = \{\alpha_1,\ldots,\alpha_j\}$ for $j = 1,\ldots,k$ and $H^{(0)} = \varnothing$ (the empty set). Then each $\mathbb{P}_{H_i\|\mathbf{V}\setminus H_i} : i = 1,\ldots,n$ is given by:*

   $$\mathbb{P}_{H_i\|\mathbf{V}\setminus H_i} = \prod_{j=:\alpha_j\in H_i} \frac{\mathbb{P}_{H^{(j)}\|\mathbf{V}\setminus H^{(j)}}}{\mathbb{P}_{H^{(j-1)}\|\mathbf{V}\setminus H^{(j-1)}}}.$$

   Each $\mathbb{P}_{H^{(j)}\|\mathbf{V}\setminus H^{(j)}} : j = 0,1,\ldots,k$ is given by the marginalisation:

   $$\mathbb{P}_{H^{(j)}\|\mathbf{V}\setminus H^{(j)}} = \left(\mathbb{P}_{H\|\mathbf{V}\setminus H}\right)^{\downarrow H\setminus H^{(j)}}. \tag{3.14}$$

Furthermore, the results of this lemma are a consequence of Rules 2 and 3 of Theorem 3.22.

Proof The first statement is proved first, then Equation (3.14) is established and finally the second statement is proved.

The proof is by induction. When H includes exactly one node from $\mathbf{V}$, the result is clearly true, from the definition.

Suppose that the two statements are true for $H : |H| \leq k$ for an integer k and consider an arbitrary set $E \subset \mathbf{V}$ of size $|E| = k+1$. Let $H = \{\alpha_1, \ldots, \alpha_k\}$ and $E = H \cup \{\alpha_{k+1}\}$, where the indices correspond to the topological order. Let $H'_1, \ldots, H'_n$ be the c-components of $H \cup (\mathrm{Pa}(H) \cap \mathbf{U})$ in $\mathcal{G}_{H \cup (\mathrm{Pa}(H) \cap \mathbf{U})}$ and let $H_i = H'_i \cap H$ for $1 \leq i \leq n$. Let $Y = \mathbf{V} \backslash E$.

Now consider the c-components of $E \cup (\mathrm{Pa}(E) \cap \mathbf{U})$ in $\mathcal{G}_{E \cup (\mathrm{Pa}(E) \cap \mathbf{U})}$.

If $\mathrm{Pa}(\alpha_{k+1}) \cap \mathrm{Pa}(H) \cap \mathbf{U} = \varnothing$, then the c-components are $E'_1, \ldots, E'_{n+1}$, where $E'_i = H'_i$ for $i = 1, \ldots, n$ and $E'_{n+1} = \{\alpha_{k+1}\} \cup (\mathrm{Pa}(\alpha_{k+1}) \cap \mathbf{U})$. It follows that $E_i := E'_i \cap E = H_i$ for $i = 1, \ldots, n$ and $E_{n+1} = \{\alpha_{k+1}\}$. In this case, let $m = n+1$.

If $\mathrm{Pa}(\alpha_{k+1}) \cap \mathrm{Pa}(H) \cap \mathbf{U} \neq \varnothing$, then α_{k+1} shares at least one parent in $\mathbf{U}$ with a node in H. Let $E'_1, \ldots, E'_m$ denote the c-components of $E \cup (\mathrm{Pa}(E) \cap \mathbf{U})$ in $\mathcal{G}_{E \cup (\mathrm{Pa}(E) \cap \mathbf{U})}$ and let $E_i = E'_i \cap E$. It follows that, relabelling if necessary, $E_i = H_i$ for $i = 1, \ldots, m-1$ and $E_m = \{\alpha_{k+1}\} \cup \bigcup_{i=m}^{n} H_i$.

By the inductive hypothesis,

$$\mathbb{P}_{H \| \mathbf{V} \backslash H} = \prod_{i=1}^{n} \mathbb{P}_{H_i \| \mathbf{V} \backslash H_i},$$

which may be rewritten as:

$$\mathbb{P}_{H \| (\mathbf{V} \backslash E) \cup \{\alpha_{k+1}\}} = \prod_{i=1}^{n} \mathbb{P}_{H_i \| (\mathbf{V} \backslash E) \cup \{\alpha_{k+1}\} \cup H_1^{i-1} \cup H_{i+1}^{n}}$$

where the notation H_i^j for $i < j$ means: $\cup_{k=i}^{j} H_k$. For the first statement, it is required to prove that:

$$\mathbb{P}_{H, \{\alpha_{k+1}\} \| \mathbf{V} \backslash (H \cup \{\alpha_{k+1}\})} = \mathbb{P}_{H_m^n, \{\alpha_{k+1}\} \| (\mathbf{V} \backslash E) \cup H_1^{m-1}} \prod_{j=1}^{m-1} \mathbb{P}_{H_j \| (\mathbf{V} \backslash H_j)}.$$

A straightforward factorisation gives:

$$\mathbb{P}_{H, \{\alpha_{k+1}\} \| (\mathbf{V} \backslash (H \cup \{\alpha_{k+1}\}))} = \mathbb{P}_{\{\alpha_{k+1}\} | H \| (\mathbf{V} \backslash (H \cup \{\alpha_{k+1}\}))} \mathbb{P}_{H \| (\mathbf{V} \backslash (H \cup \{\alpha_{k+1}\}))}. \tag{3.15}$$

The notation $Y = V \backslash E$ will be used. The D-separation statement: $H \perp\!\!\!\perp \{\alpha_{k+1}\} \|_{\mathcal{G}_{\overline{Y}, \overline{\alpha_{k+1}}}} Y$ holds. This is because any Y-active trail in $\mathcal{G}_{\overline{Y}, \overline{\alpha}}$ between α_{k+1} and a node in H does not include any node in Y since nodes in Y are instantiated fork nodes. Since the arrows from parents of α_{k+1} have been removed and the nodes of H are of a lower topological order, any active trail contains an instantiated collider connection. But the instantiated nodes are in Y, hence links to their parents have been removed, hence such a trail does not exist.

Using Rule 3, it follows that:

$$\mathbb{P}_{H \| Y} = \mathbb{P}_{H \| \alpha_{k+1}, Y},$$

from which (3.15) gives:

$$\mathbb{P}_{H,\alpha_{k+1}\|Y} = \mathbb{P}_{\alpha_{k+1}|H\|Y}\mathbb{P}_{H\|\alpha_{k+1},Y} = \mathbb{P}_{\alpha_{k+1}|H\|Y}\prod_{j=1}^{m-1}\mathbb{P}_{H_j\|H_1^{j-1},H_{j+1}^n,Y,\alpha_{k+1}} \times \prod_{j=m}^{n}\mathbb{P}_{H_j\|H_1^{j-1},H_{j+1}^n,Y,\alpha_{k+1}}.$$

By the inductive hypothesis,

$$\prod_{j=1}^{m-1}\mathbb{P}_{H_j\|H_1^{j-1},H_{j+1}^n,Y,\alpha_{k+1}} = \mathbb{P}_{H_1^{m-1}\|H_m^n,Y,\alpha_{k+1}}.$$

As before,

$$H_1^{m-1} \perp \alpha_{k+1}\|_{\mathcal{G}_{\overline{Y},\underline{H_m^n},\underline{\alpha_{k+1}}}} Y \cup H_m^n.$$

This is because if there is a $Y \cup H_m^n$-active trail in $\mathcal{G}_{\overline{Y,H_m^n},\overline{\alpha}_{k+1}}$ between α_{k+1} and H_1^{m-1}, the trail does not contain any nodes of Y since all nodes of Y are instantiated forks. Since the links between α_{k+1} and its parents have been removed and since the nodes of H_1^{m-1} are of lower topological order than α_{k+1}, the node must contain at least one collider which is either instantiated or has an instantiated descendant. But all instantiated nodes have had links to their parents removed, hence no such path exists.

It now follows from Rule 3 that:

$$\mathbb{P}_{H_1^{m-1}\|H_m^n,Y,\alpha_{k+1}} = \mathbb{P}_{H_1^{m-1}\|H_m^n,Y}.$$

The next step is to show that

$$\alpha_{k+1} \perp H_m^n\|_{\mathcal{G}_{\overline{Y},\underline{H_m^n}}} Y \cup H_1^{m-1}.$$

If there is a $Y \cup H_1^{m-1}$-active trail in $\mathcal{G}_{\overline{Y},\underline{H_{m+1}^n}}$ between α_{k+1} and H_m^n, the path does not contain any node of Y, since nodes in Y are instantiated forks. Furthermore, there are no arrows into a node of H_m^n since links between H_m^n and its descendants have been removed. Consider the shortest $Y \cup H_1^{m-1}$-active trail between α_{k+1} and a node in H_m^n. Let the node be designated β. Except for the end points, the trail contains no nodes in $\mathbf{V}$, hence all are in $\mathbf{U}$. Since the nodes of H_m^n are of a lower topological order than α_{k+1}, the trail contains at least one fork node. It follows from the definition of c-component that α_{k+1} and β belong to the same c-component, which is a contradiction.

Using Rule 2, it therefore follows that:

$$\mathbb{P}_{\alpha_{k+1}|H\|Y} = \mathbb{P}_{\alpha_{k+1}|H_1^{m-1},H_m^n\|Y} = \mathbb{P}_{\alpha_{k+1}|H_1^{m-1}\|H_m^n,Y}.$$

Putting these together, it follows that:

$$\begin{aligned}\mathbb{P}_{H,\alpha_{k+1}\|Y} &= \mathbb{P}_{\alpha_{k+1}|H\|Y}\prod_{j=1}^{m-1}\mathbb{P}_{H_j\|H_1^{j-1},H_{j+1}^n,Y}\prod_{j=m}^{n}\mathbb{P}_{H_j\|H_1^{j-1},H_{j+1}^n,Y,\alpha_{k+1}}\\ &= \mathbb{P}_{\alpha_{k+1}|H_1^{m-1}\|H_m^n,Y}\mathbb{P}_{H_1^{m-1}\|H_m^n,Y}\prod_{j=m}^{n}\mathbb{P}_{H_j\|H_1^{j-1},H_{j+1}^n,Y,\alpha_{k+1}}\\ &= \mathbb{P}_{H_1^{m-1},\alpha_{k+1}|H_m^n,Y}\prod_{j=m}^{n}\mathbb{P}_{H_j\|H_1^{j-1},H_{j+1}^n,Y,\alpha_{k+1}}.\end{aligned}$$

This establishes the first statement, since $\mathbb{P}_{H,\alpha_{k+1}\|Y} = \mathbb{P}_{E\|\mathbf{V}\backslash E}$.

Now Equation (3.14) is established. From the first part,

$$\left(\mathbb{P}_{H\|\mathbf{V}\backslash H}\right)^{\downarrow H\backslash H^{(j)}} = \prod_{i=1}^{n}\left(\mathbb{P}_{H_i\|\mathbf{V}\backslash H_i}\right)^{\downarrow H_i\backslash H^{(j)}}.$$

Now let $\widetilde{H}_i^{(j)} = (H^{(j)}\cap H_i)\cup(\mathrm{Pa}(H_i)\cap\mathbf{U})$ and let $\widetilde{H}_i = H_i\cup(\mathrm{Pa}(H_i)\cap\mathbf{U})$. Then, by construction, $\widetilde{H}_i^{(j)}$ is an ancestral subset of $H_i\cup(\mathrm{Pa}(H_i)\cap\mathbf{U})$ in $\mathcal{G}_{H_i\cup(\mathrm{Pa}(H_i)\cap\mathbf{U})}$ and hence (by extending the set $\mathbf{V}$ for a moment to include all the nodes of $\widetilde{H}_j^{(j)}$), it follows by Lemma 3.27 that:

$$\mathbb{P}_{\widetilde{H}_i^{(j)}\|\mathbf{V}\backslash H^{(j)}} = \left(\mathbb{P}_{\widetilde{H}_i\|\mathbf{V}\backslash H_i}\right)^{\downarrow H_i\backslash H^{(j)}}.$$

Marginalising both sides over $\mathrm{Pa}(H_i)\cap\mathbf{U}$ gives:

$$\mathbb{P}_{H^{(j)}\cap H_i\|\mathbf{V}\backslash H^{(j)}} = \left(\mathbb{P}_{H_i\|\mathbf{V}\backslash H_i}\right)^{\downarrow H_i\backslash H^{(j)}}.$$

It follows that

$$\left(\mathbb{P}_{H\|\mathbf{V}\backslash H}\right)^{\downarrow H\backslash H^{(j)}} = \prod_{i=1}^{n}\mathbb{P}_{H^{(j)}\cap H_i\|\mathbf{V}\backslash H^{(j)}}.$$

Equation 3.14 now follows because do-conditioned on $\mathbf{V}\backslash H^{(j)}$, the node sets $H^{(j)}\cap H_i : i = 1,\ldots,n$ are D-separated from each other.

It follows from 3.14 (by considering $H^{(k)} = H$ and $H^{(k+1)} = E$) that

$$\mathbb{P}_{H\|\mathbf{V}\backslash H} = \left(\mathbb{P}_{H,\alpha_{k+1}\|\mathbf{V}\backslash(H\cup\{\alpha\})}\right)^{\downarrow\{\alpha_{k+1}\}} = \left(\mathbb{P}_{E\|\mathbf{V}\backslash E}\right)^{\downarrow\{\alpha_{k+1}\}}. \tag{3.16}$$

By the inductive hypothesis, H satisfies statement 2, so that:

$$\mathbb{P}_{H_i\|\mathbf{V}\backslash H_i} = \prod_{j|\alpha_j\in H_i}\frac{\mathbb{P}_{H^{(j)}\|\mathbf{V}\backslash H^{(j)}}}{\mathbb{P}_{H^{(j-1)}\|\mathbf{V}\backslash H^{(j-1)}}} \qquad i = 1,\ldots,n \tag{3.17}$$

By construction, $E^{(j)} = H^{(j)}$ for $j = 1,\ldots,k$ and $E^{(k+1)} = E = \{\alpha_1,\ldots,\alpha_{k+1}\}$. For $j = 1,\ldots,k$, it follows from (3.16) that the following are equal:

$$\mathbb{P}_{E^{(j)}\|\mathbf{V}\backslash E^{(j)}} = \left(\mathbb{P}_{E\|\mathbf{V}\backslash E}\right)^{H\cup\{\alpha\}\backslash H^{(j)}} = \left(\mathbb{P}_{H\|\mathbf{V}\backslash H}\right)^{H\backslash H^{(j)}} = \mathbb{P}_{H^{(j)}\|\mathbf{V}\backslash H^{(j)}}.$$

From (3.15) (end of proof of Part 1), it follows that:

$$\mathbb{P}_{E\|\mathbf{V}\backslash E} = \mathbb{P}_{E^{(k+1)}\|\mathbf{V}\backslash E^{(k+1)}} = \mathbb{P}_{H_m^n,\alpha_{k+1}\|\mathbf{V}\backslash(H_m^n\cup\{\alpha_{k+1}\})} \prod_{i=1}^{m-1} \mathbb{P}_{H_i\|\mathbf{V}\backslash H_i}.$$

From this, it follows from (3.17) that:

$$\mathbb{P}_{E_m\|\mathbf{V}\backslash E_m} = \mathbb{P}_{H_m^n,\alpha_{k+1}\|\mathbf{V}\backslash(H_m^n\cup\{\alpha_{k+1}\})} = \frac{\mathbb{P}_{E^{(k+1)}\|\mathbf{V}\backslash E^{(k+1)}}}{\prod_{i=1}^{m-1}\mathbb{P}_{H_i\|\mathbf{V}\backslash H_i}}.$$

Now,

$$\mathbb{P}_{E^{(k+1)}\|\mathbf{V}\backslash E^{(k+1)}} = \prod_{j=0}^{k} \frac{\mathbb{P}_{E^{(k+1)}\|\mathbf{V}\backslash E^{(k+1)}}}{\mathbb{P}_{E^{(k)}\|\mathbf{V}\backslash E^{(k)}}} = \prod_{i=1}^{m} \prod_{j:\alpha_j\in E_i} \frac{\mathbb{P}_{E^{(j)}\|\mathbf{V}\backslash E^{(j)}}}{\mathbb{P}_{E^{(j-1)}\|\mathbf{V}\backslash E^{(j-1)}}}$$

so that

$$\mathbb{P}_{E_m\|\mathbf{V}\backslash E_m} \prod_{j|\alpha_j\in E_m} \frac{\mathbb{P}_{E^{(j)}|\mathbf{V}\backslash E^{(j)}}}{\mathbb{P}_{E^{(j-1)}\|\mathbf{V}\backslash E^{(j-1)}}}$$

as required, and the lemma is proved. □

Based on the first statement of Lemma 3.28, establishing the *non-identifiability* of a statement may be reduced to establishing the non-identifiability of a statement within a c-component. The relevant result is the following:

Theorem 3.29. *Let $\mathcal{G}$ be a semi-Markovian model. If*

1. *$\mathcal{G}$ is itself a c-component,*
2. *$S \subset \mathbf{V}$ in $\mathcal{G}$ and $\mathcal{G}_{S\cup(Pa(S)\cap\mathbf{U})}$ has only one c-component,*
3. *all the nodes of $\mathbf{V}\backslash S$ are ancestors of S in $\mathcal{G}$,*

then $\mathbb{P}_{S\|\mathbf{V}\backslash S}$ is not identifiable in $\mathcal{G}$.

Proof Non-identifiability is established if it can be shown that there is a back-door path between a node in S and a node in $\mathbf{V}\backslash S$. Assume there is no back-door path, then there does not exist a node $v \in \mathbf{U}$ which is an ancestor of both a node in S and a node in $\mathbf{V}\backslash S$. It follows that $\mathcal{G}$ is not itself a c-component, which is a contradiction. □

Lemmas 3.27 and 3.28 provide the basis of a complete identification algorithm for computing do-conditioning statements $\mathbb{P}_{S\|\mathbb{V}\backslash S}$ for $S \subseteq \mathbf{V}$ in terms of see-conditioning statements, in the sense that when it does not give an output **fail**, it returns the correct answer. Theorem 3.29 establishes that the algorithm is complete, in the sense that it returns an output **fail** when and only when the statement is not identifiable.

Algorithm 1 Algorithm: Compute $\mathbb{P}_{S\|\mathbf{V}\setminus S}$

INPUT: $S \subseteq \mathbf{V}$

OUTPUT: Expression for $\mathbb{P}_{S\|\mathbf{V}\setminus S}$ or **fail**.

Let $V_1, \ldots, V_n$ be a partition of $\mathbf{V}$, where $V_j = V'_j \cap \mathbf{V}$ and $V'_1, \ldots, V'_n$ are the c-components of the sub-graph $\mathcal{G}_{\mathbf{V}\cup \mathrm{Pa}(\mathbf{V})}$. Let $S_1, \ldots, S_l$ be a partition of S where $S'_1, \ldots, S'_l$ are the c-components of the sub-graph $\mathcal{G}_{S\cup(\mathrm{Pa}(S)\cap\mathbf{U})}$ and $S_j = S'_j \cap \mathbf{V}$ for $j = 1, \ldots, l$. The subsets are labelled such that $S_j \subseteq V_j$ for $j = 1, \ldots, l$; this can clearly be done without loss of generality. Now

1. Compute each $\mathbb{P}_{V_j\|\mathbf{V}\setminus V_j}$ with Lemma 3.28;
2. Compute each $\mathbb{P}_{S_j\|\mathbf{V}\setminus S_j}$ using Algorithm 3.8 (**identify** (C, T) below), with $C = S_j$, $T = V_j$ and $Q = \mathbb{P}_{V_j\|\mathbf{V}\setminus V_j}$.
3. If in part 2. Algorithm 3.8 gives the output **fail** for any of the $S_j : j = 1, \ldots, l$, then $\mathbb{P}_{S\|\mathbf{V}\setminus S}$ is not identifiable and the output given is **fail**. Otherwise, $\mathbb{P}_{S\|\mathbf{V}\setminus S}$ is identifiable and is given by:

$$\mathbb{P}_{S\|\mathbf{V}\setminus S} = \prod_{j=1}^{l} \mathbb{P}_{S_j\|\mathbf{V}\setminus S_j}.$$

This follows from Lemma 3.28

Algorithm 2 Algorithm: Identify (C, T)

INPUT: C and T where $C \subseteq T \subseteq \mathbf{V}$, where $\mathcal{G}_{C \cup (\mathrm{Pa}(C) \cap \mathbf{U})}$ and $\mathcal{G}_{T \cup (\mathrm{Pa}(T) \cap \mathbf{U})}$ are both composed of a single c-component.

OUTPUT: Expression for $\mathbb{P}_{C \| \mathbf{V} \setminus C}$ or **fail**.

Let $A = (\mathrm{an}(C) \cup C)_{\mathcal{G}_{T \cup \left(\mathrm{Pa}_{(T) \cap \mathbf{U}}\right)}}$.

1. If $A = C$, then the output is $\mathbb{P}_{C \| \mathbf{V} \setminus C}$. By Lemma 3.27, this is given by:

$$\mathbb{P}_{C \| \mathbf{V} \setminus C} = \left(\mathbb{P}_{T \| \mathbf{V} \setminus T}\right)^{T \setminus C}.$$

2. Else: if $A = T$ (and $T \neq C$) then output **fail**.

3. Else: (if $C \subset A \subset T$).

   (a) By Lemma 3.27, compute:

$$\mathbb{P}_{A \| \mathbf{V} \setminus A} = \left(\mathbb{P}_{T \| \mathbf{V} \setminus T}\right)^{\downarrow T \setminus A}.$$

   (b) Assume that, in $\mathcal{G}_{A \cup (\mathrm{Pa}(A) \cap \mathbf{U})}$, C is contained in a c-component T_1'. Set $T_1 = T_1' \cap A$.

   (c) Compute $\mathbb{P}_{T_1 \| \mathbf{V} \setminus T_1}$ from $\mathbb{P}_{A \| \mathbb{V} \setminus A}$ by Lemma 3.28.

   (d) The output is the output of algorithm **identify** applied to (C, T_1).

Algorithm 3.8 is therefore recursive, until either it finds an expression for $\mathbb{P}_{C\|\mathbf{V}\setminus C}$ or else returns the output **fail**.

It now follows that if $\mathbb{P}_{\mathbf{V}\setminus T\|T}$ is identifiable, then so is $\mathbb{P}_{S\|T}$ for any $S \subseteq \mathbf{V}\setminus T$, by marginalisation:

$$\mathbb{P}_{S\|T} = \left(\mathbb{P}_{\mathbf{V}\setminus T|T}\right)^{\downarrow(\mathbf{V}\setminus T)\setminus S}.$$

Algorithm 3.8 is based on the following consideration: let $D = (S \cup \mathrm{an}(S))_{\mathcal{G}_{\mathbf{V}\setminus T}} \cap \mathbf{V}$. Then D is an ancestral set in $\mathbf{V}\setminus T$ and hence

$$\left(\mathbb{P}_{\mathbf{V}\setminus T\|T}\right)^{\downarrow\mathbf{V}\setminus(T\cup D)} = \mathbb{P}_{D\|\mathbf{V}\setminus D}.$$

It follows that:

$$\mathbb{P}_{S\|T} = \left(\left(\mathbb{P}_{\mathbf{V}\setminus T\|T}\right)^{\downarrow\mathbf{V}\setminus(T\cup D)}\right)^{\downarrow D\setminus S} = \left(\mathbb{P}_{D\|\mathbf{V}\setminus D}\right)^{\downarrow D\setminus S}.$$

Algorithm 3 Algorithm: Compute $\mathbb{P}_{S\|T}$

INPUT: Two disjoint observable variable sets $S \subseteq \mathbf{V}$ and $T \subset \mathbf{V}$, where T is interventional.

OUTPUT: The expression for $\mathbf{P}_{S\|T}$ or **fail**.

1. Let $D = (S \cup \mathrm{an}(S))_{\mathcal{G}_{\mathbf{V}\setminus T}} \cap \mathbf{V}$.
2. Use the algorithm: **Computing $\mathbf{P}_{S\|\mathbf{V}\setminus S}$** to compute $\mathbb{P}_{D\|\mathbf{V}\setminus D}$.
3. If the algorithm returns **fail**, then the output is **fail**.
4. Else, output
$$\mathbb{P}_{S\|T} = \left(\mathbb{P}_{D\|\mathbf{V}\setminus D}\right)^{\downarrow D\setminus S}.$$

The converse of Corollary 3.23 can now be stated:

Theorem 3.30. *The three inference rules of Theorem 3.22, together with standard probability manipulations, are* complete *for determining the identifiability of $\mathbb{P}_{H\|\mathbf{V}\setminus H}$ for all $H \subset \mathbf{V}$.*

Proof Lemmas 3.27 and 3.28 follow from the inference rules of Theorem 3.22 (as proved by Huang and Valtorta [64]). These form the basis of Algorithms 3.8, 3.8 and 3.8. By Theorem 3.29, it follows that the algorithms give the output **fail** if and only if the statement is not identifiable; by standard probability manipulations, they give the correct answer otherwise. □

3.8.1 Example: Front Door Criterion

Suppose that $\mathbb{P}_{U,X,Z,Y}$ factorises according to the DAG in Figure 3.13 and that U is hidden.

1. $\mathbb{P}_{Z\|X}$ $X \perp\!\!\!\perp Z\|_{\mathcal{G}_{\underline{X}}}\varnothing$ and hence Rule 2 gives:

$$\mathbb{P}_{Z\|X} = \mathbb{P}_{Z|X}. \tag{3.18}$$

2. $\mathbb{P}_{Y\|Z}$: $\mathcal{G}_{\underline{Z}}$ contains a back-door path from Z to Y, which is: $Z \leftarrow X \leftarrow U \rightarrow Y$. This path is blocked if X is instantiated.

$$\mathbb{P}_{Y\|Z} = \left(\mathbb{P}_{Y|X\|Z}\mathbb{P}_{X\|Z}\right)^{\downarrow X}.$$

Since $Z \perp\!\!\!\perp X\|_{\mathcal{G}_{\overline{Z}}}\varnothing$, it follows that Rule 3 may be applied:

$$\mathbb{P}_{X\|Z} = \mathbb{P}_X.$$

Since $Z \perp\!\!\!\perp Y\|_{\mathcal{G}_{\underline{Z}}}\varnothing$, Rule 2 gives:

$$\mathbb{P}_{Y|X\|Z} = \mathbb{P}_{Y|X,Z}.$$

It follows that

$$\mathbb{P}_{Y\|Z} = \left(\mathbb{P}_{Y|X,Z}\mathbb{P}_X\right)^{\downarrow X}. \tag{3.19}$$

This is a special case of the back-door formula (3.10).

3. $\mathbb{P}_{Y\|X}$. Writing

$$\mathbb{P}_{Y\|X} = \left(\mathbb{P}_{Y|Z\|X}\mathbb{P}_{Z\|X}\right)^{\downarrow Z}$$

it follows from (3.18) that $\mathbb{P}_{Z\|X} = \mathbb{P}_{Z|X}$. Rule 2 may be applied to give $\mathbb{P}_{Y|Z\|X} = \mathbb{P}_{Y\|X,Z}$, since $Y \perp\!\!\!\perp X\|_{\mathcal{G}_{\overline{X},\underline{Z}}}Z$. Rule 3 may be applied, since $Y \perp\!\!\!\perp X\|_{\mathcal{G}_{\overline{X},\overline{Z}}}Z$, to give:

$$\mathbb{P}_{Y|Z\|X} = \mathbb{P}_{Y\|Z},$$

which was computed in terms of see-conditioning in (3.19). Putting all this together gives:

$$\mathbb{P}_{Y\|X} = \left(\mathbb{P}_{Z|X}\left(\mathbb{P}_{Y|X,Z}\mathbb{P}_X\right)^{\downarrow X}\right)^{\downarrow Z}.$$

All the other causal effects (for example $\mathbb{P}_{Y,Z\|X}$ and $\mathbb{P}_{X,Z\|Y}$) can be derived from the rules of Theorem 3.22.

3.8.2 Causal Inference by Surrogate Experiments

Suppose that the causal effect of X on Y, $\mathbb{P}_{Y\|X}$ is of interest, but it is not identifiable and the variable X cannot be controlled via a randomised experiment. For example, if we are interested in assessing the effect of cholestorol X on heart disease Y, it may be possible to exercise control over the subject's diet Z rather than directly controlling the quantity of cholestorol in the subject's blood.

Formally, this problem amounts to transforming the problem $\mathbb{P}_{Y\|X}$ into expressions where do-conditioning is only on variables in Z. By Theorem 3.22, the following conditions are sufficient:

1. X intercepts all directed paths from Z to Y.
2. $\mathbb{P}_{Y\|X}$ is identifiable in $\mathcal{G}_{\overline{Z}}$.

If the first of these holds, it follows that $Y \perp\!\!\!\perp Z\|_{\mathcal{G}_{\overline{X},\overline{Z}}}X$ and hence $\mathbb{P}_{Y\|X} = \mathbb{P}_{Y\|X,Z}$. This represents the causal effects of X on Y in a model that factorises along $\mathcal{G}_{\overline{Z}}$ which is identifiable by the second condition. These conditions are satisfied by the two models in Figure 3.16. Translated to the cholestorol example, they require that there be no direct effect of diet on heart disease and no confounding effect between cholestorol level and heart disease unless there is an intermediate variable between the two which can be measured. For the first figure, the conditions are clear. For the second figure, $\mathbb{P}_{Y\|X}$ is identifiable in $\mathcal{G}_{\overline{Z}}$ because

$$\mathbb{P}_{Y\|X,Z} = \left(\mathbb{P}_{U_2}\mathbb{P}_{U_3}\mathbb{P}_{U_4}\mathbb{P}_{Y|W,U_3,U_4}\mathbb{P}_{W|X,U_2}\right)^{\downarrow W,U_2,U_3,U_4} = \left(\mathbb{P}_{Y|W}\mathbb{P}_{W|X}\right)^{\downarrow W}.$$

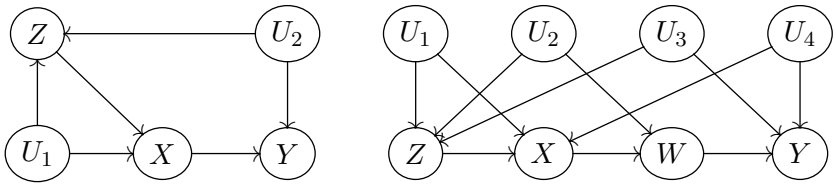

Figure 3.16: Do-condition on surrogate variable Z

3.9 Measurement Bias and Effect Restoration

Consider a situation where we would like to compute the *causal* effect of X on Y (namely $\mathbb{P}_{Y\|X}$), in the situation where there is a *sufficient confounder* U. *Confounder* means that, as a result of its presence, the effect $\mathbb{P}_{Y\|X}$ is not identifiable; *sufficient* in this context means that $\mathbb{P}_{Y\|X}$ could be computed if U were observable. There are situations where an observable W may give sufficient information about U to enable $\mathbb{P}_{Y\|X}$ to be identified from data.

The material for this section is taken from Kuroki and Pearl [77] (2014) and deals with two situations: firstly, the situation where $\mathbb{P}_{W|U}$ is known and W gives sufficient information about U to identify

$\mathbb{P}_{Y\|X}$. Secondly, $\mathbb{P}_{W|U}$ is unknown, but there are two observable variables (Z, W) which together give sufficient information to identify $\mathbb{P}_{Y\|X}$ without bias.

Example 3.31.

The Head Start Program is discussed in Madgison [87] (1977). This was a government programme within the United States of America aimed at giving assistance to children. Magidson's sample consists of 148 children who received the programme and 155 control children.

Let X be an indicator variable, indicating whether or not the child received the programme. Y is the outcome variable of the Metropolitan Readiness Test (a test which supposedly measures cognitive ability). U represents socio-economic status. This is unobserved and may be considered, following the discussion of Madgison, as a sufficient confounder. Figure 3.17 gives three possible situations; the first where W is measured as a proxy variable for U, the second and third where W and Z (family income) are measured as proxy variables of U.

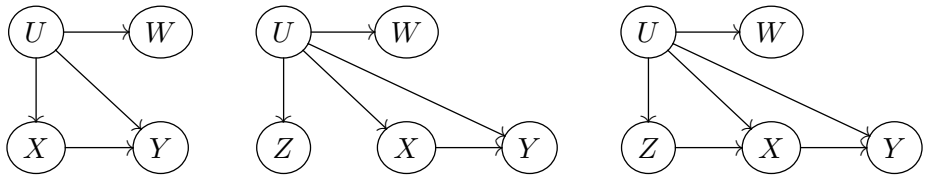

Figure 3.17: U hidden; causal models with proxy variables on U. For (a), $\mathbb{P}_{W|U}$ is required to identify $\mathbb{P}_{Y\|X}$. For (b) and (c), under further assumptions on Z and W, the effect $\mathbb{P}_{Y\|X}$ may be estimated from data.

In Figure 3.17, U satisfies the back-door criterion relative to (X, Y), but its proxy variables W and Z do not. For each of the models,

$$\mathbb{P}_{Y\|X} = \left(\mathbb{P}_{Y|X,U}\mathbb{P}_U\right)^{\downarrow U}.$$

If the conditional distribution $\mathbb{P}_{W|U}$ is known (and W is observable) then, under additional assumptions on $\mathbb{P}_{W|U}$, it is possible to construct an asymptotically unbiased estimator of $\mathbb{P}_{Y\|X}$.

3.9.1 The Matrix Adjustment Method

We now consider the model in Figure 3.17 (a), under the assumption that $\mathbb{P}_{W|U}$ is known, and show how to compute $\mathbb{P}_{Y\|X}$. The method is known as the *matrix adjustment method*.

Assume that U and W both have finite state space, with k elements. Without loss of generality, let U and W both have state space $\{1, \ldots, k\}$. The main idea for recovering $\mathbb{P}_{X,Y,U}$ from both $\mathbb{P}_{X,Y,W}$ and $\mathbb{P}_{W|U}$, the *matrix adjustment method*, found in Greenland and Lash [57] (2008) p. 360 and discussed in Pearl [113] (2010). The discussion here is taken from Kuroki and Pearl [77] (2014).

$$\mathbb{P}_{Y,W|X} = \left(\mathbb{P}_{Y,U|X}\mathbb{P}_{W|U}\right)^{\downarrow U}$$

Set

$$V_{U:Y|X}(.:y|x) = \begin{pmatrix} \mathbb{P}_{Y,U|X}(1,y|x) \\ \dots \\ \mathbb{P}_{Y,U|X}(k,y|x) \end{pmatrix}, \quad V_{W:Y|X}(.:y|x) = \begin{pmatrix} \mathbb{P}_{Y,W|X}(1,y|x) \\ \dots \\ \mathbb{P}_{Y,W|X}(k,y|x) \end{pmatrix}$$

and

$$M_{W|U} = \begin{pmatrix} \mathbb{P}_{W|U}(1|1) & \dots & \mathbb{P}_{W|U}(1|k) \\ \vdots & \ddots & \vdots \\ \mathbb{P}_{W|U}(k|1) & \dots & \mathbb{P}_{W|U}(k|k) \end{pmatrix}.$$

If $M_{W|U}$ is invertible, then:

$$V_{U:Y|X}(.:y|x) = M_{W|U}^{-1} V_{W:Y|X}(.:y|x)$$

Similarly, set $V_{U:|X} = \left(V_{U:Y|X}\right)^{\downarrow Y}$ so that $V_{U:|X}(u|x) = \mathbb{P}_{U|X}(u|x)$ and similarly $V_{W:|X} = \left(W_{W:Y|X}\right)^{\downarrow Y}$, then

$$V_{U:|X} = M_{W|U}^{-1} V_{W:|X}.$$

It follows that if $\mathbb{P}_{W|U}$ is known, then the causal effect of manipulating X, i.e. $\mathbb{P}_{Y\|X}$, is estimable and is given by:

$$\mathbb{P}_{Y\|X} = \left(\frac{\mathbb{P}_{Y,U\|X}\mathbb{P}_U}{\mathbb{P}_{U|X}}\right)^{\downarrow U} = \left(\frac{\left(M_{W,U}^{-1t}\mathbb{P}_{Y,W|X}\left(M_{W,U}^{-1t}\mathbb{P}_W\right)^{\downarrow W}\right)^{\downarrow W}}{\left(M_{W,U}^{-1t}\mathbb{P}_{W|X}\right)^{\downarrow W}}\right)^{\downarrow U}.$$

where $\mathbb{P}_W$ and $\mathbb{P}_{W|X}(.|x)$, for each $x \in \mathcal{X}$, are taken as column k-vectors. □

3.9.2 Effect Restoration Without External Studies

Now consider the more difficult problem of estimating causal effects *without* prior knowledge of $\mathbb{P}_{W|U}$. This is not possible for the first of the models of Figure 3.17, but may be possible, under additional assumptions for the second and third models in that figure.

For each given (x,y), let σ be a permutation of $1,\dots,k$ such that

$$\mathbb{P}_{Y|X,U}(y|x,\sigma(1)) \geq \dots \geq \mathbb{P}_{Y|X,U}(y|x,\sigma(k)).$$

For the models under consideration, $W \perp\!\!\!\perp \{X,Y,Z\}\|_{\mathcal{G}} U$ and $Y \perp\!\!\!\perp \{W,Z\}\|_{\mathcal{G}}\{U,X\}$.

$$\mathbb{P}_{Z,W|X} = \left(\mathbb{P}_{Z,W,U|X}\right)^{\downarrow U} = \left(\mathbb{P}_{W|Z,U,X}\mathbb{P}_{Z|U,X}\mathbb{P}_{U|X}\right)^{\downarrow U} = \left(\mathbb{P}_{W|U}\mathbb{P}_{Z|U}\mathbb{P}_{U|X}\right)^{\downarrow U}.$$

Similarly,

$$\mathbb{P}_{Y,W|X} = \left(\mathbb{P}_{W|U}\mathbb{P}_{Y|X,U}\mathbb{P}_{U|X}\right)^{\downarrow U}$$

$$\mathbb{P}_{Y,Z|X} = \left(\mathbb{P}_{Y|X,U}\mathbb{P}_{Z|X,U}\mathbb{P}_{X|U}\right)^{\downarrow U}$$

$$\mathbb{P}_{Y,Z,W|X} = \left(\mathbb{P}_{W|U}\mathbb{P}_{Z|X,U}\mathbb{P}_{Y|X,U}\mathbb{P}_{U|X}\right)^{\downarrow U}$$

Let

$$\begin{cases} \mathcal{P}_{Z,W} = \begin{pmatrix} 1 & \mathbb{P}_{W|X}(1|x) & \dots & \mathbb{P}_{W|X}(k-1|x) \\ \mathbb{P}_{Z|X}(1|x) & \mathbb{P}_{Z,W|X}(z_1,w_1|x) & \dots & \mathbb{P}_{Z,W|X}(z_1,w_{k-1}|x) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{P}_{Z|X}(k-1|x) & \mathbb{P}_{Z,W|X}(k-1,1|x) & \dots & \mathbb{P}_{Z,W|X}(z_{k-1},w_{k-1}|x) \end{pmatrix}. \\ \mathcal{Q}_{Z,W} = \begin{pmatrix} 1 & \mathbb{P}_{Y,W|X}(y,1|x) & \dots & \mathbb{P}_{Y,W|X}(y,k-1|x) \\ \mathbb{P}_{Y,Z|X}(y,1|x) & \mathbb{P}_{Y,Z,W|X}(y,z_1,w_1|x) & \dots & \mathbb{P}_{Y,Z,W|X}(y,z_1,w_{k-1}|x) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{P}_{Y,Z|X}(y,k-1|x) & \mathbb{P}_{Y,Z,W|X}(y,k-1,1|x) & \dots & \mathbb{P}_{Y,Z,W|X}(y,z_{k-1},w_{k-1}|x) \end{pmatrix}. \end{cases} \tag{3.20}$$

$$\mathcal{U}_{W,U} = \begin{pmatrix} 1 & \mathbb{P}_{W|U}(1|\sigma(1)) & \dots & \mathbb{P}_{W|U}(k-1|\sigma(1)) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \mathbb{P}_{W|U}(1|\sigma(k)) & \dots & \mathbb{P}_{W|U}(k-1|\sigma(k)) \end{pmatrix}$$

$$\mathcal{R}_{Z,U} = \begin{pmatrix} 1 & \mathbb{P}_{Z|X,U}(1|x,\sigma(1)) & \dots & \mathbb{P}_{Z|X,U}(k-1|x,\sigma(1)) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \mathbb{P}_{Z|X,U}(1|x,\sigma(k)) & \dots & \mathbb{P}_{Z|X,U}(k-1|x,\sigma(k)) \end{pmatrix}$$

$$\Delta_U = \operatorname{diag}(\mathbb{P}_{Y|X,U}(y|x,\sigma(1)),\dots,\mathbb{P}_{Y|X,U}(y|x,\sigma(k))) \tag{3.21}$$

$$\mathcal{M}_U = \operatorname{diag}(\mathbb{P}_{U|X}(\sigma(1)|x),\dots,\mathbb{P}_{U|X}(\sigma(k)|x)).$$

Note that $\mathcal{P}$ and $\mathcal{Q}$ can be written:

$$\mathcal{P}_{Z,W} = \mathcal{R}_{Z,U}^t\mathcal{M}_U\mathcal{U}_{W,U}, \qquad \mathcal{Q}_{Z,W} = \mathcal{R}_{Z,U}^t\mathcal{M}_U\Delta_U\mathcal{U}_{W,U}.$$

Provided both $\mathcal{P}_{Z,W}$ is invertible, it follows that:

$$\mathcal{P}_{Z,W}^{-1}\mathcal{Q}_{Z,W} = \mathcal{U}_{W,U}\Delta_U\mathcal{U}_{W,U}.$$

It follows that the recovery problem of $\mathbb{P}_{W|U}$ from $\mathcal{U}_{W,U}$ rests on the eigenvalue decomposition of $\mathcal{P}_{Z,W}^{-1}\mathcal{Q}_{Z,W}$. Once $\mathbb{P}_{W|U}$ is known, the matrix adjustment method may be used to evaluate the causal effect on Y of manipulating X. This requires additionally that $\mathcal{Q}_{Z,W}$ be invertible and the probabilities $\mathbb{P}_{Y|X,U}(y|x,1),\dots,\mathbb{P}_{Y|X,U}(y|x,k)$ take distinct values for given (x,y).

The result is presented in the following theorem:

Theorem 3.32. *Suppose U is a sufficient confounder relative to (X,Y) and suppose that*

1. *Two proxy variables of U that are conditionally independent of each other given U can be observed; call them W and Z. Both $W \perp \{X, Y, Z\}|U$ and $Y \perp \{W, Z\}|\{U, X\}$ hold.*

2. *W, Z and the counfounder U are discrete variables, with a given finite nuber of categories, k.*

3. *Both $\mathcal{P}_{Z,W}$ and $\mathcal{Q}_{Z,W}$ defined by (3.20) are invertible.*

4. *The probabilities $\mathbb{P}_{Y|X,U}(y|x,1), \ldots, \mathbb{P}_{Y|X,U}(y|x,k)$ take distinct values for given x and y,*

then the causal effect $\mathbb{P}_{Y\|X}$ of X on Y is identifiable.

Proof The proof is based on the following two-step procedure that recovers $\mathbb{P}_{X,Y,U}$ from $\mathbb{P}_{X,Y,Z,W}$.

- Stage 1: Solve an eigenvalue problem of $\mathcal{P}_{Z,W}^{-1}\mathcal{Q}_{Z,W}$ to recover $\mathbb{P}_{W|U}$ from $\mathcal{U}_{W,U}$
- Recover $\mathbb{P}_{X,Y,U}$ using the matrix adjustment method.

Step 1 First solve $|\mathcal{P}_{Z,W}^{-1}\mathcal{Q}_{Z,W} - \lambda I_k| = 0$, where I_k denotes the $k \times k$ identity matrix. The solutions $\lambda_1, \ldots, \lambda_k$ are the eigenvalues of $\mathcal{P}_{Z,W}^{-1}\mathcal{Q}_{Z,W}$. They satisfy:

$$|\mathcal{P}_{Z,W}^{-1}\mathcal{Q}_{Z,W} - \lambda I_k| = |\Delta_U - I_k| = 0$$

where Δ_U is defined by (3.21). It follows that $\lambda_i = \mathbb{P}_{Y|X,U}(y|x, \sigma(i))$ and hence the elements of Δ_U are estimable.
To obtain the eigenvector η_i corresponding to λ_i, let $H = (\eta_1, \ldots, \eta_k)$, then H satisfies:

$$\mathcal{P}_{Z,W}^{-1}\mathcal{Q}_{Z,W}H = H\Delta_U.$$

By the condition that λ_i take different values, it follows that $\eta_1, \ldots, \eta_k$ are uniquely determined.

Let $A = \mathcal{U}_{W,U}^{-1}E$ where $E = \text{diag}(\alpha_1, \ldots, \alpha_k)$ for non-zero values of $(\alpha_1, \ldots, \alpha_k)$, then:

$$\mathcal{P}_{Z,W}^{-1}\mathcal{Q}_{Z,W}A = \mathcal{U}_{W,U}^{-1}\Delta_U E = \mathcal{U}_{W,U}^{-1}E\Delta_U = A\Delta_U.$$

It follows that A is also a matrix of eigenvectors of $\mathcal{P}_{X,Z}^{-1}\mathcal{Q}_{X,Z}$ and hence, with a particular choice of $\alpha_1, \ldots, \alpha_k$, $A = \mathcal{U}_{W,U}^{-1}E = H$.

It follows that for the inverse H^{-1} of the estimable matrix H satisfies (using $\mathcal{U}_{W,U}^{-1}E = H$):

$$\mathcal{U}_{W,U} = \begin{pmatrix} 1 & \mathbb{P}_{W|U}(1|\sigma(1)) & \ldots & \mathbb{P}_{W|U}(k-1|\sigma(1)) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \mathbb{P}_{W|U}(1|\sigma(k)) & \ldots & \mathbb{P}_{W|U}(1|\sigma(k)) \end{pmatrix} = EH^{-1} = \begin{pmatrix} \alpha_1 H_{11}^{-1} & \ldots & \alpha_1 H_{1k}^{-1} \\ \vdots & \ddots & \vdots \\ \alpha_k H_{k1}^{-1} & \ldots & \alpha_k H_{kk}^{-1} \end{pmatrix}$$

It follows, equating the first column, that $\alpha_j = \frac{1}{H_{j1}^{-1}} : j = 1, \ldots, k$. This shows that $\mathcal{U}_{W,U}$ is identifiable from EH^{-1} since H^{-1} is estimable. It follows that every element $\mathbb{P}_{W|U}$ of $\mathcal{U}_{W,U}$ can be obtained.

Step 2 Since

$$\mathbb{P}_{X,Y,W} = \left(\mathbb{P}_{X,Y,U}\mathbb{P}_{W|U}\right)^{\downarrow U}$$

it now follows that

$$\mathbb{P}_{Y\|X} = \left(\mathbb{P}_{Y|X,U}\mathbb{P}_U\right)^{\downarrow U} = \left(\frac{\mathbb{P}_{X,Y,U}}{\mathbb{P}_{X,U}}\mathbb{P}_U\right)^{\downarrow U}$$

is identifiable. □

3.10 Identification of Counterfactuals

A *counterfactual* is simply a hypothetical statement that cannot be tested directly. For example, following the network of Arthur Cayley (chapter 4), it is not possible to act to cause a storm-force gale, nor to dissociate experimentally by an intervention the effects of wind and rain from their common causes; such an intervention is not possible.

Another example of a counterfactual is the following: a given dose of treatment was administered to a patient. The dose was decided upon as a result of standard diagnostic procedures. It failed to cure the disease and the patient died. Would a stronger dose have cured the patient? Or would a stronger dose have still failed to control the disease? Or would the patient have died of side effects from the treatment?

Such questions, of course, have importance. One would like to know whether or not the wrong treatment was administered for future reference; what to do in future cases that exhibit similar symptoms and whether there are possibilities of adjusting the treatment, once administered, if it does not seem to be having the desired effect.

Let X denote 'treatment', taking values in $\mathcal{X}$ and let $x \in \mathcal{X}$ denote a generic element. Let Y denote the 'effect'. This could be binary (0 or 1) if the only question of interest is whether or not the patient was cured, or it could (for example) be a real valued random variable, denoting the quantity of an enzyme after treatment.

To formulate the counterfactual query, Y should no longer be considered as a single random variable, but rather as a *stochastic process* Y' indexed by $\mathcal{X}$. A stochastic process, in its greatest generality, is defined as follows:

Definition 3.33 (Stochastic Process)**.** *Let $\mathcal{X}$ be a set and $(E, \mathcal{E})$ a measurable space. A* stochastic process *Y indexed by $\mathcal{X}$ with state space $(E, \mathcal{E})$, indexed by a set $\mathcal{X}$, is a family of measurable mappings $\{Y(x) : x \in \mathcal{X}\}$ from a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ into $(E, \mathcal{E})$. The space $(E, \mathcal{E})$ is called the state space.*

There is no requirement from the definition of 'process' that the state space $\mathcal{X}$ should represent 'time'.

In the counterfactual set-up, Y' has state space $\mathcal{X}_Y$, the same state space as Y. Attention is restricted to the situation where $E = \mathcal{X}_Y$ is either finite, in which case $\mathcal{E}$ is simply the set of all possible subsets, or else $E = \mathbb{R}$, in which case $\mathcal{E} = \mathcal{B}(\mathbb{R})$, the Borel σ-algebra over $\mathbb{R}$, the smallest collection of subsets necessary to define integration (and hence a probability measure).

Suppose that the state space of Y is $\mathcal{Y} = \{0, 1\}$, where 0 represents 'death' and 1 represent 'cure'. Suppose that x_1 was the dose administered and the outcome was 'death'. Consider the counterfactual query: 'would the patient have survived if we had given a treatment dose x_2?' In terms of the counterfactual process, the quantity to be computed is therefore:

$$\mathbb{P}(Y'(x_2) = 1 | Y'(x_1) = 0).$$

In some limited cases, with serious additional modelling assumptions, this quantity can be computed from the one-dimensional marginal distributions. For example, suppose we assume that, for $x_1 < x_2$, $\{Y'(x_1) = 1\} \subseteq \{Y'(x_2) = 1\}$. This means that we assume that if the patient survives a low dose of the treatment, he will also survive a higher dose. The treatment does not have side effects which kill the patient; increasing the treatment dose increases the chance of success.

Under this assumption,

$$\mathbb{P}(Y'(x_2) = 0 | Y'(x_1) = 0) = \begin{cases} \frac{\mathbb{P}(Y'(x_2)=0)}{\mathbb{P}(Y'(x_1)=0)} & x_2 > x_1 \\ 1 & x_2 \leq x_1. \end{cases}$$

Several types of counterfactual query can be considered; if X is a cause and Y an effect within a larger network, x_1 could either be *observed*, or *forced* by intervention; the query is then 'we observed effect $Y = y$ when we observed $X = x_1$. What would have happened if we had forced $X \leftarrow x_2$ by intervention?'

To construct the appropriate counterfactual probability distribution, we add the counterfactual process Y', indexed by $\mathcal{X}$, which does not have X as a parent. At the same time, the original variable Y, with parent X remains in the graph and the counterfactual query is to compute

$$\mathbb{P}(Y'(x_2) = y | Y = y, X = x_1).$$

3.10.1 Counterfactual Graphs

A *countefactual Bayesian Network* is a *Bayesian network* that is obtained by extending the original network in a way that can be used to answer the counterfactual query.

Firstly, it is important that the same 'random' component is considered. In other words, if we observe an instantiation of Y when we do $X \leftarrow x_1$ and we are asking about the probability distribution, conditional on this information, of the distribution of Y if we had done $X \leftarrow x_2$ instead, we need a network which contains both $Y(x_1)$ and $Y(x_2)$ and also assumes the *same* random influence.

Therefore, we start with a formulation of the DAG which involves functional equations. That is, if X_j has parents $\mathrm{Pa}(j)$, then $X_j = f_j(\mathrm{Pa}(j), U_j)$ where $U_1, \ldots, U_d$ are i.i.d. $U(0, 1)$ random variables, the functions $f_j : j = 1, \ldots, d$ are deterministic functions and the parents of X_j in the DAG are $\mathrm{Pa}(j) \cup \{U_j\}$.

We extend the DAG to answer the counterfactual query in the following way: If the query involves (a) do-conditioning on a subset $A \subseteq \mathbf{V}$ and (b) asking a counterfactual question about the causal effect on a set of nodes Y, then for each node β on the causal path from A to Y

1. add a *c-process node* β'. These are the *counterfactual process* nodes, corresponding to the counterfactual process, enumerating the value taken by the variable for each $x \in \mathcal{X}_A$.

   A c-process node β' has all the parents of α except the nodes in A; there are no links from nodes in A to c-process nodes.

2. For each $\alpha \in A$ and each β on the causal path between A and Y (including all the nodes in Y), add in an arrow $\alpha \to \beta$. If β and γ, where $\beta, \gamma \notin A$ are on the causal path between A and Y and there is an arrow $\beta \to \gamma$, add in a *process arrow* $\beta' \Rightarrow \gamma'$.

3. Add in a *process to variable* arrow $\beta' \twoheadrightarrow \beta$ for each process node and its corresponding variable node. This is shorthand for an arrow $\beta'(x) \to \beta$ for each $x \in \mathcal{X}_A$.

4. Add in *variable to process* nodes $\gamma \rightarrowtail \beta'$ for each $\gamma \in \mathrm{Pa}(\beta) \backslash A$. This is shorthand for $\gamma \to \beta'(x)$ for each $x \in \mathcal{X}_A$.

5. If there is an arrow $\beta' \Rightarrow \gamma'$, then remove the arrow $\beta \to \gamma$ (if it exists).

Within a Process Node A *process node* α' is shorthand for $\{\alpha'(x) : x \in \mathcal{X}_A\}$.

Between Process Nodes If $\alpha = f(\beta_1, \ldots, \beta_k, v)$ in the original DAG, where $\mathrm{Pa}(\alpha) = \{\beta_1, \ldots, \beta_k\}$ and v is the random effect, then

$$\alpha'(j) = f(\beta_1'(j), \ldots, \beta_k'(j), v)$$

for each j.

Example 3.34.

Suppose we are interested in how likely a patient would be to have a certain symptom Y (1 = yes, 0 = no), given a dose x of a drug X assuming we know that the patient took dose x' of the drug and exhibited the symptom. Suppose there is a mediating variable W, for example: blood pressure, and that it is the blood pressure which is the cause of the symptom. Furthermore, we also know that the patient took dose d of a drug D and we have measured a symptom $Z = z$. We know $\mathbb{P}_{Z|D}$, the conditional probability distribution for symptom Z given drug D.

The blood pressure / symptom (W, Y) may therefore considered as a *counterfactual process* indexed by the dose of drug. The random variable $Y(x)$ indicates whether or not the patient exhibits the symptom when dose x is administered. In this language, problem is therefore to compute $\mathbb{P}_{Y(x)|Y(x'),Z,D}$. Figure 3.18 (a) shows the original DAG for the Bayesian Network; (b) shows the network in terms of functional relations.

3.10.2 Joint Counterfactual Probabilities and Intervention

Notes The 'do - calculus' is due to Judea Pearl in [109] and [107]. It enables conclusions to be drawn about the effects of *active* interventions, based on passive observations. The other main sources

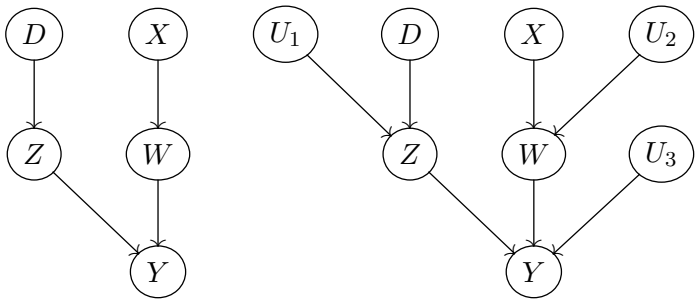

Figure 3.18: (a) A DAG, (b) the graph expressed as a functional relations graph

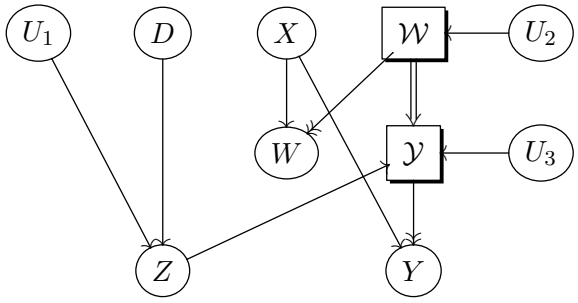

Figure 3.19: (c) the graph extended to answer a counterfactual query

for the presentation here are Edwards [40] (2000) chapter 9 and Lauritzen (2001) [82]. The idea of deletion of connections (in terms of wiping out equations in a multivariate model) is found in Strotz and Wold (1960) [128]. The intervention formula is due to J. Pearl, but is also given independently in the first edition of Spirtes, Glymour and Scheines (2002) [127]. The designation *semi-Markovian model* follows [134]. The paper [64] (2006) summarises the recent developments in the problem of identifiability and presents an algorithmic solution. The results by Y. Huang, M. Valtorta in [64] show that the do-calculus rules of Pearl [107] and [108] (1995) are complete in the sense that if a causal effect is identifiable, then the causal effects can be computed in terms of observational quantities. The article [43] by Freedman and Humphreys makes the obvious point that causality cannot be learned from data and is a necessary response to errors that inexplicably crept into the literature.

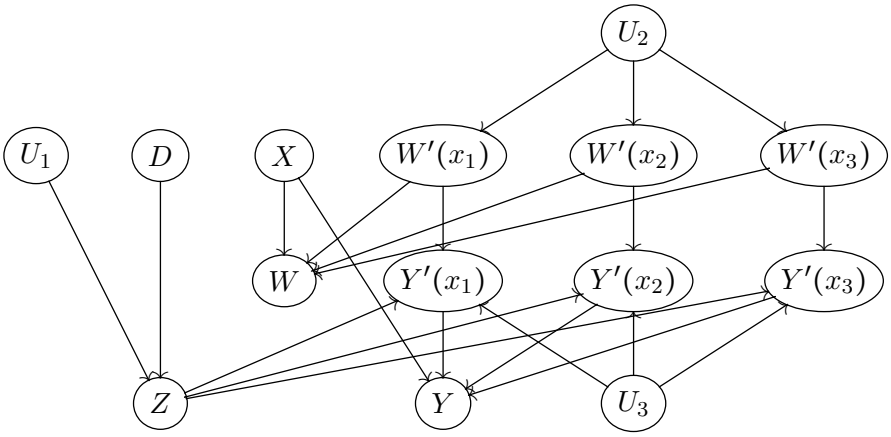

Figure 3.20: Graph of Figure 3.19 (b) with the process nodes written out

3.11 Exercises

1. The two parts of this exercise are very similar and straightforward, illustrating how d-separation in the mutilated graph corresponds to conditional independence in the remaining variables after do-conditioning.

   (a) Let $\mathcal{G}$ be a Directed Acyclic Graph, and suppose that a probability distribution $\mathbb{P}$ may be factorised along $\mathcal{G}$. Let $\mathcal{G}^{-X}$ denote the graph obtained by deleting from $\mathcal{G}$ all arrows pointing towards X (that is, all links between X and its parents are deleted). Prove that if Y and Z are d-separated in $\mathcal{G}^{-X}$ by X, then

   $$\mathbb{P}_{Y|Z\|X}(.|.\|x) = \mathbb{P}_{Y\|X}(.\|x),$$

   where the conditioning is taken from right to left.

   (b) Let A, B, C, W be disjoint sets of nodes in a Bayesian Network. Let $\mathcal{G}$ denote the Directed Acyclic Graph describing the causal network, and let $\mathcal{G}^{-C}$ denote the graph with all edges between C and parents of C removed.

   Prove that if A and B are d-separated by (C, W) on the graph $\mathcal{G}^{-C}$, then

   $$\mathbb{P}_{A|W,B\|C}(x_A|x_W, x_B, \|x_C) = \mathbb{P}_{A|W\|C}(x_A|x_W\|x_C),$$

   where the conditioning is performed from right to left.

2. Suppose the causal relations between the variables $(X_1, X_2, X_3, X_4, X_5, X_6, Y, Z)$ may be expressed by the DAG given in Figure 3.21. Which of the following sets satisfy the back door criterion with respect to the ordered pair of nodes (Y, Z)? $C_1 = \{X_1, X_2\}$, $C_2 = \{X_4, X_5\}$, $C_3 = \{X_4\}$.

   State all sets of nodes that satisfy the back door criterion with respect to the ordered set of nodes (Z, Y).

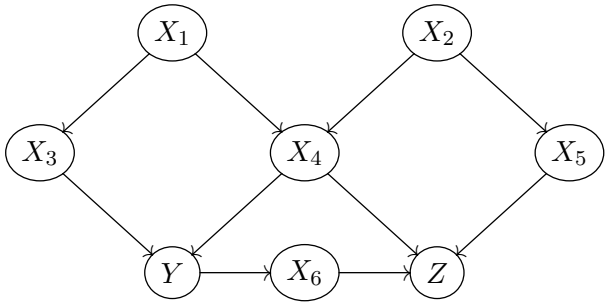

Figure 3.21: Causal Relations between Variables

3. Let a set of variables C satisfy the back door criterion relative to (X, Y). Prove that

$$\mathbb{P}_{Y\|X}(y\|x) = \sum_c \mathbb{P}_{Y|C,X}(y|c,x)\mathbb{P}_C(c).$$

4. Let C be a set of variables in a Bayesian Network and let X be a variable such that C contains no descendants of X. Prove, from the definition, that

$$\mathbb{P}_{C\|X}(c\|x) = \mathbb{P}_C(c).$$

5. Let $V = \{X_1, \ldots, X_d\}$ denote a set of variables. Let $V = Z \cup U$, where the variables in Z are observable and the variables in U are unobservable. Assume that the probability distribution over the variables in V may be factorised along a Directed Acyclic Graph $\mathcal{G} = (V, D)$, where no variable in U is a descendant of any variable in Z. That is, the model is semi-Markovian. Consider a single variable, say $X_j \in Z$. Assume that there is no trail between X_j and X_k for $X_k \in Z$ with only fork and chain connections which contains a variable $X_i \in U$. Show that

$$\mathbb{P}_{Z\setminus\{X_j\}\|X_j}(\underline{x}_{\widetilde{Z}\setminus\{j\}}\|x_j) = \mathbb{P}_{Z\setminus(\{X_j\}\cup \mathrm{Pa}_j)|X_j,\mathrm{Pa}_j\cap Z}\left(\underline{x}_{\widetilde{Z}\setminus(\{j\}\cup\widetilde{\mathrm{Pa}}_j)}|x_j, \underline{x}_{\widetilde{\mathrm{Pa}}_j\cap\widetilde{Z}}\right)\mathbb{P}_{\mathrm{Pa}_j\cap\widetilde{Z}}(\underline{x}_{\widetilde{\mathrm{Pa}}_j\cap\widetilde{Z}}).$$

3.12 Answers

1. (a) Let $\widetilde{\mathbb{P}}_{V\backslash\{X\}} = \mathbb{P}_{V\backslash\{X\}\|X}(.\|x)$. Then $\widetilde{\mathbb{P}}$ factorises along $\mathcal{G}^{-X}_{V\backslash\{X\}}$, the subgraph of $\mathcal{G}^{-X}$ over the variables $V\backslash\{X\}$. The probability tables are, for $Y \neq X$ and parent sets $\widetilde{\mathrm{Pa}}_Y = \mathrm{Pa}_Y\backslash\{X\}$ ($\widetilde{\mathrm{Pa}}_Y$ is the original parent set of Pa_Y with X removed) $\widetilde{\mathbb{P}}_{Y|\widetilde{\mathrm{Pa}}_Y} = \mathbb{P}_{Y|\mathrm{Pa}_Y}$ with the instantiation $X \leftarrow x$ for every appearance of X in Pa_Y. If $Y \perp\!\!\!\perp Z\|_{\mathcal{G}^{-X}} X$, then all trails between Y and Z in $\mathcal{G}^{-X}$ have either X as a fork or chain node, or else have a collider node that is not X and which does not have X as a descendant. It follows that all trails between Y and Z in $\mathcal{G}^{-X}_{V\backslash\{X\}}$ have at least one collider node and hence that $Y \perp\!\!\!\perp Z\|_{\mathcal{G}^{-X}_{V\backslash\{X\}}} \varnothing$ (d separated when none of the other variables are instantiated. It follows that, under probability distribution $\widetilde{\mathbb{P}}$, $Y \perp Z$, so that

$$\mathbb{P}_{Y|Z\|X}(.|.\|x) = \widetilde{\mathbb{P}}_{Y|Z} = \widetilde{\mathbb{P}}_Y = \mathbb{P}_{Y\|X}(.\|x).$$

   (b) Let V denote the variable set and let $\widetilde{\mathbb{P}}_{V\backslash C} = \mathbb{P}_{V\backslash C\|C}(.\|x_C)$. Then $\widetilde{\mathbb{P}}$ factorises along the graph $\mathcal{G}^{-C}_{V\backslash C}$ (the subgraph of $\mathcal{G}^{-C}$ with the nodes C removed) and, for $X \notin C$, conditional probability potentials $\widetilde{\mathbb{P}}_{X|\widetilde{\mathrm{Pa}}_X} = \mathbb{P}_{X|\mathrm{Pa}_X}$ where $\widetilde{\mathrm{Pa}}_X = \mathrm{Pa}_X\backslash C$, Pa_X denotes the original neighbour set, and the variables in $\mathrm{Pa}_X \cap C$ instantiated with the appropriate values.

   If $A \perp\!\!\!\perp B\|_{\mathcal{G}^{-C}} C \cup W$ then any trail from A to B either has a fork or chain node in $C \cup W$ or a collider node that is not in $C \cup W$ with no descendants in $C \cup W$. It follows that, on the graph $\mathcal{G}^{-C}_{V\backslash C}$, any trail from A to B either has a fork or chain node in W or a collider node that is not in W with no descendants in W; edges are deleted, but not added, by taking the subgraph restricted to the variables of $V\backslash C$ and hence no new trails are added by removing the nodes in C. It follows that $A \perp\!\!\!\perp B\|_{\mathcal{G}^{-C}_{V\backslash C}} W$ and hence that

$$\mathbb{P}_{A|W,B\|C}(x_A|x_W, x_B\|x_C) = \widetilde{\mathbb{P}}_{A|W,B}(x_A|x_W, x_B) = \widetilde{\mathbb{P}}_{A|W}(x_A|x_W) = \mathbb{P}_{A|W\|C}(x_A|x_W\|x_C)$$

   which is the result.

2. $C_1 = \{X_1, X_2\}$ does not satisfy the back door criterion; $Y - X_4 - Z$ is a trail between Y and Z with an edge pointing to Y which is not blocked by C_1.

   $C_2 = \{X_4, X_5\}$ satisfies the back door criterion; trail $Y - X_6 - Z$ does not have an edge pointing towards Y. The other trails pass through X_4. For the trails $Y - X_4 - Z$ and $Y - X_3 - X_1 - X_4 - Z$, X_4 is an instantiated fork or chain respectively, hence C_2 blocks the trail. For $Y - X_1 - X_4 - X_2 - X_5 - Z$, X_5 is an instantiated chain and hence the trail is blocked. All trails between Y and Z have been considered.

   For the backdoor criterion with respect to (Z, Y), the sets have to block all trails with an arrow pointing towards Z. This means that any set that contains X_6, X_4 and any node from $\{X_3, X_1, X_2, X_5\}$ will satisfy the backdoor criterion with respect to (Z, Y); any set that does not will not.

3. It is clear that

$$\mathbb{P}_{Y\|X}(y\|x) = \sum_c \mathbb{P}_{Y|C\|X}(y|c\|x)\mathbb{P}_{C\|X}(c\|x).$$

Since C blocks all trails between Y and X that have an edge pointing towards X, it follows that $Y \perp\!\!\!\perp (\mathrm{Pa}_X \backslash C)\|_{\mathcal{G}} C$. It follows, with notation that should be clear, using Proposition 3.12 that

$$\begin{aligned}
\mathbb{P}_{Y|C\|X}(y|c\|x) &= \sum_{\pi\backslash c} \mathbb{P}_{Y|C,\mathrm{Pa}_X\|X}(y|c,\pi\backslash c\|x)\mathbb{P}_{\mathrm{Pa}_X\backslash C\|X}(\pi\backslash c\|x) \\
&= \sum_{\pi\backslash c} \mathbb{P}_{Y|C,\mathrm{Pa}_X,X}(y|c,\pi\backslash c,x)\mathbb{P}_{\mathrm{Pa}_X\backslash C}(\pi\backslash c) \\
&= \sum_{\pi\backslash c} \mathbb{P}_{Y|C,X}(y|c,x)\mathbb{P}_{\mathrm{Pa}_X\backslash C}(\pi\backslash c) \\
&= \mathbb{P}_{Y|C,X}(y|c,x).
\end{aligned}$$

Furthermore, since none of the variables in C are descendants of X, it follows (again, using Proposition 3.12) that

$$\mathbb{P}_{C\|X}(c\|x) = \mathbb{P}_C(c)$$

and the result follows. The fact that $\mathbb{P}_{C\|X}(c\|x) = \mathbb{P}_C(c)$,

$$\mathbb{P}_{Y|C,\mathrm{Pa}_X\|X}(y|c,\pi\backslash c\|x) = \mathbb{P}_{Y|C,\mathrm{Pa}_X,X}(y|c,\pi\backslash c,x)$$

and $\mathbb{P}_{\mathrm{Pa}_X\backslash C\|X}(\pi\backslash c\|x) = \mathbb{P}_{\mathrm{Pa}_X\backslash C}(\pi\backslash c)$ is clear by comparing the original DAG and the mutilated graph. A formal algebraic proof that $\mathbb{P}_{C\|X}(c\|x) = \mathbb{P}_C(c)$ is given in the next exercise.

4. The variables may be ordered as $V = \{Y_1, \ldots, Y_n, X, Y_{n+1}, \ldots, Y_{n+m}\}$ where the ordering is chosen such that $\mathrm{Pa}(Y_j) \subseteq \{Y_1, \ldots, Y_{j-1}\}$ for $j \leq n$, $\mathrm{Pa}(X) \subseteq \{Y_1, \ldots, Y_n\}$,

$$\mathrm{Pa}(Y_j) \subseteq \{Y_1, \ldots, Y_n, X, Y_{n+1}, \ldots, Y_{j-1}\}$$

for $j \in \{m+1, \ldots, n+m\}$ and where $C \subseteq \{Y_1, \ldots, Y_n\}$. From the intervention formula,

$$\mathbb{P}_{V\backslash X\|X}(y_1, \ldots, y_{m+n}\|x) = \prod_{j=1}^{m+n} \mathbb{P}_{Y_j|\mathrm{Pa}_j}(y_j|\pi_j)$$

while

$$\mathbb{P}_V(y_1, \ldots, y_{m+n}, x) = \mathbb{P}_{X|\mathrm{Pa}(X)}(x|\pi_X) \prod_{j=1}^{m+n} \mathbb{P}_{Y_j|\mathrm{Pa}_j}(y_j|\pi_j).$$

Now, sum over variables $Y_{n+1}, \ldots, Y_{n+m}$ in both expressions, then sum over X in the second expression. Then sum over all remaining variables not in C. The same answer obtains for both expressions, so that

$$\mathbb{P}_{C\|X} = \mathbb{P}_C.$$

5. Firstly,

$$\mathbb{P}_{V\backslash\{X_j\}\|\{X_j\}} = \mathbb{P}_{V\backslash(\{X_j\}\cup \mathrm{Pa}_j)|X_j,\mathrm{Pa}_j}\mathbb{P}_{\mathrm{Pa}_j}.$$

Now let $\mathrm{Pa}_U = \mathrm{Pa}_j \cap U$, U_a denote ancestors of X_j in U, U_b ancestors of $Z\backslash(\{X_j\} \cup \mathrm{Pa}_j)$ in U and $U_c = U\backslash(\mathrm{Pa}_U \cup U_a \cup U_b)$.

Sum over the variables in $V\backslash(Z \cup \mathrm{Pa}_U)$, then, from the condition that there are no trails between X_j and other variables in Z that contain only fork or chain connections, $Z\backslash\{X_j\} \cup \mathrm{Pa}_j$ is d-separated from Pa_U by $\{X_j\}$. It follows that

$$\begin{aligned}\mathbb{P}_{Z\cup\mathrm{Pa}_U\backslash\{X_j\}\|\{X_j\}} &= \mathbb{P}_{Z\backslash(\{X_j\}\cup(\mathrm{Pa}_j\cap Z))|X_j,(\mathrm{Pa}_j\cap Z),\mathrm{Pa}_U}\mathbb{P}_{\mathrm{Pa}_j} \\ &= \mathbb{P}_{Z\backslash(\{X_j\}\cup(\mathrm{Pa}_j\cap Z))|X_j,(\mathrm{Pa}_j\cap Z)}\mathbb{P}_{\mathrm{Pa}_j}\end{aligned}$$

so that

$$\mathbb{P}_{Z\backslash\{X_j\}\|\{X_j\}} = \mathbb{P}_{Z\backslash(\{X_j\}\cup(\mathrm{Pa}_j\cap Z)\cup\mathrm{Pa}_U)|X_j,(\mathrm{Pa}_j\cap Z)}\mathbb{P}_{\mathrm{Pa}_j\cap Z}.$$

Chapter 4

The Pioneering Work of Arthur Cayley

4.1 Cayley's Contribution

Arthur Cayley F.R.S. (16 August 1821 - 26 January 1895) was a British mathematician, known for his work in pure mathematics. His contributions include the so-called Cayley-Hamilton theorem, that every square matrix satisfies its own characteristic polynomial, which he verified for matrices of order 2 and 3 (1858) [16]. He was the first to define the concept of a group in the modern way, as a set with a binary operation satisfying certain laws. From group theory, he is known for *Cayley's theorem*, which states that every group G is isomorphic to a subgroup of the symmetric group acting on G (1854) [15].

In the context of Bayesian networks, attention is drawn to a short article by Arthur Cayley from 1853, where in an example that takes less than one page, he seems to develop several principles that later formed the basis of the subject of Bayesian networks, in particular, the 'noisy or' gate.

Here is the article in its entirety.

> XXXVII. *Note on a Question in the Theory of Probabilities.*
>
> *By A. Cayley*.*
>
> The following question was suggested to me, either by some of Prof. Boole's memoirs on the subject of probabilities, or in conversation with him, I forget which; it seems to me a good instance of the class of questions to which it belongs.
>
> Given the probability α that a cause A will act, and the probability p that A acting the effect will happen; also the probability β that a cause B will act, and the probability q that B acting the effect will happen; required the total probability of the effect.
>
> As an instance of the precise case contemplated, take the following: say a day is called *windy* if there is at least w of wind, and a day is called *rainy* if there is at least r of rain, and a day is called *stormy* if there is at least W of wind, *or* if there is at least R of rain. The day may therefore be stormy because of there being at least W of wind, or because of there being at least R of rain, or on both accounts; but if there is less than W of wind *and* less than R of rain, the day will not be stormy. Then α is the probability that a day

chosen at random will be windy, p the probability that a windy day chosen at random will be stormy, β the probability that a day chosen at random will be rainy, q the probability that a rainy day chosen at random will be stormy. The quantities λ, μ introduced in the solution of the question mean in this particular instance, λ the probability that a windy day chosen at random will be stormy by reason of the quantity of wind, or in other words, that there will be at least W of wind, μ the probability that a rainy day chosen at random will be stormy by reason of the quantity of rain, or in other words, that there will be at least R of rain.

The sense of the terms being clearly understood, the problem presents of course no difficulty. Let λ be the probability that the cause A acting will act efficaciously; μ the probability that the cause B acting will act efficaciously; then

$$p = \lambda + (1 - \lambda)\mu\beta$$

$$q = \mu + (1 - \mu)\alpha\lambda,$$

which determine λ, μ; and the total probability ρ of the effect is given by

$$\rho = \lambda\alpha + \mu\beta - \lambda\mu\alpha\beta,$$

suppose, for instance, $\alpha = 1$, then

$$p = \lambda + (1 - \lambda)\mu\beta, \quad q = \mu + \lambda - \lambda\mu, \quad \rho = \lambda + \mu\beta - \lambda\mu\beta,$$

that is, $\rho = p$, for p is in this case the probability that (acting as a cause which is certain to act) the effect will happen, or what is the same thing, p is the probability that the effect will happen.

Machynlleth, August 16, 1853.

*Communicated by the Author.

In this short note, Cayley gives a prototype example of a causal network; rain and wind both have causal effects on the state of the day (stormy or not), which may be *inhibited*. He demonstrates the key principle of *modularity*, taking a problem with several variables and splitting it into its simpler component conditional probabilities, by considering the direct causal influences for each variable and considering the natural factorisation of the probability distribution in this problem into these conditional probabilities.

It should also be pointed out that Cayley was no stranger to graph theory; he proved *Cayley's tree formula*, that there are n^{n-2} distinct labelled trees of order n (1889) [19] and established links between graph theory and group theory, representing groups by graphs. The *Cayley graph* is named after him.

The variables here may be taken as

$$C = \begin{cases} 1 & \text{wind} \\ 0 & \text{no wind} \end{cases} \qquad D = \begin{cases} 1 & \text{rain} \\ 0 & \text{no rain} \end{cases}$$

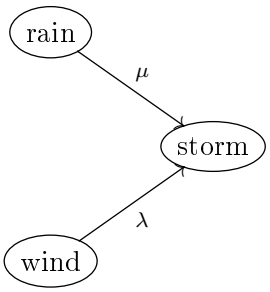

Figure 4.1: Rain and wind causing a storm

with

$$\alpha = \mathbb{P}_C(1) \qquad \beta = \mathbb{P}_D(1).$$

Let Y be the variable denoting whether there is a storm;

$$Y = \begin{cases} 1 & \text{storm} \\ 0 & \text{no storm} \end{cases}$$

Then, in Cayley's notation, if there is rain, it causes a storm with probability μ; if there is wind, it causes a storm with probability λ. The corresponding 'network', on three variables, is seen in Figure 4.1. The subscripts μ and λ on the arrows indicate the probability that the cause, if active, will trigger the effect.

This is a *noisy 'or' gate*, which can be expressed as a logical 'or' gate by the addition of two variables, R and W. The variable R denotes *severe rain*, that is that the 'rain' variable reaches the threshold to trigger a storm. This happens if the quantity of rain is above a threshold. The W variable denotes *severe wind*; that is, that the 'wind' variable reaches the threshold to trigger a storm. This happens if the strength of wind is above a threshold. The variables, to form the logical or gate have conditional probability values given below; $\mathbb{P}_{W|C}$ denotes the conditional probability function for the variable W given C and $\mathbb{P}_{R|D}$ denotes the conditional probability function for the variable R given D.

$$\mathbb{P}_{W|C} = \begin{array}{c|cc} C\backslash W & 1 & 0 \\ \hline 1 & \lambda & 1-\lambda \\ 0 & 0 & 1 \end{array} \qquad \mathbb{P}_{R|D} = \begin{array}{c|cc} D\backslash R & 1 & 0 \\ \hline 1 & \mu & 1-\mu \\ 0 & 0 & 1 \end{array}$$

The network may now be expressed graphically according to Figure 4.2. This DAG is a representation of the factorisation that Cayley is using;

$$\mathbb{P}_{C,D,R,W,Y} = \mathbb{P}_C \mathbb{P}_D \mathbb{P}_{R|C} \mathbb{P}_{W|D} \mathbb{P}_{Y|W,R}$$

where $\mathbb{P}_{Y|W,R}$ denotes the CPP for the variable Y, given W and R. For $Y = 1$, these values are given in the following table:

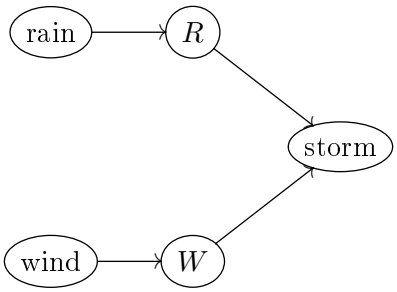

Figure 4.2: Rain and wind: logical 'or' gate

$$\mathbb{P}_{Y|W,R}(1|.,.) = \begin{array}{c|cc} W\backslash R & 1 & 0 \\ \hline 1 & 1 & 1 \\ 0 & 1 & 0 \end{array}.$$

From the factorisation,

$$\mathbb{P}_W(1) = \sum_x \mathbb{P}_{W|C}(1|x)\mathbb{P}_C(x) = \lambda\alpha, \quad \mathbb{P}_R(1) = \mu\beta,$$

From Cayley, p is the probability that a windy day, chosen at random, will be stormy; $\mathbb{P} = \mathbb{P}_{Y|D}(1|1)$.

$$\begin{aligned} p &= \mathbb{P}_{Y|D}(1|1) = \sum_{x_1}\mathbb{P}_A(x_1)\sum_{x_2}\mathbb{P}_{R|C}(x_2|x_1)\sum_{x_3}\mathbb{P}_{Y|R,W}(1|x_2,x_3)\mathbb{P}_{W|D}(x_3|1) \\ &= \beta\lambda\mu + \beta\mu(1-\lambda) + \beta(1-\mu)\lambda + (1-\beta)\lambda \\ &= \beta\mu - \beta\lambda\mu + \lambda = \lambda + (1-\lambda)\beta\mu. \end{aligned}$$

Similarly, q, the probability that a rainy day, chosen at random, will be stormy; $q = \mathbb{P}_{Y|C}(1|1)$, is given by

$$q = \mu + (1-\mu)\alpha\lambda,$$

as computed by Cayley. Cayley is deriving the expression for the marginal probability of a stormy day, $\rho = \mathbb{P}_Y(1)$;

$$\begin{aligned} \mathbb{P}_Y(1) &= \sum_{x_1}\mathbb{P}_C(x_1)\sum_{x_2}\mathbb{P}_D(x_2)\sum_{x_3}\mathbb{P}_{R|C}(x_3|x_1)\sum_{x_4}\mathbb{P}_{W|D}(x_4|x_2)\mathbb{P}_{Y|R,W}(1|x_3,x_4) \\ &= \sum_{x_3}\mathbb{P}_R(x_3)\sum_{x_4}\mathbb{P}_W(x_4)\mathbb{P}_{Y|R,W}(1|x_3,x_4) \\ &= \mathbb{P}_R(1)\mathbb{P}_W(1) + \mathbb{P}_R(1)\mathbb{P}_W(0) + \mathbb{P}_R(0)\mathbb{P}_W(1) \\ &= \alpha\lambda + \beta\mu - \alpha\beta\lambda\mu. \end{aligned}$$

This simple construction from 1853 possibly represents the first example of a causal network and the first construction of a noisy-or gate, with the concept of an inhibitor.

4.2 Arthur Cayley and Judea Pearl's intervention calculus

There is the cryptic remark towards the end of Arthur Cayley's paper, which indicates that he may already have had the framework of Judea Pearl's intervention calculus in mind when considering causal probabilistic models. The phrase ' .... acting a cause which is certain to act' may be a clumsy way of expressing a brilliant insight into the intervention calculus, if by 'acting' he means intervening to force the state of the variable.

This reading may be somewhat strained; in Arthur Cayley's example, no human intervention is possible to force the states of the wind or rain variables. Since 'wind' and 'rain' are both ancestor variables, no links are removed from the DAG and in Pearl's framework, intervention conditioning is the same as the standard conditioning on an observation. The wording suggests, though, that he understood, from causal principles, that the two equations relating λ and μ to p and q remain valid if the conditioning on an ancestor variable is forced by intervention, rather than simply observed, one of the features of Pearl's intervention calculus.

4.3 Arthur Cayley: algebraic geometry and Bayesian networks

The emerging field of algebraic statistics (Pistone et al. (2001) [115], Drton et. al. (2009) [39]) advocates polynomial algebra as a tool in the statistical analysis of experiments and discrete data; the connection between algebraic geometry and Bayesian networks is discussed by Garcia et. al. (2005) in [50].

For a probability distribution over a set of variables, the conditional independence statements for subsets X, Y, Z, W satisfy the logical relations of decomposition, contraction, weak union and intersection, described on page 32 chapter 2.

A *factorisation* is equivalent to a set of conditional independence statements;

$$\{X_{\sigma(j)} \perp X_{\Xi^\sigma(j)} | X_{\mathrm{Pa}^\sigma(j)} \quad j = 1, \ldots, d\},$$

where $\mathrm{Pa}^\sigma(j) \subset \{\sigma(1), \ldots, \sigma(j-1)\}$ is the parent set of node $\sigma(j)$ when ordering σ is employed and $\Xi^\sigma(j) = \{\sigma(1), \ldots, \sigma(j-1)\} \backslash \mathrm{Pa}^\sigma(j)$.

Let $V = \{1, \ldots, d\}$ denote the node set which indexes the variables, $X = (X_1, \ldots, X_d)$ the random vector, let the indexing set for the state space for variable X_j be $\mathcal{X}_j = \{0, 1, \ldots, k_j - 1\}$ and the indexing set for the state space for X be $\mathcal{X} = \times_{j=1}^d \mathcal{X}_j$. Let $\mathbb{R}(\mathcal{X})$ the ring of polynomial functions on $\mathbb{R}^{\mathcal{X}}$.

A conditional independence statement $X_A \perp X_B | X_C$, where A, B and C are disjoint subsets of V, translates using proposition 8.1 from Sturmfels (2002) [130], into a set of homogeneous quadratic polynomials on $\mathbb{R}(\mathcal{X})$, and these polynomials generate an *ideal*. Let $\mathcal{I}_{A \perp B|C}$ denote the ideal generated by the statement $X_A \perp X_B | X_C$. The ideal for a collection of independence statements, for example those corresponding to a factorisation, is defined as the sum of the ideals; let $\mathcal{M} = \{X_{A_i} \perp X_{B_i} | X_{C_i} \qquad i = 1, \ldots, m\}$, then

$$\mathcal{I}_{\mathcal{M}} = \mathcal{I}_{A_1 \perp B_1 | C_1} + \ldots + \mathcal{I}_{A_m \perp B_m | C_m}.$$

Cayley is using the expression of the conditional independence statements that define the factorisation in terms of polynomials to obtain the two polynomial equations

$$\begin{cases} p = \lambda + (1-\lambda)\mu\beta \\ q = \mu + (1-\mu)\lambda\alpha \end{cases} \tag{4.1}$$

and writes, '.... which determine λ and μ'. This amounts to finding roots of the two polynomials in λ, μ

$$\begin{cases} f_1(\lambda,\mu) = \lambda + (1-\lambda)\mu\beta - p \\ f_2(\lambda,\mu) = \mu + (1-\mu)\lambda\alpha - q \end{cases}$$

In terms of algebraic geometry, equation (4.1) defines the *affine variety*

$$V(f_1, f_2) = \left\{(\lambda,\mu) \in \mathbb{R}^2 | f_1(\lambda,\mu) = f_2(\lambda,\mu) = 0\right\}.$$

In his brief note, Cayley has pointed out the connections between Bayesian networks and algebraic geometry, a subject that he knew well. Cayley did much to clarify a large number of interrelated theorems in algebraic geometry and is known for the Cayley surface (1869) [17].

Chapter 5

Moral Graph, Independence Graph, Chain Graphs

The definition of a chain graph is given below and it is shown that an essential graph is a chain graph, although not vice versa. The study of chain graphs will be developed in 5.2.

Definition 5.1 (Chain Graph)**.** *A* chain graph *is a graph $\mathcal{G} = (V, E)$, where the edge set contains both directed and undirected edges, $E = D \cup U$, where D is the set of directed edges and U the set of undirected edges. The node set V can be partitioned into n disjoint subsets $V = V_1 \cup \ldots \cup V_n$ where the sets $V_1, \ldots, V_n$ are the node sets of the connected components of (V, U), the graph obtained by removing all the directed edges.*

1. *$\mathcal{G}_{V_j}$ is an undirected graph for all $j = 1, \ldots, n$*
2. *For any $i \neq j$, and any $\alpha \in V_i$, $\beta \in V_j$, there is no cycle in $\mathcal{G} = (V, E)$ (Definition 1.9) containing both α and β.*

The chain graph consists of components where the edges are undirected, which are connected by directed edges. The components with undirected edges are known as *chain components*, which are defined below.

Definition 5.2 (Chain Component)**.** *Let $\mathcal{G} = (V, E)$ be a chain graph, where $E = D \cup U$, D is the set of directed edges. Let $\widehat{\mathcal{G}} = (V, U)$ denote the graph obtained by* removing *all the* directed *edges from E. Each connected component of $\widehat{\mathcal{G}}$ is known as a* chain component.

The *chain components* (V_j, U_j), $j = 1, \ldots, n$ of $\mathcal{G}$ therefore satisfy the following conditions:

1. $V_j \subseteq V$ and U_j is the edge set obtained by retaining all *undirected* edges $\langle \alpha, \beta \rangle \in E$ such that $\alpha \in V_j$ and $\beta \in V_j$.
2. There is no undirected edge in E from any node in $V \backslash V_j$ to any node in V_j.

Theorem 5.3 states any essential graph is necessarily a chain graph and presents the additional features required to ensure that a chain graph is an essential graph corresponding to a directed acyclic graph. It gives a characterisation for essential graphs that is useful for structure learning algorithms.

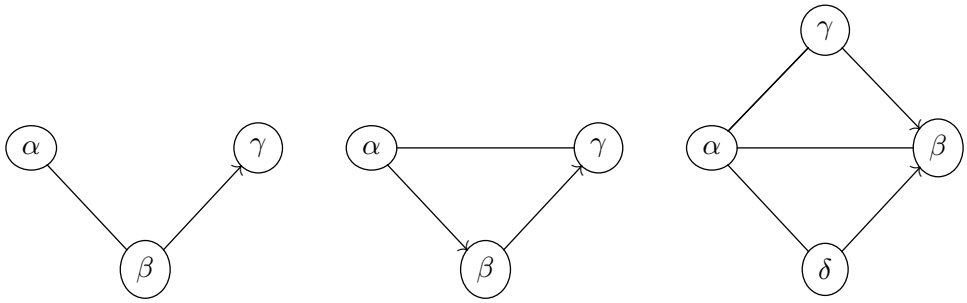

Figure 5.1: Forbidden subgraphs

Theorem 5.3. *Let $\mathcal{G} = (V, E)$ be a graph, where $E = D \cup U$. There exists a directed acyclic graph $\mathcal{G}^*$ for which $\mathcal{G}$ is the corresponding essential graph if and only if $\mathcal{G}$ satisfies the following conditions:*

1. *$\mathcal{G}$ is a chain graph,*
2. *Each chain component of $\mathcal{G}$ is triangulated,*
3. *The configurations shown in Figure 5.1 do not occur in any induced sub-graph of a three variable set $\{\alpha, \beta, \gamma\} \subset V$ for the first two configurations or a four variable set $\{\alpha, \beta, \gamma, \delta\}$ for the third configuration.*
4. *Every directed edge $(\alpha_1, \alpha_2) \in D$ is compelled in $\mathcal{G}$.*

Proof Proof that an essential graph satisfies the conditions. To prove that it is a chain graph, the first part of the definition is easily satisfied and it is sufficient to show that there is no cycle in (V, E) containing $\alpha \in V_i$ and $\beta \in V_j$ for two distinct chain components V_i and V_j.

Recall that the edges of a cycle $\tau_0, \ldots, \tau_n$ are either directed (τ_i, τ_{i+1}) or undirected $\langle \tau_i, \tau_{i+1} \rangle$. Let (τ, γ) denote a directed edge in the cycle where $\gamma \in V_j$. Both connected components will have a node γ with this property. If there is an undirected edge $\langle \gamma, \gamma_1 \rangle$ in the cycle, then there is also an undirected edge $\langle \tau, \gamma_1 \rangle$ in the graph. If there is a directed edge (τ, γ_1) or (γ_1, τ) then the edge between γ and γ_1 is compelled contradicting the fact that it is undirected. If there is an undirected edge $\langle \tau, \gamma_1 \rangle$, then $\tau \in V_j$. Proceeding inductively, it is clear that if there is a cycle, then there is an undirected edge $\langle \tau_1, \tau_2 \rangle$ where $\tau_1 \in V_i$ and $\tau_2 \in V_j$ contradicting the fact that the two chain components are distinct. It follows that an essential graph is a chain graph.

Secondly, if there is a cycle of length ≥ 4 of undirected edges without a chord, then the DAG will have a directed cycle, otherwise additional immoralities will appear when the edges are directed, hence the chain components are triangulated.

Thirdly, the configuration stated cannot appear in an essential graph. The fourth requirement follows from the definition of an essential graph.

For the other direction: suppose a graph satisfies the four conditions stated. All the directed edges appear in configurations that are compelled and from the forbidden subgraphs, no undirected edges appear in compelled configurations where there should be a directed edge. It remains to show that the undirected edges may be oriented in a way that produces a directed acyclic graph.

For each chain component, orient the edges so that the chain component is a directed acyclic triangulated graph. This can be done. Then, since the first structure is forbidden, this operation does not produce additional immoralities in the whole graph. Furthermore, since there are no cycles containing two nodes α and β with $\alpha \in V_j$ and $\beta \in V_k$ for $j \neq k$, this operation does not produce directed cycles. The graph is therefore the essential graph of a DAG. □

5.1 The Moral Graph and the Independence Graph

Let $(\mathbb{P}, \mathcal{G})$ be a Bayesian network; that is, a probability distribution $\mathbb{P}$ over a random vector $\underline{X} = (X_1, \ldots, X_d)$, such that $\mathbb{P}$ factorises along a directed acyclic graph $\mathcal{G} = (V, D)$, and this is no longer true if any variable is eliminated from any of the parent sets.

Definition 5.4 (Moral Graph)**.** *Let $\mathcal{G} = (V, D)$ be a directed acyclic graph. The moral graph $\mathcal{G}^{(m)} = (V, U)$ is the undirected graph such that for any $\alpha, \beta \in V$, $\langle \alpha, \beta \rangle \in U$ if and only if either $(\alpha, \beta) \in D$ or $(\beta, \alpha) \in D$ or $\{\alpha, \beta\} \in Pa(\gamma)$ for some $\gamma \in V$. That is, the moral graph is the graph obtained by firstly for each node adding links between all the parent variables of the node and then undirecting all the directed edges.*

The moral graph satisfies the following property:

Theorem 5.5. *Let $\mathcal{G} = (V, D)$ be a directed acyclic graph and let $\mathcal{G}^{(m)} = (V, U)$ be its moral graph. There is an edge $\langle \alpha, \beta \rangle \in U$ if and only if $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} V \backslash \{\alpha, \beta\}$. That is, the moral graph has an edge if and only if α and β are not D-separated by the remaining variables.*

Proof The proof of this is left as an exercise (Exercise 6 page 352). □

The *independence graph* is defined as follows:

Definition 5.6 (Independence Graph)**.** *Let $X = (X_1, \ldots, X_d)$ be a random vector. The* independence graph *$\mathcal{G} = (V, U)$ is the undirected graph with vertex set $V = \{1, \ldots, d\}$ and where $\langle \alpha, \beta \rangle \in U$ for $\alpha \neq \beta$ if and only if $X_\alpha \not\perp\!\!\!\perp X_\beta | X_{-(\alpha,\beta)}$ where the notation $X_{-(\alpha,\beta)}$ denotes X without components X_α and X_β.*

Recall the definition of separator (Definition 7.15). The independence graph satisfies the following property:

Theorem 5.7. *Let $X = (X_1, \ldots, X_d)$ be a random vector, let $V = \{1, \ldots, d\}$ be the indexing set for X and let $\mathcal{G} = (V, U)$ be the independence graph of X. Then for three disjoint sets A, B and S such that $V = A \cup B \cup S$, it holds that $A \perp B | S$ (A and B are conditionally independent given S) if and only if $A \perp\!\!\!\perp B \,[\!]\, S$ (A and B separated by S).*

Proof Firstly, assume that for three disjoint sets A, B and S such that $A \cup B \cup S$, $A \perp\!\!\!\perp B \,[\!]\, S$ in the independence graph. Then, for each $\alpha_1, \alpha_2 \in A$ and $\beta \in B$, set $C = V \backslash \{\alpha_1, \alpha_2, \beta\}$. From the definition of the independence graph,

$$X_{\alpha_1} \perp X_\beta | X_{C \cup \{\alpha_2\}} \qquad \text{and} \qquad X_{\alpha_2} \perp X_\beta | X_{C \cup \{\alpha_1\}}.$$

It follows from the **intersection** property, which states that if $X \perp Y | W \cup Z$ and $X \perp W | Y \cup Z$ then $X \perp W \cup Y | Z$, that

$$(X_{\alpha_1}, X_{\alpha_2}) \perp X_\beta | X_C.$$

By successive applications of the **intersection** property to each variable, it follows that

$$X_A \perp X_\beta | X_{-(A \cup \{\beta\})}.$$

This holds for all $\beta \in B$. The *intersection* property gives:

$$X_A \perp X_{\beta_1} | X_{-(A \cup \{\beta_1\})} \qquad \text{and} \qquad X_A \perp X_{\beta_2} | X_{-(A \cup \{\beta_2\})} \Rightarrow X_A \perp (X_{\beta_1}, X_{\beta_2}) | X_{-(A \cup \{\beta_1, \beta_2\})}.$$

Successive applications of the intersection property to the variables with indices in B give

$$X_A \perp X_B | X_S.$$

Now assume that $X_A \perp X_B | X_S$. Then, for each $\alpha \in A$ and $\beta \in B$, this may be rewritten as

$$(X_\alpha, X_{A \backslash \{\alpha\}}) \perp (X_\beta, X_{B \backslash \beta}) | X_S.$$

Using the **weak union** result, that $X \perp Y \cup Z | W \Rightarrow X \perp Y | Z \cup W$ it follows that

$$X_\alpha \perp X_B | X_{S \cup A \backslash \{\alpha\}}$$

and another application gives

$$X_\alpha \perp X_\beta | X_{-(\alpha, \beta)}.$$

□

Theorem 5.8. *Let $\mathbb{P}$ be a probability distribution that factorises along a DAG $\mathcal{G} = (V, D)$. Let $\mathcal{G}^{(m)} = (V, U^{(m)})$ denote its moral graph and let $\mathcal{G}^{(i)} = (V, U^{(i)})$ denote the independence graph of $\mathbb{P}$. Then $U^{(i)} \subseteq U^{(m)}$. Furthermore, if (V, D) is faithful to $\mathbb{P}$, then $U^{(i)} = U^{(m)}$.*

Proof From Theorem 5.5, the moral graph has an edge $\langle\alpha,\beta\rangle$ if and only if $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} V\backslash\{\alpha,\beta\}$; there is no edge $\langle\alpha,\beta\rangle$ if and only if $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} V\backslash\{\alpha,\beta\}$. Since D-separation implies conditional independence (Theorem 1.25), it follows that the lack of an edge $\langle\alpha,\beta\rangle$ imples $X_\alpha \perp X_\beta | X_{-(\alpha,\beta)}$. From this, it follows directly that $U^{(i)} \subseteq U^{(m)}$.

For a faithful DAG, D-separation and conditional independence are equivalent, from which it follows that $U^{(i)} = U^{(m)}$ when $\mathbb{P}$ and $\mathcal{G} = (V, D)$ are faithful. □

If a distribution $\mathbb{P}$ does not have a faithful representation, then for any DAG $U^{(i)} \subset U^{(m)}$.

The following corollary is an obvious consequence of the preceeding.

Corollary 5.9. *Let $X = (X_1, \ldots, X_d)$ be a random vector and $V = \{1, \ldots, d\}$ be its indexing set. Let $\mathcal{G} = (V, D)$ be a directed acyclic graph, along which $\mathbb{P}$, the probability distribution of X, factorises and let $\mathcal{G}^{(m)}$ be the moral graph. Let $V = A \cup B \cup S$ where A, B and S are disjoint subsets. Then $A \perp\!\!\!\perp B \,[\!]\, S$ (A and B separated by S in $\mathcal{G}^{(m)}$) implies $X_A \perp X_B | X_S$ (A and B conditionally independent given S).*

Proof A clear consequence of the preceeding arguments. □

5.2 Chain Graphs

5.2.1 Motivation

Consider the problem of finding a graphical model where each graphical separation statement implies the corresponding conditional independence statement, and the aim is to locate a graph structure which encodes as much of the conditional independence structure as possible. When there does not exist a faithful DAG, a Bayesian Network cannot encode the complete set of conditional independence statements. Chain graphs give a substantially broader class of graphical models which can encode more of the conditional independence structure.

Example 5.10 (Chain Graph (1))**.**

Consider the situation where a probability distribution is constructed out of pairwise potentials:

$$\mathbb{P}_{X_1,X_2,X_3,X_4}(x_1,x_2,x_3,x_4) = C \exp\left\{-\beta_{12}(x_1-x_2) - \beta_{23}(x_2-x_3) - \beta_{34}(x_3-x_4) - \beta_{14}(x_1-x_4)\right\}.$$

Consider a factorisation of this distribution

$$\mathbb{P}_{X_1,X_2,X_3,X_4} = \mathbb{P}_{X_1}\mathbb{P}_{X_2|X_1}\mathbb{P}_{X_3|X_1,X_2}\mathbb{P}_{X_4|X_1,X_2,X_3}.$$

Note that

$$\mathbb{P}_{X_1,X_2,X_3} = C \exp\left\{-\beta_{12}(x_1-x_2) - \beta_{23}(x_2-x_3)\right\} \sum_{x_4} \exp\left\{-\beta_{34}(x_3-x_4) - \beta_{14}(x_1-x_4)\right\}.$$

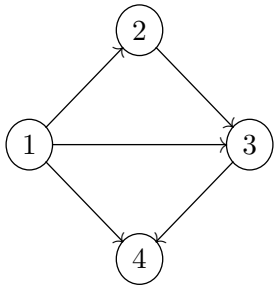

Figure 5.2: DAG for 4 variable example

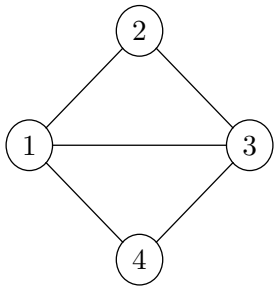

Figure 5.3: Moral graph for 4 variable example

It follows that the BN is given by the following factorisation:

$$\mathbb{P}_{X_1,X_2,X_3,X_4} = \mathbb{P}_{X_1}\mathbb{P}_{X_2|X_1}\mathbb{P}_{X_3|X_1,X_2}\mathbb{P}_{X_4|X_1,X_3}.$$

with DAG given by Figure 5.2. The *moral graph* of the DAG of Figure 5.2 is given in Figure 5.3.

Whatever ordering of the variables, the moral graph of the resulting Bayesian network will be *triangulated*. The cliques of the moral graph are the parent/variable sets of factorisation.

More of the independence structure is revealed in this example by the *factor graph* shown in Figure 5.4, which is a chain graph. □

Example 5.11 (Chain Graph (2))**.**

Figure 5.5 gives an example of a chain graph which is not an essential graph, where the chain components are nevertheless triangulated. Its chain components are shown in Figure 5.6.
Figure 5.5 is the chain graph of a probability distribution which has factorisation:

$$\mathbb{P}_{X_1,X_2,X_3,X_4} = \mathbb{P}_{X_1}\mathbb{P}_{X_2}\mathbb{P}_{X_3,X_4|X_1,X_2},$$

but where neither $X_1 \perp X_4|X_3$ nor $X_2 \perp X_3|X_4$ hold. Such a distribution could arise, for example, with a probability distribution

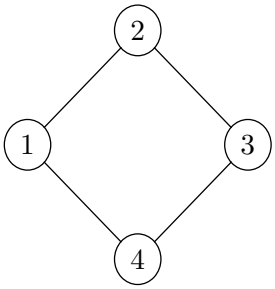

Figure 5.4: Factor graph for 4 variable example

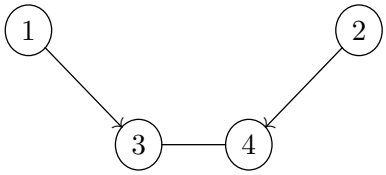

Figure 5.5: Chain Graph, Not an Essential Graph

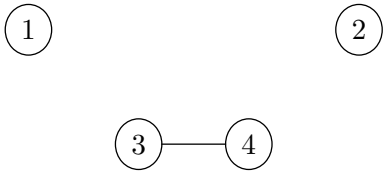

Figure 5.6: Chain Components of Chain Graph, Figure 5.5

$$\mathbb{P}_{U,X_1,X_2,X_3,X_4} = \mathbb{P}_U \mathbb{P}_{X_1} \mathbb{P}_{X_2} \mathbb{P}_{X_3|X_1,U} \mathbb{P}_{X_4|X_2,U}$$

where U is a hidden variable. □

The additional flexibility available for modelling when chain graphs are used should be clear. Chain graphs, however, still satisfy the *composition* property and therefore separation statements in a chain graph do not characterise the independence structure; there does not exist a faithful chain graph for Example 2.7, the three-coin example.

5.2.2 Factorisation along a Chain Graph

Let $X = (X_1, \ldots, X_d)$ be a random vector indexed by $V = \{1, \ldots, d\}$. Let $\mathbb{P}$ denote the probability distribution of X. The probability distribution is said to factorise along the chain graph $\mathcal{G} = (V, E)$ if and only if there exist functions $\varnothing_C : C \in \mathcal{C}$ and it has a decomposition of the form:

$$\mathbb{P}_{X_1,\ldots,X_d} = \prod_{j \in \mathcal{A}} \mathbb{P}_{X_j|X_{\mathrm{Pa}(j)}} \prod_{C \in \mathcal{C}} \varnothing_C$$

where $\mathcal{A} = \{j : \exists k : (k, j) \in D\}$ and $\mathcal{C}$ denotes the collection of cliques of the chain components; clique C is the domain of the function $\varnothing_C$ for each $C \in \mathcal{C}$.

To generalise from DAGs to chain graphs, some additional definitions and machinery are necessary. The approach taken here follows Ma-Xie-Geng (2008) [86].

A *head-to-head section* in a chain graph plays the same role as an immorality in a DAG.

Definition 5.12 (Section, Terminal)**.** *The* terminals *of a trail $\rho = (\rho_0, \ldots, \rho_k)$ are simply the nodes at each end, ρ_0 and ρ_k. A* section *of a trail $\rho = (\rho_0, \ldots, \rho_k)$ is a* maximal undirected subroute *$\sigma = (\rho_i, \ldots, \rho_j)$. In other words, either $\rho_i = \rho_0$ or else $i \neq 0$ and there is a directed edge $\rho_{i-1} \mapsto \rho_i$ or $\rho_i \mapsto \rho_{i-1}$; similarly, either $j = k$ or else there is a directed edge $\rho_j \mapsto \rho_{j+1}$ or $\rho_{j+1} \mapsto \rho_j$.*

The vertices ρ_i and ρ_j are called terminals*. ρ_i (ρ_j) is a* head terminal *if $i > 0$ and $\mathcal{G}$ contains the directed edge $\rho_{i-1} \mapsto \rho_i$ (or $j < k$ and $\mathcal{G}$ contains the edge $\rho_{j+1} \mapsto \rho_j$ and a* tail terminal *if $i > 0$ and the graph $\mathcal{G}$ contains the edge $\rho_i \mapsto \rho_{i-1}$. (or $j < k$ and the graph contains the edge $\rho_j \mapsto \rho_{j+1}$).*

A section σ of ρ is a head-to-head section *if it has two head-terminals, otherwise it is a* non head-to-head section.

For a set of vertices $S \subset V$, a section σ is outside *S if $\{\rho_i, \ldots, \rho_j\} \cap S = \varnothing$; otherwise we say that σ is* hit *by S.*

A *complex* within a trail in a chain graph plays a similar role to a collider node in a trail in a DAG.

Definition 5.13 (Complex)**.** *A* complex *in $\mathcal{G}$ is a trail $\rho = (\rho_0, \ldots, \rho_k)$ such that $\rho_0 \mapsto \rho_1$ and $\rho_k \mapsto \rho_{k-1}$ are in $\mathcal{G}$ and, for $i = 1, \ldots, k-2$ $\mathcal{G}$ contains the undirected edges $\rho_i - \rho_{i+1}$. The vertices ρ_0 and ρ_k are the* parents *of the complex and $\{\rho_1, \ldots, \rho_{k-1}\}$ the* region *of the complex.*

The *pattern* of a chain graph corresponds to taking the skeleton of a DAG and directing those edges which belong to immoralities.

Definition 5.14 (Complex Arrow, Pattern, Moral Graph)**.** *A directed edge in the chain graph is known as a* complex arrow *if it belongs to a complex of $\mathcal{G}$. The* pattern *of $\mathcal{G}$, denoted $\mathcal{G}^*$ is the graph obtained by undirecting all directed edges which are not complex arrows. The* moral graph $\mathcal{G}^{(m)}$ *of a chain graph is the graph obtained by first, for each complex, adding an undirected edge between each pair of parents of the complex, and then undirecting all the edges.*

For a chain graph, the descendants of a node are those for which there is a trail where each edge is either undirected or directed from the node to the descendant.

Definition 5.15 (Descendant)**.** *A node β is a* descendant *of a node α if there is a path $\rho = (\rho_0, \rho_1, \ldots, \rho_k)$ such that $\rho_0 = \alpha$, $\rho_k = \beta$ and for $i = 0, \ldots, k-1$ either there is either an undirected edge $\langle \rho_i, \rho_{i+1} \rangle$ or a directed edge (ρ_i, ρ_{i+1}) in $\mathcal{G}$.*

For a DAG, a connection is open if it is an uninstantiated fork or chain, or if it is a collider which is either instantiated or has an instantiated descendant. In chain graphs, this has to be developed slightly.

Definition 5.16 (Intervented)**.** *A trail ρ in $\mathcal{G}$ is* intervented *by a subset S of V if and only if there exists a section σ of ρ such that:*

1. *either σ is a head to head section with respect to ρ and σ and all its descendants are outside S, or*
2. *σ is a non-head-to-head section with respect to ρ and σ is hit by S.*

Note In [86], the requirement in *1.* that the descendants are also outside S is not given. It is clear that this is necessary, by considering the situation where the chain graph is a DAG.

The notion of C-separation for chain graphs corresponds to D-separation for DAGs.

Definition 5.17 (C-Separation)**.** *Let A, B and S be three disjoint subsets of V of a chain graph $\mathcal{G}$ such that A and B are non-empty. The sets A and B are C-separated by S, written $A \perp\!\!\!\perp B \| _{\mathcal{G}} S$ if and only if every trail with one of its terminals in A and another in B is intervented by S. The set S is a C-separator for A and B.*

The definition of *Markov equivalence* is the same, with C-separation substituted for D-separation.

Definition 5.18 (Markov Equivalence)**.** *Two chain graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are said to be Markov equivalent if for any three disjoint subsets A, B and S with both A and B non-empty,*

$$A \perp\!\!\!\perp B \| _{\mathcal{G}_1} S \Leftrightarrow A \perp\!\!\!\perp B \| _{\mathcal{G}_2} S.$$

Having formulated the concepts for chain graphs that correspond to those for DAGs, the key result for chain graphs corresponds directly to Theorem 2.11.

Theorem 5.19. *Two chain graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are Markov equivalent if and only if they have the same skeleton and the same complexes. That is, they have the same* pattern.

Proof Frydenberg [48] (1990). It is similar to the proof of Theorem 2.11 for DAGs. □

A distribution that factorises according to a chain graph is said to be *Markovian* with respect to the chain graph.

Definition 5.20 (Markovian)**.** *A distribution* $\mathbb{P}$ *is said to be* Markovian *with respect to a chain graph* $\mathcal{G}$ *if* C*-separation statements imply the corresponding independence statements:*

$$A \perp\!\!\!\perp B \| _{\mathcal{G}} S \Rightarrow X_A \perp X_B | X_S.$$

The definition of faithfulness for chain graphs is analogous to faithfulness for DAGs.

Definition 5.21 (Chain Graph Faithfulness)**.** *A distribution* $\mathbb{P}$ *is said to be* faithful *with respect to a chain graph* $\mathcal{G}$ *if* C*-separation statements and independence statements are equivalent;*

$$A \perp\!\!\!\perp B \| _{\mathcal{G}} S \Leftrightarrow X_A \perp X_B | X_S.$$

5.2.3 Separation Trees for Chain Graphs

A DAG can be moralised, the moral graph triangulated and the triangulated moral graph decomposed into a junction tree. This is the basis of the Aalborg inference engine. The moral graph for a chain graph is given by Definition 5.14. In Ma-Xie-Geng [86], the *separation tree* is proposed as the analogous object to the junction tree.

Definition 5.22. *Let* $\mathcal{G} = (V, E)$ *be a chain graph. Let* $\mathcal{C} = \{C_1, \ldots, C_H\}$ *be a collection of distinct sets of variables such that* $V = \cup_{j=1}^{H} C_j$*. Let* $\mathcal{T}$ *denote the graph* $(\mathcal{C}, \mathcal{U})$ *where* $\mathcal{U}$ *is a set of labelled undirected edges.* $U_{ij} \in \mathcal{U}$ *if and only if* $C_i \cap C_j \neq \varnothing$*; the label is* $C_i \cap C_j$ *and* U_{ij} *is the* separator*.*

$\mathcal{T}$ is said to be a tree *if removal of the nodes of* U_{ij} *for any pair* $i \neq j$ *splits* $\mathcal{T}$ *into two disjoint trees* $\mathcal{T}_i$ *(with node set denoted* $\mathcal{C}_i$*) and* $\mathcal{T}_j$ *(with node set denoted* $\mathcal{C}_j$*). Let* $V_i = \cup_{C \in \mathcal{C}_i} C$ *and* $V_j = \cup_{C \in \mathcal{C}_j} C$*.*

A tree $\mathcal{T}$ *with node set* $\mathcal{C}$ *is a* separation tree *for chain graph* $\mathcal{G}$ *if and only if:*

1. $\cup_{C \in \mathcal{C}} C = V$ *and*
2. *For any separator* $S \in \mathcal{U}$*, with* V_1 *and* V_2 *defined above by removing* S*,*

$$V_1 \backslash S \perp\!\!\!\perp V_2 \backslash S \| _{\mathcal{G}} S.$$

The separation tree has similarities to the junction tree, but it does not require that the collection $\{C_1, \ldots, C_H\}$ are cliques or that every separator is complete.

A separation tree can be constructed quite easily from the independence graph.

Theorem 5.23. *Let* $X = (X_1, \ldots, X_d)$ *be a random vector and let* $\mathcal{G}^{(i)}$ *denote the independence graph. Any junction tree constructed from any triangulation of* $\mathcal{G}^{(i)}$ *is a separation tree.*

Proof This is obvious, since any separation statement in the independence graph implies the corresponding C-separation statement in the chain graph. □

Lemma 5.24. *Let α and β be two adjacent nodes in a chain graph $\mathcal{G}$, then any separation tree $\mathcal{T}$ for $\mathcal{G}$ contains a tree-node C such that $\{\alpha, \beta\} \subseteq C$.*

Proof Assume not, then there exists a separator K on $\mathcal{T}$ such that $\alpha \in V_1 \backslash K$ and $\beta \in V_2 \backslash K$, where V_i denotes the variable set of the subtree $\mathcal{T}_i$ obtained by removing the edge attached by separator K, for $i = 1, 2$. This implies that $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} K$, which is false. □

The separation tree satisfies several properties which will be useful in 16.12 for learning a chain graph. Some of them are collected in the following theorem.

Theorem 5.25. *Let $\mathcal{T}$ be a separation tree for a chain graph $\mathcal{G} = (V, E)$. Nodes α and β are C-separated by some set $S_{\alpha\beta} \subset V$ in $\mathcal{G}$ if and only if one of the following conditions hold:*

1. *α and β are not both contained in the same node C for any $C \in \mathcal{C}$.*
2. *$\alpha, \beta \in C$ for some $C \in \mathcal{C}$, but for any separator $S \subset C$, $\{\alpha, \beta\} \notin S$ and there exists a set $S'_{\alpha,\beta} \subset C$ such that*

$$\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} S'_{\alpha\beta}.$$

3. *There is a $C \in \mathcal{C}$ such that $\{\alpha, \beta\} \subseteq C$, there is a separator $S \subset C$ such that $\{\alpha, \beta\} \subseteq S$, but there is a subset $S'_{\alpha\beta}$ of either $\cup_{C:\alpha\in C} C$ or $\cup_{C:\beta\in C} C$ such that*

$$\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} S'_{\alpha\beta}.$$

The following proposition shows that, similarly to the situation with DAGs, the parents for each complex are all contained within the same tree node.

Proposition 5.26. *Let $\mathcal{G}$ be a chain graph and $\mathcal{T}$ a separation tree of $\mathcal{G}$. For any complex ρ in $\mathcal{G}$, there exists a tree-node $C \in \mathcal{C}$ such that $Pa(\rho) \subseteq C$.*

The proofs of Theorem 5.25 and Proposition 5.26 are given after the following example.

Example 5.27 (Chain Graph, Moral Graph, Separation Tree)**.**

Figure 5.7 (a) shows a chain graph, while (b) shows the moralised graph. Figure 5.8 shows a separation tree. The vertex set for the separation tree here is:

$$\mathcal{C} = \{\{A, B, C\}, \{B, C, D\}, \{C, D, E\}, \{D, E, F\}, \{E, I\}, \{I, J\}, \{D, F, G\}, \{F, G, K, H\}\}.$$

In this case, the separation tree is the junction tree corresponding to a triangulation of the moral graph, but a separation tree does not necessarily have to satisfy this property.

Lemma 5.28. *Let $\mathcal{G} = (V, E)$ be a chain graph and let $\alpha, \beta \in V$. There exists an edge $\alpha \sim \beta$ in E if and only if $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} S$ for any $S \subseteq V \backslash \{\alpha, \beta\}$.*

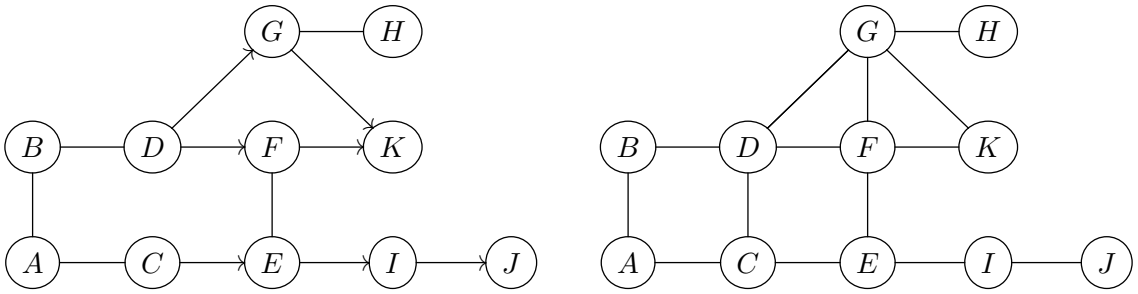

Figure 5.7: Chain Graph, Moralised Graph

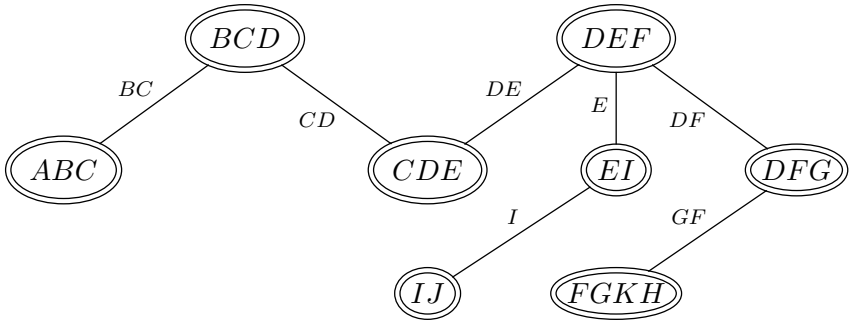

Figure 5.8: A Separation Tree for Figure 5.7

Proof If there is an edge $\alpha \sim \beta$, whether directed or undirected, then clearly $\alpha \not\perp\!\!\!\perp \beta \| S$ for any $S \subseteq V \backslash \{\alpha, \beta\}$. Let $\mathrm{Pa}(\alpha) = \{\gamma : (\gamma, \alpha) \in E\} \cup \{\gamma : \langle \gamma, \alpha \rangle \in E\}$. In other words, the *parents* of a node α are all nodes for which there is either a directed edge *from* the node to α or an *undirected* edge between the node and α.

Suppose there is no edge $\alpha \sim \beta$, then $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \mathrm{Pa}(\alpha)$ if β is an ancestor of α, $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \mathrm{Pa}(\beta)$ if α is an ancestor of β and both statements are true if α is not an ancestor of β and β is not an ancestor of α. □

Proof of Theorem 5.25 Clearly, if any of the three conditions hold, then there is a C-sep-set $S_{\alpha\beta}$ such that $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} S_{\alpha\beta}$.

Assume that, for a given separation tree $\mathcal{T}$, none of the conditions hold. That is, there exists an α, β and C such that $\alpha, \beta \in C$, there is a separator $S \subset C$ such that $\alpha, \beta \in S$ and for every subset S of either $\cup_{C:\alpha \in C} C$ or $\cup_{C:\beta \in C}$, $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} S$.

Note that, using the definitions from the proof of Lemma 5.28, if there is no edge $\alpha \sim \beta$, then either $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \mathrm{Pa}(\alpha)$ or $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \mathrm{Pa}(\beta)$ or both. Since $\mathrm{Pa}(\alpha) \subset \cup_{C:\alpha \in C} C$ and $\mathrm{Pa}(\beta) \subset \cup_{C:\beta \in C} C$, this is a contradiction, hence there is an edge $\alpha \sim \beta$ in $\mathcal{G}$, hence (by Lemma 5.28 there is no set R such that $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} R$ and the theorem is proved. □

Proof of Proposition 5.26 Suppose that α and β are parents of a complex $\kappa = (\alpha, \gamma_1, \ldots, \gamma_k, \beta)$ where $k \geq 1$. Suppose that for every tree-node $C \in \mathcal{C}$, $\{\alpha, \beta\} \cap C \neq \{\alpha, \beta\}$. Consider two tree-nodes C_1 and C_2 such that $\alpha \in C_1$ and $\beta \in C_2$. Let $C_1 - D_1 - \ldots, D_n - C_2$ denote the path in the tree from C_1 to C_2 and let $S = C_1 \cap D_1$. If $S \cap \{\alpha, \beta\} = \varnothing$, then $\{\gamma_1, \ldots, \gamma_k\} \cap S \neq \varnothing$. This implies that $\alpha \not\perp\!\!\!\perp \beta \| S$, since instantiation of any non-empty subset of the set $\{\gamma_1, \ldots, \gamma_k\}$ opens the connection. This contradicts the fact that S is a separator in the separation tree. It follows that either $\alpha \in S$ and hence $\alpha \in D_1$ or $\beta \in S$ and hence $\beta \in D_1$; hence, inductively, it follows that there is a tree-node C such that $\{\alpha, \beta\} \subseteq C$. □

Chapter 6

Evidence and Metrics

6.1 Probability Updates

Let $V = \{X_1, \ldots, X_d\}$ denote the set of random variables, $\mathcal{X}_j = (x_j^{(1)}, \ldots, x_j^{(k_j)})$ denote the state space for variable X_j and let $\mathcal{X} = \times_{j=1}^{d} \mathcal{X}_j$ denote the state space for the collection V. Let $\mathbb{P} : \mathcal{X} \to [0,1]$ denote the probability function of $(X_1, \ldots, X_d)$. Let $x = (x_1^{(i_1)}, \ldots, x_d^{(i_d)})$ denote an element of $\mathcal{X}$. For subsets $A \subseteq \mathcal{X}$, $\mathbb{P}(A)$ will be used to denote

$$\mathbb{P}(A) = \sum_{x \in A} \mathbb{P}(x).$$

The space $\mathcal{X}$ contains a finite number of elements and the event algebra $\mathcal{A}$ is simply the set of all possible subsets of $\mathcal{X}$. If an event $A \subset \mathcal{X}$ is observed, then the probability $\mathbb{P}$ is updated to a probability measure $\mathbb{P}^*$ using the definition of conditional probability

$$\mathbb{P}(B|A) = \mathbb{P}^*(B) = \frac{\mathbb{P}(AB)}{\mathbb{P}(A)}$$

to a probability function $\mathbb{P}^*$ over $\mathcal{X}$ that satisfies

$$\mathbb{P}^*(x) = \begin{cases} \frac{\mathbb{P}(x)}{\mathbb{P}(A)} & x \in A \\ 0 & x \notin A. \end{cases}$$

6.1.1 Jeffrey's Rule

There may be evidence that is not expressed in the form that an event $A \subseteq \mathcal{X}$ has occurred. It often happens in experimental settings that the probability space and event space are determined in advance and then information is acquired that is not of the form that an event, as a subset of the original probability space has occurred.

Jeffrey's rule is for the particular situation where the additional information leads to a re-assessment of the probabilities for a collection $(G_j)_{j=1}^{r}$ of mutually exclusive and exhaustive events from $\mathbb{P}(G_j)$ to $\mathbb{P}^*(G_j)$ and where it may be assumed that the conditional probabilities $\mathbb{P}(A|G_j)$ remain unaltered for $j = 1, \ldots, r$ and all $A \subseteq \mathcal{X}$.

Definition 6.1 (Jeffrey's Update). *The Jeffrey's rule for computing the update of the probability for any $A \subseteq \mathcal{X}$ is given by*

$$\mathbb{P}^*(A) = \sum_{j=1}^{r} \mathbb{P}^*(G_j)\mathbb{P}(A|G_j) \tag{6.1}$$

Let $\mathbb{P}(G_j) = \mu_j$ for $j = 1, \ldots, r$ and $\mathbb{P}^*(G_j) = \lambda_j$ for $j = 1, \ldots, r$. Then, for $x \in \mathcal{X}$,

$$\mathbb{P}^*(x) = \frac{\lambda_j}{\mu_j}\mathbb{P}(x) \qquad x \in G_j, \quad j = 1, \ldots, r. \tag{6.2}$$

The information leading to the update may be considered as an event Ξ such that $\Xi \notin \mathcal{X}$. The probability measure $\mathbb{P}$ is extended to acommodate the event Ξ in the following way: for the set of mutually exclusive and exhaustive events $G_1, \ldots, G_r$ and any $A \subseteq \mathcal{X}$, $\Xi \perp A|G_j$ for each $j = 1, \ldots, r$. The conditional probabilities of the events $G_1, \ldots, G_r$ given Ξ are specified as $\mathbb{P}(G_j|\Xi) = \lambda_j$. Then, for any $A \subseteq \mathcal{X}$, the probability update is

$$\widetilde{\mathbb{P}}(A) = \mathbb{P}(A|\Xi) = \sum_{j=1}^{r} \mathbb{P}(A|G_j, \Xi)\mathbb{P}(G_j|\Xi) = \sum_{j=1}^{r} \lambda_j \mathbb{P}(A|G_j).$$

Using $\mathbb{P}(A|G_j) = \frac{\mathbb{P}(A \cap G_j)}{\mathbb{P}(G_j)}$ and $\mu_j = \mathbb{P}(G_j)$, this gives

$$\widetilde{\mathbb{P}}(x) = \frac{\lambda_j}{\mu_j}\mathbb{P}(x) \qquad x \in G_j, \qquad j = 1, \ldots, r.$$

Pearl's Update Pearl's update is a re-expression of Jeffrey's update, where the information is presented in a slightly different format. Information received is that an event $\Xi \notin \mathcal{X}$ has happened, where $\Xi \perp A|G_j$ for each $j = 1, \ldots, r$, where $(G_j)_{j=1}^r$ are a set of mutually exclusive and exhaustive events. The information, though, is given in terms of likelihood ratios. Instead of $\lambda_j = \mathbb{P}(G_j|\Xi)$, the information is expressed as a collection of likelihood ratios $\rho_j = \frac{\mathbb{P}(\Xi|G_j)}{\mathbb{P}(\Xi|G_1)}$ for $j = 1, \ldots, r$, ratios of the likelihood of Ξ given G_j compared with the likelihood of Ξ given G_1. That is, λ_j represents the likelihood ratio that the event A occurs given that G_j occurs, compared with G_1. Note that $\lambda_1 = 1$. Using the same notation $\mu_j = \mathbb{P}(G_j)$, for any $A \subseteq \mathcal{X}$, an application of Bayes rule gives

$$\begin{aligned}
\widetilde{\mathbb{P}}(A) &= \mathbb{P}(A|\Xi) = \sum_{j=1}^{r} \mathbb{P}(A|G_j)\mathbb{P}(G_j|\Xi) \\
&= \sum_{j=1}^{r} \mathbb{P}(A|G_j)\frac{\mathbb{P}(\Xi|G_j)\mathbb{P}(G_j)}{\mathbb{P}(\Xi)} \\
&= \sum_{j=1}^{r} \mathbb{P}(A|G_j)\frac{\mathbb{P}(\Xi|G_j)\mathbb{P}(G_j)}{\sum_{k=1}^{r}\mathbb{P}(\Xi|G_k)\mathbb{P}(G_k)} \\
&= \sum_{j=1}^{r} \mathbb{P}(A|G_j)\frac{\rho_j\mu_j}{\sum_{k=1}^{r}\rho_j\mu_j}.
\end{aligned}$$

Definition 6.2 (Pearl's update). *Let $\mathbb{P}$ denote a probability distribution over $\mathcal{X}$ and let $G_1, \ldots, G_r$ be a mutually exclusive (that is $G_i \cap G_j = \phi$ for all $i \neq j$) and exhaustive (that is $\cup_{j=1}^{n} G_j = \mathcal{X}$) events, where*

$\mathbb{P}(G_j) = \mu_j$. *Let* $\rho_1 = 1$ *and let* ρ_j, $j = 2, \ldots, r$ *denote a collection of numbers. Then, for each* $x \in \mathcal{X}$, *the* Pearl update $\widetilde{\mathbb{P}}$ *is defined as*

$$\widetilde{\mathbb{P}}(x) = \mathbb{P}(x)\frac{\rho_j}{\sum_{j=1}^{r}\rho_j\mu_j} \qquad x \in G_j, \qquad j = 1, \ldots, r. \tag{6.3}$$

This is clearly a well defined probability function over $\mathcal{X}$. The numbers ρ_j are interpreted as likelihood ratios where Ξ is an event $\Xi \notin \mathcal{X}$ and $\mathbb{P}$ is extended to include Ξ such that $\Xi \perp A|G_j$ for each $j = 1, \ldots, r$, $A \subseteq \mathcal{X}$ and $\rho_j = \frac{\mathbb{P}(\Xi|G_j)}{\mathbb{P}(\Xi|G_1)}$.

Pearl's update and Jeffrey's rule are equivalent. The original probability space has been extended; information has been received of a form that cannot be expressed in terms of events, or subsets, of the original probability space.

Example 6.3.

A piece of cloth is to be sold on the market. The colour C is either green (c_g), blue (c_b) or violet (c_v). Tomorrow, the piece of cloth will either be sold (s) or not (s^c); this is denoted by the variable S. Experience gives the following probability distribution over C, S

$$\mathbb{P}_{C,S} = \begin{array}{c|ccc} S\backslash C & c_g & c_b & c_v \\ \hline s & 0.12 & 0.12 & 0.32 \\ s^c & 0.18 & 0.18 & 0.08 \end{array}$$

The marginal distribution over C is

$$\mathbb{P}_C = \begin{array}{ccc} c_g & c_b & c_v \\ \hline 0.3 & 0.3 & 0.4 \end{array}.$$

The piece of cloth is inspected by candle light. From the inspection by candle light, the probability over C is assessed as:

$$\mathbb{Q}_C = \begin{array}{ccc} c_g & c_b & c_v \\ \hline 0.7 & 0.25 & 0.05 \end{array}.$$

This is a situation where Jeffrey's rule may be used to update the probability.

$$\mathbb{Q}_{S,C} = \mathbb{Q}_C\mathbb{P}_{S|C} = \frac{\mathbb{Q}_C}{\mathbb{P}_C}\mathbb{P}_{S,C}.$$

This gives, for example,

$$\mathbb{Q}_{S,C}(s, c_g) = \frac{\lambda_g}{\mu_g}\mathbb{P}(s, c_g) = \frac{0.7}{0.3} \times 0.12 = 0.28.$$

Updating the whole distribution in this way gives

$$\mathbb{Q}_{C,S} = \begin{array}{c|ccc} S\backslash C & c_g & c_b & c_v \\ \hline s & 0.28 & 0.10 & 0.04 \\ s^c & 0.42 & 0.15 & 0.01 \end{array}$$

6.2 Evidence

For a Bayesian network, three different types of evidence will be discussed; *hard evidence*, *soft evidence* and *virtual evidence*. The definitions used are as follows:

Definition 6.4 (Hard Evidence, Soft Evidence, Virtual Evidence)**.** *The definitions are:*

- *A* hard finding *is an* instantiation, $\{X_i = x_i^{(l)}\}$ *for a particular value of* $i \in \{1, \ldots, d\}$ *and a particular value of* $l \in \{1, \ldots, k_i\}$. *This specifies that variable* X_i *is in state* $x_i^{(l)}$.
- Hard evidence *is a collection of hard findings.*
- *A* soft finding *on a variable* X_j *specifies the probability distribution of the variable* X_j. *That is, the conditional probability function* $\mathbb{P}_{X_j|Pa_j}$ *is replaced by a probability function* $\mathbb{P}^*_{X_j}$ *with domain* $\mathcal{X}_j$.
- Soft evidence *is a collection of soft findings.*
- *A* virtual finding *on variable* X_j *is a collection of values* $\{L(x_j^{(m)}),\ m = \{1, \ldots, k_j\}\}$ *such that the updated conditional probability function for* $X_j|Pa_j = \pi_j^{(n)}$ *is, for* $m = 1, \ldots, k_j$,

$$\mathbb{P}^*_{X_j|Pa_j}(x_j^{(m)}|\pi_j^{(n)}) = \frac{1}{\sum_{q=1}^{k_j} \mathbb{P}_{X_j|Pa_j}(x_j^{(q)}|\pi_j^{(n)})L(x_j^{(q)})} \mathbb{P}_{X_j|Pa_j}(x_j^{(m)}|\pi_j^{(n)})L(x_j^{(m)}). \tag{6.4}$$

- Virtual evidence *is a collection of virtual findings.*

Soft evidence and virtual evidence are different. When soft evidence is received on a variable, the links between the variable and its parents are severed; if soft evidence is received on variable X_j, then the conditional probability function $\mathbb{P}_{X_j|\mathrm{Pa}_j}$ is replaced by a new probability function $\mathbb{P}^*_{X_j}$.

Soft evidence basically applies to the situation described in the discussion of intervention calculus; it is assumed that the Bayesian network has been derived from causal principles, where the parents of a variable are direct causes. The soft evidence gives a new distribution over the variable, where the new distribution is not influenced by its parents. The state of the variable is forced, as in a controlled experiment, without reference to the other variables, while the new distribution $\mathbb{P}^*_{X_j}$ describes the probability of which state of X_j is enforced.

When virtual evidence is received, the links are preserved; the evidence is interpreted as an additional variable, which is instantiated.

6.3 Virtual Evidence

A virtual finding on variable X_j affects the probability $\mathbb{P}_{X_j|\mathrm{Pa}_j}$, without affecting any other conditional probabilities. The following discussion shows how to incorporate virtual evidence by extended the probability space by the addition of a *virtual* variable.

Virtual Evidence and the DAG The following shows how, in general, virtual evidence can be considered as an additional node E in the DAG. Consider a set of variables $V = \{X_1, \ldots, X_d\}$, where the joint probability distribution is factorised as

$$\mathbb{P}_{X_1,\ldots,X_d} = \prod_{j=1}^{d} \mathbb{P}_{X_j|\mathrm{Pa}_j}.$$

Suppose that virtual evidence is received on variable X_j. This may be expressed as a variable E and, by d-separation properties, the updated distribution $\mathbb{P}_{X_1,\ldots,X_d,E}$ has a factorisation

$$\mathbb{P}_{X_1,\ldots,X_d,E} = \left(\prod_k \mathbb{P}_{X_k|\mathrm{Pa}_k}\right) \mathbb{P}_{E|X_j}. \tag{6.5}$$

The variable E is a 'dummy variable', in the sense that its state space and distribution do not need to be defined; the virtual evidence is interpreted as a particular instantiation $\{E = e\}$ for this variable and this is the only information that is needed. From Equation (6.5),

$$\mathbb{P}_{X_1,\ldots,X_d|E}(.,\ldots,.|e) = \left(\prod_k \mathbb{P}_{X_k|\mathrm{Pa}_k}\right) \frac{\mathbb{P}_{E|X_j}(e|.)}{\mathbb{P}_E(e)}.$$

From Equation (6.4),

$$\frac{L(x_j^{(m)})}{\sum_{i=1}^{k_j} L(x_j^{(i)}) \mathbb{P}_{X_j|\mathrm{Pa}_j}(x_j^{(i)}|\pi_j^{(n)})} = \frac{\mathbb{P}_{E|X_j}(e|x_j^{(m)})}{\mathbb{P}_E(e)} \qquad m = 1, \ldots, k_j,$$

so that for m_1 and m_2 in $\{1, \ldots, k_j\}$,

$$\frac{L(x_j^{(m_1)})}{L(x_j^{(m_2)})} = \frac{\mathbb{P}_{E|X_j}(e|x_j^{(m_1)})}{\mathbb{P}_{E|X_j}(e|x_j^{(m_2)})}.$$

When applying virtual evidence, create an extra node on the network, with conditional probabilities $\mathbb{P}_{E|X_j}(1|x_j^{(m)}) \propto L(x_j^{(m)})$; any values satisfying $0 < \mathbb{P}_{E|X_j}(1|x_j^{(m)}) < 1$ for $L(x_j^{(m)}) > 0$ will suffice and $\mathbb{P}_{E|X_j}(0|x_j^{(m)}) = 1 - \mathbb{P}_{E|X_j}(1|x_j^{(m)})$. For a Bayesian networks programme, these values need to be defined, although the only conditional probability values used are those for $E = 1$. Then update the network with the hard evidence $E = 1$.

Equivalence with Pearl's Update Represented on a DAG, the virtual evidence node E satisfies $E \perp\!\!\!\perp V \backslash \{X_j\} \| _{\mathcal{G}} X_j$. The virtual evidence $\{E = e\}$ may be expressed as Pearl's update with $\Xi = \{E = e\}$ and the partition events $G_m = \{X_j = x_j^{(m)}\}$ for $m = 1, \ldots, k_j$. The collection $(G_m)_{j=1}^{k_j}$ are mutually exclusive and exhaustive events. Set $\rho_1 = 1$ and $\rho_m = \frac{\mathbb{P}_{E|X_j}(e|x_j^{(m)})}{\mathbb{P}_{E|X_j}(e|x_j^{(1)})}$ for $m = 2, \ldots, k_j$. Set

$$\mu_m = \mathbb{P}_{X_j}(x_j^{(m)}) \qquad m = 1, \ldots k_j.$$

Then, after extending $\mathbb{P}$ to accommodate the new variable E, the probability distribution $\mathbb{P}_{X_1,\ldots,X_d}$ is updated to $\widetilde{\mathbb{P}}_{X_1,\ldots,X_d} = \mathbb{P}_{X_1,\ldots,X_d|E}(.,\ldots,.|e)$ where

$$\widetilde{\mathbb{P}}_{X_1,\ldots,X_d}(x_1^{(i_1)}, \ldots, x_d^{(i_d)}) = \mathbb{P}_{X_1,\ldots,X_d}(x_1^{(i_1)}, \ldots, x_d^{(i_d)}) \frac{\rho_{i_j}}{\sum_{m=1}^{k_j} \mu_m \rho_m}.$$

Example 6.5.

Consider a DAG on five variables, X_1, X_2, X_3, X_4 and X_5, given in Figure 6.1. Suppose that a piece of virtual evidence is received on the variable X_3. This evidence may be modelled by a variable E, that is inserted to the DAG giving the DAG in Figure 6.2. The state of X_3 affects the virtual evidence that is observed.

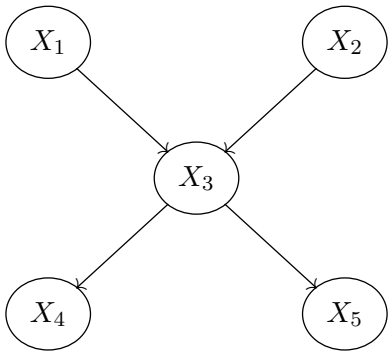

Figure 6.1: Before Virtual Evidence is Added

From Figure 6.2, it is clear that $(X_1, X_2, X_4, X_5) \perp\!\!\!\perp E \| _{\mathcal{G}} X_3$. The decomposition along the DAG gives $\mathbb{P}(E|X_1, X_2, X_3, X_4, X_5) = \mathbb{P}(E|X_3)$ and $\mathbb{P}(X_1, X_2, X_4, X_5|X_3, E) = \mathbb{P}(X_1, X_2, X_4, X_5|X_3)$. □

Example 6.6 (Burglary)**.**

Suppose that on any given day, there is a burglary at any given house with probability 10^{-4}. If there is a burglary, then the alarm will go off with probability 0.95; if there is no burglary, then it does not go off. One day, Professor Noddy receives a call from his neighbour Margarita, saying that she may have heard Professor Noddy's burglar alarm going off. Professor Noddy decides that it is four times more likely that Margarita did hear the alarm going off than that she was mistaken.

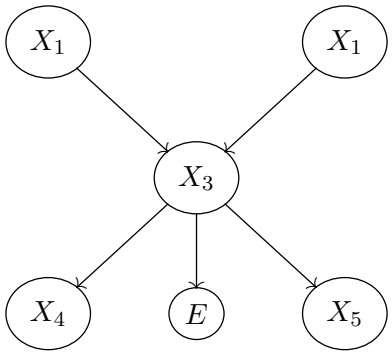

Figure 6.2: After the Virtual Evidence Node is Added

Let A take value 1 to denote the alarm going off and 0 otherwise, $B = 1$ to denote that a burglary takes place and 0 otherwise and let E denote the variable 'telephone call'; $E = 1$ is the evidence that Noddy received the call from Jemima. This evidence can be interpreted by extending $\mathbb{P}$ to include the variable E, where $B \perp E|A$ (the virtual evidence is received on A; B is the remainder of the network) and the relevant quantity is

$$\lambda = \frac{\mathbb{P}_{E|A}(1|1)}{\mathbb{P}_{E|A}(1|0)} = 4.$$

Then, the update of $\mathbb{P}_{B,A}$ requires $\mathbb{P}_A$. The conditional probabilities are

$$\mathbb{P}_B = \begin{array}{cc} 1 & 0 \\ 10^{-4} & 1-10^{-4} \end{array} \qquad \mathbb{P}_{A|B} = \begin{array}{c|cc} B\backslash A & 1 & 0 \\ \hline 1 & 0.95 & 0.05 \\ 0 & 0 & 1 \end{array}$$

Using $\mathbb{P}_{B,A} = \mathbb{P}_B \mathbb{P}_{A|B}$, the joint probabilities are

$$\mathbb{P}_{B,A} = \begin{array}{c|cc} B\backslash A & 1 & 0 \\ \hline 1 & 0.95\times 10^{-4} & 0.05\times 10^{-4} \\ 0 & 0 & 1-10^{-4} \end{array}$$

so that

$$\mathbb{P}_A = \begin{array}{cc} 1 & 0 \\ 0.95\times 10^{-4} & 1-0.95\times 10^{-4} \end{array}$$

and hence, using Pearl's update,

$$\widetilde{\mathbb{P}}_{B,A}(.,1) = \mathbb{P}_{B,A|E}(.,1|1) = \mathbb{P}_{B,A}(.,1)\frac{4}{4\times 0.95\times 10^{-4} + 1 - 0.95\times 10^{-4}}$$

$$\widetilde{\mathbb{P}}_{B,A}(.,0) = \mathbb{P}_{B,A|E}(.,0|1) = \mathbb{P}_{B,A}(.,0)\frac{1}{4\times 0.95\times 10^{-4} + 1 - 0.95\times 10^{-4}}$$

Exactly the same thing may be computed directly; using $\lambda_0 = 1$ and $\lambda_1 = 4$,

$$\begin{aligned} \widetilde{\mathbb{P}}_{B,A} &= \mathbb{P}_{B,A|E}(.,.|1) = \frac{\mathbb{P}_{B,A,E}(.,.,1)}{\mathbb{P}_E(1)} \\ &= \frac{\mathbb{P}_B \mathbb{P}_{A|B} \mathbb{P}_{E|A}(1|.)}{\mathbb{P}_{E|A}(1|1)\mathbb{P}_A(1) + \mathbb{P}_{E|A}(1|0)\mathbb{P}_A(0)} = \mathbb{P}_B \mathbb{P}_{A|B} \frac{\lambda_a}{\lambda_1 \mathbb{P}_A(1) + \lambda_0 \mathbb{P}_A(0)} \end{aligned}$$

where a denotes the value taken by variable A.
It follows that

$$\widetilde{\mathbb{P}}_B(1) = \widetilde{\mathbb{P}}_{B,A}(1,1) + \widetilde{\mathbb{P}}_{B,A}(1,0) = 10^{-4} \times \left(\frac{3.80 + 0.05}{1 + 3.85\times 10^{-4}}\right) \simeq 3.85\times 10^{-4}.$$

6.4 Measures of Divergence between Probability Distributions

A *distance* is a more specific measure of divergence, which satisfies the properties given in the following definition.

Definition 6.7 (Distance)**.** *A measure of divergence D between probability distributions is a* distance *if it satisfies the following three properties: for any three probability distributions $\mathbb{P}_1, \mathbb{P}_2$ and $\mathbb{P}_3$ over the same space $\mathcal{X} = (x_1, \ldots, x_k)$,*

- *Positivity: $D(\mathbb{P}_1, \mathbb{P}_2) \geq 0$. Furthermore, $D(\mathbb{P}_1, \mathbb{P}_2) = 0 \Leftrightarrow \mathbb{P}_1 \equiv \mathbb{P}_2$*
- *Symmetry: $D(\mathbb{P}_1, \mathbb{P}_2) = D(\mathbb{P}_2, \mathbb{P}_1)$*
- *Triangle Inequality: $D(\mathbb{P}_1, \mathbb{P}_3) \leq D(\mathbb{P}_1, \mathbb{P}_2) + D(\mathbb{P}_2, \mathbb{P}_3)$.*

Consider two common measures of divergence between probability distributions. Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability functions over the same finite state space $\mathcal{X} = (x_1, \ldots, x_k)$ and let $p_j = \mathbb{P}(x_j)$ and $q_j = \mathbb{Q}(x_j)$ for $j = 1, \ldots, k$.

Definition 6.8 (Euclidean Distance)**.** *The* quadratic *or* Euclidean distance *is defined as*

$$D_2(\mathbb{P}, \mathbb{Q}) = \sqrt{\sum_{j=1}^{k} (p_j - q_j)^2}.$$

Definition 6.9 (Kullback Leibler Divergence)**.** *The* Kullback Leibler divergence *between two probability distributions $\mathbb{P}$ and $\mathbb{Q}$ over the same state space $\mathcal{X}$ is defined as*

$$D_{KL}(\mathbb{P} \| \mathbb{Q}) = \sum_{j=1}^{k} p_j \ln \frac{p_j}{q_j}.$$

The Kullback Leibler divergence is non negative (left as an exercise) and $D_{KL}(\mathbb{P} \| \mathbb{Q}) = 0 \Leftrightarrow \mathbb{P} \equiv \mathbb{Q}$, but it is not a *distance* in the sense of Definition 6.7; it does not, in general, satisfy $D_{KL}(\mathbb{P} \| \mathbb{Q}) = D_{KL}(\mathbb{Q} \| \mathbb{P})$.

Example 6.10.

Let

$$\mathbb{P}^{(1)} = (0.02, 0.98),\ \mathbb{Q}^{(1)} = (0.0364, 0.9636), \qquad \mathbb{P}^{(2)} = (0.01, 0.99),\ \mathbb{Q}^{(2)} = (0.00471, 0.99529).$$

Then

$$D_2(\mathbb{P}^{(1)}, \mathbb{Q}^{(1)}) = \sqrt{(0.02 - 0.0364)^2 + (0.98 - 0.9636)^2} = 0.0232$$
$$D_2(\mathbb{P}^{(2)}, \mathbb{Q}^{(2)}) = \sqrt{(0.00471 - 0.01)^2 + (0.99529 - 0.99)^2} = 0.00748,$$

so the change represented by the second adjustment is less than one third of the change represented by the first if the change is measured using the quadratic distance measure. For the Kullback-Leibler,

$$D_{KL}(\mathbb{P}^{(1)}\|\mathbb{Q}^{(1)}) = 0.02\ln\frac{0.02}{0.0364} + 0.98\ln\frac{0.98}{0.9636} = 0.004562,$$
$$D_{KL}(\mathbb{P}^{(2)}\|\mathbb{Q}^{(2)}) = 0.01\ln\frac{0.01}{0.00471} + 0.99\ln\frac{0.99}{0.99529} = 0.00225,$$

so the change represented by the second adjustment is approximately one half of the change represented by the first. Clearly, different distance measures give different impressions of the relative importance of parameter changes.

6.5 The Chan - Darwiche Distance Measure

The problem with *both* the Kullback Leibler and the Quadratic distance measure is that they do not emphasise the *proportional* difference between two probability values when they are close to zero. The following distance measure was proposed by Chan and Darwiche. It will be seen that it is particularly useful when comparison of odds ratios are in view.

Definition 6.11 (Chan - Darwiche Distance)**.** *Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability functions over a finite state space $\mathcal{X}$. That is, $\mathbb{P}:\mathcal{X}\to[0,1]$ and $\mathbb{Q}:\mathcal{X}\to[0,1]$, $\sum_{x\in\mathcal{X}}\mathbb{P}(x)=1$ and $\sum_{x\in\mathcal{X}}\mathbb{Q}(x)=1$. The Chan - Darwiche distance is defined as*

$$D_{CD}(\mathbb{P},\mathbb{Q}) = \ln\max_{x\in\mathcal{X}}\frac{\mathbb{Q}(x)}{\mathbb{P}(x)} - \ln\min_{x\in\mathcal{X}}\frac{\mathbb{Q}(x)}{\mathbb{P}(x)},$$

where, by definition, $\frac{0}{0}=1$.

Unlike the Kullback - Leibler divergence, the Chan - Darwiche distance is a *distance*; it satisfies the three requirements of Definition 6.7. This result is stated in Theorem 6.13.

The *support* of a probability function defined on a finite state space; namely, those points where it is strictly positive (relating to outcomes that can happen) is important when comparing two different probability functions over the same state space.

Definition 6.12 (Support)**.** *Let $\mathbb{P}$ be a probability function over a countable state space $\mathcal{X}$; that is, $\mathbb{P}:\mathcal{X}\to[0,1]$ and $\sum_{x\in\mathcal{X}}\mathbb{P}(x)=1$. The* support *of $\mathbb{P}$ is defined as the subset $\mathcal{S}_{\mathbb{P}}\subseteq\mathcal{X}$ such that*

$$\mathcal{S}_{\mathbb{P}} = \{x\in\mathcal{X}|\mathbb{P}(x)>0\}. \tag{6.6}$$

Theorem 6.13. *The Chan - Darwiche distance measure is a distance measure, in the sense that for any three probability functions $\mathbb{P}_1$, $\mathbb{P}_2$, $\mathbb{P}_3$ over a state space $\mathcal{X}$, the following three properties hold:*

- *Positivity: $D_{CD}(\mathbb{P}_1,\mathbb{P}_2)\geq 0$ and $D_{CD}(\mathbb{P}_1,\mathbb{P}_2)=0\Leftrightarrow\mathbb{P}_1\equiv\mathbb{P}_2$.*
- *Symmetry: $D_{CD}(\mathbb{P}_1,\mathbb{P}_2)=D_{CD}(\mathbb{P}_2,\mathbb{P}_1)$*
- *Triangle Inequality: $D_{CD}(\mathbb{P}_1,\mathbb{P}_2)+D_{CD}(\mathbb{P}_2,\mathbb{P}_3)\geq D_{CD}(\mathbb{P}_1,\mathbb{P}_3)$.*

Proof Positivity and symmetry are clear and are left as exercises. It only remains to prove the triangle inequality. Since the state space is discrete and *finite*, it follows that there exist $y, z \in \mathcal{X}$ such that

$$\begin{aligned}
D_{CD}(\mathbb{P}_1, \mathbb{P}_3) &= \ln \max_{x\in\mathcal{X}} \frac{\mathbb{P}_3(x)}{\mathbb{P}_1(x)} - \ln \min_{x\in\mathcal{X}} \frac{\mathbb{P}_3(x)}{\mathbb{P}_1(x)} = \ln \frac{\mathbb{P}_3(y)}{\mathbb{P}_1(y)} - \ln \frac{\mathbb{P}_3(z)}{\mathbb{P}_1(z)} \\
&= \ln \frac{\mathbb{P}_3(y)}{\mathbb{P}_2(y)} + \ln \frac{\mathbb{P}_2(y)}{\mathbb{P}_1(y)} - \ln \frac{\mathbb{P}_3(z)}{\mathbb{P}_2(z)} - \ln \frac{\mathbb{P}_2(z)}{\mathbb{P}_1(z)} \\
&= \left(\ln \frac{\mathbb{P}_3(y)}{\mathbb{P}_2(y)} - \ln \frac{\mathbb{P}_3(z)}{\mathbb{P}_2(z)} \right) + \left(\ln \frac{\mathbb{P}_2(y)}{\mathbb{P}_1(y)} - \ln \frac{\mathbb{P}_2(z)}{\mathbb{P}_1(z)} \right) \\
&\leq \left(\ln \max_{x\in\mathcal{X}} \frac{\mathbb{P}_3(x)}{\mathbb{P}_2(x)} - \ln \min_{x\in\mathcal{X}} \frac{\mathbb{P}_3(x)}{\mathbb{P}_2(x)} \right) + \left(\ln \max_{x\in\mathcal{X}} \frac{\mathbb{P}_2(x)}{\mathbb{P}_1(x)} - \ln \min_{x\in\mathcal{X}} \frac{\mathbb{P}_2(x)}{\mathbb{P}_1(x)} \right) \\
&= D_{CD}(\mathbb{P}_1, \mathbb{P}_2) + D_{CD}(\mathbb{P}_2, \mathbb{P}_3).
\end{aligned}$$

□

This distance is relatively easy to compute. It has the advantage over the Kullback Leibler divergence (which is not a true distance measure) that it may be used to obtain bounds on *odds ratios*.

Definition 6.14 (Odds)**.** *Let $\mathbb{P}$ be a probability measure over $\mathcal{X}$ and let $A \subset \mathcal{X}$ and $B \subset \mathcal{X}$. The* odds *for A versus A^c given B is defined as*

$$O_{\mathbb{P}}(A|B) = \frac{\mathbb{P}(A|B)}{\mathbb{P}(A^c|B)}.$$

Comparison with the Kullback Leibler Divergence and Euclidean Distance Consider two probability distributions $\mathbb{P} = (p_1, p_2, p_3)$ and $\mathbb{Q} = (q_1, q_2, q_3)$ over $\{1, 2, 3\}$ defined by

$$p_1 = a, \qquad p_2 = b - a, \qquad p_3 = 1 - b$$

$$q_1 = ka, \qquad q_2 = b - ka, \qquad q_3 = 1 - b$$

Then

$$D_{KL}(\mathbb{P}\|\mathbb{Q}) = -a \ln k - (b - a) \ln \frac{b - ka}{b - a}.$$

Consider the events $A = \{1\}$, $B = \{1, 2\}$, then $O_{\mathbb{P}}(A|B) = \frac{a}{b-a}$ and $O_{\mathbb{Q}}(A|B) = \frac{ka}{b-ka}$ and the odds ratio is given by

$$\frac{O_{\mathbb{Q}}(A|B)}{O_{\mathbb{P}}(A|B)} = \frac{k(b - a)}{b - ka}.$$

As $a \to 0$, $D_{KL}(\mathbb{P}\|\mathbb{Q}) \to 0$, while $\frac{O_{\mathbb{Q}}(A|B)}{O_{\mathbb{P}}(A|B)} \to k$. It is therefore not possible to find a bound on the odds ratio in terms of the Kullback Leibler divergence.

Similarly, in this example, the Euclidean distance is

$$D_2(\mathbb{P}, \mathbb{Q}) = \sqrt{2}a(1 - k) \xrightarrow{a\to 0} 0,$$

while

$$D_{CD}(\mathbb{P},\mathbb{Q}) = \ln\max\left(\frac{1}{k},\frac{b-a}{b-ka},1\right) - \ln\min\left(\frac{1}{k},\frac{b-a}{b-ka},1\right) \xrightarrow{a\to 0} \ln k.$$

Neither the *Kullback Leibler* divergence nor the *Euclidean* distance can be used to provide uniform bounds on the odds ratios; even if there is a large *relative* difference between pairs of probability values for $\mathbb{P}$ and $\mathbb{Q}$, they will be *ignored* if the absolute values of these probabilities are small.
The Chan Darwiche distance measure is useful, because it can be used to obtain sharp bounds on the way that odds change as the probability distribution changes.

Theorem 6.15. *Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability distributions over the same finite state space $\mathcal{X}$ and let A and B be two subsets of $\mathcal{X}$. Let $A^c = \mathcal{X}\backslash A$ and $B^c = \mathcal{X}\backslash B$. Let $O_{\mathbb{P}}(A|B) = \frac{\mathbb{P}(A|B)}{\mathbb{P}(A^c|B)}$ and $O_{\mathbb{Q}}(A|B) = \frac{\mathbb{Q}(A|B)}{\mathbb{Q}(A^c|B)}$. Then*

$$e^{-D_{CD}(\mathbb{P},\mathbb{Q})} \le \frac{O_{\mathbb{Q}}(A|B)}{O_{\mathbb{P}}(A|B)} \le e^{D_{CD}(\mathbb{P},\mathbb{Q})}.$$

The bound is sharp *in the sense that for any pair of distributions $(\mathbb{P},\mathbb{Q})$ there are subsets A and B of $\mathcal{X}$ such that*

$$\frac{O_{\mathbb{Q}}(A|B)}{O_{\mathbb{P}}(A|B)} = \exp\{D_{CD}(\mathbb{P},\mathbb{Q})\}, \qquad \frac{O_{\mathbb{Q}}(A^c|B)}{O_{\mathbb{P}}(A^c|B)} = \exp\{-D_{CD}(\mathbb{P},\mathbb{Q})\}.$$

Proof of Theorem 6.15 Without loss of generality, it may be assumed that $\mathbb{P}$ and $\mathbb{Q}$ have the same support; that is, $\mathbb{P}(x) > 0 \Leftrightarrow \mathbb{Q}(x) > 0$. Otherwise $D_{CD}(\mathbb{P},\mathbb{Q}) = +\infty$ and the statement is trivially true; for any $A, B \subseteq \mathcal{X}$, $0 \le \frac{O_{\mathbb{Q}}(A|B)}{O_{\mathbb{P}}(A|B)} \le +\infty$. For $\mathbb{P}$ and $\mathbb{Q}$ such that $\mathbb{P}$ and $\mathbb{Q}$ have the same support, let $r(x) = \frac{\mathbb{Q}(x)}{\mathbb{P}(x)}$. For any two subsets $A, B \subseteq \mathcal{X}$,

$$\begin{aligned}
\frac{O_{\mathbb{Q}}(A|B)}{O_{\mathbb{P}}(A|B)} &= \frac{\mathbb{Q}(A|B)}{1-\mathbb{Q}(A|B)}\frac{1-\mathbb{P}(A|B)}{\mathbb{P}(A|B)} = \frac{\mathbb{Q}(AB)}{\mathbb{Q}(A^cB)}\frac{\mathbb{P}(A^cB)}{\mathbb{P}(AB)} = \frac{\sum_{x\in AB}\mathbb{Q}(x)}{\sum_{x\in A^cB}\mathbb{Q}(x)}\frac{\sum_{x\in A^cB}\mathbb{P}(x)}{\sum_{x\in AB}\mathbb{P}(x)} \\
&= \frac{\sum_{x\in AB} r(x)\mathbb{P}(x)}{\sum_{x\in A^cB} r(x)\mathbb{P}(x)}\frac{\sum_{x\in A^cB}\mathbb{P}(x)}{\sum_{x\in AB}\mathbb{P}(x)} \le \frac{\max_{z\in\mathcal{X}} r(z)\sum_{x\in AB}\mathbb{P}(x)}{\min_{z\in\mathcal{X}} r(z)\sum_{x\in A^cB}\mathbb{P}(x)}\frac{\sum_{x\in A^cB}\mathbb{P}(x)}{\sum_{x\in AB}\mathbb{P}(x)} \\
&= \frac{\max_{z\in\mathcal{X}} r(z)}{\min_{z\in\mathcal{X}} r(z)}.
\end{aligned}$$

Similarly,

$$\frac{O_{\mathbb{Q}}(A|B)}{O_{\mathbb{P}}(A|B)} \ge \frac{\min_{z\in\mathcal{X}} r(z)}{\max_{z\in\mathcal{X}} r(z)}.$$

From the definition of $D_{CD}(\mathbb{P},\mathbb{Q})$, it follows directly that

$$e^{D_{CD}(\mathbb{P},\mathbb{Q})} = \frac{\max_{z\in\mathcal{X}} r(z)}{\min_{z\in\mathcal{X}} r(z)},$$

hence

$$e^{-D_{CD}(\mathbb{P},\mathbb{Q})} \le \frac{O_{\mathbb{Q}}(A|B)}{O_{\mathbb{P}}(A|B)} \le e^{D_{CD}(\mathbb{P},\mathbb{Q})},$$

as required, thus proving the first part. □

To prove that the bound is tight, consider x such that $r(x) = \max_{z\in\mathcal{X}} r(z)$ and y such that $r(y) = \min_{z\in\mathcal{X}} r(z)$. Set $A = \{x\}$ and $B = \{x, y\}$. Then

$$O_{\mathbb{Q}}(A|B) = \frac{r(x)\mathbb{P}(x)}{r(y)\mathbb{P}(y)}.$$

Since $O_{\mathbb{P}}(A|B) = \frac{\mathbb{P}(x)}{\mathbb{P}(y)}$ and $e^{D_{CD}(\mathbb{P},\mathbb{Q})} = \frac{\max_{z\in\mathcal{X}} r(z)}{\min_{z\in\mathcal{X}} r(z)}$, it follows that

$$\frac{O_{\mathbb{Q}}(A|B)}{O_{\mathbb{P}}(A|B)} = e^{D_{CD}(\mathbb{P},\mathbb{Q})}.$$

Similarly, let $C = \{y\}$, then

$$\frac{O_{\mathbb{Q}}(C|B)}{O_{\mathbb{P}}(C|B)} = e^{-D_{CD}(\mathbb{P},\mathbb{Q})}.$$

□

Theorem 6.15 may be used to obtain bounds on arbitrary queries $\mathbb{Q}(A|B)$ for the measure $\mathbb{Q}$ in terms of $\mathbb{P}(A|B)$.

Corollary 6.16. *Set $d = D_{CD}(\mathbb{P}, \mathbb{Q})$, then*

$$\frac{\mathbb{P}(A|B)e^{-d}}{1 + (e^{-d} - 1)\mathbb{P}(A|B)} \leq \mathbb{Q}(A|B) \leq \frac{\mathbb{P}(A|B)e^{d}}{1 + (e^{d} - 1)\mathbb{P}(A|B)}. \tag{6.7}$$

Proof Equation (6.7) is a straight forward consequence of Theorem 6.15. The computation is left as an exercise. □

6.5.1 Soft Evidence and Virtual Evidence

Jeffrey's Rule Let $\mathbb{P}$ denote a probability distribution over a finite state space $\mathcal{X}$ and let $\mathbb{Q}$ denote the distribution obtained by updating according to Jeffrey's rule. The following formula may be established.

Theorem 6.17. *Let $\mathbb{P}$ be a probability distribution over a countable state space $\mathcal{X}$ and let $G_1, \ldots, G_n$ be a collection of mutually exclusive and exhaustive events. Let $\lambda_j = \mathbb{P}(G_j)$ for $j = 1, \ldots, n$. Let $\mathbb{Q}$ denote the probability distribution such that $\mathbb{Q}(G_j) = \mu_j$ for $j = 1, \ldots, n$ and such that for all $x \in \mathcal{X}$*

$$\mathbb{Q}(x) = \frac{\mu_j}{\lambda_j}\mathbb{P}(x) \qquad x \in G_j.$$

In other words, $\mathbb{Q}$ is the Jeffrey's update of $\mathbb{P}$, defined by $\mathbb{Q}(G_j) = \mu_j$, $j = 1, \ldots, n$. Then

$$D_{CD}(\mathbb{P}, \mathbb{Q}) = \ln \max_j \frac{\lambda_j}{\mu_j} - \ln \min_j \frac{\lambda_j}{\mu_j}.$$

Proof This follows directly and is left as an exercise. □

This immediately gives the following bound.

Corollary 6.18. *Let $O_{\mathbb{P}}$ and $O_{\mathbb{Q}}$ denote the odds function before and after applying Jeffreys rule. Let*

$$d = \ln \max_j \frac{\lambda_j}{\mu_j} - \ln \min_j \frac{\lambda_j}{\mu_j}.$$

Then for any two events A and B,

$$e^{-d} \leq \frac{O_{\mathbb{P}}(A|B)}{O_{\mathbb{Q}}(A|B)} \leq e^d.$$

Proof This follows directly. □

Under the Chan - Darwiche distance measure, Jeffrey's rule may be considered *optimal*, in the following sense.

Theorem 6.19. *Let $\mathbb{P}$ denote a probability distribution over $\mathcal{X}$ and let $G_1, \ldots, G_r$ denote a collection of mutually exclusive and exhaustive events. Let $\mu_j = \mathbb{P}(G_j)$, let $\lambda_1, \ldots, \lambda_r$ be a collection of non negative numbers such that $\sum_{j=1}^r \lambda_j = 1$ and let $\mathbb{Q}$ be the probability distribution over $\mathcal{X}$ defined by*

$$\mathbb{Q}(x) = \frac{\lambda_j}{\mu_j}\mathbb{P}(x) \qquad x \in G_j.$$

Then $D_{CD}(\mathbb{P}, \mathbb{Q})$ minimises $D_{CD}(\mathbb{P}, \mathbb{R})$ subject to the constraint that $\mathbb{R}$ is a probability distribution over $\mathcal{X}$ such that $\mathbb{R}(G_i) = \lambda_i$ for $i = 1, \ldots, r$.

Proof Let $\mathbb{Q}$ denote the distribution generated by Jeffrey's rule and let $\mathbb{R}$ be any distribution that satisfies the constraint $\mathbb{R}(G_j) = \mathbb{Q}(G_j) = \lambda_j$, $j = 1, \ldots, \mathbb{R}$. If $\mathbb{P}$ and $\mathbb{R}$ do not have the same support (Definition 6.12), then $+\infty = D_{CD}(\mathbb{P}, \mathbb{R}) \geq D_{CD}(\mathbb{P}, \mathbb{Q})$. If they have the same support, let j denote the value such that $\frac{\lambda_j}{\mu_j} = \max_i \frac{\lambda_i}{\mu_i}$ and let k denote the value such that $\frac{\lambda_k}{\mu_k} = \min_i \frac{\lambda_i}{\mu_i}$. Let $\alpha = \max_{x \in \mathcal{X}} \frac{\mathbb{R}(x)}{\mathbb{P}(x)}$. Then

$$\alpha\mu_j = \alpha \sum_{x \in G_j} \mathbb{P}(x) \geq \sum_{x \in G_j} \frac{\mathbb{R}(x)}{\mathbb{P}(x)}\mathbb{P}(x) = \mathbb{R}(G_j) = \lambda_j,$$

so that

$$\alpha \geq \frac{\lambda_j}{\mu_j}.$$

Set $\beta = \min_{x \in \mathcal{X}} \frac{\mathbb{R}(x)}{\mathbb{P}(x)}$, then a similar argument gives $\beta \leq \frac{\lambda_k}{\mu_k}$. It follows that the distance between $\mathbb{P}$ and $\mathbb{R}$ is

$$\begin{aligned} D_{CD}(\mathbb{P}, \mathbb{R}) &= \ln \max_{x \in \mathcal{X}} \frac{\mathbb{R}(x)}{\mathbb{P}(x)} - \ln \min_{x \in \mathcal{X}} \frac{\mathbb{R}(x)}{\mathbb{P}(x)} = \ln \alpha - \ln \beta \\ &\geq \ln \frac{\lambda_j}{\mu_j} - \ln \frac{\lambda_j}{\mu_j} = \ln \max_i \frac{\lambda_i}{\mu_i} - \ln \min_i \frac{\lambda_i}{\mu_i} = D_{CD}(\mathbb{P}, \mathbb{Q}). \end{aligned}$$

Therefore $\mathbb{Q}$ gives the smallest distance. □

Pearl's Method of Virtual Evidence Recall Pearl's Method of Virtual Evidence. The CD distance between the original distribution and the updated distribution has a convenient expression.

Theorem 6.20. *Let $\mathbb{P}$ be a probability distribution over a finite state space $\mathcal{X}$ and let $\lambda_1 = 1$ and $\lambda_2, \ldots, \lambda_r$ be positive numbers. Let $G_1, \ldots, G_r$ be a collection of mutually exclusive and exhaustive subsets of $\mathcal{X}$. Let $\mu_j = \sum_{x \in G_j} \mathbb{P}(x)$ for $j = 1, \ldots, r$. Let $\mathbb{Q}$ be defined as*

$$\mathbb{Q}(x) = \mathbb{P}(x) \frac{\lambda_j}{\sum_{k=1}^{r} \mu_k \lambda_k} \qquad x \in G_j.$$

Then $\mathbb{Q}$ is a probability distribution over $\mathcal{X}$ and

$$D_{CD}(\mathbb{P}, \mathbb{Q}) = \ln \max_i \lambda_i - \ln \min_i \lambda_i.$$

Proof Firstly, it is clear from the construction that $\sum_{x \in \mathcal{X}} \mathbb{Q}(x) = 1$ and that $\mathbb{Q}(x) \geq 0$ for all $x \in \mathcal{X}$, so that $\mathbb{Q}$ is a probability function. From the definition,

$$\frac{\mathbb{Q}(x)}{\mathbb{P}(x)} = \frac{\lambda_j}{\sum_k \mu_k \lambda_k} \qquad x \in G_j.$$

It follows that

$$\begin{aligned} D_{CD}(\mathbb{P}, \mathbb{Q}) &= \ln \max_{x \in \mathcal{X}} \frac{\mathbb{Q}(x)}{\mathbb{P}(x)} - \ln \min_{x \in \mathcal{X}} \frac{\mathbb{Q}(x)}{\mathbb{P}(x)} \\ &= \ln \max_j \frac{\lambda_j}{\sum_k \mu_k \lambda_k} - \ln \min_j \frac{\lambda_j}{\sum_k \mu_k \lambda_k} \\ &= \ln \max_j \lambda_j - \ln \min_j \lambda_j \end{aligned}$$

as required. □

This immediately gives the following bound.

Corollary 6.21. *Let $O_{\mathbb{Q}}$ and $O_{\mathbb{P}}$ denote the odds functions associated with the probability measures defined in Theorem 6.20 and let*

$$d = D_{CD}(\mathbb{P}, \mathbb{Q}) = \ln \max_i \lambda_i - \ln \min_i \lambda_i.$$

Then for any events $A, B \subseteq \mathcal{X}$,

$$e^{-d} \leq \frac{O_{\mathbb{Q}}(A|B)}{O_{\mathbb{P}}(A|B)} \leq e^{d}.$$

Proof This follows directly and is left as an exercise. □

Example 6.22.

The 'Burglary' example may be developed to illustrate these results. Let A denote the event that the alarm goes off, B the event that a burglary takes place and let E denote the evidence of the telephone call from Jemima. According to Pearl's method, this evidence can be interpreted as

$$\lambda = \frac{\mathbb{P}_{E|A}(1|1)}{\mathbb{P}_{E|A}(1|0)} = 4.$$

Therefore, the distance between the original distribution $\mathbb{P}$ and the update $\mathbb{Q}(.) = \mathbb{P}(.|E = 1)$ derived according to Pearl's method is $D_{CD}(\mathbb{P}, \mathbb{Q}) = \ln 4 \simeq 1.386$. This distance may be used to bound $\mathbb{Q}_B(1)$, the probability of a Burglary, after the update to incorporate the evidence. Using the bound stated in the corollary,

$$\frac{\mathbb{P}_B(1)e^{-d}}{1 + (e^{-d} - 1)\mathbb{P}_B(1)} \leq \mathbb{Q}_B(1) \leq \frac{\mathbb{P}_B(1)e^{d}}{1 + (e^{d} - 1)\mathbb{P}_B(1)},$$

so that $2.50 \times 10^{-5} \leq \mathbb{Q}_B(1) \leq 4.00 \times 10^{-4}$. An application of Pearl's virtual evidence rule gives $\mathbb{Q}_B(1) = 3.85 \times 10^{-4}$. □

Notes The article [37] discusses probability updates when the information received does not fit into the framework of the standard definition. The Chan Darwiche distance measure is proposed in [20]. The article [22] by Chan and Darwiche discusses the application of Jeffrey's update rule and Pearl's method to virtual evidence. These two articles provide the basis for the chapter.

6.6 Exercises

1. **Jeffrey's Rule** In a certain country, people use only two car models, Volvo and Saab, which come in two colours, red and blue. The sales statistics suggest $\mathbb{P}(\text{Volvo}) = \mathbb{P}(\text{Saab}) = 1/2$. Furthermore, $\mathbb{P}(\text{red}|\text{Volvo}) = 0.7$ and $\mathbb{P}(\text{red}|\text{Saab}) = 0.2$. You are on holiday in this region and you are standing outside a large underground garage, which you may not enter. The attendant of the garage communicates his impression that 40% of the cars in the garage are red. What is the probability that the first car leaving the garage is a Volvo?

2. **Pearl's Method** The two parts of this question are virtually identical.

   (a) Let A denote an event that gives uncertain information (or virtual / soft evidence) about the partition (that is a collection of mutually exclusive and exhaustive events) $\{G_j\}_{j=1}^n$. Suppose that A satisfies

   $$\mathbb{P}(A \mid G_j, B) = \mathbb{P}(A \mid G_j), \quad j = 1, 2, \ldots, n$$

   for *every* event B. This is an assumption of conditional independence; the event A is independent of all other events given the partition G_j. Set $\lambda_j = \mathbb{P}(A|G_j)$ and show that for any event B,

   $$\mathbb{P}(B \mid A) = \frac{\sum_{j=1}^n \lambda_j \mathbb{P}(B \cap G_j)}{\sum_{j=1}^n \lambda_j \mathbb{P}(G_j)}.$$

   Check that $\mathbb{P}(.|A)$ satisfies the definition of the Pearl update (Definition 6.2).

   (b) Let $\mathbb{P}$ denote a probability distribution before evidence is obtained and suppose that a piece of evidence Ξ gives uncertain information about the partition (that is, the collection of mutually exclusive and exhaustive events) $\{G_j\}_{j=1}^n$. Suppose that Ξ is not in the original event space and that for any event A in the original event space, $\Xi \perp A|G_j$ for each $j = 1, \ldots, n$. Suppose that this evidence is specified by the posterior probabilities

   $$\mathbb{P}^*(G_j) = \mathbb{P}(G_j|\Xi) = q_j, \quad j = 1, 2, \ldots, n.$$

   Let

   $$\rho_j = \frac{\mathbb{P}(\Xi|G_j)}{\mathbb{P}(\Xi|G_1)} \qquad j = 1, 2, \ldots, n$$

   and

   $$\lambda_j = \frac{q_j}{\mathbb{P}(G_j)}, \quad j = 1, 2, \ldots, n.$$

   For any event C, compute the probability $\mathbb{P}(C|\Xi)$ obtained by Pearl's method of virtual evidence and show that this gives the same result as Jeffrey's rule of update.

3. Let X_1, X_2, X_3 be three binary random variables, each taking values in $\{0,1\}$, such that

$$\mathbb{P}_{X_1,X_2,X_3}(x_1,x_2,x_3) = \frac{1}{8},$$

for $(x_1,x_2,x_3) \in \{0,1\}^3$.

Now let V be an additional binary random variable and let $E = \{V = 1\}$. Here V stands for virtual information. Suppose that the conditional probability function of V given X_3 satisfies

$$\mathbb{P}_{V|X_3}(1 \mid 1) = \lambda \mathbb{P}_{V|X_3}(1 \mid 0).$$

Let G_1 and G_2 be the two events

$$G_1 = \left\{(x_1,x_2,x_3) \in \{0,1\}^3 \mid x_3 = 0\right\}$$

and

$$G_2 = \left\{(x_1,x_2,x_3) \in \{0,1\}^3 \mid x_3 = 1\right\}.$$

The events G_1 and G_2 are mutually exclusive and exhaustive. Use Pearl's method of virtual evidence to obtain the updated probability distribution

$$\widetilde{\mathbb{P}}_{X_1,X_2,X_3}(x_1,x_2,x_3) = \mathbb{P}_{X_1,X_2,X_3|V}(x_1,x_2,x_3|1) \qquad (x_1,x_2,x_3) \in \{0,1\}^3.$$

4. Let $\mathcal{G} = (V,E)$ be a Directed Acyclic Graph, where $V = (X_1,\ldots,X_d)$, and let $\mathbb{P}$ and $\mathbb{Q}$ be two probability distribution factorised along $\mathcal{G}$. Let

$$\theta_{jil} = \mathbb{P}_{X_j|\mathrm{Pa}_j}(x_j^{(i)}|\pi_j^{(l)}).$$

Suppose that the conditional probabilities for $\mathbb{P}$ and $\mathbb{Q}$ are the same except for one single (j,l) variable / parent configuration, where $\mathbb{P}_{X_j|\mathrm{Pa}_j}(.|\pi_j^{(l)})$ is given by $\theta_{j.l}$ and $\mathbb{Q}_{X_j|\mathrm{Pa}_j}(.|\pi_j^{(l)})$ is given by $\widetilde{\theta}_{j.l}$. Let D_{KL} denote the Kullback Leibler distance. Show that

$$D_{KL}(\mathbb{P}\|\mathbb{Q}) = \mathbb{P}(\{\mathrm{Pa}_j = \pi_j^{(l)}\}) d_{KL}(\theta_{j.l},\widetilde{\theta}_{j.l}).$$

5. Let D_{CD} denote the Chan Darwiche distance. Prove the remaining two statements of Theorem 6.13; that for any $\mathbb{P}$ and $\mathbb{Q}$,

$$D_{CD}(\mathbb{P},\mathbb{Q}) \geq 0 \qquad D_{CD}(\mathbb{P},\mathbb{Q}) = 0 \Rightarrow \mathbb{P} = \mathbb{Q}$$

and

$$D_{CD}(\mathbb{P},\mathbb{Q}) = D_{CD}(\mathbb{Q},\mathbb{P}).$$

6. Let $\mathbb{P}$ be a probability distribution over a countable state space $\mathcal{X}$ and let $G_1, \ldots, G_n$ be a collection of mutually exclusive and exhaustive events. Let $\lambda_j = \mathbb{P}(G_j)$ for $j = 1, \ldots, n$. Let $\mathbb{Q}$ denote the probability distribution such that $\mathbb{Q}(G_j) = \mu_j$ for $j = 1, \ldots, n$ and such that for any other event A,

$$\mathbb{Q}(A) = \sum_{j=1}^{n} \mu_j \mathbb{P}(A|G_j).$$

In other words, $\mathbb{Q}$ is the Jeffrey's update of $\mathbb{P}$, defined by $\mathbb{Q}(G_j) = \mu_j$, $j = 1, \ldots, n$. Prove that

$$D_{CD}(\mathbb{P}, \mathbb{Q}) = \ln \max_j \frac{\lambda_j}{\mu_j} - \ln \min_j \frac{\lambda_j}{\mu_j},$$

where D_{CD} denotes the Chan Darwiche distance.

7. (a) Find a calibration of the Chan-Darwiche distance in terms of the distance between two Bernoulli trials. That is, let $\mathbb{P} = (p_0, p_1)$ and $\mathbb{Q} = (q_0, q_1)$. Find the number $cd(k)$ such that if $q_0 = 1 - cd(k)$ and $q_1 = cd(k)$ and $p_0 = p_1 = \frac{1}{2}$, then

$$D_{CD}(\mathbb{P}, \mathbb{Q}) = k.$$

You should obtain

$$cd(k) = \frac{e^{\pm k}}{1 + e^{\pm k}}.$$

(b) Find a calibration of the Kullback Leibler distance; that is, the number $KL(k)$ such that if $q_0 = 1 - KL(k)$, $q_1 = KL(k)$ and $p_0 = p_1 = \frac{1}{2}$, then $D_{KL}(\mathbb{P}\|\mathbb{Q}) = k$. You should obtain

$$KL(k) = \frac{1}{2} \pm \frac{1}{2}\sqrt{1 - e^{-2k}}.$$

8. **Jensen's inequality** Let $\phi(x)$ be a convex function and X finite discrete real valued random variable, defined on a finite space $\mathcal{X}$. Prove, by induction, that

$$\mathbb{E}[\phi(X)] \geq \phi(\mathbb{E}[X])$$

Hence prove that $D_{KL}(\mathbb{P}\|\mathbb{Q}) \geq 0$ with equality if and only if $\mathbb{P} = \mathbb{Q}$.

9. **The Chan-Darwiche Distance between Two Multivariate Bernoulli Distributions** Consider d independent Bernoulli trials, $X = (X_1, \ldots, X_d)$, where the 'success' probabilities for each trial may differ. The distribution of the random vector X is known as a *multivariate Bernoulli distribution*. This example considers the distance between two multivariate Bernoulli distributions where the 'success' probabilities for the two distributions are given by the vectors $p = (p_1, \ldots, p_d)$ and $q = (q_1, \ldots, q_d)$ respectively.

Let $\mathcal{X}$ be the binary hypercube; that is, $\mathcal{X} = \{0, 1\}^d$ and let $x \in \mathcal{X}$ denote an element in $\mathcal{X}$. Then $x = (x_i)_{i=1}^d$, where $x_i \in \{0, 1\}$. Let $\mathbb{Q}$ and $\mathbb{P}$ be two *multivariate Bernoulli probability functions* over $\mathcal{X}$. That is, $\mathbb{Q} : \mathcal{X} \to [0, 1]$ and $\mathbb{P} : \mathcal{X} \to [0, 1]$ are defined such that each $x \in \mathcal{X}$,

$$\mathbb{Q}(x) = \prod_{i=1}^{d} q_i^{x_i} (1-q_i)^{1-x_i}$$

and

$$\mathbb{P}(x) = \prod_{i=1}^{d} p_i^{x_i} (1-p_i)^{1-x_i}$$

where, for this example, it is assumed that $0 < q_i < 1$ and $0 < p_i < 1$ (i.e. the inequalities are strict) for all $i \in \{1, \ldots, d\}$.

(a) Show that

$$D_{CD}(\mathbb{P}, \mathbb{Q}) = \sum_{i=1}^{d} \ln \max\left(\frac{O_{q,i}}{O_{p,i}}, \frac{O_{p,i}}{O_{q,i}}\right). \tag{6.8}$$

where $O_{p,i} = \frac{p_i}{1-p_i}$ and $O_{q,i} = \frac{q_i}{1-q_i}$.

(b) Let $q_i = q$ and $p_i = p$ for all i, and $0 < q < 1$ and $0 < p < 1$. Show that, in this case,

$$D_{DC}(\mathbb{P}, \mathbb{Q}) = d \ln \max\left(\frac{O_{\mathbb{Q}}}{O_{\mathbb{P}}}, \frac{O_{\mathbb{P}}}{O_{\mathbb{Q}}}\right), \tag{6.9}$$

10. A piece of cloth is to be sold on the market. The colour C is either green (c_g), blue (c_b) or violet (c_v). Tomorrow, the piece of cloth will either be sold (s) or not (s^c); this is denoted by the variable S. Experience gives the following probability distribution over C, S

$$\mathbb{P}_{C,S} = \begin{array}{c|ccc} S\backslash C & c_g & c_b & c_v \\ \hline s & 0.12 & 0.12 & 0.32 \\ s^c & 0.18 & 0.18 & 0.08 \end{array}$$

The marginal distribution over C is

$$\mathbb{P}_C = \begin{array}{ccc} c_g & c_b & c_v \\ \hline 0.3 & 0.3 & 0.4 \end{array}.$$

The piece of cloth is inspected by candle light. Since it cannot be seen perfectly, this only gives *soft evidence*. From the inspection by candle light, the probability over C is assessed as:

$$\mathbb{Q}_C = \begin{array}{ccc} c_g & c_b & c_v \\ \hline 0.7 & 0.25 & 0.05 \end{array}.$$

The Jeffrey's update gives $\mathbb{Q}_{S,C} = \mathbb{Q}_C \mathbb{P}_{S|C} = \frac{\mathbb{Q}_C}{\mathbb{P}_C} \mathbb{P}_{S,C}$ which is

$$\mathbb{Q}_{S,C} = \begin{array}{c|ccc} S\backslash C & c_g & c_b & c_v \\ \hline s & 0.28 & 0.10 & 0.04 \\ s^c & 0.42 & 0.15 & 0.01 \end{array}.$$

(a) Compute $D_{CD}(\mathbb{P}, \mathbb{Q})$, the Chan-Darwiche distance between the original and updated distributions.

(b) Compute the bounds on the odds ratios given by Corollary 6.18 in this example. Compare with $\frac{O_{\mathbb{Q}}(c_g|s)}{O_{\mathbb{P}}(c_g|s)}$.

(c) Suppse that $\mathbb{Q}^*_C = (0.25, 0.25, 0.50)$. Compute $D_{CD}(\mathbb{P}, \mathbb{Q}^*)$ and the bounds on the odds ratios given by Corollary 6.18. Again, compare with $\frac{O_{\mathbb{Q}^*}(c_g|s)}{O_{\mathbb{P}}(c_g|s)}$
The distribution $\mathbb{Q}^*$ is closer to $\mathbb{P}$ than $\mathbb{Q}$ and hence the bounds are tighter.

(d) Now consider the following problem: the probability that the piece of cloth is green, given that it is sold tomorrow is, before updating, 0.214. What evidence would satisfy the constraint that the updated probability that the cloth is green, given that it is sold tomorrow, does not exceed 0.3?

6.7 Answers

1. Let A denote car type and C colour. Events to be updated: $\mathbb{P}^*_C(\text{red}) = 0.4$, $\mathbb{P}^*_C(\text{blue}) = 0.6$
Original *joint* probability function:

$$\mathbb{P}_{A,C} = \begin{array}{c|cc} \text{car\colour} & R & B \\ \hline V & 0.35 & 0.15 \\ S & 0.1 & 0.4 \end{array}$$

so

$$\mathbb{P}_C(\text{red}) = 0.45 \qquad \mathbb{P}_C(\text{blue}) = 0.55$$

and

$$\mathbb{P}_{A|C} = \begin{array}{c|cc} \text{car\colour} & R & B \\ \hline V & 7/9 & 3/11 \\ S & 2/9 & 8/11 \end{array}$$

Jeffrey's rule:

$$\mathbb{P}^*_{A,C} = \mathbb{P}_{A|C}\mathbb{P}^*_C = \begin{array}{c|cc} \text{car\colour} & R & B \\ \hline V & 14/45 & 9/55 \\ S & 4/45 & 24/55 \end{array}$$

$$\mathbb{P}^*(\text{volvo}) = \frac{47}{99}$$

2. (a) $A \perp B|G_j$ for each G_j, $j = 1, \ldots n$ so $\mathbb{P}(A|G_j, B) = \mathbb{P}(A|G_j)$. It follows, using $\mathbb{P}(B|G_j)\mathbb{P}(G_j) = \mathbb{P}(BG_j)$ and $\lambda_j = \mathbb{P}(A|G_j)$ that

$$\mathbb{P}(B|A) = \sum_j \mathbb{P}(B|A, G_j)\mathbb{P}(G_j|A) = \sum_j \mathbb{P}(B|G_j)\frac{\mathbb{P}(A|G_j)\mathbb{P}(G_j)}{\mathbb{P}(A)} = \frac{\sum_j \lambda_j \mathbb{P}(B \cap G_j)}{\sum_j \lambda_j \mathbb{P}(G_j)}.$$

For an outcome x,

$$\mathbb{P}(x|A) = \mathbb{P}(x)\frac{\lambda_j}{\sum_k \lambda_k \mathbb{P}(G_k)} = \mathbb{P}(x)\frac{\rho_j}{\sum_k \rho_k \mathbb{P}(G_k)} \qquad x \in G_j, \quad j = 1, \ldots, n$$

where $\rho_k = \frac{\lambda_k}{\lambda_1} = \frac{\mathbb{P}(A|G_k)}{\mathbb{P}(A|G_1)}$
which is the definition of the Pearl update.

(b) The Jeffrey's rule is valid for a piece of information Ξ that alters the probabilities on the partition events $G_1, \ldots, G_n$ and such that $\mathbb{P}(\Xi|G_j, B) = \mathbb{P}(\Xi|G_j)$ for any event B. Let $\mathbb{P}^*(C) = \mathbb{P}(C|\Xi)$, the updated probability for an event C. Then the update under Jeffrey's rule is, for any outcome x,

$$\mathbb{P}^*(x) = \sum_{j=1}^{n} \mathbb{P}(x|G_j)\mathbb{P}^*(G_j) = \sum_{j=1}^{n} q_j \mathbb{P}(x|G_j) = q_k \mathbb{P}(x|G_k) \qquad x \in G_k,$$

Pearl's method for updating given a piece of information A and a partition $G_1, \ldots, G_n$ is to set

$$\rho_j = \frac{\mathbb{P}(\Xi|G_j)}{\mathbb{P}(\Xi|G_1)} \qquad j = 1, \ldots, n$$

The Pearl update is defined by

$$\mathbb{P}^*(x) = \mathbb{P}(x|\Xi) = \mathbb{P}(x)\frac{\rho_j}{\sum_{k=1}^n \rho_k \mathbb{P}(G_k)} \qquad x \in G_j.$$

Using $\mathbb{P}(\Xi|G) = \frac{\mathbb{P}(G|\Xi)\mathbb{P}(\Xi)}{\mathbb{P}(G)}$,

$$\rho_j = \frac{\mathbb{P}(G_j|\Xi)}{\mathbb{P}(G_j)}\frac{\mathbb{P}(G_1)}{\mathbb{P}(G_1|\Xi)} = \frac{\lambda_j}{\lambda_1}$$

so that

$$\mathbb{P}^*(x) = \mathbb{P}(x)\frac{\lambda_j}{\sum_{k=1}^n \lambda_k \mathbb{P}(G_k)} \qquad x \in G_j,$$

which gives an expression for Pearl's update in terms of the $\lambda_j = \frac{q_j}{\mathbb{P}(G_j)} = \frac{\mathbb{P}^*(G_j)}{\mathbb{P}(G_j)} = \frac{\mathbb{P}(G_j|A)}{\mathbb{P}(G_j)}$. To show that it is the same as Jeffrey's rule, for $x \in G_j$,

$$\mathbb{P}^*(x) = \mathbb{P}(x)\frac{\mathbb{P}(\Xi|G_j)}{\mathbb{P}(\Xi)} = \frac{\mathbb{P}(x \cap G_j)}{\mathbb{P}(G_j)}\frac{\mathbb{P}(\Xi|G_j)\mathbb{P}(G_j)}{\mathbb{P}(\Xi)} = \mathbb{P}(x|G_j)\mathbb{P}(G_j|\Xi) = q_j\mathbb{P}(x|G_j),$$

which is the Jeffrey's update.

3. In this example, X_1, X_2, X_3 are mutually independent; $\mathbb{P}_{X_1,X_2,X_3} = \mathbb{P}_{X_1}\mathbb{P}_{X_2}\mathbb{P}_{X_3}$, $\mathbb{P}_{X_j}(1) = \mathbb{P}_{X_j}(0) = \frac{1}{2}$ for $j = 1, 2, 3$. Virtual evidence on X_3 is treated as a node with a single parent X_3, so (using $\mathbb{P}_{X_3}(1) = \mathbb{P}_{X_3}(0) = \frac{1}{2}$),

$$\begin{aligned}\widetilde{\mathbb{P}}_{X_3}(1) &= \mathbb{P}_{X_3}(1)\frac{\mathbb{P}_{V|X_3}(1|1)}{\mathbb{P}_{V|X_3}(1|1)\mathbb{P}_{X_3}(1) + \mathbb{P}_{V|X_3}(1|0)\mathbb{P}_{X_3}(0)} \\ &= \mathbb{P}_{X_3}(1)\frac{2\lambda}{\lambda+1} = \frac{\lambda}{\lambda+1}\end{aligned}$$

so

$$\widetilde{\mathbb{P}}_{X_1,X_2,X_3}(x_1, x_2, 1) = \mathbb{P}_{X_1}(x_1)\mathbb{P}_{X_2}(x_2)\widetilde{\mathbb{P}}_{X_3}(1) = \frac{\lambda}{8(\lambda+1)}$$

$$\widetilde{\mathbb{P}}_{X_1,X_2,X_3}(x_1, x_2, 0) = \mathbb{P}_{X_1}(x_1)\mathbb{P}_{X_2}(x_2)\widetilde{\mathbb{P}}_{X_3}(x_3) = \frac{1}{8(\lambda+1)}$$

for each value of $(x_1, x_2) \in \{(0,0), (0,1), (1,0), (1,1)\}$.

4. Assume that the variables are ordered so that $\mathrm{Pa}_j \subseteq \{X_1, \ldots, X_{j-1}\}$. Then

$$\begin{aligned}
D_{KL}(\mathbb{P},\mathbb{Q}) &= \sum_x \mathbb{P}(x)\ln\frac{\mathbb{P}(x)}{\mathbb{Q}(x)} \\
&= \sum_{x=(x_1^{(i_1)},\ldots,x_d^{(i_d)})} \prod_{k=1}^d \mathbb{P}_{X_k|\mathrm{Pa}_k}(x_k^{(i_k)}|\pi_k(x))\ln\frac{\prod_{k=1}^d \mathbb{P}_{X_k|\mathrm{Pa}_k}(x_k^{(i_k)}|\pi_k(x))}{\prod_{k=1}^d \mathbb{Q}_{X_k|\mathrm{Pa}_k}(x_k^{(i_k)}|\pi_k(x))} \\
&= \sum_{x|\pi_j(x)=\pi_j^l} \prod_{k=1}^d \mathbb{P}_{X_k|\mathrm{Pa}_k}(x_k^{(i_k)}|\pi_k(x))\ln\frac{\theta_{ji_jl}}{\widetilde{\theta}_{ji_jl}} \\
&= \sum_{i_j} \theta_{ji_jl}\ln\frac{\theta_{ji_jl}}{\widetilde{\theta}_{ji_jl}} \sum_{(x_1^{(i_1)},\ldots,x_{j-1}^{(i_{j-1})})|\pi_j(x)=\pi_j^l} \prod_{k=1}^{j-1} \mathbb{P}_{X_k|\mathrm{Pa}_k}(x_k^{(i_k)}|\pi_k(x) \\
&= d_{KL}(\theta_{j.l},\widetilde{\theta}_{j.l})\mathbb{P}_{\mathrm{Pa}_j}(\pi_j^l).
\end{aligned}$$

5.
$$D_{CD}(\mathbb{P},\mathbb{Q}) = \max_x \ln\frac{\mathbb{P}(x)}{\mathbb{Q}(x)} - \min_x \ln\frac{\mathbb{P}(x)}{\mathbb{Q}(x)}.$$
Clearly, for any function f, $\max_x f(x) \geq \min_x f(x)$ so the distance is non negative. If $D_{CD}(\mathbb{P},\mathbb{Q}) = 0$, it follows that $\frac{\mathbb{P}(x)}{\mathbb{Q}(x)} = \alpha$, a constant, for all $x \in \mathcal{X}$. It follows that $\mathbb{P}(x) = \alpha\mathbb{Q}(x)$ so that
$$1 = \sum_x \mathbb{P}(x) = \alpha\sum_x \mathbb{Q}(x) = \alpha$$
and hence $\alpha = 1$, so that $\mathbb{P}(x) = \mathbb{Q}(x)$ for all $x \in \mathcal{X}$.

For the second point, if $\mathbb{P}(x) > 0$ for a point where $\mathbb{Q}(x) = 0$, or $\mathbb{P}(x) = 0$ for a point where $\mathbb{Q}(x) > 0$, then $D_{CD}(\mathbb{P},\mathbb{Q}) = +\infty$.

Now consider $\mathbb{P}(x) > 0 \Leftrightarrow \mathbb{Q}(x) > 0$. For a strictly positive function f, $\max_x f(x) = \frac{1}{\min_x(1/f(x))}$ since the point where the maximum of $f(x)$ is attained is the point where the minimum of $1/f(x)$ is attained. It follows that
$$\begin{aligned}
D_{CD}(\mathbb{P},\mathbb{Q}) &= \max\ln\frac{\mathbb{P}(x)}{\mathbb{Q}(x)} - \min\ln\frac{\mathbb{P}(x)}{\mathbb{Q}(x)} \\
&= \ln\frac{1}{\min_x \frac{\mathbb{Q}(x)}{\mathbb{P}(x)}} - \ln\frac{1}{\max_x \frac{\mathbb{Q}(x)}{\mathbb{P}(x)}} \\
&= -\min_x \ln\frac{\mathbb{Q}(x)}{\mathbb{P}(x)} + \max_x \ln\frac{\mathbb{Q}(x)}{\mathbb{P}(x)} \\
&= D_{CD}(\mathbb{P},\mathbb{Q}).
\end{aligned}$$

6. Take any point $x \in \mathcal{X}$, then $x \in G_j$ for exactly one j. It follows that $\mathbb{Q}(x) = \mu_j\mathbb{P}(x|G_j) = \mu_j\frac{\mathbb{P}(x)}{\lambda_j}$ for j such that $x \in G_j$. Therefore
$$\begin{aligned}
D_{CD}(\mathbb{P},\mathbb{Q}) &= \max_x \ln\frac{\mathbb{P}(x)}{\mathbb{Q}(x)} - \min_x \ln\frac{\mathbb{P}(x)}{\mathbb{Q}(x)} \\
&= \max_j \ln\frac{\lambda_j}{\mu_j} - \min_j \ln\frac{\lambda_j}{\mu_j}
\end{aligned}$$
as required.

7. (a) Let $\mathbb{P} = (p_0, p_1)$ be a Bernoulli trial with success probability $p_1 = \frac{1}{2}$ and $\mathbb{Q}$ a Bernoulli trial with success probability $q_1 = \theta > \frac{1}{2}$. Then

$$D_{CD}(\mathbb{Q}, \mathbb{P}) = \ln 2\theta - \ln 2(1-\theta) = \ln \frac{\theta}{1-\theta}.$$

Hence, let $\theta(k)$ denote the value of θ such that $D_{CD}(\mathbb{Q}, \mathbb{P}) = k$, then

$$e^k = \frac{\theta}{1-\theta}$$

giving

$$\theta(k) = \frac{e^k}{1+e^k}.$$

Considering $0 \le \theta \le \frac{1}{2}$ gives, for a Chan Darwiche distance k,

$$\theta(k) = \frac{e^{-k}}{1+e^{-k}}.$$

(b)

$$D_{CD}(\mathbb{Q}, \mathbb{P}) = \frac{1}{2} \ln \frac{1}{2\theta(k)} + \frac{1}{2} \ln \frac{1}{2\theta(k)} = k$$

$$\theta(1-\theta) = \frac{1}{4} e^{-2k}$$

$$\left(\theta - \frac{1}{2}\right)^2 = \frac{1}{4}\left(1 - e^{-2k}\right)$$

$$\theta(k) = \frac{1}{2} \pm \frac{1}{2}\sqrt{1 - e^{-2k}}.$$

8. Definition of convexity for a function ϕ: for any $\lambda \in [0,1]$ and any (x, y),

$$\phi(\lambda x + (1-\lambda)y) \le \lambda\phi(x) + (1-\lambda)\phi(y).$$

Proof of result by induction: if $\mathcal{X} = \{x_1, x_2\}$, set $p_1 = \lambda$, $p_2 = 1 - \lambda$, then $\mu = E[X] = p_1 x_1 + p_2 x_2$ so definition of convexity gives

$$\phi(\mu) \le p_1\phi(x_1) + p_2\phi(x_2) = E[\phi(X)],$$

with equality if and only if $\phi(x) = a + bx$ for $x \in \{x_1, x_2, \mu\}$.

Assume result is true for any probability distribution over $\{x_1, \ldots, x_n\}$. Consider a probability distribution $(p_1, \ldots, p_{n+1})$ over $(x_1, \ldots, x_{n+1})$. Then

$$\phi(\mu) = \phi\left(\sum_{j=1}^{n+1} p_j x_j\right) \le p_{n+1}\phi(x_{n+1}) + (1 - p_{n+1})\phi\left(\sum_{j=1}^{n} \frac{p_j}{1-p_{n+1}} x_j\right)$$

and, by the inductive hypothesis (since $\sum_{j=1}^{n} \frac{p_j}{1-p_{n+1}} = 1$) ,

$$\phi\left(\sum_{j=1}^{n} \frac{p_j}{1-p_{n+1}} x_j\right) \le \sum_{j=1}^{n} \frac{p_j}{1-p_{n+1}} \phi(x_j)$$

so that

$$\phi(\mu) \leq \sum_{j=1}^{n+1} \phi(x_j) p_j$$

with equality if and only if $\phi(x) = ax + b$ for $x \in \{\mu, x_1, \ldots, x_n\}$ as required.

It follows that

$$D_{KL}(\mathbb{P} \| \mathbb{Q}) = -\sum_j p_j \ln \frac{q_j}{p_j} \geq -\ln \sum_j p_j \frac{q_j}{p_j} = -\ln \sum_j q_j = -\ln 1 = 0$$

with equality if and only if $p = q$.

9. (a) The *likelihood ratio* between $\mathbb{Q}$ and $\mathbb{P}$ is well defined and is given by

$$\mathrm{LR}(x) = \frac{\mathbb{Q}(x)}{\mathbb{P}(x)} = \prod_{i=1}^{d} \left(\frac{q_i}{p_i}\right)^{x_i} \left(\frac{1-q_i}{1-p_i}\right)^{1-x_i}.$$

For each $i \in \{1, \ldots, d\}$, let m_i be defined as

$$m_i = \begin{cases} 1 & \text{if } \frac{q_i}{p_i} \geq \frac{1-q_i}{1-p_i} \\ 0 & \text{otherwise.} \end{cases} \tag{6.10}$$

Then $m = (m_i)_{i=1}^d \in \mathcal{X}$ and, by construction, it follows that for all $x \in \mathcal{X}$,

$$\mathrm{LR}(x) \leq \mathrm{LR}(m) = \prod_{i=1}^{d} \max\left(\frac{q_i}{p_i}, \frac{1-q_i}{1-p_i}\right). \tag{6.11}$$

Next let $\bar{m}$ be the *binary complement* of m defined by Equation (6.10). That is, for each $i \in \{1, \ldots, d\}$, $\bar{m}_i = 1 - m_i$, giving $\bar{m}_i = 0$, if $m_i = 1$ and $\bar{m}_i = 1$ if $m_i = 0$. Then it holds that

$$\mathrm{LR}(x) \geq \mathrm{LR}(m) = \prod_{i=1}^{d} \min\left(\frac{q_i}{p_i}, \frac{1-q_i}{1-p_i}\right). \tag{6.12}$$

It now follows from the definition of the Chan - Darwiche distance measure (Definition 6.11) that

$$\begin{aligned} D_{DC}(p,q) &= \ln \mathrm{LR}(m) - \ln \mathrm{LR}(\widetilde{m}) \\ &= \sum_{i=1}^{d} \ln \frac{\max\left(\frac{q_i}{p_i}, \frac{1-q_i}{1-p_i}\right)}{\min\left(\frac{q_i}{p_i}, \frac{1-q_i}{1-p_i}\right)}. \end{aligned}$$

For i such that $m_i = 1$ it clearly holds that

$$\frac{\max\left(\frac{q_i}{p_i}, \frac{1-q_i}{1-p_i}\right)}{\min\left(\frac{q_i}{p_i}, \frac{1-q_i}{1-p_i}\right)} = \frac{\frac{q_i}{p_i}}{\frac{1-q_i}{1-p_i}} = \frac{O_{q,i}}{O_{p,i}},$$

where O denotes the odds;

$$O_{q,i} = \frac{q_i}{1-q_i}, O_{p,i} = \frac{p_i}{1-p_i}.$$

Similarly for i such that $m_i = 0$ it holds that

$$\frac{\max\left(\frac{q_i}{p_i}, \frac{1-q_i}{1-p_i}\right)}{\min\left(\frac{q_i}{p_i}, \frac{1-q_i}{1-p_i}\right)} = \frac{O_{p,i}}{O_{q,i}},$$

from which the result follows.

(b) Let $q_i = q$ and $p_i = p$ for all i, and $0 < q < 1$ and $0 < p < 1$.

$$\mathbb{Q}(x) = q^k (1-q)^{d-k}, \qquad \mathbb{P}(x) = p^k (1-p)^{d-k},$$

where k is the number of digital ones in x. It follows that

$$D_{DC}(\mathbb{P}, \mathbb{Q}) = d \ln \max\left(\frac{O_{\mathbb{Q}}}{O_{\mathbb{P}}}, \frac{O_{\mathbb{P}}}{O_{\mathbb{Q}}}\right).$$

If, say, $\frac{O_{\mathbb{Q}}}{O_{\mathbb{P}}} > \frac{O_{\mathbb{P}}}{O_{\mathbb{Q}}}$, then

$$D_{DC}(\mathbb{P}, \mathbb{Q}) = d\left(\ln O_{\mathbb{Q}} - \ln O_{\mathbb{P}}\right),$$

10. (a) Theorem 6.17 gives

$$D_{CD}(\mathbb{P}, \mathbb{Q}) = \ln \max_i \frac{\lambda_i}{\mu_i} - \ln \min_i \frac{\lambda_i}{\mu_i} = \ln \frac{0.7}{0.3} - \ln \frac{0.05}{0.4} = 2.93,$$

(b) Corollary 6.18 gives

$$0.05 \leq \frac{O_{\mathbb{Q}}(c_g|s)}{O_{\mathbb{P}}(c_g|s)} \leq 18.73.$$

This suggests that the distributions have changed dramatically. Note that $\mathbb{P}_{C|S}(c_g|s) = \frac{0.12}{0.56} = 0.214$, while $\mathbb{Q}_{C|S}(c_g|s) = \frac{0.28}{0.42} = 0.667$.

$$\frac{O_{\mathbb{Q}}(c_g|s)}{O_{\mathbb{P}}(c_g|s)} = \frac{0.667/0.333}{0.214/0.786} = 7.34.$$

(c) If the new distribution over colour is $\mathbb{Q}^*_C = (0.25, 0.25, 0.50)$, then $D_{CD}(\mathbb{P}, \mathbb{Q}^*) = 0.406$ and

$$0.513 \leq \frac{O_{\mathbb{Q}^*}(c_g|s)}{O_{\mathbb{P}}(c_g|s)} \leq 1.946$$

The evidence is weaker and the bounds are therefore tighter. In this case,

$$\begin{aligned}
&\frac{O_{\mathbb{Q}^*}(c_g|s)}{O_{\mathbb{P}}(c_g|s)} \\
&\quad = \frac{\mathbb{Q}^*(c_g|s)}{\mathbb{Q}^*(c_g|s^c)} \frac{\mathbb{P}(c_g|s^c)}{\mathbb{P}(c_g|s)} = \frac{\mathbb{Q}^*(c_g, s)}{\mathbb{Q}^*(s)} \frac{\mathbb{Q}^*(s^c)}{\mathbb{Q}^*(c_g, s^c)} \frac{\mathbb{P}(c_g, s^c)}{\mathbb{P}(s^c)} \frac{\mathbb{P}(s)}{\mathbb{P}(c_g, s)} \\
&\quad = \frac{\mathbb{Q}^*(c_g)\mathbb{P}(s|c_g)\left(\mathbb{Q}^*(c_g)\mathbb{P}(s^c|c_g) + \mathbb{Q}^*(c_v)p(s^c|c_v) + \mathbb{Q}^*(c_b)\mathbb{P}(s^c|c_b)\right)}{\left(\mathbb{Q}^*(c_g)\mathbb{P}(s|c_g) + \mathbb{Q}^*(c_v)\mathbb{P}(s|c_v) + \mathbb{Q}^*(c_b)\mathbb{P}(s|c_b)\right)\mathbb{Q}^*(c_g)\mathbb{P}(s^c|c_g)} \frac{\mathbb{P}(c_g, s^c)}{\mathbb{P}(s^c)} \frac{\mathbb{P}(s)}{\mathbb{P}(c_g, s)} \\
&\quad = 1.756
\end{aligned}$$

(d) Inequality (6.7) gives

$$\frac{0.214e^{-d}}{1+(e^{-d}-1)\times 0.214} \le \mathbb{Q}(c_g|s) \le \frac{0.214e^{d}}{1+(e^{d}-1)\times 0.214}.$$

The constraint $\mathbb{Q}_{C|S}(c_g|s) \le 0.3$ is satisfied if

$$\frac{0.214e^{d}}{1+(e^{d}-1)\times 0.214} \le 0.3$$

giving $d \le 0.454$. The current distribution over colour is $(\mu_g, \mu_b, \mu_v) = (0.3, 0.3, 0.4)$. The problem now reduces to finding $(\lambda_g, \lambda_b, \lambda_v)$ such that $\mathbb{Q}_{C|S}(c_g|s) = 0.3$ and

$$\ln \max\left(\frac{\lambda_g}{0.3}, \frac{\lambda_b}{0.3}, \frac{\lambda_v}{0.4}\right) - \ln \min\left(\frac{\lambda_g}{0.3}, \frac{\lambda_b}{0.3}, \frac{\lambda_v}{0.4}\right) = 0.454.$$

Since

$$\mathbb{Q}_{C,S}(c_j, s) = \frac{\lambda_j}{\mu_j}\mathbb{P}_{C,S}(c_j, s), \qquad j = g, b, v$$

it follows that

$$\mathbb{Q}_{C|S}(c_g|s) = \frac{0.4\lambda_g}{0.4\lambda_g + 0.4\lambda_b + 0.4\lambda_v}.$$

With $\mathbb{P}_{C|S}(c_g|s) = 0.3$,

$$0.28\lambda_g - 0.12\lambda_b - 0.24\lambda_v = 0,$$

with constraint $\lambda_g + \lambda_b + \lambda_v = 1$, so that $10\lambda_g - 3\lambda_v = 3$.

If the maximum and minimum are then given by λ_g and λ_v respectively, then

$$0.454 = \ln \frac{\lambda_g}{0.3} - \ln \frac{10\lambda_g - 3}{1.2}$$

giving

$$\lambda_g = \frac{3e^{0.454}}{10e^{0.454} - 4} = 0.402$$
$$\lambda_v = 0.34, \qquad \lambda_b = 0.258.$$

Finally, to check that the solution is valid, $\frac{0.4}{\lambda_v} = \frac{0.4}{0.34} = 1.176 > 1.163 = \frac{0.3}{0.258} = \frac{0.3}{\lambda_b}$. Similarly, $\frac{\lambda_g}{0.3}$ clearly gives the maximum in the first term.

Chapter 7

Marginalisation, Triangulated Graphs and Junction Trees

7.1 Functions and Domains

Notation Let $V = \{X_1, \ldots, X_d\}$ denote the set of random variables, where variable X_j has state space $\mathcal{X}_j = (x_j^{(1)}, \ldots, x_j^{(k_j)})$ for $j = 1, \ldots, d$. Let $\mathcal{X} = \times_{j=1}^{d} \mathcal{X}_j$ denote the state space for the random vector $\underline{X} = (X_1, \ldots, X_d)$. Let $\tilde{V} = \{1, \ldots, d\}$ denote the indexing set for the variables. For $D \subset \tilde{V}$, where $D = \{j_1, \ldots, j_m\}$, let $\mathcal{X}_D = \times_{j \in D} \mathcal{X}_j$ and let $\underline{X}_D = (X_{j_1}, \ldots, X_{j_m})$. Let $\underline{x} \in \mathcal{X}$ denote a generic element of $\mathcal{X}$ and let $\underline{x}_D = (x_{j_1}, \ldots, x_{j_m}) \in \mathcal{X}_D$, when $\underline{x} = (x_1, \ldots, x_d) \in \mathcal{X}$. The notation

$$\underline{x}_D = \times_{v \in D} x_v$$

is used to denote a configuration (or a collection of outcomes) on the nodes in D. Furthermore, for any set $W \subset V$, let $\tilde{W}$ denote the indexing set for W. The notation $\mathcal{X}_W$ will also be used to denote $\mathcal{X}_{\tilde{W}}$, $\underline{X}_W$ to denote $\underline{X}_{\tilde{W}}$ and $\underline{x}_W$ to denote $\underline{x}_{\tilde{W}}$. Suppose $D \subseteq W \subseteq \tilde{V}$ and that $\underline{x}_W \in \mathcal{X}_W$. That is, $\underline{x}_W = \times_{v \in W} x_v$. Then, ordering the variables of W so that $\mathcal{X}_W = \mathcal{X}_D \times \mathcal{X}_{W \setminus D}$, the projection of $\underline{x}_W$ onto D is defined as the variable $\underline{x}_D$ that satisfies

$$\underline{x}_W = (\underline{x}_D, \underline{x}_{W \setminus D}),$$

where the meaning of the notation '$(,)$' is clear from the context. Here $A \setminus B$ denotes the set difference; i.e. the elements in the set A not included in B.

Definition 7.1 (Function, Domain)**.** *Consider a function $\phi : \mathcal{X}_D \to \mathbb{R}_+$. The space $\mathcal{X}_D$ is known as the* domain *of the function. If the domain is the state space of a random vector $\underline{X}_D$, then $\underline{X}_D$ may also be referred to as the domain of the function.*

In this setting, a function over a domain $\mathcal{X}_D$ has $\prod_{j \in D} k_j$ entries. For $W \subset V$, the domain of a function $\mathcal{X}_W$ may also be denoted by the collection of random variables W.

Addition, Multiplication, Division For functions defined on the same domain, addition, multiplication and division are defined pointwise where, by definition,

$$a(\underline{x}) = 0, \quad b(\underline{x}) = 0 \implies \frac{a(\underline{x})}{b(\underline{x})} = 0.$$

□

Functions over different domains If function ϕ_1 is defined over domain $\mathcal{X}_{D_1}$ and function ϕ_2 is defined over domain $\mathcal{X}_{D_2}$, then multiplication and division of functions may be defined by first extending both functions to the domain $\mathcal{X}_{D_1 \cup D_2}$.

Definition 7.2 (Extending the Domain)**.** *Let the function ϕ be defined on a domain $\mathcal{X}_D$, where $D \subset \tilde{W} \subseteq \tilde{V}$. Then ϕ, defined over a domain $\mathcal{X}_D$, is extended to the domain $\mathcal{X}_{\tilde{W}}$ in the following way. For each $\underline{x}_{\tilde{W}} \in \mathcal{X}_{\tilde{W}}$,*

$$\phi(\underline{x}_{\tilde{W}}) = \phi(\underline{x}_D),$$

where $\underline{x}_D$ is the projection of $\underline{x}_{\tilde{W}}$ onto $\mathcal{X}_D$, using the definition of $\underline{x}_D$ (and hence $\underline{x}_{\tilde{W}}$) from the beginning of the section, page 141. In other words, the extended function depends on $\underline{x}_{\tilde{W}}$ only through $\underline{x}_D$.

Addition, Multiplication and Division of Functions over Different Domains Addition, multiplication and division of functions over different domains is defined as first, extending the domains of definition using Definition 7.2 so that they are defined over the same domain, followed by standard pointwise addition, multiplication or division. □

Multiplication of functions may be expressed in the following terms: the product $\phi_1.\phi_2$ of functions ϕ_1 and ϕ_2, defined over domains $\mathcal{X}_{D_1}$ and $\mathcal{X}_{D_2}$ is defined as

$$(\phi_1.\phi_2)(\underline{x}_{D_1 \cup D_2}) = \phi_1(\underline{x}_{D_1 \cup D_2})\phi_2(\underline{x}_{D_1 \cup D_2}),$$

where ϕ_1 and ϕ_2 have *first* been extended to $\mathcal{X}_{D_1 \cup D_2}$.

Let D_ϕ denote the index set for the domain variables of a function ϕ. Then for two functions ϕ_1 and ϕ_2, $D_{\phi_1.\phi_2} = D_{\phi_1} \cup D_{\phi_2}$.

Marginalisation The operation of marginalisation is now considered more generally. Let $U \subseteq W \subseteq V$ and let ϕ be a function defined over $\mathcal{X}_W$. The expression $\sum_{\mathcal{X}_{W \setminus U}} \phi$ denotes the *margin* (or the *sum margin*) of ϕ over $\mathcal{X}_U$ and is defined for $\underline{x}_U \in \mathcal{X}_U$ by

$$\left(\sum_{W \setminus U} \phi \right)(\underline{x}_U) = \sum_{\underline{z} \in \mathcal{X}_W \setminus \mathcal{X}_U} \phi(\underline{z}, \underline{x}_U),$$

where the arguments have been rearranged so that those corresponding to $W \setminus U$ appear first, $\underline{z} \in \mathcal{X}_{W \setminus U}$ is the projection of $(\underline{z}, \underline{x}_U) \in \mathcal{X}_W$ onto $\mathcal{X}_{W \setminus U}$ and $\underline{x}_U \in \mathcal{X}_U$ the projection of $(\underline{z}, \underline{x}_U) \in \mathcal{X}_W$ onto $\mathcal{X}_U$. The following notation is also used for marginalising a function with domain $\mathcal{X}_W$.

$$\phi^{\downarrow U} = \left(\sum_{W \setminus U} \phi \right).$$

The marginalisation operation obeys the following rules:

1. The Commutative Law: for any two sets of variables $U \subset V$ and $W \subset V$,

$$(\phi^{\downarrow U})^{\downarrow W} = (\phi^{\downarrow W})^{\downarrow U}.$$

2. The Distributive Law:

   If $\mathcal{X}_{D_1}$ is the domain of ϕ_1 and $D_1 \subseteq \tilde{V}$, then $(\phi_1 \phi_2)^{\downarrow D_1} = \phi_1 (\phi_2)^{\downarrow D_1}$.

Definition 7.3 (Charge, Contraction)**.** *A* charge

$$\Phi = \{\phi_1, \ldots, \phi_m\}$$

is defined as a set of functions on $\mathcal{X}$*.*

A contraction *of a charge, or set of functions is an operation of multiplication of functions, after extending them to* $\mathcal{X}$*, that returns the function*

$$\Phi(\underline{x}) = \prod_{j=1}^{m} \phi_j(\underline{x}).$$

The same notation is often used to denote the *contraction* of a charge and of the *set of functions* (the charge). The context makes it clear which is intended.

Probability function factorised along a DAG The *joint probability* function $p_{X_1,\ldots,X_d}$ is itself a function, with domain $\mathcal{X}$. If the joint probability function may be factorised according to a DAG $\mathcal{G} = (V, D)$, the decomposition is written as

$$p_{X_1,\ldots,X_d} = \prod_{j=1}^{d} p_{X_j|\Pi_j}.$$

Then for each $j = 1, \ldots, d$, ϕ_j defined by $\phi_j = p_{X_j|\Pi_j}$ is a function with domain $\mathcal{X}_{D_j} = \mathcal{X}_j \times \mathcal{X}_{\tilde{\Pi}_j}$ and $D_j = \{j\} \cup \tilde{\Pi}_j$.

Example 7.4.

Consider a probability function over six variables that may be factorised along the directed acyclic graph in Figure 7.1. The functions corresponding to the conditional probabilities are

$$\phi_1 = p_{X_1}, \phi_2 = p_{X_2|X_1}, \phi_3 = p_{X_3|X_1},$$

$$\phi_4 = p_{X_4|X_2}, \phi_5 = p_{X_5|X_2,X_3}, \phi_6 = p_{X_6|X_3}.$$

The corresponding *domains* are

$$\mathcal{X}_{D_1} = \mathcal{X}_1, \quad \mathcal{X}_{D_2} = \mathcal{X}_2 \times \mathcal{X}_1, \quad \mathcal{X}_{D_3} = \mathcal{X}_3 \times \mathcal{X}_1$$
$$\mathcal{X}_{D_4} = \mathcal{X}_4 \times \mathcal{X}_2, \quad \mathcal{X}_{D_5} = \mathcal{X}_5 \times \mathcal{X}_2 \times \mathcal{X}_3, \quad \mathcal{X}_{D_6} = \mathcal{X}_6 \times \mathcal{X}_3.$$

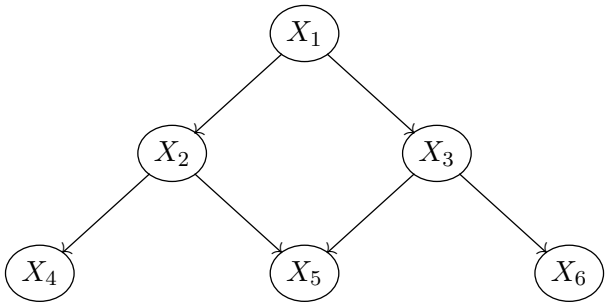

Figure 7.1: A Bayesian Network on 6 variables

Definition 7.5 (Domain Graph)**.** *The* domain graph *for the set of functions in* Φ *is an undirected graph with the variables as nodes and the links between any pair of variables which are members of the same domain.*

Figure 7.2 illustrates the domain graph associated with DAG of Figure 7.1. The domain graph of a DAG is the *moral graph*, Definition 5.4. The *maximal cliques* of the moral graph are illustrated in Figure 7.3.

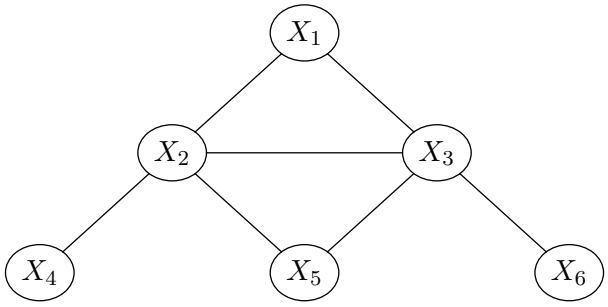

Figure 7.2: Domain graph of Bayesian Network in Figure 7.1

It is clear that the domain graph of a Bayesian network is the moral graph, since by definition all the parents are connected to each other and to the variable.

7.2 Marginalisation and Graphical Representations

Let ϕ_1 be a function with domain $\mathcal{X}_{D_1}$ and let ϕ_2 be a function with domain $\mathcal{X}_{D_2}$. Suppose that $A \subset D_1 \cup D_2$ and their product $\phi_1\phi_2$ is to be marginalised over $\mathcal{X}_A$. If $A \cap D_1 = \phi$ (the empty set), then

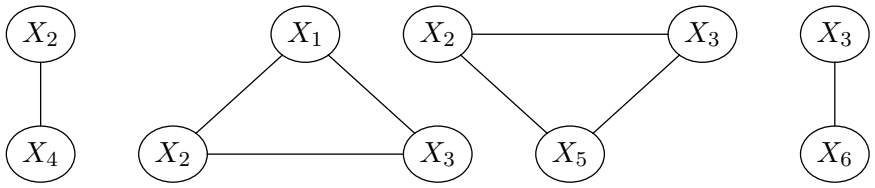

Figure 7.3: Maximal Cliques of the Graph in Figure 7.2

$$\sum_{\mathcal{X}_A} \phi_1 \phi_2 = \phi_1 \sum_{\mathcal{X}_A} \phi_2.$$

In coordinates, let ϕ_1 have domain $\mathcal{X}_{D_1 \cup D_3}$ and ϕ_2 domain $\mathcal{X}_{D_2 \cup D_3 \cup D_4}$, where D_1, D_2, D_3 and D_4 are disjoint. By the distributive law, the marginalisation may be written as

$$\sum_{\underline{x}_2 \in \mathcal{X}_{D_2}} \phi_1(\underline{x}_1, \underline{x}_3)\phi_2(\underline{x}_2, \underline{x}_3, \underline{x}_4) = \phi_1(\underline{x}_1, \underline{x}_3) \sum_{\underline{x}_2 \in \mathcal{X}_{D_2}} \phi_2(\underline{x}_2, \underline{x}_3, \underline{x}_4).$$

The function over $\mathcal{X}_{D_1} \times \mathcal{X}_{D_3} \times \mathcal{X}_{D_4}$ is first marginalised down to a function over $\mathcal{X}_{D_3} \times \mathcal{X}_{D_4}$. The function is transmitted to the function over $\mathcal{X}_{D_2} \times \mathcal{X}_{D_3}$, to which it is multiplied. The domains of the two functions to be multiplied have to be extended to $\mathcal{X}_{D_1} \times \mathcal{X}_{D_3} \times \mathcal{X}_{D_4}$. Using X_1, X_2, X_3, X_4 to denote the associated domains $\mathcal{X}_{D_1}, \mathcal{X}_{D_2}, \mathcal{X}_{D_3}$ and $\mathcal{X}_{D_4}$, the domains under consideration for the operations are illustrated in Figure 7.4. First, the function ϕ_2, defined over (X_2, X_3, X_4) is considered. This is marginalised to a function over (X_3, X_4) and is then extended, by multiplying with ϕ_1, to a function over (X_1, X_3, X_4).

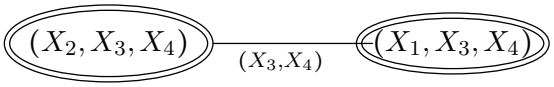

Figure 7.4: The Distributive Law

Example 7.6 (Example of a Marginalisation)**.**

Consider the computation for marginalising a contraction of a charge Φ defined over a state space $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \mathcal{X}_3 \times \mathcal{X}_4 \times \mathcal{X}_5$ where

$$\Phi(\underline{x}) = \phi_1(x_1, x_3, x_5)\phi_2(x_1, x_2)\phi_3(x_3, x_4)\phi_4(x_5, x_6).$$

More particularly, consider the computation of

$$\Phi^{\downarrow 0} = \sum_{\underline{x} \in \mathcal{X}} \Phi(\underline{x}),$$

where the notation $\Phi^{\downarrow U}$ is defined on page 142. With the order of summation: x_2, x_4, x_6, x_5, x_3 ,x_1, the sum may be written (taking sums from right to left) as

$$\sum_{x_1 \in \mathcal{X}_1} \sum_{x_3 \in \mathcal{X}_3} \sum_{x_5 \in \mathcal{X}_5} \phi_1(x_1, x_3, x_5) \sum_{x_6 \in \mathcal{X}_6} \phi_4(x_5, x_6) \sum_{x_4 \in \mathcal{X}_4} \phi_3(x_3, x_4) \sum_{x_2 \in \mathcal{X}_2} \phi_2(x_1, x_2).$$

The computation, carried out in this order (right to left), may be represented by the graph in Figure 7.5; a computational tree, according to the distributive law, is given in Figure 7.6. □

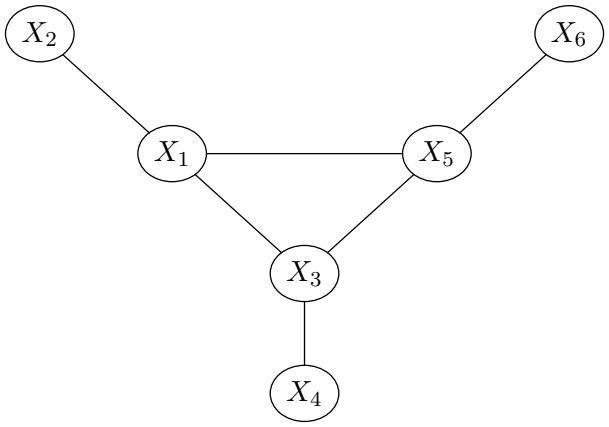

Figure 7.5: Associations of Variables

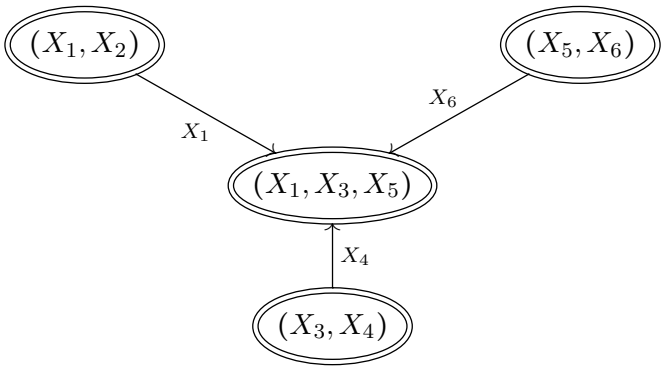

Figure 7.6: A Computational Tree for the Marginalisation

Recall (page 142) that the operation $\Phi^{\downarrow U}(\underline{x})$ means marginalising Φ over all variables not in the set U.

Definition 7.7 (Elimination of a Variable)**.** *The variable X_v, with index $v \in \tilde{W} = \tilde{V} \backslash \tilde{U}$ is* eliminated *from $\sum_{\underline{x}_{V \backslash U} \in \mathcal{X}_{V \backslash U}} \Phi(\underline{x}_{V \backslash U}, \underline{x}_U)$ by the following procedure, where* contraction *means multiplying together all the functions in the charge.*

1. *Let* Φ_v *(or* Φ_{X_v}*) denote the* contraction *of the functions in* Φ *that have* X_v *in their domain; that is,*

$$\Phi_v = \prod_{j|v\in D_j} \phi_j.$$

2. *Let* $\phi^{(v)}$ *(or* $\phi^{(X_v)}$*) denote the function* $\sum_{x_v\in\mathcal{X}_v} \Phi_v$.

3. *Find a new set of functions* Φ^{-v} *(or* Φ^{-X_v}*) by setting*

$$\Phi^{-v} = (\Phi \cup \{\phi^{(v)}\})\backslash\Phi_v.$$

This is the definition of Φ^{-v}*, also denoted by* Φ^{-X_v}.

Those functions that do *not* contain X_v in their domain have been retained; the others have been multiplied together and then marginalised over $\mathcal{X}_v$ (thus eliminating the variable) to give $\phi^{(v)}$. This function has been added to the collection, and all those containing X_v (other than $\phi^{(v)}$) have been removed.

(Note that the notation Φ^{-X_v} has two meanings: it is used to the collection of functions, and it is also used to denote the contraction of the charge obtained by multiplying together the functions in the collection. The meaning is determined by the context.) Having removed X_v, it remains to compute

$$\sum_{\underline{x}_{W\backslash\{X_v\}}} \Phi^{-X_v}(\underline{x}_U, \underline{x}_{W\backslash\{X_v\}}).$$

The quantity

$$\Phi^{\downarrow U}(\underline{x}_U) = \sum_{\underline{x}_{W\backslash U}\in\mathcal{X}_{W\backslash U}} \Phi(\underline{x}_{W\backslash U}, \underline{x}_U)$$

can be computed through successive elimination of the variables $X_v \in W\backslash U$. The task, of course, is to find a sequence for marginalising the variables such that, at each stage, the variable is to be eliminated from as small a domain as possible. The procedure outlined above may be considered graphically in terms of undirected graphs and their triangulations.

7.3 Decomposable Graphs and Node Elimination

Recall the definition of an *induced sub graph* (Definition 1.5); a subgraph induced by a subset $A \subset V$ is the graph $\mathcal{G}_A = (A, E_A)$ where $E_A = E \cap A \times A$. The following definitions are necessary.

Definition 7.8 (Complete Graph, Complete Subset)**.** *A graph* $\mathcal{G}$ *is* complete *(or a* clique*) if every pair of nodes is joined by an undirected edge. That is, for each* $(\alpha, \beta) \in V \times V$ *with* $\alpha \neq \beta$, $(\alpha, \beta) \in E$ *and* $(\beta, \alpha) \in E$*. In other words,* $\langle\alpha, \beta\rangle \in U$*, where* U *denotes the set of undirected edges. A subset of nodes is called* complete *if it induces a complete sub graph.*

Definition 7.9 (Maximal Clique)**.** *A* maximal clique *is a* complete sub graph *that is maximal with respect to* $\subseteq$*. In other words, a maximal clique is not a sub graph of any other complete graph.*

Definition 7.10 (Simplicial Node). *Recall the definition of family, found in Definition 1.2. For an undirected graph, the family of a node β is $F(\beta) = \{\beta\} \cup N(\beta)$, where $N(\beta)$ denotes the set of neighbours of β. A node β in an undirected graph is called* simplicial *if its family $F(\beta)$ is a maximal clique.*

This means that, in an undirected graph, a node β is *simplicial* if all its neigbours are neighbours of each other.

Definition 7.11 (Connectedness, Strong Components). *Let $\mathcal{G} = (V, E)$ be a simple graph, where $E = U \cup D$. That is, E may contain both directed and undirected edges. Let $\alpha \to \beta$ denote that there is a path (Definition 1.7) from α to β. If there is both $\alpha \to \beta$ and $\beta \to \alpha$ then α and β are said to be* connected. *This is written:*

$$\alpha \leftrightarrow \beta.$$

This is clearly an equivalence relation. The equivalence class *for α is denoted by $[\alpha]$. In other words, $\beta \in [\alpha]$ if and only if $\beta \leftrightarrow \alpha$. These equivalence classes are called* strong components *of $\mathcal{G}$.*

Note that a *graph* is connected if between any two nodes there exists a trail (Definition 1.6), but any two nodes α and β are only said to be connected if there is path from α to β and a path from β to α, where the definition of a 'path' is given in Definition 1.7.

Definition 7.12 (Chord). *Let $\mathcal{G} = (V, E)$ be a graph. Let σ be an n cycle in $\mathcal{G}$. A* chord *of this cycle is a pair (α_i, α_j) of* non consecutive *nodes in σ such that $\alpha_i \sim \alpha_j$ in $\mathcal{G}$.*

Definition 7.13 (Triangulated). *An undirected graph is said to be* triangulated *if every cycle of length ≥ 4 has a chord.*

Lemma 7.14. *If $\mathcal{G} = (V, E)$ is triangulated, then the induced graph $\mathcal{G}_A$ is also triangulated.*

Proof Consider any cycle of length ≥ 4 in the restricted graph. All the edges connecting these nodes remain. If the cycle possessed a chord in the original graph, the chord remains in the restricted graph. □

Definition 7.15 (Separator). *Let $\mathcal{G} = (V, E)$ be a graph. Let $\alpha, \beta \in V$ be two nodes. A subset $S \in V$ is called an α, β separator if every trail between α and β has at least one node in S. Let $A \subset V$, $B \subset V$. A set $S \subset V$ separates A and B if it is an α, β separator for each $(\alpha, \beta) \in A \times B$. A and B are said to be* separated *by S. The notation used in this text is $A \perp\!\!\!\perp B \,[\!]\, S$.*

Definition 7.16 (Minimal Separator). *Let $A \subseteq V$, $B \subseteq V$ and $S \subseteq V$ be three disjoint subsets of V. Let S separate A and B. The separator S is said to be a* minimal *separator of A and B if no proper subset of S is itself a separator of A and B.*

Definition 7.17 (Decomposition, Weak Decomposition). *Let $\mathcal{G} = (V, U)$ be an undirected graph. A triple (A, B, S) of disjoint subsets of the node set V of an undirected graph is said to form a* decomposition *of $\mathcal{G}$ or to* decompose *$\mathcal{G}$ if*

$$V = A \cup B \cup S$$

and

- *S separates A from B,*
- *S is a* complete *subset of V,*
- *Both $\mathcal{G}_{A\cup S}$ and $\mathcal{G}_{B\cup S}$ are decomposable.*

A, B or S may be the empty set. If both A and B are non empty, then the decomposition is proper.

A triple (A, B, S) of disjoint subsets of the node set V of an undirected graph is said to form a weak decomposition *of $\mathcal{G}$ or to weakly decompose $\mathcal{G}$ if $V = A \cup B \cup S$, S separates A from B and both $\mathcal{G}_{A\cup S}$ and $\mathcal{G}_{B\cup S}$ are weakly decomposable.*

A *weak decomposition* differs from a decomposition in that the separator set S is not necessarily complete. Clearly, every graph can be decomposed to its connected components (Definition 1.6). If the graph is undirected, then the connected components are the strong components (Definition 7.11).

Definition 7.18 (Decomposable Graph)**.** *An undirected graph $\mathcal{G}$ is* decomposable *if either*

1. *it is complete, or*
2. *it possesses a proper decomposition (A, B, S) such that both sub graphs $\mathcal{G}_{A\cup S}$ and $\mathcal{G}_{B\cup S}$ are decomposable.*

This is a *recursive* definition, which is permissible, since the decomposition (A, B, S) is required to be proper, so that $\mathcal{G}_{A\cup S}$ and $\mathcal{G}_{B\cup S}$ have fewer nodes than the original graph $\mathcal{G}$.

Example 7.19 (Decomposable Graph)**.**

Consider the graph in Figure 7.7. In the first stage, set $S = \{\alpha_3\}$, with $A = \{\alpha_1, \alpha_2\}$ and $B = \{\alpha_4, \alpha_5, \alpha_6\}$. Then S is a maximal clique and S separates A from B. Then $A \cup S = \{\alpha_1, \alpha_2, \alpha_3\}$ and $\mathcal{G}_{A\cup S}$ is a maximal clique. $B \cup S = \{\alpha_3, \alpha_4, \alpha_5, \alpha_6\}$. The graph $\mathcal{G}_{B\cup S}$ is decomposable; take $S_2 = \{\alpha_3, \alpha_5\}$, $A_2 = \{\alpha_4\}$ and $B_2 = \{\alpha_6\}$. Then $\mathcal{G}_{A_2\cup S_2}$ and $\mathcal{G}_{B_2\cup S_2}$ are maximal cliques. □

Theorem 7.20. *Let $\mathcal{G} = (V, U)$ be an undirected graph. The following conditions 1), 2) and 3) are equivalent.*

1. *$\mathcal{G}$ is decomposable.*
2. *$\mathcal{G}$ is triangulated.*
3. *For every pair of nodes $(\alpha, \beta) \in V \times V$, their minimal separator is complete.*

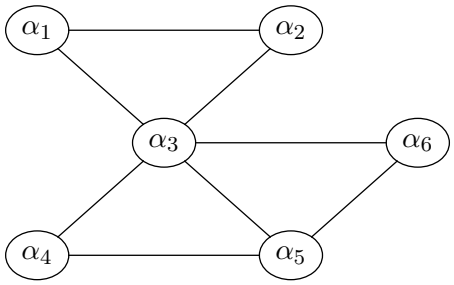

Figure 7.7: Example of a Decomposable Graph

Proof of Theorem 7.20: 1) $\Longrightarrow$ 2) **Inductive hypothesis**: All undirected decomposable graphs with n nodes or less are triangulated. This is true for one node.
Let $\mathcal{G}$ be a decomposable graph with $n+1$ nodes. There are two alternatives:
Either $\mathcal{G}$ is complete, in which case it is triangulated,
Or: by the definition of decomposable, there are three disjoint subsets A, B, S such that S is a complete subset, S separates A from B, $V = A \cup B \cup S$ and $\mathcal{G}_{A\cup S}$ and $\mathcal{G}_{B\cup S}$ are decomposable. The decomposition is proper, hence $\mathcal{G}_{A\cup S}$ and $\mathcal{G}_{B\cup S}$ have less than or equal to n nodes. Therefore, by the *inductive hypothesis* $\mathcal{G}_{A\cup S}$ and $\mathcal{G}_{B\cup S}$ are triangulated. Therefore, a cycle of length ≥ 4 without a chord, will be a cycle from A which passes through B. By decomposability, S separates A from B and therefore any such cycle must pass S at least twice. But then this cycle has a chord, since S is a complete subset. □

Proof of Theorem 7.20: 2) $\Longrightarrow$ 3) Assume that $\mathcal{G} = (V, U)$ is an undirected, triangulated graph. Let S be a minimal separator for two nodes α and β. Let A denote the set such that $\alpha \in A$ and $\mathcal{G}_A$ is the largest connected sub-graph of $\mathcal{G}_{V\backslash S}$ such that α is in the node set. Let $B = V\backslash(A \cup S)$. For every node $\gamma \in S$, there is a node $\tau \in A$ such that $\langle \gamma, \tau \rangle \in U$ and there is a node $\sigma \in B$ such that $\langle \gamma, \sigma \rangle \in U$. Otherwise $S\backslash\{\gamma\}$ would be a separator for α and β, contradicting the minimality of S. Hence, for any pair $(\gamma, \delta) \in S \times S$, there exist paths $\gamma, \tau_1, \ldots, \tau_m, \delta$ and $\gamma, \sigma_1, \ldots, \sigma_n, \delta$ where all the nodes $\{\tau_1, \ldots, \tau_m\}$ are in A and all the nodes $\{\sigma_1, \ldots \sigma_n\}$ are in B. Then $\gamma, \tau_1, \ldots, \tau_m, \delta, \sigma_n, \ldots, \sigma_1, \gamma$ is a cycle of length ≥ 4 and therefore has a chord. Assume that $\tau_1, \ldots, \tau_m$ and $\sigma_1, \ldots, \sigma_n$ have been chosen so that the paths are as short as possible (that is, there is no shorter path from γ to δ with all intervening nodes in A and no shorter path from γ to δ with all intervening nodes in B).

The chord cannot be of the form $\langle \tau_i, \tau_j \rangle$ for some (i, j) or $\langle \sigma_k, \sigma_l \rangle$ for any (k, l) because of the minimality of the lengths of the chosen paths. Therefore, $\langle \gamma, \delta \rangle \in U$. Therefore, γ and δ are adjacent for every pair $(\gamma, \delta) \in S \times S$. It follows that S is complete. □

Proof of Theorem 7.20: 3) $\Longrightarrow$ 1) If $\mathcal{G}$ is complete, then the result is clear. If $\mathcal{G}$ is not complete, then choose two distinct nodes $(\alpha, \beta) \in V \times V$ that are not adjacent. Let $S \subseteq V\backslash\{\alpha, \beta\}$ denote the minimal separator for the pair (α, β). Let A denote the node set of the maximal connected component

of $\mathcal{G}_{V\backslash S}$ and let $B = V\backslash(A \cup S)$. Then (A, B, S) provides three disjoint subsets, where S is complete. We have to show that $\mathcal{G}_{A\cup S}$ and $\mathcal{G}_{B\cup S}$ are decomposable. The procedure can be repeated on both $\mathcal{G}_{A\cup S}$ and $\mathcal{G}_{B\cup S}$ and repeated recursively, stopping when $\mathcal{G}_{A'\cup S'}$ is complete for a set A' and corresponding separator S', hence the graph is decomposable. □

Definition 7.21 (Perfect Node Elimination Sequence)**.** *Let $V = \{\alpha_1, \ldots, \alpha_d\}$ denote the node set of a graph $\mathcal{G}$. A* perfect node elimination sequence *of a graph $\mathcal{G}$ is an ordering of the node set $\{\alpha_1, \ldots, \alpha_d\}$ such that for each j in $1 \leq j \leq d-1$, α_j is a simplicial node of the sub graph of $\mathcal{G}$ induced by $\{\alpha_j, \alpha_{j+1}, \ldots, \alpha_d\}$*

Lemma 7.22. *Every triangulated graph $\mathcal{G}$ has a simplicial node. Moreover, if $\mathcal{G}$ is* not *complete, then it has two* non adjacent *simplicial nodes.*

Proof The lemma is trivial if either $\mathcal{G}$ is complete, or else $\mathcal{G}$ has two or three nodes. Assume that $\mathcal{G}$ is not complete. Suppose the result is true for all graphs with fewer nodes than $\mathcal{G}$. Consider two non adjacent nodes α and β. Let S denote the minimal separator of α and β. Let $\mathcal{G}_A$ denote the largest connected component of $\mathcal{G}_{V\backslash S}$ such that $\alpha \in A$ and let $B = V\backslash(A \cup S)$, so that $\beta \in B$.

By induction, either $\mathcal{G}_{A\cup S}$ is complete, or else it has two non adjacent simplicial nodes. Since $\mathcal{G}_S$ is complete, it follows that at least one of the two simplicial nodes is in A. Such a node is therefore also simplicial in $\mathcal{G}$, because none of its neighbours is in B.

If $\mathcal{G}_{A\cup S}$ is complete, then any node of A is a simplicial node of $\mathcal{G}$.

In all cases, there is a simplicial node of $\mathcal{G}$ in A. Similarly, there is a simplicial node in B. These two nodes are then non adjacent simplicial nodes of $\mathcal{G}$. □

Theorem 7.23. *A graph $\mathcal{G}$ is* triangulated *if and only if it has a perfect node elimination sequence.*

Proof Suppose that $\mathcal{G}$ is triangulated. Assume that every triangulated graph with fewer nodes than $\mathcal{G}$ has a perfect elimination sequence. By the previous lemma, $\mathcal{G}$ has a simplicial node α. Removing α returns a triangulated graph. (Consider any cycle of length ≥ 4 with a chord. If the cycle remains after the node is removed, then the chord is not removed). By proceeding inductively, it follows that $\mathcal{G}$ has a perfect elimination sequence.

Conversely, assume that $\mathcal{G}$ has a perfect sequence, say $\{\alpha_1, \ldots, \alpha_d\}$. Consider any cycle of length ≥ 4. Let j be the first index such that α_j is in the cycle. Let $V(C)$ denote the node set of the cycle and let $V_j = \{\alpha_j, \ldots, \alpha_d\}$. Then $V(C) \in V_j$. Since α_j is simplicial in $\mathcal{G}_{V_{j+1}}$, the neighbours of α_j in the cycle are adjacent, hence the cycle has a chord. Therefore $\mathcal{G}$ is triangulated. □

Definition 7.24 (Eliminating a Node)**.** *Let $\mathcal{G} = (V, E)$ be an undirected graph. A node α is eliminated from an undirected graph $\mathcal{G}$ in the following way:*

1. *For all pairs of neighbours (β, γ) of α add a link if $\mathcal{G}$ does not already contain one. The added links are called* fill ins.

2. *Remove α.*

The resulting graph is denoted by $\mathcal{G}^{-\alpha}$.

For example, consider the graph in Figure 7.8. This graph is already triangulated. But suppose one did not notice this and one decided to eliminate node α_3 from the graph in Figure 7.8. The resulting graph is given in Figure 7.9.

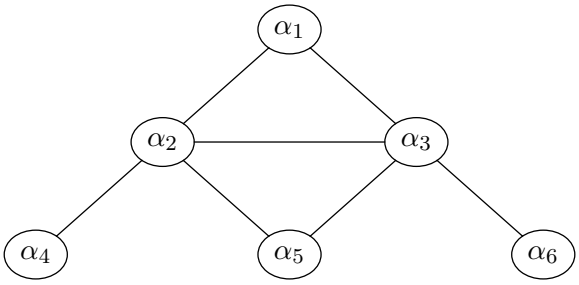

Figure 7.8: Example for Eliminating a Node

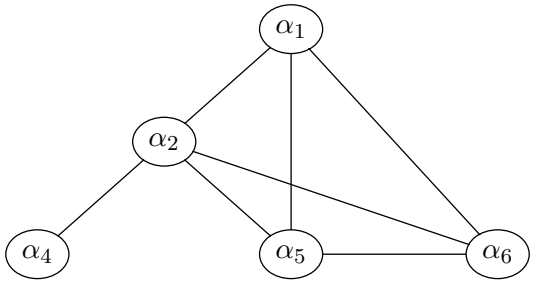

Figure 7.9: Graph 7.8 with α_3 Eliminated

Definition 7.25 (Elimination Sequence)**.** *An* elimination sequence *of* $\mathcal{G}$ *is a linear ordering of its nodes.*

Let σ be an elimination sequence and let Λ denote the fill ins produced by eliminating a node of $\mathcal{G}$ in the order σ. Denote by $\mathcal{G}^{\sigma}$ the graph $\mathcal{G}$ extended by Λ.

Example 7.26.

Consider the graph in Figure 7.8. Suppose the elimination sequence $\alpha_3, \alpha_2, \alpha_4, \alpha_5, \alpha_6$ is employed. Then the fill ins, for each stage, will be $\langle\alpha_1, \alpha_6\rangle, \langle\alpha_1, \alpha_5\rangle, \langle\alpha_2, \alpha_6\rangle$, $\langle\alpha_5, \alpha_6\rangle$ for α_3, then $\langle\alpha_1, \alpha_4\rangle, \langle\alpha_4, \alpha_6\rangle$ for α_2. No further fill ins are required. The graph $\mathcal{G}^{\sigma}$ is given in Figure 7.10.

Definition 7.27 (Elimination sequence, elimination domains)**.** *An elimination sequence* σ *is a linear ordering of the set of nodes* $V = \{\alpha_1, \ldots, \alpha_d\}$ *where for each* $\alpha \in \{1, \ldots, d\}$, $\sigma(\alpha)$ *denotes the number assigned to variable* X_α. *A node* β *is said to be of higher elimination order than* α *if* $\sigma(\beta) > \sigma(\alpha)$. *The* elimination domain *of a node* α *is the set of neighbours of* α *of higher elimination order.*

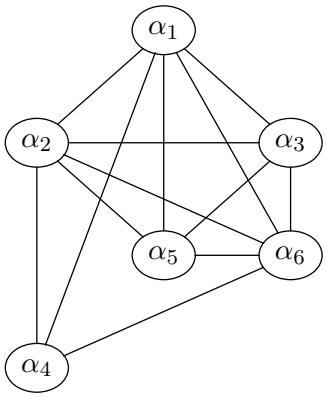

Figure 7.10: $\mathcal{G}^\sigma$. Elimination sequence $(\alpha_3, \alpha_2, \alpha_4, \alpha_1, \alpha_5, \alpha_6)$

In $\mathcal{G}^\sigma$, any node α together with its neighbours of higher elimination order form a *complete* subset. The neighbours of α of higher elimination order are denoted by $N_{\sigma(\alpha)}$. The sets $N_{\sigma(\alpha)}$ are the *elimination domains* corresponding to the elimination sequence σ.

An efficient algorithm clearly tries to minimise the number of fill ins. If possible, one should find an elimination sequence that does not introduce fill ins.

Proposition 7.28. *All maximal cliques in a $\mathcal{G}^\sigma$ are a $N_{\sigma(\alpha)}$ for some $\alpha \in V$.*

Proof Let C be a maximal clique in $\mathcal{G}^\sigma$ and let α be a variable in C of the lowest elimination order. Then $C = N_{\sigma(\alpha)}$. □

An efficient algorithm ought to find an elimination sequence for the domain graph that yields maximal cliques of minimal total size.

The following proposition is clear.

Proposition 7.29. *Any $\mathcal{G}^\sigma$ is a triangulation of $\mathcal{G}$.*

Proof By construction, the elimination sequence σ for graph $\mathcal{G}^\sigma$ does not require any fill-ins. □

Recall that a graph is *triangulated* if and only if it has an elimination sequence without fill ins. This is equivalent to the statement that an undirected graph is triangulated if and only if all nodes can be eliminated by successively eliminating a node α such that the family $F_\alpha = \{\alpha\} \cup N_\alpha$ is complete. From the definition, such a node α is a *simplicial node*.

Marginalisation and triangulation of graphs Let $\mathcal{G} = (V, U)$ be an undirected graph, where $V = \{X_1, \ldots, X_d\}$. Recall the definition of the *domain graph* (Definition 7.5) and note that the *maximal*

cliques of the domain graph are the domains of the functions of the charge. Recall Definition 7.7, which describes the procedure for eliminating a variable in a marginalisation. When a node X_v is eliminated from the graph $\mathcal{G}$, the resulting graph is denoted by $\mathcal{G}^{-X_v}$. Graphically, the procedure described in Definition 7.7 is the same as Definition 7.24, eliminating a node. If $\mathcal{G}$ is the domain graph for a set of functions Φ, then it is clear from Definition 7.24 that the graph $\mathcal{G}^{-X_v}$ is the *domain graph* for the set of functions Φ^{-X_v}. Therefore, if the domain graph is triangulated, there is a perfect elimination sequence; there is an order for eliminating the variables that, at each stage, the elimination domain corresponds to a maximal clique in the current domain graph.

7.4 Junction Trees

Decomposable graphs provide the basis for one of the key methods for updating a probability distribution described in terms of a Bayesian network. The DAG is moralised and then triangulated using the most efficient triangulation algorithms available. The triangulated graph is then decomposed and organised to form a *junction tree*, which supports a effective algorithms. The purpose of this section is to define junction trees and to show how to construct them. They provide a key tool for updating a Bayesian network.

Definition 7.30 (Junction Trees)**.** *Let $\mathcal{C}$ be a collection of subsets of a finite set V and $\mathcal{T}$ be a tree with $\mathcal{C}$ as its node set. Then $\mathcal{T}$ is said to be a* junction tree *(or* join *tree) if any intersection $C_1 \cap C_2$ of a pair C_1, C_2 of sets in $\mathcal{C}$ is contained in every node on the unique path in $\mathcal{T}$ between C_1 and C_2. Let $\mathcal{G}$ be an undirected graph and $\mathcal{C}$ the family of its maximal cliques. If $\mathcal{T}$ is a junction tree with $\mathcal{C}$ as its node set, then $\mathcal{T}$ is known as* junction tree for the graph $\mathcal{G}$.

Theorem 7.31. *There exists a junction tree $\mathcal{T}$ of maximal cliques for the graph $\mathcal{G}$ if and only if $\mathcal{G}$ is decomposable.*

Proof Firstly, we prove that if the graph is decomposable, then there exists a junction tree of the maximal cliques. The proof is by construction; a sequence is established in the following way. Firstly, a *simplicial node* α is chosen; F_α is therefore a maximal clique. The algorithm continues by choosing nodes from F_α that *only have neighbours in* F_α. The set of nodes F_α is labelled C_1 and the set of those nodes in F_α that have neighbours *not* in F_α is labelled S_1. This set is a *separator*.

Now remove the nodes in F_α that do not have neighbours outside F_α and name the new graph $\mathcal{G}'$. Choose a new node α in the graph $\mathcal{G}'$ such that F_α is a maximal clique. Repeat the process, with the index j, where j is the previous index, *plus* 1.

When the parts have been established (as indicated in the diagram below), each separator S_i is then connected to a maximal clique S_j with $j > i$ and such that $S_i \subset C_j$. This is always possible, because S_i is a *complete* set and, in the elimination sequence described above, the first point of S_i is eliminated when dealing with a maximal clique of index greater than i.

It is necessary to prove that the structure constructed is a tree and that it has the junction tree property.

Firstly, each maximal clique has at most one parent, so there are not multiple paths. The structure is therefore a tree.

To prove the *junction tree* condition, consider two maximal cliques, C_i and C_j with $i > j$ and let α be a member of both. There is a *unique* path between C_i and C_j.

Because α is not eliminated when dealing with C_j, it is a member of S_j. By construction, it is also a member of the child of C_j, say C_k. Arguing similarly, it is also a member of the child of C_k and, by induction it is also a member of C_i and, of course, all the separators in between.

The converse is trivial; if the maximal cliques can be arranged as a junction tree, then we can construct a perfect elimination sequence by: take a simplicial node from a maximal clique which is a leaf of the junction tree and remove the node. If this is not the only simplicial node in the chosen maximal clique, the maximal clique remains as a leaf of the junction tree, otherwise the maximal clique is removed from the junction tree; the resulting maximal clique tree is a junction tree. Hence there is a perfect elimination sequence, hence the graph is triangulated (and decomposable). □

Example 7.32.

Consider the directed acyclic graph in Figure 7.11. The corresponding moral graph is given in Figure 7.12.

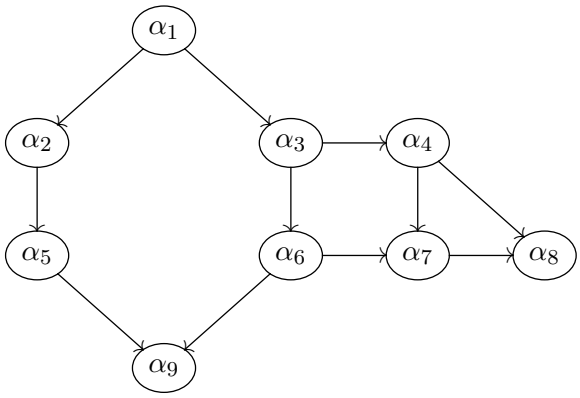

Figure 7.11: A Directed Acyclic Graph

An appropriate elimination sequence for this moral graph is

$$(\alpha_8, \alpha_7, \alpha_4, \alpha_9, \alpha_2, \alpha_3, \alpha_1, \alpha_5, \alpha_6).$$

There are two fill-ins; these are $\langle \alpha_1, \alpha_5 \rangle$ corresponding to the elimination of α_2 and $\langle \alpha_1, \alpha_6 \rangle$, corresponding to the elimination of α_3. The corresponding triangulated graph is given in Figure 7.13.
The junction tree construction may be applied. The maximal cliques and separators, with the labels resulting from the diagram, are shown in Figure 7.14 and put together to form the junction tree, or join tree, shown in Figure 7.15.
Later, when using the algorithm for updating, it will be useful to designate one node as the *root*.

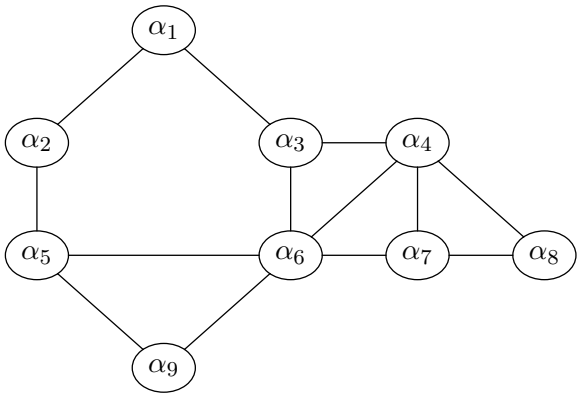

Figure 7.12: Moral Graph corresponding to Figure 7.11

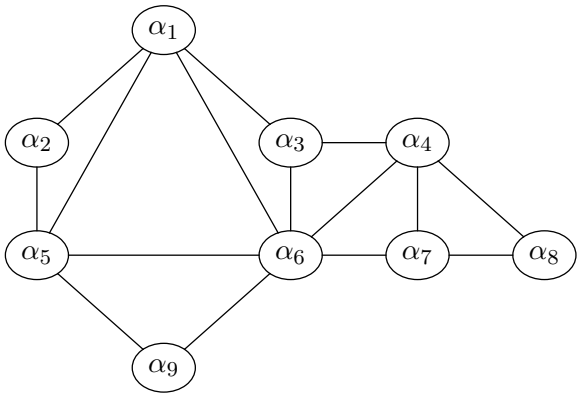

Figure 7.13: The triangulated graph corresponding to Figure 7.12

Definition 7.33 (Rooted Tree)**.** *A* rooted tree $\mathcal{T}$ *is a tree graph with a designated node* ρ *called the* root*. A* leaf *of a tree is a node that is joined to at most one other node.*

7.5 Perfect Orders of Maximal Cliques

Closely associated with the concept of a Junction Tree is the concept of *running intersection property* and *perfect order of maximal cliques.* Let $\mathcal{C} = \{C_1, \ldots, C_n\}$ denote a collection of maximal cliques.

Definition 7.34 (Running Intersection Property)**.** $\mathcal{C}$ *is said to have* running intersection property *(r.i.p.) if there is an order* σ *of* $\{1, \ldots, n\}$ *such that for each* $j \geq 2$ *there is an* l *such that* $\sigma(l) < \sigma(j)$ *and*

$$C_{\sigma(j)} \cap \left(\cup_{i=1}^{j-1} C_{\sigma(i)} \right) \subseteq C_{\sigma(j)} \cap C_{\sigma(l)}. \tag{7.1}$$

An order of the maximal cliques that satisfies r.i.p. is said to be a perfect order *of the maximal cliques.*

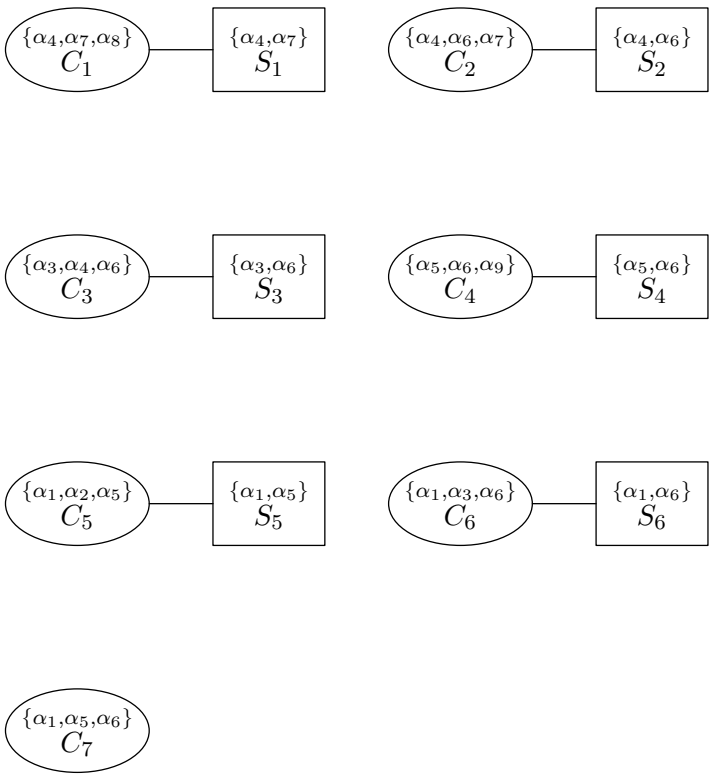

Figure 7.14: The Maximal Cliques and Separators from Figure 7.13

Theorem 7.35. *For an undirected graph $\mathcal{G} = (V, U)$ with maximal cliques $\mathcal{C} = \{C_1, \ldots, C_n\}$, there exists a perfect order of the maximal cliques if and only if $\mathcal{G}$ is triangulated. Furthermore, for any order such that (7.2) holds, the tree constructed by adding the edge $\sigma(j) \sim \sigma(l(j))$, where for each $j \geq 2$ a single $l(j) \in \{1, \ldots, j-1\}$ is chosen such that*

$$C_{\sigma(j)} \cap \left(\cup_{i=1}^{j-1} C_{\sigma(i)} \right) \subseteq C_{\sigma(j)} \cap C_{\sigma(l(j))} \tag{7.2}$$

is a junction tree.

Proof The graph is triangulated if and only if the maximal cliques can be arranged as a junction tree. If there is a perfect order of the maximal cliques, then clearly the method described for constructing a tree from these maximal cliques (edge between $C_{\sigma(j)}$ and maximal clique $C_{\sigma(l(j))}$ such that (7.2) holds) gives the junction tree property; namely, that for any two cliques C_α, C_β, $C_\alpha \cap C_\beta$ is contained in each separator on the unique path $C_\alpha \leftrightarrow C_\beta$ on the tree. On the other hand, if there is a junction tree, then we may choose arbitrarily one node as root, call it $\sigma(1)$ and then proceed by choosing $\sigma(j)$ as any neighbour of $\sigma(1), \ldots, \sigma(j-1)$ that has not yet appeared in the order. This order of the maximal cliques satisfies r.i.p.. □

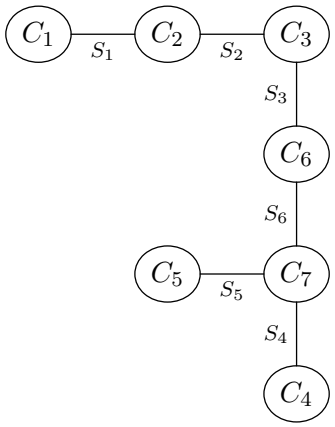

Figure 7.15: A Junction Tree (or join tree) constructed from the triangulated graph in Figure 7.13

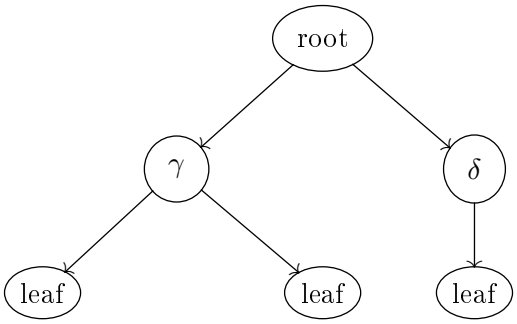

Figure 7.16: Illustration of a Rooted Tree

Notes The material is standard from algorithmic graph theory. See, for example, [55]. The proof of Theorem 7.20 follows the lines of Cowell, Dawid, Lauritzen and Spiegelhalter in [32].

Chapter 8

Junction trees and message passing

The task is to describe a scheme of message passing (propagation) between the maximal cliques of a junction tree to compute the marginal distribution over a set of variables $A \subset V \backslash E$, given hard evidence on a set of variables E; $\{\underline{X}_E = \underline{x}_E\}$;

$$\mathbb{P}_{V\backslash E|E}(\underline{x}_{V\backslash E}|\underline{x}_E)^{\downarrow A} = \left(\sum_{\underline{x}_{V\backslash(A\cup E)} \in \mathcal{X}_{V\backslash A}} \mathbb{P}_{V\backslash E|E}(\underline{x}_A, \underline{x}_{\tilde{V}\backslash(A\cup E)}|\underline{x}_E) \right).$$

The message passing algorithm described here is the one used by the R packages **gRain** and **bnlearn** and also many other software programmes that deal with Bayesian Networks; the algorithm is based on representing joint distribution of a Bayesian network using the so - called *Aalborg formula*

$$\mathbb{P}_{\underline{X}}(x_1, \ldots, x_n) = \frac{\prod_{C\in\mathcal{C}} \phi_C(\underline{x}_C)}{\prod_{S\in\mathcal{S}} \phi_S(\underline{x}_S)},$$

(which will be established later in this section), where

$$\mathcal{C} = \text{maximal cliques of the triangulated moral graph}$$

and

$$\mathcal{S} = \text{separators of the junction tree}$$

and each ϕ_C and ϕ_S is the function over the respective maximal clique C and separator S. The propagation presented is the approach of Lauritzen and Spiegelhalter, discussed in [83]; the technicalities may differ slightly in the various implementations available.

8.1 Factorisation along an Undirected Graph

Let $\mathcal{G} = (V, U)$ be an undirected graph, where $V = \{X_1, \ldots X_d\}$ is a set of discrete variables.

Definition 8.1. *A joint probability* $\mathbb{P}_{\underline{X}}$ *over a random vector* $\underline{X} = (X_1, \ldots, X_d)$ *is said to be* factorised *according to* $\mathcal{G}$ *if there exist* functions *or* factors, ϕ_A *defined on* $\times_{v\in\tilde{A}}\mathcal{X}_v$ *where* A *is a* complete *set of nodes in* $\mathcal{G}$ *such that*

$$\mathbb{P}_{\underline{X}}(\underline{x}) = \prod_A \phi_A(\underline{x}_A)$$

where the notation is clear (see section 7.1, page 141); the product is over all the functions.

Recall Definition 7.15 of a separator and Definition 7.17 of a decomposition. In the definition, A, B or S may be the empty set, ϕ.

Proposition 8.2. *Let $\mathcal{G}$ be a decomposable undirected graph and let (A, B, S) decompose $\mathcal{G}$. Then the following two statements are equivalent:*

1. *$\mathbb{P}$ factorises along $\mathcal{G}$ and*
2. *both $\mathbb{P}_{A\cup S}$ and $\mathbb{P}_{B\cup S}$ factorise along $\mathcal{G}_{A\cup S}$ and $\mathcal{G}_{B\cup S}$ respectively and*

$$\mathbb{P}(\underline{x}) = \frac{\mathbb{P}_{A\cup S}(\underline{x}_{A\cup S})\mathbb{P}_{B\cup S}(\underline{x}_{B\cup S})}{\mathbb{P}_S(\underline{x}_S)}.$$

Proof of 1) $\Longrightarrow$ 2) Since the graph is decomposable, its maximal cliques can be organised as a junction tree. Hence, without loss of generality, the factorisation can be taken to be of the form

$$\mathbb{P}(\underline{x}) = \prod_{C\in\mathcal{C}} \phi_C(\underline{x}_C),$$

where the product is over the maximal cliques of $\mathcal{G}$. Since (A, B, S) decomposes $\mathcal{G}$, any maximal clique of $\mathcal{G}$ can either be taken as a subset of $A \cup S$ or as a subset of $B \cup S$. Furthermore, S is a *strict* subset of any maximal clique of $A \cup S$ containing S and S is a *strict* subset of any maximal clique of $B \cup S$ containing S. Letting C denote a maximal clique, it follows that

$$\mathbb{P}(\underline{x}) = \prod_{C\subseteq A\cup S} \phi_C(\underline{x}_C) \prod_{C\subseteq B\cup S} \phi_C(\underline{x}_C).$$

Since S is itself complete, it is a subset of any maximal clique containing S, so that no maximal clique in the decomposition will appear in both $A \cup S$ and $B \cup S$. Set

$$h(\underline{x}_{A\cup S}) = \prod_{C\subseteq A\cup S} \phi_C(\underline{x}_C), \qquad k(\underline{x}_{B\cup S}) = \prod_{C\subseteq B\cup S} \phi_C(\underline{x}_C).$$

Then

$$\mathbb{P}(\underline{x}) = h(\underline{x}_{A\cup S})k(\underline{x}_{B\cup S})$$

and the marginal distribution is given by

$$\mathbb{P}_{A\cup S}(\underline{x}_{A\cup S}) = h(\underline{x}_{A\cup S}) \sum_{\mathcal{X}_B} k(\underline{x}_{B\cup S}) = h(\underline{x}_{A\cup S})k_S(\underline{x}_S)$$

and

$$\mathbb{P}_{B\cup S}(\underline{x}_{B\cup S}) = k(\underline{x}_{B\cup S})h_S(\underline{x}_S),$$

where k_S is defined as $k_S(\underline{x}_S) = \sum_{\mathcal{X}_B} k(\underline{x}_{B\cup S})$ and $h_S(\underline{x}_S) = \sum_{\mathcal{X}_A} h(\underline{x}_{A\cup S})$. It follows that

$$\mathbb{P}(\underline{x}) = h(\underline{x}_{A\cup S})k(\underline{x}_{B\cup S}) = \frac{\mathbb{P}(\underline{x}_{A\cup S})\mathbb{P}(\underline{x}_{B\cup S})}{k(\underline{x}_S)h(\underline{x}_S)}.$$

Since

$$\mathbb{P}_S(\underline{x}_S) = \sum_{\mathcal{X}_A} h(\underline{x}_{A\cup S}) \sum_{\mathcal{X}_B} k(\underline{x}_{B\cup S}) = h(\underline{x}_S)k(\underline{x}_S),$$

it follows that

$$\mathbb{P}(\underline{x}) = \frac{\mathbb{P}_{A\cup S}(\underline{x}_{A\cup S})\mathbb{P}_{B\cup S}(\underline{x}_{B\cup S})}{\mathbb{P}_S(\underline{x}_S)}.$$

Since $\mathbb{P}_{A\cup S}(\underline{x}_{A\cup S}) = \prod_{C\subset A\cup S} \phi_C(\underline{x}_C)$ and $\mathbb{P}_{B\cup S}(\underline{x}_{B\cup S}) = \prod_{C\subset B\cup S} \phi_C(\underline{x}_C)$, it follows that $\mathbb{P}_{A\cup S}$ and $\mathbb{P}_{B\cup S}$ factorise along the corresponding graphs. This establishes the proof of 1) $\implies$ 2). □

Proof of 2) $\implies$ 1) If both $\mathbb{P}_{A\cup S}$ and $\mathbb{P}_{B\cup S}$ factorise along $\mathcal{G}_{A\cup S}$ and $\mathcal{G}_{B\cup S}$ respectively and the given formula holds, then

$$\mathbb{P}(\underline{x}) = \frac{1}{\mathbb{P}_S(\underline{x}_S)} \prod_{C\subset A\cup S} \phi_C(\underline{x}_C) \prod_{C\subset B\cup S} \phi_C(\underline{x}_C). \tag{8.1}$$

For the maximal clique C that satisfies $C \subset A \cup S$ such that $C \cap S \neq \phi$ and set $\psi_C = \frac{\phi_C}{p_S}$. For all other C, set $\psi_C = \phi_C$, then

$$\mathbb{P}(\underline{x}) = \prod_{C\subset V} \psi_C(\underline{x}_C),$$

so that $\mathbb{P}$ factorises along $\mathcal{G}$. □

Since $\mathbb{P}_{A\cup S} = \prod_{C\subset A\cup S} \phi_C$ and $\mathbb{P}_{B\cup S} = \prod_{C\subset B\cup S} \phi_C$ in Equation (8.1), it follows by a recursive application of the proposition that

$$\mathbb{P}(\underline{x}) = \frac{\prod_{C\in\mathcal{C}} \mathbb{P}_C(\underline{x}_C)}{\prod_{S\in\mathcal{S}} \mathbb{P}_S(\underline{x}_S)},$$

where $\mathcal{C}$ denotes the set of maximal cliques and $\mathcal{S}$ denotes the set of separators. □

8.2 Factorising along a Junction Tree

Let $\mathbb{P}$ be a probability distribution that factorises along a directed acyclic graph $\mathcal{G} = (V, E)$. The factorisation is given by

$$\mathbb{P}_{\underline{X}}(\underline{x}) = \prod_{v=1}^{d} \mathbb{P}_{X_v|\Pi_v}(x_v|\pi_v(\underline{x})),$$

where $\pi_v(\underline{x})$ denotes the parent set of X_v for an instantiation $\underline{x}$. It is clear that this may be expressed as a factorisation according to the moralised graph $\mathcal{G}^{mor}$, which is undirected:

$$\mathbb{P}_{\underline{X}}(\underline{x}) = \prod_{v=1}^{d} \phi_{A_v}(\underline{x}_{A_v})$$

where $A_v = \{X_v\} \cup \Pi_v$ and

$$\phi_{A_v}(\underline{x}_{A_v}) = \mathbb{P}_{X_v|\Pi_v}(x_v|\pi_v(\underline{x})).$$

Hence a probability distribution factorised along the DAG is also factorised along the moral graph $\mathcal{G}^{mor}$. For implementing algorithms, the problem is that it may not be possible to represent the sets $(A_v)_{v=1}^d$ on a tree. To enable this, $\mathcal{G}^{mor}$ is *triangulated* to give $(\mathcal{G}^{mor})^t$. Recall that $(\mathcal{G}^{mor})^t$ is decomposable and its maximal cliques can be organised into a junction tree $\mathcal{T}$. The probability distribution can clearly be factorised as

$$\mathbb{P}_{\underline{X}}(\underline{x}) = \prod_{C \in \mathcal{C}} \phi_C(\underline{x}_C),$$

where $\phi_C(\underline{x}_C)$ is the product of all those $\mathbb{P}(x_v|\underline{x}_{\tilde{\Pi}_v})$, *all* of whose arguments belong to C. This factorisation is not necessarily unique. It corresponds to a triangulation of the moral graph, where C are the maximal cliques. It follows that

$$\mathbb{P}_{\underline{X}}(\underline{x}) = \frac{\prod_{C \in \mathcal{C}} \mathbb{P}_C(\underline{x}_C)}{\prod_{S \in \mathcal{S}} \mathbb{P}_S(\underline{x}_S)}, \tag{8.2}$$

where $\mathcal{C}$ denotes the set of maximal cliques and $\mathcal{S}$ denotes the set of separators of $(\mathcal{G}^{mor})^t$, which may be organised according to a junction tree. This is the definition of a factorisation along a junction tree.

Definition 8.3 (Factorisation along a Junction Tree, Marginal Charge)**.** *Let $\mathbb{P}_{\underline{X}}$ be a probability distribution over a random vector $\underline{X} = (X_1, \ldots, X_d)$. Suppose that the variables can be organised as a junction tree, with maximal cliques $\mathcal{C}$ and separators $\mathcal{S}$ such that $\mathbb{P}_{\underline{X}}$ has representation given in Equation* (8.2)*, where $\mathbb{P}_C$ and $\mathbb{P}_S$ denote the marginal probability functions over the maximal clique variables $C \in \mathcal{C}$ and separator variables $S \in \mathcal{S}$ respectively. The representation in Equation* (8.2) *is known as the* factorisation along the junction tree*, and the charge*

$$\Phi = \{\mathbb{P}_S : S \in \mathcal{S}, \mathbb{P}_C : C \in \mathcal{C}\}$$

is known as the marginal charge*.*

From the foregoing discussion, it is clear that Definition 8.3 is a special case of Definition 8.1, with appropriate choice of functions in Definition 8.1.

Entering Evidence Equation (8.2) expresses the probability distribution in terms of functions over the maximal cliques and separators of $(\mathcal{G}^{mor})^t$, or the junction tree. Suppose that hard evidence is obtained on the variables U; namely, that for $U \subseteq V$, $\{\underline{X}_U = \underline{y}_U\}$ and the probability over the variables $V \backslash U$ has to be updated accordingly.

The algorithm described below describes a procedure such that for any function $f : \mathcal{X} \to \mathbb{R}_+$ (not necessarily a probability function) that is expressed as

$$f(\underline{x}) = \frac{\prod_{C\in\mathcal{C}} \phi_C(\underline{x}_C)}{\prod_{S\in\mathcal{S}} \phi_S(\underline{x}_S)}, \tag{8.3}$$

for a collection of functions $\Phi = \{\phi_C,\ C \in \mathcal{C},\ \phi_S,\ S \in \mathcal{S}$ where $\mathcal{C}$ and $\mathcal{S}$ are the maximal cliques and separators of a junction tree, the algorithm updates Φ to a collection of functions $\Phi^* = \{f_C,\ C \in \mathcal{C},\ f_S,\ S \in \mathcal{S}$ that satisfy

$$f_C(\underline{x}_C) = \sum_{\underline{z}\in\mathcal{X}_{V\setminus C}} f(\underline{z}, \underline{x}_C)$$

and

$$f_S(\underline{x}_S) = \sum_{\underline{z}\in\mathcal{X}_{V\setminus S}} f(\underline{z}, \underline{x}_S)$$

for each $C \in \mathcal{C}$ and each $S \in \mathcal{S}$. It follows that if the algorithm is applied using

$$\phi_C(\underline{x}_C) = \begin{cases} \mathbb{P}_C(\underline{x}_C) & \underline{x}_{C\cap U} = \underline{y}_{C\cap U} \\ 0 & \underline{x}_{C\cap U} \neq \underline{y}_{C\cap U} \end{cases}$$

and

$$\phi_S(\underline{x}_S) = \begin{cases} \mathbb{P}_S(\underline{x}_S) & \underline{x}_{S\cap U} = \underline{y}_{S\cap U} \\ 0 & \underline{x}_{S\cap U} \neq \underline{y}_{S\cap U} \end{cases}$$

then

$$\mathbb{P}_{\underline{X}_U}(\underline{y}_U) = \sum_{\underline{z}\in\mathcal{X}_{C\setminus(U\cap C)}} f_C(\underline{z}, \underline{y}_{U\cap C}) = \sum_{\underline{z}\in\mathcal{X}_{S\setminus(U\cap S)}} f_S(\underline{z}, \underline{y}_{U\cap S})$$

for all $S \in \mathcal{S}$ and all $C \in \mathcal{C}$. This quantity may therefore be computed by marginalising the maximal clique or separator with the smallest domain; dividing f_C and f_S by this quantity will give $\mathbb{P}_{C|U}(.|\underline{y}_U)$ and $\mathbb{P}_{S|U}(.|\underline{y}_U)$ respectively and hence a representation of the conditional distribution in terms of marginal distributions over the maximal cliques and separators.

8.3 Flow of Messages

8.3.1 First Example

Consider a non-negative function with domain $X \times Y \times Z$, $F : X \times Y \times Z \to \mathbb{R}_+$, which may be written as

$$F(x, y, z) = \frac{f(x, z)g(y, z)}{h(z)}, \tag{8.4}$$

for non negative functions $f : X \times Z \to \mathbb{R}_+$, $g : Y \times Z \to \mathbb{R}_+$ and $h : Z \to \mathbb{R}_+$.
Decomposition (8.4) for the function F is of the form given in Equation (8.3), with maximal cliques $C_1 = \{X, Z\}$, $C_2 = \{Z, Y\}$ and separator $S = \{Z\}$ arranged according to the junction tree in Figure 8.1.

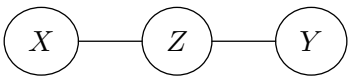

Figure 8.1: Undirected Graph for the Three Variables

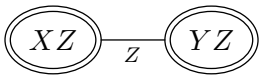

Figure 8.2: Junction Tree for message passing

The following procedure returns a representation $F(x,y,z) = \frac{F_1(x,z)F_2(y,z)}{F_3(z)}$, where

$$F_1(x,z) = \sum_{y\in Y} F(x,y,z), \quad F_2(y,z) = \sum_{x\in X} F(x,y,z), \quad F_3(z) = \sum_{(x,y)\in X\times Y} F(x,y,z).$$

Firstly,

$$F_1(x,z) = \sum_{y\in Y} F(x,y,z) = \sum_{y} \frac{f(x,z)g(y,z)}{h(z)} = \frac{f(x,z)}{h(z)} \sum_{y\in Y} g(y,z).$$

Define the auxiliary function $h^*(z) = \sum_y g(y,z)$, and the update $f^*(x,y) = f(x,y)\frac{h^*(z)}{h(z)}$, then clearly

$$f^*(x,z) = f(x,z)\frac{h^*(z)}{h(z)} = F_1(x,z).$$

The calculation of the marginal function $F_1(x,z)$ by means of the auxiliary function $h^*(z)$ may be described as passing a *local message flow* from ZY to XZ through their separator Z. The factor

$$\frac{h^*(z)}{h(z)}$$

is called the *update ratio*. It follows that

$$F(x,y,z) = \frac{f(x,z)g(y,z)}{h(z)} = \frac{f(x,z)g(y,z)h^*(z)}{h^*(z)h(z)} = F_1(x,z)\frac{1}{h^*(z)}g(y,z).$$

The passage of the flow has resulted in a new representation of $F(x,y,z)$ similar to the original, but where one of the factors is a marginal function.

Similarly, a message can be passed in the other direction, i.e. from XZ to ZY Using the same procedure, set

$$\tilde{h}(z) = \sum_{x\in X} F_1(x,z) = \sum_{(x,y)\in X\times Y} F(x,y,z) = F_3(z).$$

Next, set

$$\tilde{g}(y,z) = g(y,z)\frac{\tilde{h}(z)}{h^*(z)}.$$

It then follows that $\tilde{g}(y,z) = F_2(y,z)$, because

$$F(x,y,z) = F_1(x,z)\frac{1}{\tilde{h}(z)}\tilde{g}(y,z) = F_1(x,z)\frac{1}{F_3(z)}\tilde{g}(y,z)$$

and hence, since $F_3(z) = \sum_{x\in X} F_1(x,z)$, that

$$F_2(y,z) = \sum_{x\in X} F(x,y,z) = \tilde{g}(y,z)\sum_{x\in X} F_1(x,z)\frac{1}{F_3(z)} = \tilde{g}(y,z).$$

□

Passing messages in both directions results in a new overall representation of the function $F(x,y,z)$; .

$$\begin{aligned} F(x,y,z) &= f^*(x,z)\frac{1}{h^*(z)}g(y,z) = f^*(x,z)\frac{1}{h^*(z)}\frac{h^*(z)}{\tilde{h}(z)}\tilde{g}(y,z) \\ &= f^*(x,z)\frac{1}{\tilde{h}(z)}\tilde{g}(y,z) \\ &= F_1(x,z)\frac{1}{F_3(z)}F_2(y,z). \end{aligned}$$

The original representation using functions has been transformed into a new representation where all the functions are marginal functions.

This idea is now extended to arbitrary non negative functions represented on junction trees.

8.4 Local Computation on Junction Trees

Consider a junction tree $\mathcal{T}$ with nodes $\mathcal{C}$ and separators $\mathcal{S}$ and let Φ be a charge

$$\Phi = \{\phi_C : C \in \mathcal{C}, \phi_S : S \in \mathcal{S}\} \tag{8.5}$$

be a charge; that is, a collection of non negative functions such that $\phi_C : \mathcal{X}_C \to \mathbb{R}_+$ and $\phi_S : \mathcal{X}_S \to \mathbb{R}_+$ for each $C \in \mathcal{C}$ and each $S \in \mathcal{S}$.

Definition 8.4 (Contraction of a Charge on a Junction Tree)**.** *The contraction of a charge* (8.5) *over a junction tree is defined as*

$$f(\underline{x}) = \frac{\prod_{C\in\mathcal{C}} \phi_C(\underline{x}_C)}{\prod_{S\in\mathcal{S}} \phi_S(\underline{x}_S)}. \tag{8.6}$$

Local Message Passing Let C_1 and C_2 be two adjacent neighbouring nodes in $\mathcal{T}$ separated by S. Set

$$\phi_S^*(\underline{x}_S) = \sum_{\underline{z} \in \mathcal{X}_{C_1 \setminus S}} \phi_{C_1}(\underline{z}, \underline{x}_S) \tag{8.7}$$

and set

$$\lambda_S = \frac{\phi_S^*}{\phi_S} \tag{8.8}$$

where, by definition, $\frac{0}{0} = 0$ is used in division of functions. λ_S is known as the *update ratio*. The 'message passing' is defined as the operation of updating ϕ_S to ϕ_S^* and ϕ_{C_2} to

$$\phi_{C_2}^* = \lambda_S \phi_{C_2}. \tag{8.9}$$

All other functions remain unchanged. The scheme of local message passing is illustrated in Figure 8.3.

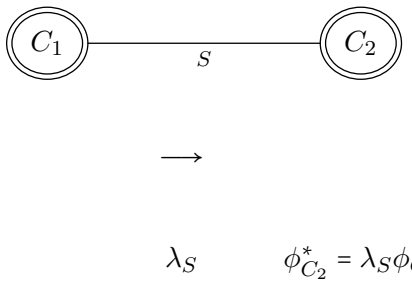

Figure 8.3: Flow from C_1 to C_2

Lemma 8.5. *Let $f : \mathcal{X} \to \mathbb{R}_+$ be the contraction of a charge $\Phi = \{\phi_S,\ S \in \mathcal{S},\ \phi_C,\ C \in \mathcal{C}\}$ on a junction tree (Definition 8.4) where $\mathcal{C}$ is the collection of maximal cliques and $\mathcal{S}$ the collection of separators. A flow does not change the contraction of the charge.*

Proof The initial contraction is given by

$$f(\underline{x}) = \frac{\prod_{C \in \mathcal{C}} \phi_C(\underline{x}_C)}{\prod_{S \in \mathcal{S}} \phi_S(\underline{x}_S)}. \tag{8.10}$$

Let the contraction after the flow be denoted by f^* and the charge, after the flow from C_1 to C_2 denoted by

$$\Phi^* = \{\phi_C^* : C \in \mathcal{C}, \phi_S^* : S \in \mathcal{S}\}$$

Then

$$f^*(\underline{x}) = \frac{\phi_{C_2}^*(\underline{x}_{C_2}) \prod_{C \in \mathcal{C}, C \neq C_2} \phi_C(\underline{x}_C)}{\phi_S^*(\underline{x}_S) \prod_{T \in \mathcal{S}, T \neq S} \phi_S(\underline{x}_T)}. \tag{8.11}$$

There are three cases to consider.

- For $\underline{x}$ such that $\phi_S(\underline{x}_S) > 0$ and $\phi^*_S(\underline{x}_S) > 0$,

$$\frac{\phi^*_{C_2}}{\phi^*_S} = \frac{\phi_{C_2}\lambda_S}{\phi^*_S} = \frac{\phi_{C_2}\left(\frac{\phi^*_S}{\phi_S}\right)}{\phi^*_S} = \frac{\phi_{C_2}}{\phi_S}$$

  and the result is proved.

- For $\underline{x}$ such that $\phi_S(\underline{x}_S) = 0$: it follows that $f(\underline{x}) = 0$ and hence that $\phi_{C_1}(\underline{x}_{C_1}) = 0$ and that $\lambda_S(\underline{x}_S) = 0$. It therefore follows from the definition of $\phi^*_{C_2}$, that $\phi^*_{C_2} = 0$ and hence that $f^*(\underline{x}) = 0$, so that $0 = f^*(\underline{x}) = f(\underline{x})$.

- For $\underline{x}$ such that $\phi_S(\underline{x}_S) > 0$, but $\phi^*_S(\underline{x}_S) = 0$, it follows directly that $\lambda_S(\underline{x}_S) = 0$, so that that $f^*(\underline{x}) = 0$. It remains to show that $f(\underline{x}) = 0$. From the definition,

$$0 = \phi^*_S(\underline{x}_S) = \sum_{\underline{z}\in\mathcal{X}_{C_1\setminus S}} \phi_{C_1}(\underline{z}, \underline{x}_S).$$

  Since $\phi_{C_1}(\underline{x}_{C_1}) \geq 0$ for all $\underline{x}_{C_1} \in \mathcal{X}_{C_1}$, it follows that $\phi_{C_1}(\underline{z}, \underline{x}_S) = 0$ for all $\underline{z} \in \mathcal{X}_{C_1\setminus S}$. Since

$$f(\underline{x}) = \phi_{C_1}(\underline{x}_{C_1})\frac{\phi_{C_2}(\underline{x}_{C_2})}{\phi_S(\underline{x}_S)}\frac{\prod_{C\in\mathcal{C}, C\neq C_1, C_2}\phi_C(\underline{x}_C)}{\prod_{S\in\mathcal{S}, S\neq S}\phi_S(\underline{x}_S)},$$

  it follows directly from the facts that the domains of the maximal cliques other than C_1 and C_2 and separators other than S do not include $\mathcal{X}_S$, and that $\frac{\phi_{C_2}(\underline{x}_{C_2})}{\phi_S(\underline{x}_S)} < +\infty$ that $f(\underline{x}) = 0$, hence $f(\underline{x}) = f^*(\underline{x})$.

In all cases, it follows that a flow does not change the contraction of a charge. □

8.5 Schedules

The aim of this section is to describe how to construct a series of transmissions between the various maximal cliques of a junction tree, to update a set of functions to a set of functions that have the same contraction as the original and which are the marginals of the contraction over the maximal cliques and separators. First, some definitions and notations are established.

Definition 8.6 (Sub-tree, Neighbouring Clique)**.** *A* sub-tree $\mathcal{T}'$ *of a junction tree* $\mathcal{T}$ *is a connected set of nodes of* $\mathcal{T}$ *together with the edges in* $\mathcal{T}$ *between them.*

A maximal clique C *of a junction tree* $\mathcal{T}$ *is a* neighbour *of a sub-tree* $\mathcal{T}'$ *if the corresponding node of* $\mathcal{T}$ *is not a node of* $\mathcal{T}'$ *but is connected to* $\mathcal{T}'$ *by an edge of* $\mathcal{T}$*.*

The following definition gives the technical terms that will be used.

Definition 8.7 (Schedule, Active Flow, Fully Active Schedule)**.** *A* schedule *is an ordered list of directed edges of* $\mathcal{T}$ *specifying which flows are to be passed and in which order.*

A flow is said to be active *relative to a schedule if before it is sent the source has already received active flows from all its neighbours in* $\mathcal{T}$*, with the exception of the* sink*; namely, the node to which it*

is sending its flow. A schedule is full *if it contains an active flow in each direction along every edge of the tree $\mathcal{T}$. A schedule is* active *if it contains* only *active flows. It is* fully active *if it is both full and active.*

It follows from this definition that the first active flow must originate in a leaf of $\mathcal{T}$.

Example 8.8 (Fully active schedule)**.**

Figure 8.4 shows a DAG and 8.5 a corresponding junction tree. An example of a fully active schedule for the junction tree given in Figure 8.5, where the maximal clique BEL is chosen as the root, would be:

$$AT \to ELT, BLS \to BEL, BDE \to BEL, EK \to ELT, ELT \to BEL$$

$$BEL \to ELT, ELT \to EK, ELT \to AT, BEL \to BLS, BEL \to BDE.$$

□

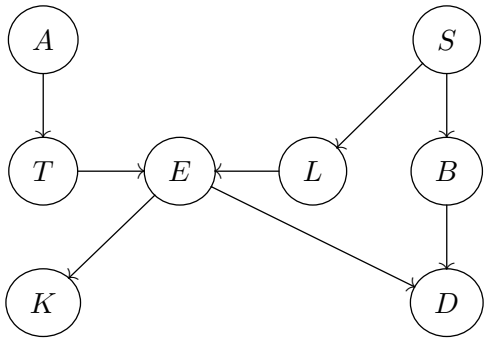

Figure 8.4: Example of a DAG

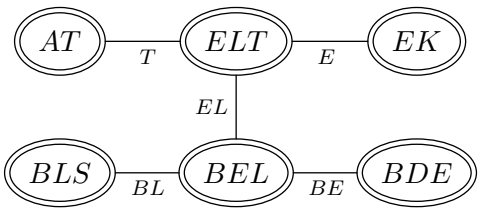

Figure 8.5: Corresponding Junction Tree

Proposition 8.9. *For any tree $\mathcal{T}$, there exists a fully active schedule.*

Proof If there is only one maximal clique, the proposition is clear; no transmissions are necessary. Assume that there is more than one maximal clique. Let C_0 denote a leaf in $\mathcal{T}$. Let $\mathcal{T}_0$ be a sub-tree of $\mathcal{T}$ obtained by removing C_0 and the corresponding edge S_0. Assume that the proposition is true for $\mathcal{T}_0$. Adding the edge

$$C_0 \rightarrow S_0 \rightarrow \mathcal{T}_0$$

to the beginning of the schedule and

$$C_0 \leftarrow S_0 \leftarrow \mathcal{T}_0$$

to the end of the schedule provides a fully active schedule for $\mathcal{T}$. □

The aim is to show that after the passage of a fully active schedule of flows over a junction tree, the resulting charge is the *marginal charge*. That is, all the functions of the charge are the marginal of the contraction of the charge over the respective maximal cliques and separators. Furthermore, there is *global* consistency after the passage of a fully active schedule of flows over a junction tree. This will be defined later, but loosely speaking, it means that if there are several apparent ways to compute a probability distribution over a set of variables using the functions of the marginal charge, they will all give the same answer.

Definition 8.10 (The Base of a Sub-tree, Restriction of a Charge, Live Sub-tree)**.** *Let $\mathcal{T}'$ be a sub-tree of $\mathcal{T}$, with nodes $\mathcal{C}' \subseteq \mathcal{C}$ and edges $\mathcal{S}' \subseteq \mathcal{S}$. The* base *of $\mathcal{T}'$ is defined as the set of variables*

$$U' := \cup_{C \in \mathcal{C}'} C.$$

Let

$$\Phi = \{\phi_C : C \in \mathcal{C}, \phi_S : S \in \mathcal{S}\}$$

be a charge for $\mathcal{T}$. Its restriction *to $\mathcal{T}'$ is defined as*

$$\Phi_{\mathcal{T}'} = \{\phi_C : C \in \mathcal{C}', \phi_S \in \mathcal{S}'\}.$$

Recall Definition 8.4. The contraction *of $\Phi_{\mathcal{T}'}$ is defined as*

$$\frac{\prod_{C \in \mathcal{C}'} \phi_C(\underline{x}_C)}{\prod_{S \in \mathcal{S}'} \phi_S(\underline{x}_S)}.$$

A sub-tree $\mathcal{T}'$ is said to be live *with respect to the schedule of flows if it has already received active flows from all its neighbours.*

Proposition 8.11. *Let*

$$\Phi^0 = \{\phi^0_C : C \in \mathcal{C}, \phi^0_S : S \in \mathcal{S}\}$$

denote an initial charge for a function f that has factorisation

$$f(\underline{x}) = \frac{\prod_{C \in \mathcal{C}} \phi^0_C(\underline{x}_C)}{\prod_{S \in \mathcal{S}} \phi^0_S(\underline{x}_S)}$$

where $\mathcal{C}$ and $\mathcal{S}$ are the sets of maximal cliques and separators for a junction tree $\mathcal{T}$. Suppose that Φ^0 is modified by a sequence of flows according to some schedule. Then, whenever $\mathcal{T}'$ is live, the contraction of the charge for $\mathcal{T}'$ is the margin of the contraction f of the charge for $\mathcal{T}$ on U'.

Proof Assume that $\mathcal{T}' \subset \mathcal{T}$ and that $\mathcal{T}'$ is live. Let C^* denote that last neighbour to have passed a flow into $\mathcal{T}'$. Let $\mathcal{T}^*$ be the sub-tree obtained by adding C^* and the associated edge S^* to $\mathcal{T}'$. Let $\mathcal{C}^*, \mathcal{S}^*$ and U^* be the maximal cliques, separators and the base of $\mathcal{T}^*$. By the junction tree property of $\mathcal{T}$, the separator associated with the edge S^* joining C^* to $\mathcal{T}'$ is

$$S^* = C^* \cap U'.$$

Also,

$$\mathcal{C}^* = \mathcal{C}' \cup \{C^*\} \qquad \text{and} \qquad \mathcal{S}^* = \mathcal{S}' \cup \{S^*\}.$$

and the set of base variables for $\mathcal{T}^*$ is

$$U^* = U' \cup C^*.$$

The induction hypothesis: The assertion holds for the contraction of the charge on $\mathcal{T}^*$. That is, using

$$f_{U^*}(\underline{x}_{U^*}) = \sum_{U \setminus U^*} f(\underline{x}),$$

Let

$$\Phi = \{\phi_C : C \in \mathcal{C}, \phi_S : S \in \mathcal{S}\}$$

denote the charge *just before the last flow from $\mathcal{C}^*$ into $\mathcal{T}'$*. It follows that

$$f_{U^*}(\underline{x}_{U^*}) = \frac{\prod_{C \in \mathcal{C}^*} \phi_C(\underline{x}_C)}{\prod_{S \in \mathcal{S}^*} \phi_S(\underline{x}_S)} = \frac{\phi_{C^*}(\underline{x}_{C^*})}{\phi_{S^*}(\underline{x}_{S^*})} \frac{\prod_{C \in \mathcal{C}'} \phi_C(\underline{x}_C)}{\prod_{S \in \mathcal{S}'} \phi_S(\underline{x}_S)}. \tag{8.12}$$

Lemma 8.5 states that a flow does not change the contraction of a charge. Let $\{\phi^*_C,\ C \in \mathcal{C}',\ \phi^*_S,\ S \in \mathcal{S}'\}$ are the updated functions over the maximal cliques and separators after the flow.
The aim is to find the marginal $f_{U'} = \sum_{U \setminus U'} f$ of f on U' and to show that after the flow,

$$f_{U'}(\underline{x}_{U'}) = \frac{\prod_{C \in \mathcal{C}'} \phi^*_C(\underline{x}_C)}{\prod_{S \in \mathcal{S}'} \phi^*_S(\underline{x}_S)}.$$

Note that $\phi_{C^*} = \phi^*_{C^*}$ and that $\phi^*_{S^*} = \sum_{C^* \setminus S^*} \phi_{C^*}$. It follows that

$$f_{U'} = \sum_{U^* \setminus U'} f_{U^*} = \left(\frac{\sum_{C^* \setminus S^*} \phi_{C^*}(\underline{x}_{C^*})}{\phi^*_{S^*}(\underline{x}_{S^*})} \right) \frac{\prod_{C \in \mathcal{C}'} \phi^*_C(\underline{x}_C)}{\prod_{S \in \mathcal{S}'} \phi^*_S(\underline{x}_S)} = \frac{\prod_{C \in \mathcal{C}'} \phi^*_C(\underline{x}_C)}{\prod_{S \in \mathcal{S}'} \phi^*_S(\underline{x}_S)}$$

and the proof is complete. □

Corollary 8.12. *Let $\{\phi_C, C \in \mathcal{C},\ \phi_S, S \in \mathcal{S}\}$ denote the current functions over the maximal cliques and separators. For any set $A \subseteq V$, let $f_A = \sum_{\mathcal{X}_{V\setminus A}} f$; the marginal over A. Whenever a maximal clique C is live, its corresponding function is $\phi_C = f_C = \sum_{\mathcal{X}_{V\setminus C}} f$.*

Proof A single maximal clique is a sub-tree. The result is immediate from the theorem. □

Corollary 8.13. *Using the notation of Corollary 8.12, whenever active flows have passed in both directions across an edge in $\mathcal{T}$, the function for the associated separator is $\phi_S = f_S = \sum_{\mathcal{X}_{V\setminus S}} f$.*

Proof The function ϕ_S for the associated separator is, by definition of the update,

$$\phi_S = \sum_{\mathcal{X}_{C\setminus S}} \phi_C,$$

so that

$$\sum_{\mathcal{X}_{C\setminus S}} \phi_C = \sum_{\mathcal{X}_{C\setminus S}} f_C = f_S,$$

because ϕ_C is f_C by the previous corollary. □

Proposition 8.14 (The Main Result)**.** *After passage of a fully active schedule of flows, the resulting charge is the marginal charge Φ and its contraction represents f. In other words, the following formula, known as the* Aalborg formula *holds;*

$$f(\underline{x}) = \frac{\prod_{C\in\mathcal{C}} f_C(\underline{x}_C)}{\prod_{S\in\mathcal{S}} f_S(\underline{x}_S)}.$$

Proof This follows from the previous two corollaries and Lemma 8.5, stating that the contraction is unaltered by the flows. □

8.6 Local and Global Consistency

Recall that $\mathcal{T}$ denotes the junction tree, the set of maximal cliques which form the nodes of $\mathcal{T}$ is denoted $\mathcal{C}$ and the *intersection of neighbours* in the tree $\mathcal{T}$ are the *separators*, denoted by $\mathcal{S}$. Recall that the functions associated with $C \in \mathcal{C}$ and $S \in \mathcal{S}$ are denoted by ϕ_C and ϕ_S respectively, and that the *charge* on $\mathcal{T}$, Φ is defined as:

$$\Phi = \{\phi_C : C \in \mathcal{C}, \phi_S : S \in \mathcal{S}\}.$$

Definition 8.15 (Local Consistency)**.** *A junction tree $\mathcal{T}$ is said to be* locally consistent *if whenever $C_1 \in \mathcal{C}$ and $C_2 \in \mathcal{C}$ are two* neighbours *with separator $S = C_1 \cap C_2$, then*

$$\sum_{\mathcal{X}_{C_1\setminus(C_1\cap C_2)}} \phi_{C_1} = \phi_S = \sum_{\mathcal{X}_{C_2\setminus(C_1\cap C_2)}} \phi_{C_2}.$$

Definition 8.16 (Global Consistency)**.** *A junction tree $\mathcal{T}$ (or its charge) is said to be* globally consistent *if for* every $C_1 \in \mathcal{C}$ *and* $C_2 \in \mathcal{C}$ *it holds that*

$$\sum_{\mathcal{X}_{C_1 \setminus (C_1 \cap C_2)}} \phi_{C_1} = \sum_{\mathcal{X}_{C_2 \setminus (C_1 \cap C_2)}} \phi_{C_2}.$$

Global consistency means that the marginalisation to $C_1 \cap C_2$ of ϕ_{C_1} and ϕ_{C_2} coincide for every C_1 and C_2 in $\mathcal{C}$. The following results show that, for a junction tree, local consistency implies global consistency.

Proposition 8.17. *After a passage of a fully active schedule of flows, a junction tree $\mathcal{T}$ is locally consistent.*

Proof The two corollaries of the main result give that for any two neighbouring C_1 and C_2,

$$\sum_{C_1 \setminus S} f_{C_1} = f_S = \sum_{C_2 \setminus S} f_{C_2}.$$

□

An equilibrium, or fixed point has been reached, in the sense that any new flows passed after passage of a fully active schedule do not alter the functions. The update ratio for another message from C_1 to C_2 becomes

$$\lambda_S = \frac{\sum_{C_1 \setminus S} f_{C_1}}{f_S} = 1.$$

Global Consistency of Junction Trees In this paragraph, it is shown that for *junction trees*, local consistency implies global consistency.

By definition, a junction tree is a tree such that the intersection $C_1 \cap C_2$ of any pair C_1 and C_2 in $\mathcal{C}$ is contained in *every* node on the *unique* trail in $\mathcal{T}$ between C_1 and C_2. The set $C_1 \cap C_2$ can be empty and, in this case it is therefore (by convention) a subset of every other set.

Proposition 8.18. *A locally consistent junction tree is globally consistent.*

Proof In a junction tree the intersection $C_1 \cap C_2$ of any pair C_1 and C_2 in $\mathcal{C}$ is contained in *every* node on the unique path in $\mathcal{T}$ between C_1 and C_2. Assume that $C_1 \cap C_2$ is non empty. Consider the unique path from C_1 to C_2. Let the nodes on the path be denoted by $\{C^{(i)}\}_{i=0}^{n}$ with $C^{(0)} = C_1$ and $C^{(n)} = C_2$, so that $C^{(i)}$ and $C^{(i+1)}$ are neighbours. Denote the separator between $C^{(i)}$ and $C^{(i+1)}$ by

$$S^{(i)} = C^{(i)} \cap C^{(i+1)}.$$

Then, for all i,

$$C_1 \cap C_2 \subseteq S^{(i)}.$$

For a set of variables C, let $\sum_C$ denote $\sum_{\mathcal{X}_C}$. The assumption of local consistency means that for any two neighbours

$$\sum_{C^{(i)}\setminus S^{(i)}} \phi_{C^{(i)}} = \sum_{C^{(i)}\setminus(C^{(i+1)}\cap C^{(i)})} \phi_{C^{(i)}} = \sum_{C^{(i+1)}\setminus(C^{(i)}\cap C^{(i+1)})} \phi_{C^{(i+1)}} = \sum_{C^{(i+1)}\setminus S^{(i)}} \phi_{C^{(i+1)}} = \phi_{S^{(i)}}.$$

Starting with the leftmost marginalisation,

$$\begin{aligned}\sum_{C^{(i)}\setminus(C_1\cap C_2\cap C^{(i)})} \phi_{C^{(i)}} &= \sum_{S^{(i)}\setminus(C_1\cap C_2\cap S^{(i)})} \sum_{C^{(i)}\setminus S^{(i)}} \phi_{C^{(i)}} \\ &= \sum_{S^{(i)}\setminus(C_1\cap C_2\cap S^{(i)})} \left(\sum_{C^{(i+1)}\setminus S^{(i)}} \phi_{C^{(i+1)}} \right) = \sum_{C^{(i+1)}\setminus(C_1\cap C_2\cap C^{(i+1)})} \phi_{C^{(i+1)}}.\end{aligned}$$

The marginalisation of ϕ_{C_1} and $\phi_{C^{(1)}}$ coincide. The procedure is continued along the path until the node C_2 is reached. The result is proved. □

Corollary 8.19. *After the passage of a fully active schedule of flows, a junction tree is globally consistent.*

Proof This follows from the proposition stating that after passage of a fully active schedule of flows a junction tree $\mathcal{T}$ is *locally* consistent, together with Proposition 8.9. □

The algorithm for updating considered the maximal cliques of a junction tree, which sent and received messages locally; the global update is performed entirely by a series of local computations. By organising the variables into maximal cliques and separators on a junction tree and determining a schedule, there is no need for global computations in the inference problem; the global update is achieved entirely by passing messages between neighbours in the tree according to a schedule and the algorithm terminates automatically when the update is completed.

8.7 Using a Junction Tree with Virtual Evidence and Soft Evidence

The junction tree may be extended to the problem of updating in the light of virtual evidence and soft evidence.

Dealing with virtual evidence is straightforward; for each virtual finding, one adds in a virtual node as illustrated in Figure 6.2, which will be instantiated according to the virtual finding. This simply adds the virtual finding node to the maximal clique containing the variable for which there is a virtual finding.

If virtual evidence is given on a variable X with state space $(x_1, \ldots, x_n)$, and the evidence is given in the form

$$\rho_1 = 1, \quad \rho_j = \frac{\mathbb{P}_{E|X}(1|x_j)}{\mathbb{P}_{E|X}(1|x_1)} \qquad j = 2, \ldots, n$$

the conditional probabilities

$$\mathbb{P}_{E|X}(1|.) = \begin{array}{c|c|c|c} x_1 & x_2 & \dots & x_n \\ \hline a & a\rho_2 & \dots & a\rho_n \end{array}$$

may be used, for some $a > 0$ such that $0 \leq \mathbb{P}_{E|X}(1|x_j) \leq 1$ for each $j = 1, \dots, n$.

To absorb soft evidence, remove the links from each variable $Y_1, \dots, Y_m$ to which soft evidence is applied. Provided the nodes on which the soft evidence is received are d-separated from each other and d-separated from the nodes on which hard evidence is received after surgery, simply replace the conditional probabilities $\mathbb{P}_{Y_j|\Pi(Y_j)}$ with $\mathbb{P}^*_{Y_j}$; the independence ensures that the marginal probabilities for these variables *after* updating will remain as $\mathbb{P}^*_{Y_j}$.

The Lazy big maximal clique algorithm If soft evidence and hard evidence are received on variables that are d-connected after sugery, then incorporating soft evidence cannot be carried out in such a straightforward manner. The problem is that the approach outline above inserts $\mathbb{P}^*_{Y_j}$, the marginal probability after updating, in the place of the a-priori assessment $\mathbb{P}_{Y_j|\Pi_j}$, without reference to other pieces of evidence. The updated distribution should have $\mathbb{P}^*_{Y_j}$ as the marginal distribution over Y_j.

One method for incorporating soft evidence is discussed in [138]. The input is a Bayesian network with a collection of soft and hard findings. The method returns a joint probability distribution with two properties:

1. The findings are the marginal distributions for the updated distribution.
2. The updated distribution is the closest to the original distribution (where the Kullback Leibler divergence is used) that satisfies this constraint (that the findings are the marginals of the updated distribution).

The junction tree is modified to incorporate soft evidence in the following way.

1. After surgery, construct a junction tree, *in which all the variables that have soft evidence are in the same maximal clique - the big maximal clique* C_1.
2. Let C_1 (the big maximal clique) be the root node, apply the hard evidence and run the first half of the fully active schedule; that is, propagating from the leafs to the root node.
3. Once the big maximal clique C_1 has been updated with the information from all the other maximal cliques, absorb all the soft evidence into C_1. This is described below.
4. Distribute the evidence according to the method described in Section 8.5, the second part; sending the messages from the updated root out to the leaves.

If the big maximal clique is updated to provide a *probability* function (namely a non negative function that sums to 1, then the distribution of evidence will update the functions over the maximal cliques and separators to probability distributions over the respective maximal cliques and separators.

Absorbing the Soft Evidence Suppose the big maximal clique C_1 has soft evidence on the variables $(Y_1, \ldots, Y_k)$. Suppose soft evidence is received that $Y_1, \ldots, Y_k$ have distributions $\mathbb{Q}_{Y_1}, \ldots, \mathbb{Q}_{Y_k}$ respectively. Let $\mathbb{Q}_{C_1}$ denote the probability function over the variables in C_1 after the soft evidence has been absorbed. Then it is required that, for each $j \in \{1, \ldots, k\}$, $\mathbb{Q}_{Y_j} = \sum_{\mathcal{X}_{C_1 \setminus \{Y_k}} \mathbb{Q}_{C_1}$. That is, the marginal of $\mathbb{Q}_{C_1}$ over all variables other than Y_k is $\mathbb{Q}_{Y_k}$.

The important feature of soft evidence (Definition 6.4) is that after soft evidence has been received, the variable has no parent variables. The *Iterative Proportional Fitting Procedure* (IPFP), therefore, may be employed. It goes in cycles of length k. Firstly, normalise the function over C_1 (after the hard evidence has been received) so that it is a probability distribution $\mathbb{P}_{C_1}$. Then

$$\mathbb{P}_{C_1}^{(0)} = \mathbb{P}_{C_1}$$

for $j = 1, \ldots, k$, set $\mathbb{P}_{Y_j}^{(mk+j-1)} = \sum_{\mathcal{X}_{C_1 \setminus \{Y_j\}}} \mathbb{P}_{C_1}^{(mk+j-1)}$, and

$$\mathbb{P}_{C_1}^{(mk+j)} = \frac{\mathbb{P}_{C_1}^{(mk+j-1)} \mathbb{Q}_{Y_j}}{\mathbb{P}_{Y_j}^{(mk+j-1)}}.$$

This is repeated until the desired accuracy is obtained. It has been well established that, for discrete distributions with finite state space, the IPFP algorithm converges to the distribution that minimises the Kullback Leibler distance from the original distribution (see, for example, [10] (1959). □

Notes The original paper describing the use of junction trees for updating a Bayesian network is by S.L. Lauritzen and D.J. Spiegelhalter [80]. The propagation presented is the approach of Lauritzen and Spiegelhalter, discussed in [83]; the technicalities differ slightly between implementations in software. The proofs or the main results for the message passing algorithm were originally presented in [33]. The Iterative Proportion Fitting Procedure dating back to Deming and Stephan (1940) [35]. This is the basis for updating a junction tree in the light of soft evidence. The basic technique is taken from M. Valtorta, Y.G. Kim, J. Vomlel (2002) [138].

8.8 Exercises

1. Let $\mathbb{P}_{X_1,X_2,X_3,X_4,X_5}$ be a probability distribution over five variables that has factorisation

$$\mathbb{P}_{X_1,X_2,X_3,X_4,X_5} = \frac{\mathbb{P}_{X_1,X_2}\mathbb{P}_{X_2,X_3,X_4}\mathbb{P}_{X_4,X_5}}{\mathbb{P}_{X_2}\mathbb{P}_{X_4}}.$$

Suppose hard evidence $X_3 = a$ is received. Let

$$f_{X_1,X_2,X_3,X_4,X_5}(x_1,x_2,a,x_4,x_5) = \begin{cases} p_{X_1,X_2,X_3,X_4,X_5}(x_1,x_2,a,x_4,x_5) & x_3 = a \\ 0 & x_3 \neq a \end{cases}$$

Work through the stages of the message passing algorithm to obtain functions ψ_{X_1,X_2}, ψ_{X_2}, $\psi_{X_2,X_3,X_4}(.,a,.)$, ψ_{X_4}, ψ_{X_4,X_5} such that such that

$$\begin{cases} \psi_{X_1,X_2} = \sum_{x_4,x_5} f_{X_1,\ldots,X_5}(.,.,a,x_4,x_5), \\ \psi_{X_2} = \sum_{x_1,x_4,x_5} f_{X_1,\ldots,X_5}(x_1,.,a,x_4,x_5), \\ \psi_{X_2,X_3,X_4} = \sum_{x_1,x_5} f_{X_1,\ldots,X_5}(x_1,.,a,.,x_5), \\ \psi_{X_4} = \sum_{x_1,x_2,x_5} f_{X_1,X_2,X_3,X_4,X_5}(x_1,x_2,a,.,x_5), \\ \psi_{X_4,X_5} = \sum_{x_1,x_2} f_{X_1,\ldots,X_5}(x_1,x_2,a,.,.) \end{cases}$$

and

$$f_{X_1,\ldots,X_5} = \frac{\psi_{X_1,X_2}\psi_{X_2,X_3,X_4}\psi_{X_4,X_5}}{\psi_{X_2}\psi_{X_4}}.$$

2. (a) Prove that *Kruskal's algorithm* returns a tree of maximal weight: Consider d nodes, labelled $(\alpha_1,\ldots,\alpha_d)$ and a weight b_{ij} corresponding to each pair of nodes $\{\alpha_i,\alpha_j\}$. The *tree of maximal weight* is the tree with nodes $\{\alpha_1,\ldots,\alpha_d\}$ such that the score $\sum_{e\in\mathcal{T}} b_e$, where the sum is taken over all edges e included in the tree, is greater than or equal to the score for any other tree.

   Krusal's algorithm proceeds as follows:

   i. The d variables yield $d(d-1)/2$ edges. The edges are indexed in decreasing order, according to their weights $b_1, b_2, \ldots, b_{d(d-1)/2}$.

   ii. The edges b_1 and b_2 are selected. Then the edge b_3 is selected *if it does not form a cycle.*

   iii. This is repeated through $b_4,\ldots,b_{d(d-1)/2}$, in that order, adding edges if they do not form a cycle and discarding them if they form a cycle.

   This may be proved by induction.

   (b) Prove that *Prim's algorithm* returns a tree of maximal weight. This proceeds by first choosing the edge of maximal weight and then subsequently choosing additional edges *to add to the tree* where the additional link has maximal weight.

3. Let $\mathcal{C}$ denote the set of maximal cliques from a triangulated graph. A *pre-I-tree* is a tree over $\mathcal{C}$ with separators $S = C_1 \cap C_2$ for adjacent maximal cliques C_1 and C_2. The *weight* of a pre-I-tree is the sum of the number of variables in the separators.

(a) Prove that a junction tree is a pre-I-tree of maximal weight.

(b) Prove that any pre-I-tree of maximal weight is a junction tree.

4. A *polytree* is a DAG whose skeleton is a tree.

(a) Prove that the moral graph of a polytree is triangulated.

(b) Prove that the separators in a junction tree for a polytree consist of exactly one variable.

8.9 Answers

1. Using $X_2X_3X_4$ as root, $X_1X_2 \to X_2X_3X_4$ - trivial message is passed. $h^*_{X_2}(x_2) = \sum_{x_1} \mathbb{P}_{X_1X_2}(x_1, x_2) = \mathbb{P}_{X_2}(x_2)$, so update ratio is $\lambda(x_2) = \frac{\mathbb{P}_{X_2}(x_2)}{\mathbb{P}_{X_2}(x_2)} = 1$; $\mathbb{P}_{X_2}(x_2)$ updated to $\mathbb{P}_{X_2}(x_2)$ and $\mathbb{P}_{X_2,X_3,X_4}$ updated to

$$\begin{cases} \mathbb{P}_{YZW}(x_2, a, x_4) & x_3 = a \\ 0 & \text{otherwise} \end{cases}$$

$X_4X_5 \to X_2X_3X_4$ - again trivial message is passed, $h^*_{X_4}(x_4) = p_{X_4}(x_4)$, update ratio is 1. After this message is passed, the maximal clique $X_2X_3X_4$ is *live*, with potential

$$\psi_{X_2,X_3,X_4}(x_2, x_3, x_4) = \begin{cases} \mathbb{P}_{YZW}(x_2, a, x_4) & x_3 = a \\ 0 & \text{otherwise} \end{cases}$$

$X_2X_3X_4 \to X_1X_2$

$$\psi_{X_2}(x_2) = \sum_{x_4} \psi_{X_2X_3X_4}(x_2, a, x_4) = \sum_{x_4} p_{X_2,X_3,X_4}(x_2, a, x_4) = p_{X_2,X_3}(x_2, a)$$

update ratio is $\lambda(x_2) = \frac{\mathbb{P}_{X_2,X_3}(x_2,a)}{\mathbb{P}_{X_2}(x_2)}$ and

$$\psi_{X_1X_2}(x_1, x_2) = \frac{\mathbb{P}_{X_2,X_3}(x_2, a)}{\mathbb{P}_{X_2}(x_2)} \mathbb{P}_{X_1,X_2}(x_1, x_2)$$

$$\psi_{X_4}(x_4) = \mathbb{P}_{X_3,X_4}(a, x_4)$$

$$\psi_{X_4,X_5}(x_4, x_5) = \frac{\mathbb{P}_{X_3,X_4}(a, x_4)}{\mathbb{P}_{X_4}(x_4)} \mathbb{P}_{X_4,X_5}(x_4, x_5)$$

2. See Lemma 15.12 page 305.

3. (a) Let T be any non - maximal spanning tree. Let $T_1 \subset T_2 \subset \ldots \subset T'$ denote a sequence of maximal trees constructed through Prim's algorithm. Let the construction be so that a link from T is chosen whenever possible. Let m be the first stage where this is not possible and let $C_1 - C_2$ with separator S be the link actually chosen ($C_1 \in T_m, C_2 \notin T_m$; the separator S $C_1 - C_2$ has maximal weight of those not used). In T, there is a path between C_1 and C_2. The path contains a link $C_3 - C_4$ with separator S' such that $C_3 \in T_m$, $C_4 \notin T_m$. Possibly C_2 is C_4. Since $C_3 - C_4$ could not be chosen, it follows that $|S'| < |S|$ and therefore S contains variables not in S'. Therefore, T does not satisfy the junction tree condition.

   (b) Consider the tree of maximal weight constructed by Prim's algorithm and let $T_1, \ldots, T_n = T$ denote the successive trees. Assume that T is not a junction tree, then at some stage m, T_m can be extended to a junction tree T' while T_{m+1} cannot. Let $C_1 - C_2$ with separator S be the link chosen at this stage; $C_2 \in T_{m+1}$. Since T_{m+1} cannot be extended to a junction tree, the link $C_1 - C_2$ is not in T', so there is a path in T' between C_1 and C_2 not containing the link $C_1 - C_2$. This path contains a link $C_3 - C_4$ with separator S' such that $C_3 \in T_m$ and $C_4 \notin T_m$. Since T' is a junction tree, it follows that $S \subseteq S'$ and since S was chosen through

Prim's algorithm, it follows that $|S| \geq |S'|$ so that $S = S'$. Now remove the link $C_3 - C_4$ from T' and add the link $C_1 - C_2$. The result is a junction tree extending T_{m+1}, contradicting the assumption that it cannot be extended to a junction tree.

4. (a) Consider any cycle length $n \geq 4$ in the moral graph. If an edge in the cycle is removed that was added at the moralisation stage, there will be a cycle of length $n+1$. Successively removing edges from the cycle that were added at the moralisation stage, the graph will still have a cycle, containing the vee structure from the original graph instead of the parent-parent edge. Hence, if the moral graph is not triangulated, the skeleton of the original graph is not a tree, hence the original graph is not singly connected.

   (b) The maximal cliques of the moral graph are the variable/parent configurations. Suppose that there are two variables U and V in a separator. If $U-V$ is a parent - parent configuration in both maximal cliques being separated, then there is a cycle in the original graph, hence contradiction. If $U - V$ represents a parent-child configuration in both maximal cliques that it is separating, then there is a contradiction; both maximal cliques taken together form a single complete subset, contradicting the fact that there are two maximal cliques. If $U - V$ represents a parent-parent configuration in one maximal clique and a parent-child configuration in another, then there is a cycle in the skeleton of the original graph, hence a contradiction.

Chapter 9

Bayesian Networks in R

9.1 Introduction

It has become clear that R is now the most effective and dominant language of statistical computing. There are excellent packages available in R for Bayesian Networks, for inference using a given Bayesian Network and for learning the structure of a Bayesian Network. This chapter introduces some of the software in R available for Bayesian Networks and discusses graphs in R and inference using networks that have already been defined. Parameter learning and structure learning are considered later.

The packages considered are **gRain** by Søren Højsgaard and **bnlearn**.

Having installed R and a suitable editor (for example Rstudio), the relevant packages have to be installed.

gRain and related packages Information for **gRain** is available on the author's web page:

`http://people.math.aau.dk/~sorenh/software/gR/`

The package, along with all the supporting packages, has to be installed. As pointed out on the web page, under '4 Installation', the package uses the packages **graph**, **RBGL** and **Rgraphviz**. These packages are *not* on CRAN, but on 'bioconductor'. To install these packages, execute

```
> source("http://bioconductor.org/biocLite.R"); biocLite(c("graph","RBGL","Rgraphviz"))
```

Warning This can take a long time. Furthermore, there may be some interactive questions requiring yes/no answers.

After this, **gRain** may be installed from CRAN in the usual way:

```
> install.packages("gRain")
```

The package **bnlearn** also has some useful inference functions, although its main consideration is learning. Install it in the usual way:

```
> install.packages("bnlearn")
```

9.2 Graphs in R

This section considers the various graphs that appear in graphical modelling and how to render them in R. In addition to the packages mentioned so far, the package **ggm**, has some useful functions for graphical Markov models.

```
>install.packages("ggm")
```

Another useful graphics package is **igraph**

```
>install.packages("igraph")
```

These packages should be activated:

```
> library("bnlearn")
> library("gRain")
> library("ggm")
> library("igraph")
> library(RBGL)
> library("gRbase")
```

9.2.1 Undirected Graphs

An undirected graph can be created using the `ug()` function. For example:

```
> ugraph <- ug(c("a","b"),c("b","c","d"),"e")
> ugraph
A graphNEL graph with undirected edges
Number of Nodes = 5
Number of Edges = 4
```

(the `gRbase` package contains the function `ug()`. It is automatically activated if `gRain` is activated). Plotting the graph requires the package **Rgraphviz**

```
> library(Rgraphviz)
> plot(ugraph)
```

The default output of `ug()` retuns a `graphNEL` object. The commands `result = ''igraph''` or `result = ''matrix''` return an igraph or adjacency matrix instead. There is a plot method for `igraph` objects in the **igraph** package.

```
> uigraph <- ug(~a:b+b:c:d+e, result="igraph")
>library("igraph")
> plot(uigraph,layout=layout.spring)
```

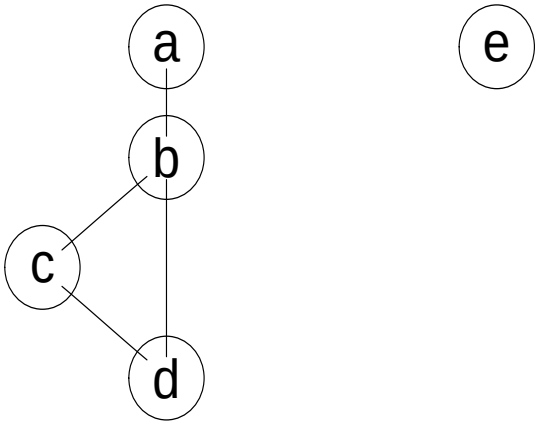

Figure 9.1: Undirected Graph

Edges can be added or deleted quite easily using the `addEdge()` and `removeEdge()` commands:

```
> ugrapha <- addEdge("a","c",ugraph)
> ugraphb <- removeEdge("c","d",ugraph)
```

The nodes and edges can be recovered quite easily:

```
> nodes(ugraph)
[1] "a" "b" "c" "d" "e"
 > str(edgeList(ugraph))
List of 4
 $ : chr [1:2] "a" "b"
 $ : chr [1:2] "b" "c"
 $ : chr [1:2] "b" "d"
 $ : chr [1:2] "c" "d"
```

The function `maxClique()` returns the cliques of the graph:

```
> maxClique(ugraph)
$maxCliques
$maxCliques[[1]]
[1] "b" "c" "d"

$maxCliques[[2]]
[1] "b" "a"

$maxCliques[[3]]
[1] "e"
```

`ugraph` is not complete; this can be seen using the `is.complete` command:

```
> is.complete(ugraph)
[1] FALSE
```

The command `separates` from the **RBGL** package, indicates whether or not there is graphical separation:

```
> separates("a","d",c("b","c"),ugraph)
[1] TRUE
```

This shows that $\{b, c\}$ separates a and d.

Subgraphs can be obtained by: `subGraph`. For example:

```
> usub <- subGraph(c("b","c","d","e"),ugraph)
> plot(usub)
```

The *boundary* $bd(\alpha)$ of a vertex α is the set of vertices adjacent to α, $adj(\alpha)$ which is equal (for an undirected graph) to the set of neighbours. The *closure* is the boundary together with the node: $cl(\alpha) = bd(\alpha) \cup \{\alpha\}$.

```
> adj(ugraph,"c")
$c
[1] "d" "b"


> closure("c",ugraph)
[1] "c" "d" "b"
```

We can also establish whether or not nodes are simplicial, if the graph is triangulated, and obtain the connected components.

```
> is.simplicial("b",ugraph)
[1] FALSE
> simplicialNodes(ugraph)
[1] "a" "c" "d" "e"
> connComp(ugraph)
[[1]]
[1] "a" "b" "c" "d"

[[2]]
[1] "e"

> is.triangulated(ugraph)
[1] TRUE
```

If we want to establish if (A, B, S) forms a *decomposition* where S is complete and separates A and B, the function is `is.decomposition`

```
> is.decomposition("a","d",c("b","c"),ugraph)
[1] FALSE
```

A *perfect elimination sequence* can be obtained by `mcs` (maximum cardinality search):

```
> mcs(ugraph)
[1] "a" "b" "c" "d" "e"
```

We can have some control over the ordering:

```
> mcs(ugraph,root=c("d","c","a"))
[1] "d" "c" "b" "a" "e"
```

It is convenient if the cliques satisfy *running intersection property* $C_j \cap (C_1 \cup \ldots \cup C_{j-1}) \subseteq C_i$ for some $i < j$. Define $S_j = C_j \cap (C_1 \cup \ldots \cup C_{j-1})$ and $R_j = C_j \backslash S_j$ with $S_1 = \phi$. Any clique C_i where $S_j \subset C_i$ with $i < j$ is a possible parent of C_i. The `rip` function returns such a list if the graph is triangulated.

```
> rip(ugraph)
cliques
  1 : b a
  2 : b c d
  3 : e
separators
  1 :
  2 : b
  3 :
parents
  1 : 0
  2 : 1
  3 : 0
```

Graphs may be triangulated using the `triangulate` function:

```
> uguntriang <- ug(~a:b:c+c:d+d:e+a:e)
> is.triangulated(uguntriang)
[1] FALSE
> plot(uguntriang)
> utriang <- triangulate(uguntriang)
> is.triangulated(utriang)
[1] TRUE
> plot(utriang)
```

9.2.2 Directed Acyclic Graphs

A DAG may be created using the `dag()` function. It can be used in several ways. For example:

```
> dgraph <- dag(~a, ~b*a, ~c*a*b, ~d*c*e, ~e*a, ~g*f)
> plot(dgraph)
```

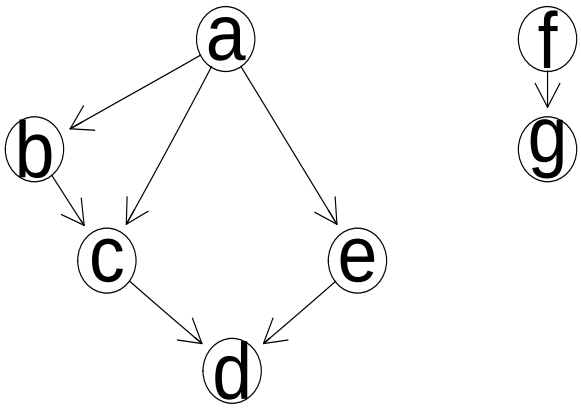

Figure 9.2: Directed Acyclic Graph

Nodes and edges may be listed as follows:

```
> nodes(dgraph)
[1] "a" "b" "c" "d" "e" "g" "f"
> str(edges(dgraph))
List of 7
 $ a: chr [1:3] "b" "c" "e"
 $ b: chr "c"
 $ c: chr "d"
 $ d: chr(0)
 $ e: chr "d"
 $ g: chr(0)
 $ f: chr "g"
```

`edges` gives a list of the *children* for each node. Alternatively, the edges are listed by:

```
> str(edgeList(dgraph))
List of 7
 $ : chr [1:2] "a" "b"
 $ : chr [1:2] "a" "c"
 $ : chr [1:2] "a" "e"
```

```
 $ : chr [1:2] "b" "c"
 $ : chr [1:2] "c" "d"
 $ : chr [1:2] "e" "d"
 $ : chr [1:2] "f" "g"
```

The `vpar()` function returns a list with an element for each node together with its parents.

```
> vpardgraph <- vpar(dgraph)
> vpardgraph$c
[1] "c" "a" "b"
```

The parents, chilren, ancestral set $an(A)$ of a set A together with all its ancestors can be obtained by:

```
> parents("d",dgraph)
[1] "c" "e"
> children("c",dgraph)
[1] "d"
> ancestralSet(c("b","e"),dgraph)
[1] "a" "b" "e"
> ag <- ancestralGraph(c("b","e"),dgraph)
> plot(ag)
```

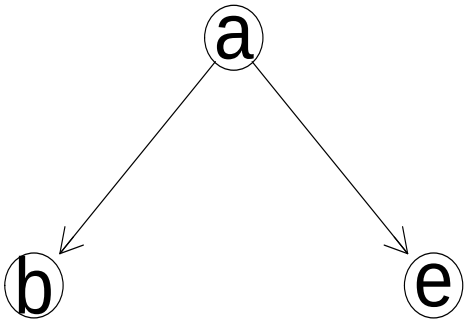

Figure 9.3: Directed Acyclic Graph

The `moralize` function moralises the graph:

```
> moral <- moralize(dgraph)
> plot(moral)
```

D-separation can be obtained by the `dSep` function from the **ggm** package.

```
> dSep(as(dgraph,"matrix"),"c","e","a")
[1] TRUE
```

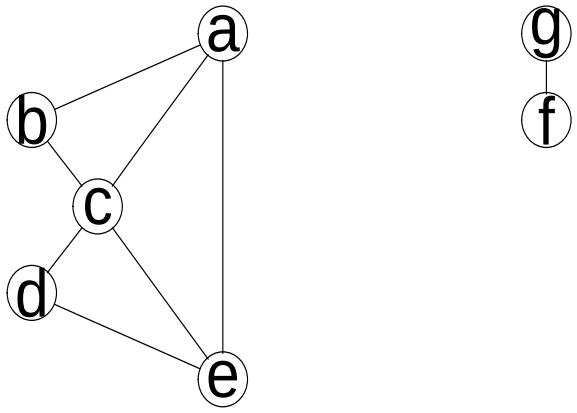

Figure 9.4: Moralised Graph

9.2.3 Mixed Graphs

Chain graphs, of which essential graphs are a subset, are mixed. They are represented in the **graph** and **igraph** package as directed graphs with multiple edges. A convenient way of defining them is to use *adjacency matrices*.

```
> adjm<-matrix(c(0,1,1,0,1,0,0,1,1,0,0,0,1,1,1,0),nrow=4)
> rownames(adjm)<-colnames(adjm)<-letters[1:4]
> adjm
  a b c d
a 0 1 1 1
b 1 0 0 1
c 1 0 0 1
d 0 1 0 0
```

This matrix can be used to create a `graphNEL` object:

```
> gG<-as(adjm,"graphNEL")
> plot(gG,"neato")
```

The graph is shown in Figure 9.5.
Note that **Rgraphviz** interprets symmetric entries as double-headed arrows. It does not distinguish between bi-directed and undirected edges. The same is true if the graph is treated as an `igraph` object. The graph from `igraph` is obtained as follows:

```
> gG1<-as(adjm,"igraph")
> plot(gG1,layout=layout.spring)
```

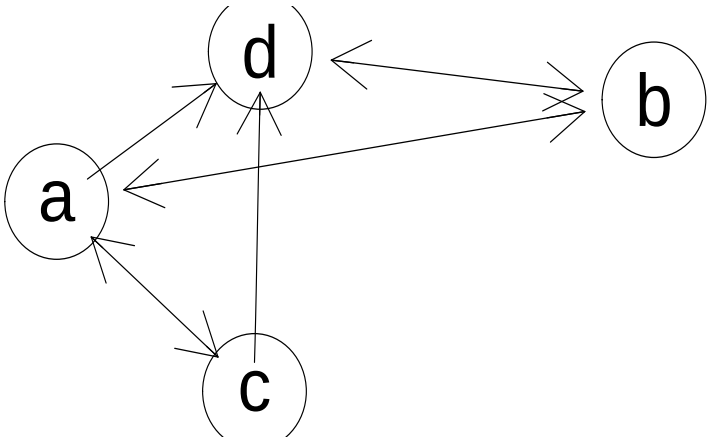

Figure 9.5: Mixed Graph

Is it a Chain Graph? The `is.chaingraph()` function from the **lcd** package determines whether a mixed graph is a chain graph. The input is an adjacency matrix.

```
> install.packages("lcd")
> library(lcd)
> is.chaingraph(as(gG1,"matrix"))
$result
[1] FALSE

$vert.order
NULL

$chain.size
NULL
```

The graph is not a chain graph; a and d are in the same chain component and therefore there should not be a directed edge $a \mapsto d$.

9.3 Bayesian Networks

9.3.1 Specifying the Conditional Probability Potentials

Consider the 'Asia' example of Lauritzen et. al. The conditional probability potentials may be specified as follows:

```
> library("gRain", lib.loc="~/R/x86_64-redhat-linux-gnu-library/3.1")
Loading required package: gRbase
> yn <- c("yes","no")
```

```
> a<-cptable(~asia, values=c(1,99),levels=yn)
> t.a<-cptable(~tub+asia,values=c(5,95,1,99),levels=yn)
> s<-cptable(~smoke, values=c(5,5),levels=yn)
> l.s<-cptable(~lung+smoke,values=c(1,9,1,99),levels=yn)
> b.s<-cptable(~bronc+smoke,values=c(6,4,3,7),levels=yn)
> e.lt<-cptable(~either+lung+tub,values=c(1,0,1,0,1,0,0,1),levels=yn)
> x.e<-cptable(~xray+either,values=c(98,2,5,95),levels=yn)
> d.be<-cptable(~dysp+bronc+either, values=c(9,1,7,3,8,2,1,9), levels = yn)
```

The + operator could be considered slightly misleading. There are other ways to enter the conditional probability potentials:

```
> t.a<-cptable(~tub|asia,values=c(5,95,1,99),levels=yn)
> t.a<-cptable(c("tub","asia"),values=c(5,95,1,99),levels=yn)
```

There are also special functions `ortable()` and `andtable`. For example, `e.lt()` could be entered by:

```
> e.lt <-ortable(~either+lung+tub, levels=yn)
```

9.3.2 Building the Network

A network is created with the function `grain()`, which returns an object of class `grain`:

```
> plist<-compileCPT(list(a, t.a, s, l.s, b.s, e.lt, x.e, d.be))
> grn1<-grain(plist)
> summary(grn1)
Independence network: Compiled: FALSE Propagated: FALSE
 Nodes : chr [1:8] "asia" "tub" "smoke" "lung" "bronc" "either" ...
> plot(grn1)
```

The plot is shown in Figure 9.6. The fictitious situation being modelled is the following: you return from a visit to Asia and find that you have a cough. A visit to Asia increases the chances of catching tuberculosis. Meanwhile, smoking causes both lung cancer and bronchitis. Tuberculosis and Lung cancer both give the same results for an x-ray. Bronchitis causes dyspnoea (shortness of breath); both lung cancer and tuberculosis have equal chances of causing dyspnoea.

9.3.3 Compilation - Finding the Clique Potentials

The network has to be compiled and propagated before queries can be made.

```
> grn1c<-compile(grn1)
> summary(grn1c)
Independence network: Compiled: TRUE Propagated: FALSE
```

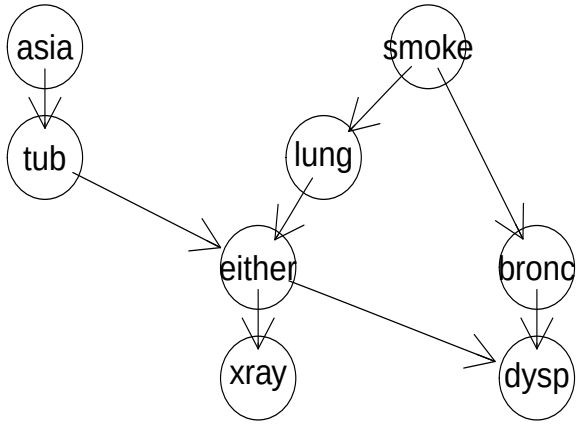

Figure 9.6: Asia Network

```
 Nodes : chr [1:8] "asia" "tub" "smoke" "lung" "bronc" "either" ...
 Number of cliques:                 6
 Maximal clique size:               3
 Maximal state space in cliques:    8
```

The various steps of `compile` can be carried out separately;

```
> g<-grn1$dag
> mg<-moralize(g)
> tmg<-triangulate(mg)
> rip(tmg)
cliques
  1 : asia tub
  2 : either lung tub
  3 : either lung bronc
  4 : smoke lung bronc
  5 : either dysp bronc
  6 : either xray
separators
  1 :
  2 : tub
  3 : either lung
  4 : lung bronc
  5 : either bronc
  6 : either
parents
  1 : 0
```

```
  2 : 1
  3 : 2
  4 : 3
  5 : 3
  6 : 5

> junctree<-rip(tmg)
> plot(junctree)
```

The plot is shown in Figure 9.7.

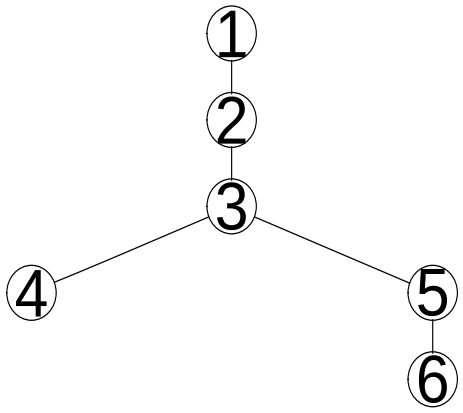

Figure 9.7: Junction Tree for Asia Network

9.3.4 Absorbing Evidence and Answering Queries

Evidence may be entered as follows: for example, suppose we have evidence that someone has visited asia and has dyspnoea. This is entered as follows:

```
> grn1c.ev<-
+ setFinding(grn1c,nodes=c("asia","dysp"),states=c("yes","yes"))
```

This creates a new `grain` object. The `grain` objects with (`grn1c.ev`) and without (`gran1c`) can be queried to give marginal probabilities:

```
> querygrain(grn1c.ev,nodes=c("lung","bronc"),type="marginal")
$lung
lung
       yes         no
0.09952515 0.90047485
```

```
$bronc
bronc
      yes        no
0.8114021 0.1885979

> querygrain(grn1c,nodes=c("lung","bronc"),type="marginal")
$lung
lung
  yes    no
0.055 0.945

$bronc
bronc
 yes   no
0.45 0.55
```

The evidence in a `grain` object can be retrieved with the `getFinding()` function, while the probability of observing the evidence is obtained using the `pFinding()` function:

```
> getFinding(grn1c.ev)
Finding:
asia: yes
dysp: yes
Pr(Finding)= 0.004501375
> pFinding(grn1c.ev)
[1] 0.004501375
```

Joint and conditional distributions may be computed as follows:

```
> querygrain(grn1c.ev,nodes=c("lung","bronc"),type="joint")
     bronc
lung         yes         no
  yes 0.06298076 0.03654439
  no  0.74842132 0.15205354
> querygrain(grn1c.ev,nodes=c("lung","bronc"),type="conditional")
     bronc
lung         yes        no
  yes 0.07761966 0.1937688
  no  0.92238034 0.8062312
```

These are both conditioned on the evidence; the former the joint distribution of `lung` and `bronc` conditioned on the evidence, while the latter is the conditional distribution of `lung` given `bronc` and the evidence.

If it is known beforehand that a specific subset U of the variables will be of interest, it is computationally faster to ensure that they are in the same clique. Consider the `grain` objects `grn1c2`, where variables of interest are forced into the root clique:

```
> grn1c2<-compile(grn1,root=c("lung","bronc","tub"),propagate=TRUE)
> grn1c2.ev<-setFinding(grn1c2,nodes=c("asia","dysp"),states=c("yes","yes"))
```

and now compare the computing times:

```
> system.time({for (i in 1:50)
+ querygrain(grn1c.ev,nodes=c("lung","bronc","tub"),type="joint")})
   user  system elapsed
  1.275   0.004   1.279
> system.time({for (i in 1:50)
+ querygrain(grn1c2.ev,nodes=c("lung","bronc","tub"),type="joint")})
   user  system elapsed
  0.012   0.000   0.013
```

The second method is much faster.

Evidence can be entered incrementally by calling `setFinding()` repeatedly. Set `propagate=FALSE` while evidence is being entered and call `propagate()` at the end:

```
> grn1c.ev<-setFinding(grn1c,nodes="asia",states="yes",propagate="FALSE")
> grn1c.ev<-setFinding(grn1c.ev,nodes="dysp",states="yes",propagate="FALSE")
> grn1c.ev<-propagate(grn1c.ev)
```

Evidence can be retracted (removed) using the `retractFinding()` function:

```
> grn1c.ev<-retractFinding(grn1c.ev,nodes="asia")
> getFinding(grn1c.ev)
Finding:
dysp: yes
Pr(Finding)= 0.4359706
```

Omitting `nodes` implies that *all* the evidence is retracted:

```
> grn1c.ev<-retractFinding(grn1c.ev)
> getFinding(grn1c.ev)
NULL
```

9.3.5 Building a Network from Data

For an $n \times d$ data matrix $\mathbf{x}$ which represents n independent instantiations of d variables $(X_1, \ldots, X_d)$, the conditional probability potentials can be estimated. Recall that `g` is the DAG for the 'Asia' network. The input is: a data frame and a DAG where the nodes are the names of the variables. To avoid 0s in the CPPs, a small smoothing number is added to all the frequencies (which are then normalised to ensure that they are probabilities). This can be, for example, 0.1.

```
> plot(g)
> simdagchest<-grain(g,data=chestSim500)
extractCPT - data.frame
> simdagchest<-compile(simdagchest,propagate=TRUE,smooth=0.1)
> querygrain(simdagchest,nodes=c("lung","bronc"),type="marginal")
$lung
lung
  yes    no
0.046 0.954

$bronc
bronc
  yes    no
0.454 0.546
```

Alternatively, a `grain` object may be built from an undirected triangulated graph. Recall that `tmg` is `g` which has been moralised and then triangulated. Then

```
> simugchest<-grain(tmg,data=chestSim500,smooth=0.1)
extractCPT - data.frame
> simugchest<-compile(simugchest,propagate=TRUE)
> plot(simugchest)
```

9.3.6 Simulation using a Network

To simulate data from the Asia network, with the evidence that a person has visited Asia and has returned with dyspnoea, the function `simulate()` may be used:

```
> simulate(grn1c.ev,nsim=5)
  asia tub smoke lung bronc either xray dysp
1  yes yes    no   no    no    yes  yes  yes
2  yes  no   yes   no   yes     no   no  yes
3  yes  no   yes   no   yes     no   no  yes
4  yes  no   yes   no   yes     no   no  yes
5  yes  no   yes   no   yes     no   no  yes
```

The `xtabs()` function may be used to obtain (approximately) the joint distribution of `lung` and `bronc` conditioned on the finding:

```
> xtabs(~lung+bronc, data=simulate(grn1c.ev,nsim=1000))/1000
     bronc
lung    yes    no
  yes 0.064 0.028
  no  0.757 0.151
```

9.3.7 Prediction

The `predict()` function is used for prediction. The default is `type = ''class''`, which gives the class with the highest probability, given the observed values of the predictors. Firstly, we generate some data:

```
> mydata<-simulate(grn1c.ev,nsim=5)
> mydata
  asia tub smoke lung bronc either xray dysp
1  yes  no   yes   no   yes     no   no  yes
2  yes  no    no   no   yes     no   no  yes
3  yes  no    no   no   yes     no   no  yes
4  yes  no   yes   no   yes     no   no  yes
5  yes  no    no   no    no     no   no  yes
```

then we try to predict the most probable configuration of `lung` and the most probable configuration of `bronc`, given all the others.

```
> predict(grn1c,response=c("lung","bronc"),newdata=mydata,
+ predictors=c("smoke","asia","tub","dysp","xray"),type="class")
$pred
$pred$lung
[1] "no" "no" "no" "no" "no"

$pred$bronc
[1] "yes" "yes" "yes" "yes" "yes"



$pFinding
[1] 0.002123915 0.001388412 0.001388412 0.002123915 0.001388412
```

These are read as follows: the variables `lung` and `bronc` are treated *individually*; this does not give the *joint* most probable configuration. The entire conditional distribution of `lung` and `bronc` is obtained as follows:

```
> predict(grn1c,response=c("lung","bronc"),newdata=mydata,
+ predictors=c("smoke","asia","tub","dysp","xray"),type="dist")
$pred
$pred$lung
              yes        no
[1,] 0.0036677551 0.9963322
[2,] 0.0005200187 0.9994800
[3,] 0.0005200187 0.9994800
[4,] 0.0036677551 0.9963322
[5,] 0.0005200187 0.9994800

$pred$bronc
           yes         no
[1,] 0.9221067 0.07789335
[2,] 0.7739757 0.22602430
[3,] 0.7739757 0.22602430
[4,] 0.9221067 0.07789335
[5,] 0.7739757 0.22602430


$pFinding
[1] 0.002123915 0.001388412 0.001388412 0.002123915 0.001388412
```

9.3.8 Buidling a Bayesian Network using bnlearn

Inference may also be carried out using the **bnlearn** package. For illustration, consider the gene expression analysis from the paper by Sachs et. al.
Sachs K.; Perez, O.; Pe'er, D.; Lauffenburger, D.A.; Nolan, G.P. (2005) *Causal Protein-Signalling Networks derived from Multi-parameter Single-cell Data* Science 308 (5721): 523-529
The relevant data is found in `sachs.interventional.txt` in the `data` directory of the course web page:

`http://www.mimuw.edu.pl/~noble/courses/BayesianNetworks/data/`

Copy the file onto your local directory, then load it into R.

```
>library(bnlearn)
>library(gRain)
> sachs.interventional <- read.table("~/data/sachs.interventional.txt", header=TRUE,
colClasses = "factor")
> isachs<-sachs.interventional
```

It is important to have `colClasses = ''factor''`. The Bayesian Network is constructed as follows:

```
> val.str=paste("[PKC][PKA|PKC][praf|PKC:PKA]",
+ "[pmek|PKC:PKA:praf][p44.42|pmek:PKA]",
+ "[pakts473|p44.42:PKA][P38|PKC:PKA]",
+ "[pjnk|PKC:PKA][plcg][PIP3|plcg]",
+ "[PIP2|plcg:PIP3]")
> val=model2network(val.str)
> isachs=isachs[, 1:11]
> for(i in names(isachs))
+ levels(isachs[, i]) = c("LOW","AVERAGE","HIGH")
> fitted = bn.fit(val, isachs, method = "bayes")
```

The variable `val` contains the DAG for the Bayesian network. Given the structure, `bn.fit` estimates the conditional probabilities. There are several methods for doing this, but the Conditional Probability Potentials simply contain the estimates from data.

Once the BN (DAG and CPPs) has been specified, we construct a junction tree for inference. The junction tree algorithm is provided by the **gRain** package.

```
> jtree <- compile(as.grain(fitted))
```

Now suppose that we have *hard evidence* or a *finding* that node `p44.42` is in state `''LOW''`. Then this is inserted quite simply by:

```
> jprop <- setFinding(jtree, nodes = "p44.42",
+ states="LOW")
```

Let us now check the marginal distribution of the node `pakts473` with and without the evidence.

```
> querygrain(jtree, nodes="pakts473")$pakts473
pakts473
       LOW    AVERAGE       HIGH
0.60893407 0.31041282 0.08065311
```

The conditional probability, conditioned on the evidence is:

```
> querygrain(jprop, nodes="pakts473")$pakts473
pakts473
        LOW     AVERAGE        HIGH
0.665161776 0.333333333 0.001504891
```

The *maximum a posteriori* states may be found by finding the largest element of the target distribution:

```
> names(which.max(querygrain(jprop,nodes=c("PKA"))$PKA))
[1] "LOW"
```

The `cpdist` and `cpquery` commands from **bnlearn** do the same thing:

```
> particles <- cpdist(fitted, nodes="pakts473",evidence=(p44.42=="LOW"))
> prop.table(table(particles))
particles
     LOW  AVERAGE     HIGH
0.669962 0.330038 0.000000
```

The `cpquery` command returns to probability of a specific event which is described by another logical expression. For example:

```
> cpquery(fitted,event=(pakts473=="LOW")&(PKA != "HIGH"),
+ evidence = (p44.42 == "LOW")|(praf=="LOW"))
[1] 0.5696073
```

9.4 Exercises

1. Professor Noddy is in his office when he reeives the news that the burglar alarm in his home has gone off. Convinced that a burglar has broken in, he starts to drive home. But, on his way, he hears on the radio that there has been a minor earth tremor in the area. Since an earth tremor can set off a burglar alarm, he therefore returns to his office.

   (a) Construct the Bayesian network associated with the situation.

   (b) Suppose that the variables are listed as R for the radio broad ast (y/n), A for the alarm (y/n), B for the burglary (y/n) and E for the earthquake (y/n), where y stands for 'yes' and n stands for 'no'. Suppose that the conditional probability potentials associated with the Bayesian Network are

$$\mathbb{P}_{R|E} = \begin{array}{c|cc} E\backslash R & y & n \\ \hline y & 0.99 & 0.01 \\ n & 0.05 & 0.95 \end{array}$$

$$\mathbb{P}_{A|B,E}(y|.,.) = \begin{array}{c|cc} E\backslash B & y & n \\ \hline y & 0.98 & 0.95 \\ n & 0.95 & 0.03 \end{array}$$

$$\mathbb{P}_B = \begin{array}{cc} y & n \\ \hline 0.01 & 0.99 \end{array}$$

$$\mathbb{P}_E = \begin{array}{cc} y & n \\ \hline 0.001 & 0.999 \end{array}$$

   Find

$$\mathbb{P}_{B|A}(y|y), \quad \mathbb{P}_{B|A}(y|y), \quad \mathbb{P}_{B|A,R}(y|y,y)$$

2. You have two CPPs from a Bayesian Network:

$$\mathbb{P}_{B|A} = \begin{array}{c|cccc} A\backslash B & b_1 & b_2 & b_3 & b_4 \\ \hline a_1 & 0.6 & 0.1 & 0.2 & 0.1 \\ a_2 & 0.2 & 0.5 & 0.1 & 0.2 \end{array}$$

   and

$$\mathbb{P}_{C|B} = \begin{array}{c|cc} B\backslash C & c_1 & c_2 \\ \hline b_1 & 0.8 & 0.2 \\ b_2 & 0.8 & 0.2 \\ b_3 & 0.2 & 0.8 \\ b_4 & 0.2 & 0.8 \end{array}$$

   Establish whether or not $A \perp C$.

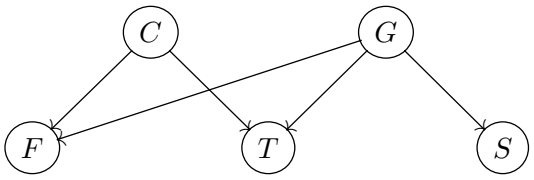

Figure 9.8: Sore throat model

3. Consider the Bayesian Network in Figure 9.8. You have a sore throat (T). There are two possible causes; either you have a cold (C), or else you have Green Monkey Disease (G). A symptom of GMD is spots (S).

   The conditional probabilities are:

$$\mathbb{P}_{F|C,G}(y|.,.) = \begin{array}{c|cc} G\backslash C & y & n \\ \hline y & 0.990 & 0.700 \\ n & 0.800 & 0.200 \end{array}$$

$$\mathbb{P}_{T|C,G}(y|.,.) = \begin{array}{c|cc} G\backslash C & y & n \\ \hline y & 0.999 & 0.900 \\ n & 0.800 & 0.300 \end{array}$$

$$\mathbb{P}_{S|G}(y|.) = \begin{array}{cc} y & n \\ \hline 0.010 & 0.001 \end{array} \qquad \mathbb{P}_C(y) = 0.20 \qquad \mathbb{P}_G(y) = 0.10$$

   (a) Let $E = (F, T, S)$ and enter the evidence $e = (n, n, y)$. That is, $\{F = n, T = n, S = y\}$.

   (b) Compute the updated joint probability distribution of C,G given the evidence, $\mathbb{P}_{C,G|E}(.,.|e)$.

   (c) Compute the *most probable explanation* of the evidence e. This is the configuration of the remaining variables $V\backslash E$ that gives the largest value for $\mathbb{P}_{E|V\backslash E}$. It therefore also maximises $\mathbb{P}_{V\backslash E|E}(.|e)$.

   (d) Consider a vector of evidence variables $\underline{E} = (E_1, \ldots, E_m)$, instantiated as $\underline{e} = (e_1, \ldots, e_m)$. The *conflict measure* of the evidence is defined as:

$$\text{conf}(\underline{e}) = \log_2 \frac{\prod_{j=1}^{m} \mathbb{P}_{E_j}(e_j)}{\mathbb{P}_{\underline{E}}(\underline{e})}.$$

   If the evidence variables are independent of each other, then the conflict measure will clearly be zero. If the pieces of evidence corroborate each other; for example $\mathbb{P}_{E_1|E_2}(e_1|e_2) > \mathbb{P}_{E_1}(e_1)$ so that given $E_2 = e_2$, the event $E_1 = e_1$ is more likely than the unconditional event, the conflict ratio will be negative. If the pieces of evidence conflict, then the conflict measure will be positive.

   Compute the conflict measure for the evidence in this example.

4. Now consider the `sachs.interventional.txt` data in the notes. Find the moral graph, triangulate it and construct a junction tree.

   For the network with the parameters given in the notes from the `sachs.interventional.txt` data, note that `PKA` is a parent of all the nodes in the `praf -> pmek -> p44.42 -> pakts473` chain. Use the junction tree algorithm to update the probabilities over these nodes when we have evidence that `PKA` is `''LOW''` and `PKA` is `''HIGH''`.

   Use any other techniques discussed on this network.

Chapter 10

Conditional Gaussian variables

10.1 Conditional Gaussian Distributions

One very important family of distributions, that is accommodated by standard Bayesian network software is the family of *conditional Gaussian distributions*.

Let $\underline{X} = (\underline{X}_\Delta, \underline{X}_\Gamma)$ where $\underline{X}_\Delta$ is a discrete random vector and $\underline{X}_\Gamma$ is a continuous random vector. Let Δ denote the indexing set for $\underline{X}_\Delta$ and Γ the indexing set for $\underline{X}_\Gamma$ variables. Let $|\Delta|$ denote the number of variables in Δ and $|\Gamma|$ denote the number of variables in Γ. Random vectors will be taken as *row* vectors. The state space is

$$\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_{|\Delta|} \times \mathcal{X}_{|\Delta|+1} \times \ldots \times \mathcal{X}_{|\Delta|+|\Gamma|},$$

where $\mathcal{X}_j$ denotes the state space for variables j. The following notation will also be used;

$$\mathcal{X}_\Delta = \mathcal{X}_1 \times \ldots \times \mathcal{X}_{|\Delta|}, \quad \mathcal{X}_\Gamma = \mathcal{X}_{|\Delta|+1} \times \ldots \times \mathcal{X}_{|\Delta|+|\Gamma|},$$

$$\mathcal{X} = \mathcal{X}_\Delta \times \mathcal{X}_\Gamma.$$

Attention is restricted to the case where the continuous variables, conditioned on the discrete variables, have Gaussian distribution, so $\mathcal{X}_\Gamma = \mathbb{R}^{|\Gamma|}$. For the discrete variables,

$$\mathcal{X}_j = \{i_j^{(1)}, \ldots, i_j^{(k_j)}\}.$$

A particular configuration $\underline{i} \in \mathcal{X}_\Delta$ is called a *cell*.

The following notation will be used to indicate that a random vector $\underline{X}_1$ conditioned on $\underline{X}_2 = \underline{x}_2$ has distribution F:

$$\underline{X}_1 \mid \underline{X}_2 = \underline{x}_2 \sim F.$$

The *moment generating function* is a useful for the definition of a multivariate normal distribution.

Definition 10.1 (Moment Generating Function). *Let* $\underline{X} = (X^1, \ldots, X^d)$ *be a random vector. Its* moment generating function *is the function* $M_{\underline{X}} : \mathbb{R}^d \to \mathbb{R}$ *is defined as*

$$M_{\underline{X}}(p_1, \ldots, p_d) = \mathbb{E}\left[\exp\left\{\sum_{j=1}^{d} p_j X_j\right\}\right].$$

The moment generating function is useful, because it *uniquely determines the distribution* of a random vector $\underline{X}$. That is, a joint probability determines a unique moment generating function, and the moment generating function uniquely determines a corresponding joint probability. The moment generating function is essentially a Laplace transform.

A *multivariate normal distribution* is defined as follows:

Definition 10.2 (Multivariate Normal Distribution). *A random vector* $\underline{X} = (X_1, \ldots, X_d)$ *is said to have a* multivariate normal *distribution, written* $\underline{X} \sim N(\underline{\mu}, C)$*, if its moment generating function is of the form*

$$\phi(p_1, \ldots, p_d) = \exp\left\{\sum_{j=1}^{p} p_j \mu_j + \frac{1}{2}\sum_{jk} p_j p_k C_{jk}\right\}, \quad \underline{p} \in \mathbb{R}^d.$$

If a random vector $\underline{X} \sim N(\underline{\mu}, C)$, then $\mathbb{E}[X_i] = \mu_i$ for each $i = 1, \ldots, n$ and $\mathrm{Cov}(X_i, X_j) = C_{ij}$ for each (i, j). If C is positive definite, then the joint density function of $\underline{X} = (X_1, \ldots, X_d)$ is given by

$$\pi_{X_1,\ldots,X_d}(x_1, \ldots, x_d) = \frac{1}{(2\pi)^{d/2}|C|^{1/2}} \exp\left\{-\frac{1}{2}(\underline{x} - \underline{\mu})C^{-1}(\underline{x} - \underline{\mu})\right\}, \quad \underline{x} \in \mathbb{R}^d,$$

where $\underline{x} = (x_1, \ldots, x_d)$ and $\underline{\mu} = (\mu_1, \ldots, \mu_d)$ are row vectors and $|C|$ denotes the determinant of C.

The *conditional Gaussian* distribution, or CG distribution may now be defined.

Definition 10.3 (CG Distribution). *A collection of random variables* $\underline{X} = (\underline{X}_\Delta, \underline{X}_\Gamma)$ *is said to follow a* CG distribution *if for each* $\underline{i} \in \mathcal{X}_\Delta$*,*

$$\underline{X}_\Gamma | \{\underline{X}_\Delta = \underline{i}\} \sim N\left(\underline{\mu}(\underline{i}), C(\underline{i})\right). \tag{10.1}$$

The notation for such a Conditional Gaussian distribution is

$$\underline{X} \sim \mathrm{CG}(|\Delta|, |\Gamma|).$$

If the numbers of discrete and continuous random variables are, respectively, $|\Delta| = p$ and $|\Gamma| = q$, then $\underline{X} \sim \mathrm{CG}(p, q)$.

If C^{-1} is well defined, then the conditional density function of $\underline{X}_\Gamma$ conditioned on $\underline{X}_\Delta = \underline{i}$ is

$$\pi_{\underline{X}_\Gamma | \underline{X}_\Delta}(\underline{x}|\underline{i}) = \frac{1}{(2\pi)^{q/2}\sqrt{\det C(i)}} e^{-\frac{1}{2}(\underline{x} - \underline{\mu}(\underline{i}))C(\underline{i})^{-1}(\underline{x} - \underline{\mu}(i))^t}, \tag{10.2}$$

for all $\underline{i} \in \mathcal{X}_\Delta$ such that

$$\mathbb{P}_{\underline{X}_\Delta}(\underline{i}) > 0.$$

For this discussion, it is assumed that $\mathbb{P}_{\underline{X}_\Delta}(\underline{i}) > 0$ for each $\underline{i} \in \mathcal{X}_\Delta$.

Directly from Equation (10.2),

$$\mathbb{P}_{\underline{X}_\Delta}(\underline{i})\pi_{\underline{X}_\Gamma|\underline{X}_\Delta}(\underline{x}|\underline{i}) = \chi(\underline{i})e^{g(\underline{i})+\underline{x}h(\underline{i})-\frac{1}{2}\underline{x}K(\underline{i})\underline{x}^t} \tag{10.3}$$

where $\chi(\underline{i}) = 1$ if $\mathbb{P}_{\underline{X}_\Delta}(\underline{i}) > 0$ and 0 if $\mathbb{P}_{\underline{X}_\Delta}(\underline{i}) = 0$,

$$h(\underline{i}) = C(\underline{i})^{-1}\underline{\mu}(\underline{i})^t \tag{10.4}$$

$$K(\underline{i}) = C(\underline{i})^{-1} \tag{10.5}$$

and

$$g(\underline{i}) = \log \mathbb{P}_{\underline{X}_\Delta}(\underline{i}) + \frac{1}{2}\left(\log\det K(\underline{i}) - |\Gamma|\log 2\pi - \underline{\mu}(\underline{i})K(\underline{i})\underline{\mu}(\underline{i})^t\right). \tag{10.6}$$

From Equation (10.2), it is clear that conditioning on the discrete variables gives a family of multivariate normal distributions. The *canonical parameters* of the Gaussian distribution are defined as $(h(\underline{i}), K(\underline{i}))$ and the *mean parameters* as $\left(\underline{\mu}(\underline{i}), C(\underline{i})\right)$. Conditioned on $\underline{X}_\Delta = \underline{i}$,

$$\mathbb{E}\left[\underline{X}_\Gamma|\underline{X}_\Delta = \underline{i}\right] = \underline{\mu}(\underline{i})$$

and

$$\mathbb{E}\left[\left(\underline{X}_\Gamma - \underline{\mu}(\underline{i})\right)\left(\underline{X}_\Gamma - \underline{\mu}(\underline{i})\right)^t|\underline{X}_\Delta = \underline{i}\right] = C(\underline{i})$$

(where the random vectors are taken to be row vectors).

Parametrisation of the CG Distribution The *canonical parameters* for the *joint* distribution, defined by the pair of functions $\left(\mathbb{P}_{\underline{X}_\Delta}, \pi_{\underline{X}_\Gamma|\underline{X}_\Delta}\right)$ are *defined* as (g, h, K), where the parameters $(h(\underline{i}), K(\underline{i}))$ are defined by Equations (10.4) and (10.5) respectively and $g(\underline{i})$ is defined by Equation (10.6).

Similarly, the *mean parameters* are *defined* as $(\mathbb{P}, \underline{\mu}, C)$, where $(\underline{\mu}(\underline{i}), C(\underline{i}))$ are the mean parameters of the conditional distribution and $\mathbb{P}(\underline{i})$ is the probability function over the discrete variables.

10.1.1 Some Results on Marginalization

The aim will be to factorise the CG distribution along an appropriate junction tree, so that evidence can be inserted and propagated. The difficulty arises that if a CG distribution is marginalised over some of its discrete variables, the resulting distribution is no longer CG. For efficient computation along a junction tree, it is desirable if the CG property can be preserved as far as possible for the marginal distributions on the cliques and separators. The following results give properties that help determine appropriate factorisations of the distribution.

Proposition 10.4. *Let $\underline{X}$ have a CG distribution. Let $V = \Delta \cup \Gamma$ denote the indexing set. Let A and B be two disjoint sets such that $V = A \cup B$, then the conditional distribution of $\underline{X}_A$ given $\underline{X}_B = \underline{x}_B$ is CG.*

Proof The following calculation shows that $\underline{X}_{A\cap\Gamma} \mid \{\underline{X}_B = \underline{x}_B\} \cup \{\underline{X}_{A\cap\Delta} = \underline{x}_{A\cap\Delta}\}$ has a multivariate Gaussian distribution. Firstly, it is clear that

$$\pi_{\underline{X}_{A\cap\Gamma}|\underline{X}_{A\cap\Delta},\underline{X}_B}(\underline{x}_{A\cap\Gamma} \mid \underline{x}_{A\cap\Delta}, \underline{x}_B) = \pi_{\underline{X}_{A\cap\Gamma}|\underline{X}_{\Delta},\underline{X}_{B\cap\Gamma}}(\underline{x}_{A\cap\Gamma} \mid \underline{x}_{\Delta}, \underline{x}_{B\cap\Gamma}).$$

The conditional density function on the right hand side is obtained by conditioning the distribution of $\underline{X}_{A\cap\Gamma} \mid \underline{X}_\Delta = \underline{x}_\Delta$ on $\underline{X}_{B\cap\Gamma} = \underline{x}_{B\cap\Gamma}$. Since $(\underline{X}_{A\cap\Gamma}, \underline{X}_{B\cap\Gamma}) \mid \underline{X}_\Delta = \underline{x}_\Delta$ has a multivariate Gaussian distribution, and the conditional distribution of a multivariate Gaussian, conditioning on some of its component variables is again multivariate Gaussian, it follows that the conditional distribution is multivariate. The proof is complete. □

If the variables to be marginalised are discrete, then complicated mixture distributions arise. The following theorem gives a situation where the marginalisation yields a CG distribution.

Proposition 10.5. *Let $A \subseteq V$ denote a subset of the indexing set for the variables. If $\underline{X}$ is CG and $B = V \setminus A$ (namely, B is the set of all indices in V that are not in A) and $B \subseteq \Delta$ and*

$$\underline{X}_B \perp \underline{X}_\Gamma \mid \underline{X}_{\Delta\setminus B},$$

then $\underline{X}_A \sim$ CG.

Proof Clearly, from the definition of a CG distribution, it is necessary and sufficient to show that

$$\underline{X}_{A\cap\Gamma} \mid \underline{X}_{\Delta\setminus B} \sim \mathcal{N}_{|A\cap\Gamma|}.$$

(multivariate normal, with dimension $|A \cap \Gamma|$). The proof requires the following identity: If $\underline{X}_B \perp \underline{X}_\Gamma \mid \underline{X}_{\Delta\setminus B}$, then

$$\pi_{\underline{X}_\Gamma|\underline{X}_\Delta}(\underline{x}_\Gamma|\underline{x}_\Delta) = \pi_{\underline{X}_\Gamma|\underline{X}_B,\underline{X}_{\Delta\setminus B}}(\underline{x}_\Gamma|\underline{x}_B, \underline{x}_{\Delta\setminus B}) = \pi_{\underline{X}_\Gamma|\underline{X}_{\Delta\setminus B}}(\underline{x}_\Gamma|\underline{x}_{\Delta\setminus B}).$$

This is a straightforward consequence of the definition of conditional independence. Recall that, from the definition of a CG distribution, $\pi_{\underline{X}_\Gamma|\underline{X}_\Delta}(\underline{x}_\Gamma \mid \underline{x}_\Delta)$ is a multivariate normal distribution. Therefore the conditional distribution of $\underline{X}_\Gamma$ conditioned on $\underline{X}_{\Delta\setminus B}$ is multivariate Gaussian, therefore the conditional distribution of $\underline{X}_{\Gamma\cap A}$ conditioned on $\underline{X}_{\Delta\setminus B}$ is multivariate Gaussian. The proof is complete. □

10.1.2 CG Regression

An important special case of CG distributions are those that follow *CG regression*. The requirement here is that the continuous variables depend *linearly* on their continuous parents. This is the situation that is treated by most softwares with a facility for CG distributions.

Definition 10.6 (CG Regression)**.** *Let $\underline{Z} = (Z_1, \ldots, Z_s)$ be a continuous random (row) vector and let $\underline{I}$ be a discrete random (row) vector with probability function $p_{\underline{I}}$. Let $\mathcal{I}$ denote the state space for I. If a random (row) vector $\underline{Y} = (Y_1, \ldots, Y_r)$ has the property that*

$$\underline{Y} \mid \{\underline{I} = \underline{i}, \underline{Z} = \underline{z}\} \sim N_r(\underline{A}(\underline{i}) + \underline{z}\mathbf{B}(\underline{i}), C(\underline{i})),$$

where for each $i \in \mathcal{I}$

- $\underline{A}(i)$ *is a* $1 \times r$ *row vector for each* $i \in \mathcal{I}$,
- $\mathbf{B}(i)$ *is an* $s \times r$ *matrix,*
- $C(i)$ *is a positive semi-definite symmetric matrix,*

then $\underline{Y}$ *is said to follow a* CG regression.

Let $\underline{X}$ denote a random vector, containing both discrete and continuous variables, which have been ordered so that the probability distribution may be factorised along a Directed Acyclic Graph $\mathcal{G} = (V, D)$. Let X_γ be a continuous variable, with parent set $\Pi(\gamma)$. Suppose that $\underline{X}$ has a CG distribution that satisfies the additional CG regression requirement. Then the conditional distribution for X_γ , conditioned on its parent nodes $\Pi(\gamma)$ is the CG regression

$$X_\gamma \mid \{\Pi_d(\gamma) = \underline{i}, \Pi_c(\gamma) = \underline{z}\} \sim N\left(\alpha(\underline{i}) + \underline{z}\underline{\beta}, \sigma^2(\underline{i})\right),$$

where the discrete variables $\Pi_d(\gamma)$ of $\Pi(\gamma)$ take values $\underline{i}$ and the continuous variables $\Pi_c(\gamma)$ of $\Pi(\gamma)$ take values $\underline{z}$. Here $\alpha(\underline{i})$ is a number, $\sigma^2(\underline{i}) = \mathbb{V}(X_\gamma|\Pi(\gamma))$ and $\underline{\beta}$ is a column vector with dimension equal to the dimension of the continuous component $\underline{z}$ so that $\underline{z}\underline{\beta}$ is a well defined inner product. Thus, the conditional density is Gaussian; $\phi(\underline{i}, \underline{z}, x_\gamma)$, equal to

$$\phi(\underline{i}, \underline{z}, x_\gamma) = \frac{1}{\sqrt{2\pi}\sigma(\underline{i})} \exp\left\{-\frac{(x_\gamma - \alpha(\underline{i}) + \underline{z}\underline{\beta})^2}{2\sigma^2(\underline{i})}\right\}. \tag{10.7}$$

Example 10.7.

This example is taken from [84]. The emissions from a waste incinerator differ because of compositional differences in incoming waste. Another important factor is the way in which the waste is burnt, which can be monitored by measuring the concentration of carbon dioxide in the emissions. The efficiency of the filter depends on its technical state and also on the amount and composition of the waste. The emission of heavy metals depends both on the concentration of metals in the incoming waste and the emission of dust particles in general. The emission of dust is monitored by measuring the penetration of light.

The situation may be modelled using a directed acyclic *marked* graph (DAMG) in Figure 10.1; *marked* because there are two types of nodes. In this case, these are discrete and continuous. In HUGIN, nodes with a double circle are continuous nodes. The categorical variables are F: filter state, W: waste type, B method of burning. The continuous variables are M_{in}: metals in the waste, M_{out}: metals emitted, E: filter efficiency, D: Dust emission, C: carbon dioxide concentration in emission and L: light penetration. The set $\Delta = \{F, W, B\}$ is the set of *discrete* variables, while $\Gamma = \{C, D, E, L, M_{in}, M_{out}\}$ is the set of *continuous* variables.

If HUGIN is being used, then inserting the graph, using double circles to indicate 'Gaussian' nodes, the conditional probability distributions can be inserted. For a conditional Gaussian distribution, the

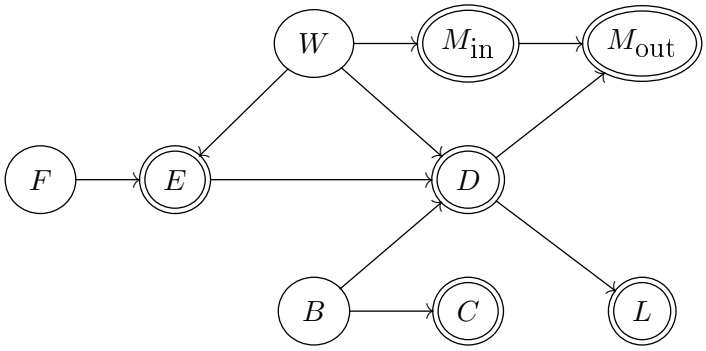

Figure 10.1: Marked Graph

continuous nodes cannot have discrete nodes as descendants. For a continuous node, HUGIN requests the mean and variance, the parameters to describe a CG regression.

10.2 The Junction Tree for Conditional Gaussian Distributions

When a CG distribution is arranged as a directed acyclic marked graph, it is done in such a way that no continuous nodes have discrete children. The assumption is that for a *continuous* variable X, with parents $\Pi(X) = (\Pi_d(X), \Pi_c(X))$, where $\Pi_d(X)$ are the discrete parents and $\Pi_c(X)$ are the continuous parents,

$$X|\{\Pi_d(X) = \underline{y}, \Pi_c(X) = \underline{z}) \sim N(\alpha(\underline{y}) + (\beta(\underline{y}), \underline{z}), \gamma(\underline{y})\}.$$

This section describes a junction tree approach due to Lauritzen [81] (1992), for finding the updated conditional Gaussian distribution when hard evidence is inserted on some of the nodes. The problem here is that while marginalising a CG distribution over one of its continuous variables gives another CG distribution, marginalising a CG distribution over one of its discrete variables does not necessarily give a CG distribution. Therefore, care has to be taken in the construction of the junction tree. Ideally, the junction tree should be constructed so that the marginal distributions over the cliques and separators are CG distributions, to enable appropriate marginalisations to be made. This requires some additional restrictions on the construction of the cliques and separators.

Definition 10.8 (Marked Graph)**.** *A* marked graph *is a graph where there are several types of nodes; the type of the node is the* mark*.*

In the context of directed acyclic graphs for conditional Gaussian distributions, there are two markings; discrete and continuous, for the types of variables represented by each type of node.

Definition 10.9 (GG Decomposition)**.** *A triple (A, B, S) of disjoint subsets of the node set V of an undirected marked graph $\mathcal{G}$ is said to form a* CG decomposition *of $\mathcal{G}$ if $V = A \cup B \cup S$ and the following three conditions hold:*

1. *S separates A from B,*
2. *S is a complete subset of V,*
3. *Either $S \subseteq \Delta$, or $B \subseteq \Gamma$ or both.*

When this holds, (A, B, S) is said to CG-decompose $\mathcal{G}$ into the components $\mathcal{G}_{A\cup S}$ and $\mathcal{G}_{B\cup S}$.

If only the first two conditions hold, then (A, B, S) is said to form a *decomposition*. Thus, a *decomposition* ignores the markings of the graph, while a *CG decomposition* takes them into account.

The logic is as follows: if B contains only continuous nodes with multivariate Gaussian distribution, then the marginal over the separator will again be multivariate Gaussian. If the separator contains only discrete nodes and B both continuous and discrete, then marginalising first over all the Gaussian nodes in B and then marginalising over the discrete nodes in B not in the separator gives the exact probability distribution over the separator.

Definition 10.10 (CG Decomposable)**.** *An undirected* marked *graph is said to be* CG decomposable *if it is complete, or if there exists a CG decomposition (A, B, S), where both A and B are non empty, into CG decomposable sub-graphs $\mathcal{G}_{A\cup S}$, and $\mathcal{G}_{B\cup S}$.*

Decomposable unmarked graphs are triangulated; any cycle of length 4 or more has a chord. CG decomposable *marked* graphs are *further* characterised by requiring that if there is a path between two discrete variable containing only continuous variables, then there is an edge between the two discrete variables.

Proposition 10.11. *For an undirected marked graph $\mathcal{G}$, the following are equivalent:*

1. *$\mathcal{G}$ is CG decomposable.*
2. *$\mathcal{G}$ is triangulated, and for any path $(\delta_1, \alpha_1, \ldots, \alpha_n, \delta_2)$ between two* discrete *nodes (δ_1, δ_2) where $(\alpha_1, \ldots, \alpha_n)$ are all continuous, δ_1 and δ_2 are neighbours.*
3. *For any α and β in $\mathcal{G}$, every minimal (α, β) separator is complete. If both α and β are discrete, then their minimal separator contains only discrete nodes.*

Proof of 1 $\Rightarrow$ 2 The proof, as before for unmarked graphs, is by induction. The inductive hypothesis is: All undirected CG decomposable graphs with n or fewer nodes are triangulated and satisfying the conditions of statement 2.

This is clearly true for a graph on one node.
Let $\mathcal{G}$ be a CG decomposable graph on $n+1$ nodes.
Either $\mathcal{G}$ is complete, in which case the properties of 2 clearly follow,
Or There exist sets A, B, S, where $V = A \cup B \cup S$, where either $B \subseteq \Gamma$ or $S \subseteq \Delta$ or both, and such that $\mathcal{G}_{A\cup S}$ and $\mathcal{G}_{B\cup S}$ are CG decomposable. Then any cycle of length 4 without a chord must pass through both A and B. By decomposability, S separates A from B. Therefore the cycle must pass through S at least twice. Since S is complete, the cycle will therefore have a chord. Since $\mathcal{G}_{A\cup S}$ and

$\mathcal{G}_{B\cup S}$ are triangulated, it follows that $\mathcal{G}$ is also triangulated. If the nodes of S are discrete, it follows that any path between two discrete variable passing through S satisfies the condition of statement 2. If $B \subseteq \Gamma$, then since all paths in $\mathcal{G}_{A\cup S}$ and all paths in $\mathcal{G}_{B\cup S}$ satisfy the condition of statement 2, it is clear that all paths passing through S will also satisfy the condition of statement 2. It follows that $\mathcal{G}$ is CG decomposable. □

Proof of 2 ⇒ 3 Assume that $\mathcal{G}$ is triangulated, with the additional property in statement 2. Consider two nodes α and β and let S be their minimal separator. Let A denote the set of all nodes that may be connected to α by a trail that does not contain nodes in S and let B denote all nodes that may be connected to β by a trail that does not contain nodes in S. Every node $\gamma \in S$ must be adjacent to some node in A and some node in B, otherwise $\mathcal{G}_{V\backslash(S\backslash\{\gamma\})}$ would not be connected. This would contradict the minimality of S, since $S\backslash\{\gamma\}$ would separate α from β. Suppose that the condition in statement 2 holds and consider the minimal separator for two discrete nodes α, β, which are not neighbours. The separator is complete. Denote the separator by S. Consider $\hat{S}$, which is S with the continuous nodes removed. Then $\hat{S}$ separates α and β on the sub graph induced by the discrete variables. But the condition of statement 2 implies that α and β are also separated on $\mathcal{G}$. Therefore, $\hat{S}$ separates α and β. It follows that the minimal separator for two discrete nodes contains only discrete nodes. □

Proof of 3 ⇒ 1 If $\mathcal{G}$ is complete, it follows that every node is discrete and the result is clear. Let α and β be two discrete nodes that are not contained within their minimal separator. Let S denote their minimal separator. Let A denote the maximal connected component of $V\backslash S$ and let $B = V\backslash(A \cup S)$. Then (A, B, S) provides a decomposition, with $S \subseteq \Delta$. Suppose that two such discrete nodes cannot be found. Let α and β be two nodes that are not contained within their minimal separator, where β is continuous. Let S denote the minimal separator. Let B denote the largest connected component of $V\backslash S$ containing β. Suppose that B contains a discrete node γ. Then S separates γ from α and therefore consists entirely of discrete nodes. Therefore, either $S \subseteq \Delta$, or $B \subseteq \Gamma$, as required. □

The construction of the junction tree has to be modified. Starting from the directed acyclic graph, the graph is first moralised by adding in the links between all the parents of each variable and then making all the edges undirected, as before. Then, sufficient edges are added in to ensure that the graph is *CG - decomposable.*

Next, a junction tree is constructed. As before, this is an organisation of a collection of subsets of the variables V into a tree, such that if A and B are two nodes on the junction tree, then the variables in $A \cap B$ appear in each node on the path between A and B.

Definition 10.12 (CG Root)**.** *A node R on a junction tree is a* CG root *if any pair of neighbours A, B, such that A lies on the path between R and B (so that A is closer to R than B) satisfies*

$$B\backslash A \subseteq \Gamma \qquad \textit{or} \qquad B \cap A \subseteq \Delta \qquad \textit{or both.}$$

This condition is equivalent to the statement that the triple $(A\backslash(A \cap B), B\backslash(A \cap B), A \cap B)$ forms a CG decomposition of $\mathcal{G}_{A\cup B}$. This means that when a separator between two neighbouring cliques

is not purely discrete, the clique furthest away from the root has only continuous nodes beyond the separator.

Theorem 10.13. *The cliques of a CG decomposable marked graph can be organised into a junction tree with at least one CG root.*

Proof As with the unmarked graph, choose simplicial nodes, one after the other. This is done in such a way that either the separator (the nodes not removed) are all discrete, or else the nodes that are removed are all continuous, until it is not possible to find any other such nodes.

The remaining graph is therefore a clique, by the following arguments: either all the remaining discrete nodes are in the same clique, or else there is not a simplicial discrete node, since the minimal separator between two discrete nodes consists entirely of discrete nodes. Assume there is not a simplicial discrete node. If there are discrete nodes remaining, then the family of any simplicial continuous node contains a discrete node that does not have neighbours outside the family and is therefore simplicial. It follows that all the discrete nodes are in the same clique, the family of any remaining continuous node.

The final clique, constructed in this way, clearly satisfies the properties of a CG root. □

10.3 Updating a CG distribution using a Junction Tree

The random vectors are taken as *row* vectors when they are several attributes measured on a single run of an experiment.

For each clique C on the CG junction tree, let $\phi_C = \prod_{X \in C \setminus S} \mathbb{P}_{X|\Pi(X)}$ where $\mathbb{P}_{X|\Pi(X)}$ is a discrete probability function if X is discrete and a conditional Gaussian if X is Gaussian, where S denotes those variables in C that were not simplicial during the junction tree construction.

For each continuous variable X,

$$\mathbb{P}_{X|\Pi_c(X),\Pi_d(X)}(x|\underline{z},\underline{y}) = \frac{1}{(2\pi\gamma(\underline{y}))^{1/2}} \exp\left\{-\frac{(x-\alpha(\underline{y})-\beta(\underline{y})\underline{z}^t)^2}{2\gamma(\underline{y})}\right\}.$$

where $\Pi_c(X)$ denotes continuous parents and $\Pi_d(X)$ denotes discrete parents. Here, α is a function, β is a (row) vector of the same length as $\underline{z}$ and γ is the conditional variance.

For the separators S, the initialisation is: $\phi_S \equiv 1$ for each $S \in \mathcal{S}$.
From this, expanding the parentheses, taking logarithms and identifying terms gives the canonical parameters (g_X, h_X, K_X) for $\mathbb{P}_{X|\Pi(X)}$. The log partition function is

$$g_X(\underline{y}) = -\frac{\alpha(\underline{y})^2}{2\gamma(\underline{y})} - \frac{1}{2}\log(2\pi\gamma(\underline{y})),$$

and the other parameters are given by

$$h_X(\underline{y}) = \frac{\alpha(\underline{y})}{\gamma(\underline{y})}\left(\ 1 \ \middle|\ -\beta(\underline{y})\ \right)$$

and

$$K_X(\underline{y}) = \frac{1}{\gamma(\underline{y})}\begin{pmatrix} 1 & -\beta(\underline{y}) \\ -\beta(\underline{y})^t & \beta(\underline{y})^t\beta(\underline{y}) \end{pmatrix}.$$

Marginalisation: Continuous Variables Suppose $\phi_{\underline{Y},\underline{X}_1,\underline{X}_2}$ is CG, where $\underline{Y}$ are discrete variables and $\underline{X}_1$ and $\underline{X}_2$ are continuous variables. That is, ϕ is given by

$$\phi_{\underline{Y},\underline{X}_1,\underline{X}_2}(\underline{y},\underline{x}_1,\underline{x}_2) = \chi(\underline{y})\exp\left\{g(\underline{y}) + h_1(\underline{y})\underline{x}_1^t + h_2(\underline{y})\underline{x}_2^t - \frac{1}{2}(\underline{x}_1,\underline{x}_2)\begin{pmatrix} K_{11} & K_{12} \\ K_{12}^t & K_{22} \end{pmatrix}\begin{pmatrix} \underline{x}_1^t \\ \underline{x}_2^t \end{pmatrix}\right\},$$

where

$$\chi(\underline{y}) = \begin{cases} 1 & \mathbb{P}_{\underline{Y}}(\underline{y}) > 0 \\ 0 & \mathbb{P}_{\underline{Y}}(\underline{y}) = 0 \end{cases}$$

K is symmetric and the triple (g,h,K) represent the canonical characteristics. Recall the standard result that, taking $\underline{z} \in \mathbb{R}^p$ as a row vector, and K a positive definite $p\times p$ symmetric matrix,

$$\frac{1}{(2\pi)^{p/2}}\int_{\mathbb{R}^p}\exp\left\{-\frac{1}{2}\underline{z}K\underline{z}^t\right\}d\underline{z} = \frac{1}{\sqrt{\det(K)}}$$

and hence that for $\underline{a} \in \mathbb{R}^p$ and K a positive definite $p\times p$ symmetric matrix

$$\int_{\mathbb{R}^p}\exp\left\{(\underline{a},\underline{z}) - \frac{1}{2}\underline{z}^tK\underline{z}\right\}d\underline{z} = \exp\left\{\frac{1}{2}a^tK^{-1}a\right\}\frac{(2\pi)^{p/2}}{\sqrt{\det(K)}}.$$

From this, it follows, after some routine calculation, that if $\underline{X}_1$ is a random p-vector with positive definite covariance matrix, then

$$\int_{\mathbb{R}^p}\phi_{\underline{Y},\underline{X}_1,\underline{X}_2}(\underline{y},\underline{x}_1,\underline{x}_2)d\underline{x}_1 = \chi(\underline{y})\exp\left\{\tilde{g}(\underline{y}) + \tilde{h}(\underline{y})\underline{x}_2^t - \frac{1}{2}\underline{x}_2\tilde{K}\underline{x}_2^t\right\},$$

where

$$\tilde{g}(\underline{y}) = g(\underline{y}) + \frac{1}{2}\left(p\log(2\pi) - \log\det(K_{11}(\underline{y})) + h_1(\underline{y})K_{11}(\underline{y})^{-1}h_1(\underline{y})^t\right),$$

$$\tilde{h}(\underline{y}) = h_2(\underline{y}), \qquad \tilde{K} = -K_{21}(\underline{y})K_{11}(\underline{y})^{-1}K_{12}(\underline{y}).$$

Marginalisation: Discrete Variables Consider a CG function $\phi_{\underline{Y}_1,\underline{Y}_2,\underline{X}}$, where $\underline{Y}_1$ and $\underline{Y}_2$ denote sets of discrete variables and $\underline{X}$ a set of continuous variables. Consider marginalisation over $\underline{Y}_2$. Firstly, if $h(\underline{y}_1,\underline{y}_2) = \tilde{h}(\underline{y}_1)$ and $K(\underline{y}_1\underline{y}_2) = \tilde{K}(\underline{y}_1)$ for some functions $\tilde{h}$ and $\tilde{K}$ (i.e. they do not depend on $\underline{y}_2$), then $\tilde{\phi}$, the marginal of $\phi_{\underline{Y}_1,\underline{Y}_2,\underline{X}}$ is simply

$$\tilde{\phi}(\underline{y}_1,\underline{x}) = \exp\left\{\tilde{h}(\underline{y}_1)\underline{x}^t - \frac{1}{2}\underline{x}\tilde{K}(\underline{y}_1)\underline{x}^t\right\}\sum_{\underline{y}_2}\chi(\underline{y}_1,\underline{y}_2)\exp\left\{g(\underline{y}_1,\underline{y}_2)\right\}.$$

The function $\tilde{\phi}$ is therefore CG with canonical characteristics $\tilde{g}(\underline{y}_1) = \log \sum_{\underline{y}_2} \exp\left\{g(\underline{y}_1, \underline{y}_2)\right\}$ and $\tilde{h}$, $\tilde{K}$ as before.

If either h or K depends on $\underline{y}_2$, then a marginalisation will not produce a CG distribution, so an *approximation* is used. For this, it is convenient to consider the *mean parameters*, $(\mathbb{P}, C, \mu)$, where $\mathbb{P}(\underline{y}_1, \underline{y}_2) = \mathbb{P}((\underline{Y}_1, \underline{Y}_2) = (\underline{y}_1, \underline{y}_2))$ and

$$\underline{X}|\{(\underline{Y}_1, \underline{Y}_2) = (\underline{y}_1, \underline{y}_2)\} = N(\mu(\underline{y}_1, \underline{y}_2), C(\underline{y}_1, \underline{y}_2)).$$

The approximation is as following: $\tilde{\phi}$ is *defined* as CG with mean parameters $(\tilde{\mathbb{P}}, \tilde{C}, \tilde{\mu})$ defined as:

$$\tilde{\mathbb{P}}(\underline{y}_1) = \sum_{\underline{y}_2} \mathbb{P}(\underline{y}_1, \underline{y}_2),$$

$$\tilde{\mu}(\underline{y}_1) = \frac{1}{\tilde{\mathbb{P}}(\underline{y}_1)} \sum_{\underline{y}_2} \mathbb{P}(\underline{y}_1, \underline{y}_2)\mu(\underline{y}_1, \underline{y}_2),$$

$$\tilde{C}(\underline{y}_1) = \frac{1}{\tilde{\mathbb{P}}(\underline{y}_1)} \sum_{\underline{y}_2} \mathbb{P}(\underline{y}_1, \underline{y}_2)\left(C(\underline{y}_1, \underline{y}_2) + (\mu(\underline{y}_1, \underline{y}_2) - \tilde{\mu}(\underline{y}_1))^t(\mu(\underline{y}_1, \underline{y}_2) - \tilde{\mu}(\underline{y}_1))\right).$$

It is relatively straightforward to compute that this approximate marginalisation has the correct expected value and second moments.

Marginalising over both Discrete and Continuous When marginalising over both types of variables, *first* the continuous variables are marginalised, and then the discrete.

Entering Evidence Two types of evidence can be entered; firstly, evidence that a continuous variable Y is instantiated as y for some $y \in \mathbb{R}$. Suppose $\mathbb{P}_{X|\Pi(X)}$ has canonical characteristics (g, h, K), where either $X = Y$ or $Y \in \Pi(X)$. The vector $(X, \Pi(X))$ may be re-ordered so that Y appears last, so that the canonical characteristics are written as

$$h(\underline{i}) = \begin{pmatrix} h_1(\underline{i}) & h_Y(\underline{i}) \end{pmatrix}, \qquad K(\underline{i}) = \begin{pmatrix} K_{11}(\underline{i}) & K_{1Y}(\underline{i}) \\ K_{Y1}(\underline{i}) & K_{YY}(\underline{i}) \end{pmatrix}.$$

It is straightforward to show that when $Y \leftarrow y$ is instantiated, $\mathbb{P}_{Y|\Pi(Y)}$ is replaced by a function $\psi_{\Pi(Y)}$ with canonical characteristics (g^*, h^*, K^*) given by

$$K^*(\underline{i}) = K_{11}(\underline{i})$$

$$h^*(\underline{i}) = h_1(\underline{i}) - yK_{1Y}(\underline{i})$$

$$g^*(\underline{i}) = g(\underline{i}) + h_Y(\underline{i})y - \frac{1}{2}y^2K_{AA}(\underline{i}).$$

The algorithm accommodates evidence on discrete variables in the form of information that certain states are impossible. If $(X, \Pi(X))$ contains discrete variables, then let $S_d = (\{X\} \cup \Pi(X)) \cap \Delta$ where Δ denotes the set of discrete variables. Then replace $\mathbb{P}_{X|\Pi(X)}$ by a function

$$\psi_{X,\Pi(X)} = f_{S_d}\mathbb{P}_{X|\Pi(X)}$$

where for each $\underline{s} \in \mathcal{X}_{S_d}$,

$$f_{S_d}(\underline{s}) = \begin{cases} 0 & \text{evidence states that } \underline{s} \text{ is impossible} \\ 1 & \text{otherwise} \end{cases}$$

The Fully Active Schedule The fully active schedule may now be applied. Firstly, the evidence is inserted. This is hard evidence, that certain states for discrete variables are excluded, or that the continuous variables take certain fixed values. The information then has to be propagated. Start at the leaves, send all messages to a CG root. A message from C to C' computes $\phi_S^* = \sum_{C\setminus S} \phi_C$, where the sum denotes an integral for a continuous variable and a sum for a discrete variable, updates $\phi_{C'}$ to $\phi_{C'}^* = \frac{\phi_S^*}{\phi_S}\phi_{C'}$ and updates ϕ_S to ϕ_{S^*}.

Note that, when two functions are multiplied or divided, this simply involves firstly: computing the canonical characteristics (either exactly, or those for the approximating function) and then if ϕ_1 has characteristics (g_1, h_1, K_1) and ϕ_2 has characteristics (g_2, h_2, K_2) then $\phi_1\phi_2$ has characteristics $(g_1 + g_2, h_1 + h_2, K_1 + K_2)$ and $\frac{\phi_1}{\phi_2}$ has characteristics $(g_1 - g_2, h_1 - h_2, K_1 - K_2)$.

When the root has received all messages, at this stage the potential over the root is *normalised*. That is, it is multiplied by a suitable constant to make it a probability. All the messages propagated to the CG root are proper marginalisations and therefore the distribution over the CG root, after the evidence is received, is an exact CG distribution.

For the propagation back out to the leaves, it will not, in general, be possible to make exact marginalinalisations. The same procedure is used; for a message C to C' separated by S, set $\phi_S^* = \sum_{C\setminus S} \phi_C$ and update $\phi_{C'}$ to $\frac{\phi_{S^*}}{\phi_S}\phi_{C'}$ and update ϕ_S to ϕ_S^*.

Having inserted hard evidence and run the schedule, since the potential over the root has been normalised, the resulting functions are probability distributions.

The approximate marginalisations give an approximate update, but by construction, since the tree has a strong root, the tree will be consistent; by construction, the exact marginalision of a clique in the direction of the strong root gives exactly the approximating distribution over the separator that is produced from by the approximate marginalisation when computing away from the root.

The Termination Although the resulting algorithm has produced approximate distributions over the cliques, which are conditional Gaussian, with the correct expectation vector and covariance structure, it should be clear from the algorithm that dividing the function over the clique by the function over the adjacent separator in the direction of the root gives the exact conditional distribution of the clique conditioned on the separator.

Notes The application of junction tree methods to conditional Gaussian distributions was taken from Lauritzen [81].

10.4 Exercises

1. Let

$$\underline{X} = (\underline{X}_\Delta, X_\Gamma) \sim \mathrm{CG}(|\Delta|, 1).$$

Let $\mathcal{I}$ denote the state space for $\underline{X}_\Delta$ and let $\mathbb{P}$ denote the probability function for the random vector $\underline{X}_\Delta$. Prove that

$$\mathbb{E}[X_\Gamma] = \sum_{i \in \mathcal{I}} \mathbb{P}(i)\mu(i)$$

and

$$\mathbb{V}(X_\Gamma) = \sum_{i \in \mathcal{I}} p(i)\sigma(i)^2 + \sum_{i \in \mathcal{I}} \mathbb{P}(i)\left(\mu(i) - \mathbb{E}[X_\Gamma]\right)^2.$$

2. Let $X \sim \mathrm{CG}(2,2)$ and let I_1 and I_2 be binary variables. Find the canonical parameters for the distribution.

3. Show that if a Conditional Gaussian Distribution is marginalised over a subset of the continuous variables, the resulting distribution is again a CG distribution. Find the canonical characteristics of the marginal distribution in terms of the original canonical characteristics, stating the standard results about multivariate normal random variables that you are using.

4. Suppose that hard evidence is entered into a subset of the *continuous* variables of a CG distribution. Show that the updated distribution is again a CG distribution and express the mean parameters (conditional expectation vector and covariance matrix) of the updated distribution in terms of the mean parameters of the original distribution.

5. This example is taken from Lauritzen [84]. It is a fictitious problem connected with controlling the emission of heavy metals from a waste incinerator. The type of incoming waste W affects the metals in the waste M_{in}, the dust emission D and the filter efficiency E. The quantity of metals in the waste M_{in} affects the metals emission M_{out}. Another important factor is the waste burning regimen B, which is monitored via the carbon dioxide concentration in the emission C. The burning regimen, the waste type and the filter efficiency E affect the dust emission D. The dust emission affects the metals emission and it is monitored by recording the light penetration L. The state of the filter F (whether it is intact or defective) affects E.

   The variables F, W, B are qualitative variables with states (the filter is either intact or defective, the waste is either industrial or household, the burning regimen is either stable or unstable). The variables E, C, D, L, M_{in} and M_{out} are continuous. The directed acyclic marked graph is given in Figure 10.1.

   (a)
   - Moralise and triangulate the graph.
   - Is it possible to construct a junction tree with a CG root?

(b)
- Moralise the graph.
- By adding in as few links as possible, construct a CG decomposable graph.
- Construct a junction tree. What are the possible strong roots for the junction tree?

(c) (HUGIN exercise) Programme the model in HUGIN, with the following conditional probabilities:

$$\mathbb{P}_B(1) = 0.85, \qquad \mathbb{P}_B(0) = 0.15 \qquad 1 = \text{stable} \qquad 0 = \text{unstable}$$

$$\mathbb{P}_F(1) = 0.95 \qquad \mathbb{P}_F(0) = 0.05 \qquad 1 = \text{intact} \qquad 0 = \text{defect}$$

$$\mathbb{P}_W(1) = 0.25 \qquad \mathbb{P}_W(0) = 0.75 \qquad 1 = \text{industrial} \qquad 0 = \text{household}$$

$$E|(F,W) = (1,0) \sim N(-3.2, 0.00002) \qquad E|(F,W) = (0,0) \sim N(-0.5, 0.0001)$$

$$E|(F,W) = (1,1) \sim N(-3.9, 0.00002) \qquad E|(F,W) = (0,1) \sim N(-0.4, 0.0001)$$

$$D|(B,W,E) = (1,1,x) \sim N(6.5 + x, 0.03) \qquad D|(B,W,E) = (1,0,x) \sim N(6.0 + x, 0.04)$$

$$D|(B,W,E) = (0,1,x) \sim N(7.5 + x, 0.1) \qquad D|(B,W,E) = (0,0,x) \sim N(7.0 + x, 0.1)$$

$$C|B = 1 \sim N(-2, 0.1) \qquad C|B = 0 \sim N(-1, 0.3)$$

$$L|D = x \sim N(3 - \frac{1}{2}d, 0.25)$$

$$M_{in}|W = 1 \sim N(0.5, 0.01) \qquad M_{in}|W = 0 \sim N(-0.5, 0.005)$$

$$M_{out}|D = x, M_{in} = y \sim N(x + y, 0.002)$$

The variable E, filter efficiency, is represented on a logarithmic scale. It is assumed that

$$\text{dust out} = \text{dust in} \times \rho$$

and $E = \log \rho$. The variable D, dust emission, is again on a logarithmic scale, as is C, the CO_2 concentration and L, the light penetrability. Light penetrability is roughly inversely proportional to the square root of dust concentration. The metal in waste M_{in} and metal emission M_{out} variables are on logarithmic scales.

Suppose that the waste burned is of industrial type ($W = 1$), the light penetration variable is measured as $L = 1.1$ and the CO_2 concentration is measured as $C = -0.9$.

Find the updated probability distributions for B and F and the updated means and variances for M_{in}, M_{out} and D.

Chapter 11

Gaussian and Conditional Gaussian Graphical Models in R

The packages **ggm**, **deal**, **glasso**, **gRc**, **pcalg**, **bnlearn**, **gRim** are useful.

Consider $\underline{X} \sim N(\underline{\mu}, \Sigma)$. The matrix $K = \Sigma^{-1}$ is known as the *concentration matrix*. The *partial correlation* between X_u and X_v given all the other variables may be derived from K as:

$$\rho_{uv|V\setminus uv} = -\frac{K_{uv}}{\sqrt{K_{uu}K_{vv}}}.$$

Thus, the independence graph does not have an edge $u \leftrightarrow v$ if and only if $K_{uv} = 0$.

Consider an illustrative example of 'carcass' data:

```
> library(gRbase)
> data(carcass)
> head(carcass)
  Fat11 Meat11 Fat12 Meat12 Fat13 Meat13 LeanMeat
1    17     51    12     51    12     61 56.52475
2    17     49    15     48    15     54 57.57958
3    14     38    11     34    11     40 55.88994
4    17     58    12     58    11     58 61.81719
5    14     51    12     48    13     54 62.95964
6    20     40    14     40    14     45 54.57870
```

The concentration matrix can be estimated as:

```
> S.carc = cov.wt(carcass,method="ML")$cov
> K.carc = solve(S.carc)
> round(100*K.carc)
         Fat11 Meat11 Fat12 Meat12 Fat13 Meat13 LeanMeat
Fat11       44      3   -20     -7   -16      4       10
Meat11       3     16    -3     -6    -6     -6       -3
```

```
Fat12      -20     -3    54      6   -21     -5        9
Meat12      -7     -6     6     14    -1     -9        0
Fat13      -16     -6   -21     -1    56      3        7
Meat13       4     -6    -5     -9     3     16       -1
LeanMeat    10     -3     9      0     7     -1       26
```

The partial correlation is obtained using `cov2pcor`:

```
> PC.carc = cov2pcor(S.carc)
> round(100*PC.carc)
         Fat11 Meat11 Fat12 Meat12 Fat13 Meat13 LeanMeat
Fat11      100    -11    41     30    32    -16      -29
Meat11     -11    100     9     41    19     35       16
Fat12       41      9   100    -24    38     18      -24
Meat12      30     41   -24    100     2     61        2
Fat13       32     19    38      2   100     -9      -18
Meat13     -16     35    18     61    -9    100        7
LeanMeat   -29     16   -24      2   -18      7      100
```

`Fat13` is conditionally independent of `Meat12` and `LeanMeat` is also conditionally independent of `Meat12`. A stepwise backward model selection procedure can be carried out as follows:

```
>library(gRim)
> sat.carc = cmod(~.^.,data=carcass)
> aic.carc = stepwise(sat.carc)
> library(Rgraphviz)
> plot(as(aic.carc,"graphNEL"),"fdp")
```

The BIC gives a higher penalty for complexity and also removes edges between `Fat13` and `Meat13`.

```
> bic.carc = stepwise(sat.carc,k=log(nrow(carcass)))
> bic.carc
Model: A cModel with 7 variables
 graphical :  TRUE  decomposable :  TRUE
 -2logL    :       11376.07 mdim :   24 aic :     11424.07
 ideviance :        2465.16 idf  :   17 bic :     11516.25
 deviance  :           8.62 df   :    4
plot(as(bic.carc,"graphNEL"),"fdp")
```

11.1 Undirected Gaussian Graphical Models

The model `gen.carc` for the carcass data specifies a UGGM (undirected Gaussian graphical model) with edges missing for all partial correlations less than or equal to 0.12.

```
> gen.carc = cmod(~Fat11*Fat12*Meat12*Meat13
+ + Fat11*Fat12*Fat13*LeanMeat
+ +Meat11*Meat12*Meat13
+ +Meat11*Fat13*LeanMeat,data=carcass)
> gen.carc
Model: A cModel with 7 variables
 graphical :  TRUE  decomposable : FALSE
 -2logL    :       11387.24 mdim :   22 aic :     11431.24
 ideviance :        2453.99 idf  :   15 bic :     11515.73
 deviance  :          19.79 df   :    6
> plot(gen.carc,"neato")
```

Alternatively, the model could be specified as follows:

```
> edge.carc=cmod(edgeList(as(gen.carc,"graphNEL")),data=carcass)
> edge.carc
Model: A cModel with 7 variables
 graphical :  TRUE  decomposable : FALSE
 -2logL    :       11387.24 mdim :   22 aic :     11431.24
 ideviance :        2453.99 idf  :   15 bic :     11515.73
 deviance  :          19.79 df   :    6
```

The matrix K is estimated by iterative proportion scaling. The point is that the estimate has to satisfy the constraint that $K_{uv} = 0$ when there is no edge $u \leftrightarrow v$.

```
> carcfit1 =
+ ggmfit(S.carc,n=nrow(carcass),edgeList(as(gen.carc,"graphNEL")))
> carcfit1[c("dev","df","iter")]
$dev
[1] 19.78537

$df
[1] 6

$iter
[1] 774
```

Hypothesis Testing A likelihood ratio test, to see whether model $\mathcal{M}_1$ gives a better fit than model $\mathcal{M}_2$ may be carried out as follows:

```
> comparemodels = function(m1,m2){}
> comparemodels=function(m1,m2){
```

```
+ lrt = m2$fitinfo$dev - m1$fitinfo$dev
+ dfdiff = m2$fitinfo$dimension[4]-m1$fitinfo$dimension[4]
+ names(dfdiff)=NULL
+ list('lrt'=lrt,'df'=dfdiff)
+ }
> comparemodels(aic.carc,bic.carc)
$lrt
[1] 8.372649

$df
[1] 2
```

This would indicate that the smaller model does not fit well.

The function `ciTest_mvn()` tests single conditional independence hypotheses. To test `LeanMeat` $\perp$ `Meat13`|`remaining variables`

```
> ciTest_mvn(list(cov=S.carc,n.obs=nrow(carcass)),
+ set=~LeanMeat+Meat13+Meat11+Meat12+Fat11+Fat12+Fat13)
Testing LeanMeat _|_ Meat13 | Meat11 Meat12 Fat11 Fat12 Fat13
Statistic (DEV):    1.687 df: 1 p-value: 0.1940 method: CHISQ
```

Gaussian conditional independence can be tested from the **pcalg** package as follows:

```
> library(pcalg)
> C.carc=cov2cor(S.carc)
> gaussCItest(7,2,c(1,3,4,5,6),list(C=C.carc,n=nrow(carcass)))
[1] 0.003077247
```

11.2 Decomposition of UGGMs

Consider the model with BIC penalisation. Let A = {`Fat13,LeanMeat`}, B = {`Meat12, Meat13`} and S = {`Fat11, Fat12, Meat11`}. Then (A, B, S) is a decomposition of its independence graph. Furthermore, both $\mathcal{M}_{A\cup S}$ and $\mathcal{M}_{B\cup S}$ are saturated. It follows that $\widehat{K}_{A\cup S} = S^{-1}_{A\cup S}$ and $\widehat{K}_{B\cup S} = S^{-1}_{B\cup S}$. The MLE of K may therefore be found using:

$$\widehat{K} = (\widehat{K}_{A\cup S})^{A\cup B\cup S} + (\widehat{K}_{B\cup S})^{A\cup B\cup S} - (S_S^{-1})^{A\cup B\cup S}$$

where the meanings of the terms are clear.

```
> K.hat = S.carc
> K.hat[]=0
```

```
> AC=c("Fat11","Fat12","Fat13","Meat11","LeanMeat")
> BC=c("Meat11","Meat12","Meat13","Fat11","Fat12")
> C=c("Fat11","Fat12","Meat11")
> K.hat[AC,AC]=K.hat[AC,AC]+solve(S.carc[AC,AC])
> K.hat[BC,BC]=K.hat[BC,BC]+solve(S.carc[BC,BC])
> K.hat[C,C]=K.hat[C,C]-solve(S.carc[C,C])
> round(100*K.hat)
         Fat11 Meat11 Fat12 Meat12 Fat13 Meat13 LeanMeat
Fat11       44      1   -20     -7   -16      6       10
Meat11       1     16    -4     -6    -4     -5       -5
Fat12      -20     -4    54      6   -20     -4        9
Meat12      -7     -6     6     14     0     -9        0
Fat13      -16     -4   -20      0    55      0        7
Meat13       6     -5    -4     -9     0     16        0
LeanMeat    10     -5     9      0     7      0       26
> Sigma.hat=solve(K.hat)
> round(Sigma.hat,2)
         Fat11 Meat11 Fat12 Meat12 Fat13 Meat13 LeanMeat
Fat11    11.34   0.74  8.42   2.06  7.66  -0.76    -9.08
Meat11    0.74  32.97  0.67  35.94  2.01  31.97     5.33
Fat12     8.42   0.67  8.91   0.31  6.84  -0.60    -7.95
Meat12    2.06  35.94  0.31  51.79  2.45  41.47     5.41
Fat13     7.66   2.01  6.84   2.45  7.62   0.89    -6.93
Meat13   -0.76  31.97 -0.60  41.47  0.89  41.44     6.43
LeanMeat -9.08   5.33 -7.95   5.41 -6.93   6.43    12.90

> round(S.carc,2)
         Fat11 Meat11 Fat12 Meat12 Fat13 Meat13 LeanMeat
Fat11    11.34   0.74  8.42   2.06  7.66  -0.76    -9.08
Meat11    0.74  32.97  0.67  35.94  2.01  31.97     5.33
Fat12     8.42   0.67  8.91   0.31  6.84  -0.60    -7.95
Meat12    2.06  35.94  0.31  51.79  2.18  41.47     6.03
Fat13     7.66   2.01  6.84   2.18  7.62   0.38    -6.93
Meat13   -0.76  31.97 -0.60  41.47  0.38  41.44     7.23
LeanMeat -9.08   5.33 -7.95   6.03 -6.93   7.23    12.90
```

Model Search using gRim Setting `search = "headlong"` causes edges to be searched in a random order, which can make the search faster.

```
> ind.carc=cmod(~.^1,data=carcass)
> set.seed(123)
```

```
> forw.carc=stepwise(ind.carc,search="headlong",
+ direction="forward",k=log(nrow(carcass)),details=0)
> forw.carc
Model: A cModel with 7 variables
 graphical :  TRUE  decomposable :  TRUE
 -2logL    :       11393.53 mdim :   23 aic :     11439.53
 ideviance :        2447.70 idf  :   16 bic :     11527.87
 deviance  :          26.08 df   :    5
> plot(forw.carc,"neato")
```

11.3 Directed Gaussian Graphical Models

A DAG may be constructed as follows: for example,

```
> gdag1=DAG(LeanMeat~Meat13:Fat11:Fat12, Meat13~Meat11:Meat12,
+ Fat12~Fat11,Fat13~Meat11:Meat12, Meat12~Meat11)
> plot(as(gdag1,"graphNEL"))
> fdag1=fitDag(gdag1,S.carc,nrow(carcass))
> fdag1$dev
[1] 552.2726
> fdag1$df
[1] 12
```

The function `essentialGraph()` from the **ggm** package returns the essential graph of a DAG. For example:

```
> eG1 = as(essentialGraph(gdag1),"igraph")
> V(eG1)$size = 40
> E(eG1)$arrow.mode=2
> E(eG1)[is.mutual(eG1)]$arrow.mode = 0
> plot(eG1,layout=layout.kamada.kawai)
```

Model Selection A DAG may be established using the package **pcalg** package. The PC algorithm may be used to find the skeleton. The `pcalg::skeleton` command ensures that the relevant version of the command `skeleton` is used.

```
> library(pcalg)
> c.carc=cov2cor(S.carc)
> suffStat=list(C=c.carc,n=nrow(carcass))
> indepTest=gaussCItest
> skeleton.carc=pcalg::skeleton(suffStat,gaussCItest,p=ncol(carcass),alpha=0.05)
```

```
> nodes(skeleton.carc@graph)=names(carcass)
> names(carcass)
[1] "Fat11"    "Meat11"   "Fat12"    "Meat12"   "Fat13"    "Meat13"
[7] "LeanMeat"
> str(skeleton.carc@sepset[[1]])
List of 7
 $ : NULL
 $ : int(0)
 $ : NULL
 $ : int(0)
 $ : NULL
 $ : int(0)
 $ : NULL
```

This is read as follows: The first variable `Fat11` was marginally independent of variables `Meat11`, `Meat12` and `Meat13`. This is seen from the designation `NULL`. Similarly,

```
> str(skeleton.carc@sepset[[2]])
List of 7
 $ : NULL
 $ : NULL
 $ : int(0)
 $ : NULL
 $ : int 4
 $ : NULL
 $ : int 6
```

This indicates that `Meat11` (the second variable) is *marginally* independent of `Fat12`, *conditionally* independent of `Fat13` (5th variable on the list) *given* `Meat12` (4th variable on the list) and conditionally independent of `LeanMeat` given `Meat13` etc.

In **pcalg**, there are several options for turning a skeleton together with sep-sets into a DAG. These are:

- `udag2pdag()`
- `udag2pdagRelaxed()`
- `udag2pdagSpecial()`

Read the help functions to find out the differences. For example:

```
> pdag.carc=udag2pdagRelaxed(skeleton.carc,verbose=0)
> nodes(pdag.carc@graph)=names(carcass)
> plot(pdag.carc@graph,"neato")
```

Undirected edges are shown as double-arrowed edges. This graph is *not* an essential graph; the arrow from `Meat12` to `Meat13` is not part of an immorality, neither is it a compelled edge.

Both steps (skeleton and edge orientation) can be called simultaneously using the function `pc()`. For example,

```
> cpdag.carc=pc(suffStat,gaussCItest,p=ncol(carcass),alpha=0.05)
> plot(cpdag.carcass@graph)
```

11.4 Gaussian Chain Graph Models

To build a chain graph model, firstly an undirected graph, corresponding to the independence graph is constructed.

The package **lcd** is useful. This uses a constraint-based algorithm due to Ma et. al. [86](2008). A junction tree for the undirected graph is then derived. The algorithm then performs a series of conditional independence tests following a scheme based on the junction tree. This may be applied to the carcass data:

```
> library(lcd)
> ug<-naive.getug.norm(carcass,0.05)
> jtree<-ug.to.jtree(ug)
> cg<-learn.mec.norm(jtree,cov(carcass),nrow(carcass),0.01,"CG")
> icg<-as(cg,"igraph")
> E(icg)$arrow.mode<-2
> E(icg)[is.mutual(icg)]$arrow.mode<-0
> V(icg)$size<-40
> plot(icg,layout=layout.kamada.kawai)
```

11.5 Conditional Gaussian Models

Recall that, for a conditional Gaussian model, the covariance of the continuous variables, conditioned on the discrete, is the same for each value of the discrete variables. That is, conditioned on the discrete variables taking configuration i, the Gaussian variables have distribution:

$$\pi_{\Gamma|\Delta}(y|i) = \frac{1}{(2\pi)^{q/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(y-\mu(i))^t\Sigma^{-1}(y-\mu(i))\right\}.$$

For illustration, consider two data sets from **gRbase**; `milkcomp1` and `wine`.

The `CGstats()` function calculates the number of observations and means of the continuous variables for each cell i, together (by default) with a common covariance matrix.

```
> data(milkcomp1,package='gRbase')
> head(milkcomp1)
```

```
  treat  fat protein    dm lactose
1     d 6.16    6.65 18.55    5.06
2     c 4.06    5.44 18.32    5.23
3     f 9.25    5.67 20.68    5.15
4     b 5.82    5.62 17.57    5.74
5     a 4.98    5.37 16.38    5.55
6     b 9.06    5.08 20.21    5.29
> library(gRim)
> SS = CGstats(milkcomp1,varnames=c("treat","fat","protein","lactose"))
> SS
$n.obs
treat
a b c d e f g
8 8 8 8 8 7 8

$center
              a       b      c       d       e        f       g
fat     6.64125 8.01000 7.0525 7.40125 8.13375 7.518571 6.97375
protein 5.48750 5.28750 5.4750 5.81750 5.26250 5.295714 5.58000
lactose 5.49125 5.48875 5.4675 5.31375 5.40625 5.382857 5.41500

$cov
                fat     protein     lactose
fat      2.31288338  0.19928422 -0.07028198
protein  0.19928422  0.12288675 -0.03035208
lactose -0.07028198 -0.03035208  0.04529896

$cont.names
[1] "fat"     "protein" "lactose"

$disc.names
[1] "treat"

$disc.levels
[1] 7
```

The coefficients of variation are:

```
> apply(SS$center,1,sd)/apply(SS$center,1,mean)
       fat    protein    lactose
0.07415672 0.03656048 0.01186589
```

The corresponding canonical parameters are:

```
> can.parms=CGstats2mmodParms(SS,type="ghk")
> print(can.parms,simplify=FALSE)
$g
treat
        a         b         c         d         e         f         g
-745.4933 -729.3707 -740.4563 -743.5508 -712.6533 -710.4957 -740.1503

$h
            [,1]       [,2]        [,3]       [,4]       [,5]       [,6]        [,7]
[1,]   0.7869693   1.628323   0.9975735   0.873605   1.686407   1.343883   0.8641906
[2,]  88.2214588  85.006209  87.6318341  90.151087  84.137184  84.816560  88.5106553
[3,] 181.5552534 180.651093 180.9626426 179.064184 178.337697 177.745064 180.1855748

$K
           [,1]       [,2]       [,3]
[1,]  0.5055681 -0.7503065  0.2816613
[2,] -0.7503065 10.8648914  6.1157915
[3,]  0.2816613  6.1157915 26.6103828

$gentype
[1] "mixed"

$gentype
[1] "mixed"

$cont.names
[1] "fat"     "protein" "lactose"

$disc.names
[1] "treat"

$disc.levels
[1] 7
```

Let j denote the level of the treatment factor, then $h(j)$ takes the form:

$$h(j) = (h^{\text{fat}}(j), h^{\text{protein}}(j), h^{\text{lactose}}(j))$$

The coefficients for h are:

```
> apply(can.parms$h,1,sd)/apply(can.parms$h,1,mean)
[1] 0.32484006 0.02614999 0.00793359
```

This suggests that h^{lactose} is a *constant* function of j. In other words,

$$\texttt{lactose} \perp \texttt{treat} | (\texttt{fat, protein}).$$

The partial correlation matrix is:

```
> conc2pcor(can.parms$K)
            [,1]       [,2]        [,3]
[1,]  1.00000000  0.3201373 -0.07679125
[2,]  0.32013725  1.0000000 -0.35967845
[3,] -0.07679125 -0.3596784  1.00000000
```

This suggests that the partial correlation between `fat` and `lactose` is zero. Therefore,

$$\texttt{lactose} \perp \texttt{fat} | (\texttt{treat, protein}).$$

The *generators* of the model are simply the cliques of the CG junction tree. The `mmod()` function from **gRim** allowed CG models to be defined using model formulae. For example, to construct a model with generators `treat, fat, protein,protein,lactose`,

```
> milkmod = mmod(~treat*fat*protein+protein*lactose, data=milkcomp1)
> milkmod
Model: A mModel with 4 variables
 graphical :  TRUE  decomposable :  TRUE
 -2logL    :         428.47 mdim :   26 aic :       480.47
 ideviance :          18.97 idf  :   15 bic :       532.66
 deviance  :           2.11 df   :    7
```

Conditional Gaussian Models To construct a marked graph, the information on marking has to be provided:

```
> uG1=ug(~a:b+b:c+c:d)
> uG2=ug(~a:b+a:d+c:d)
> mcsmarked(uG1,discrete=c("a","d"))
character(0)
> mcsmarked(uG2,discrete=c("a","d"))
[1] "a" "d" "b" "c"
> plot(uG1)
> plot(uG2)
```

For the first graph, both a and d have to be in the CG-root, hence the root contains *all* the variables; the CG-Gaussian tree contains exactly one node. For the second one, the CG-root is the clique $\{a, d\}$.

Using gRim for CG-models The function `mmod()` enables CG models to be defined and fitted.

```
> glist = ~treat:fat:protein+protein:lactose
> milk=mmod(glist,data=milkcomp1)
> milk
Model: A mModel with 4 variables
 graphical :  TRUE  decomposable :  TRUE
 -2logL    :         428.47 mdim :   26 aic :       480.47
 ideviance :          18.97 idf  :   15 bic :       532.66
 deviance  :           2.11 df   :    7
> summary(milk)
Mixed interaction model:
Generators:
  :"treat" "fat" "protein"
  :"protein" "lactose"
Discrete:   1   Continuous:   3
Is graphical: TRUE  Is decomposable: TRUE
logL: -214.233011, iDeviance: 241.774364
```

The parameters are obtained using `coef()`. The parametrisation may be specified as either canonical or mean field. For canonical parameters:

```
> coef(milk,type="ghk")
$g
treat
        a         b         c         d         e         f         g
-676.0550 -666.0859 -675.0546 -690.9918 -664.9730 -666.7805 -680.0217

$h
           [,1]        [,2]        [,3]       [,4]        [,5]        [,6]       [,7]
[1,]  -1.134727  -0.2838037  -0.9178505  -1.021725  -0.2012326  -0.5374842  -1.043008
[2,]  84.348819  81.3413696  83.8953921  86.850963  81.0040255  81.8196051  84.952805
[3,] 164.633541 164.6335413 164.6335413 164.633541 164.6335413 164.6335413 164.633541

$K
           [,1]      [,2]      [,3]
[1,]  0.5025868 -0.815040  0.000000
[2,] -0.8150400 10.762254  5.666744
[3,]  0.0000000  5.666744 24.645834

$gentype
```

```
[1] "mixed"

$cont.names
[1] "fat"     "protein" "lactose"

$disc.names
[1] "treat"

$disc.levels
[1] 7

$N
[1] 55

$SSD
           [,1]      [,2]      [,3]
[1,] 127.208586 10.960632 -3.865509
[2,]  10.960632  6.758771 -1.669364
[3,]  -3.865509 -1.669364  2.491443

$SS
             fat  protein  lactose
fat     3143.500 2227.141 2199.894
protein 2227.141 1648.827 1627.220
lactose 2199.894 1627.220 1620.993
```

Updating Models Models are updated using `update()`. A list with one or more components `add.edge`, `drop.edge`, `add.term`, `drop.term` is specified. The updates are made in the order given. For example:

```
> milk2 = update(milk,list(add.edge=~fat:lactose,drop.edge=~treat:protein))
> milk2
Model: A mModel with 4 variables
 graphical :  TRUE  decomposable :  TRUE
 -2logL    :         446.17 mdim :   21 aic :       488.17
 ideviance :          10.12 idf  :   10 bic :       530.33
 deviance  :          10.96 df   :   12
```

Inference Functions such as `ciTest()`, `testInEdges()`, `testOutEdges()` etc. have the same behaviour as with pure discrete and pure continuous networks. For example:

```
> ciTest(milkcomp1)
Testing treat _|_ fat | protein dm lactose
Statistic (DEV):    8.742 df: 6 p-value: 0.1886 method: CHISQ

> testInEdges(milk,getInEdges(milk$glist))
  statistic df     p.value        aic      V1      V2 action
1  11.06071  6 0.086518199 -0.9392919   treat     fat      +
2  18.68943  6 0.004721598  6.6894264   treat protein      -
3   8.27794  1 0.004012963  6.2779399     fat protein      -
4  10.24527  1 0.001370352  8.2452747 protein lactose      -

> testOutEdges(milk,getOutEdges(milk$glist))
  statistic df   p.value      aic    V1      V2 action
1 3.8928582  6 0.6911730 8.107142 treat lactose      -
2 0.9827155  1 0.3215293 1.017285   fat lactose      -

> milk3=update(milk,list(drop.edge=~treat:protein))
> compareModels(milk,milk3)
Large:
  :"treat" "fat" "protein"
  :"protein" "lactose"
Small:
  :"protein" "lactose"
  :"treat" "fat"
  :"fat" "protein"
-2logL:    18.69 df: 6 AIC(k= 2.0):     6.69 p.value: 0.155100

> testdelete(milk,c("treat","protein"))
dev:   18.689 df:  6 p.value: 0.00472 AIC(k=2.0):    6.7 edge: treat:protein
Notice: Test perfomed by comparing likelihood ratios
> testadd(milk,c("treat","lactose"))
dev:    3.893 df:  6 p.value: 0.69117 AIC(k=2.0):    8.1 edge: treat:lactose
Notice: Test perfomed by comparing likelihood ratios
```

Stepwise Model Selection The `stepwise()` function in `gRim` implements stepwise selection. The following starts from the saturated model and uses BIC criterion. This function can take a while to produce the output.

```
> data(wine,package='gRbase')
> mm=mmod(~.^.,data=wine)
> mm2=stepwise(mm,k=log(nrow(wine)),details=0)
> plot(mm2)
```

Chapter 12

Learning the Conditional Probability Functions

12.1 Introduction

Let $\underline{X} = (X_1, \ldots, X_d)$ be a random vector, whose probability distribution factorises along a DAG $\mathcal{G} = (V, D)$. This chapter considers the task of learning the conditional probability potentials, when the DAG $\mathcal{G}$ is given, when presented with an $n \times d$ data matrix of instantiations

$$\mathbf{x} = \begin{pmatrix} \underline{x}_{(1)} \\ \vdots \\ \underline{x}_{(n)} \end{pmatrix}$$

which are the realisation of a random matrix

$$\mathbf{X} = \begin{pmatrix} \underline{x}_{(1)} \\ \vdots \\ \underline{X}_{(n)} \end{pmatrix}.$$

Bayesian network analysis restricts itself to three settings:

- Gaussian
- Multinomial
- Conditional Gaussian.

12.2 Gaussian and Conditional Gaussian Networks

For Gaussian networks, $\underline{X} \sim N(\underline{\mu}, \Sigma)$. The model is:

$$\begin{cases} \underline{X} = \underline{\mu} + \underline{\epsilon} & \underline{\epsilon} \sim N(\underline{0}, \Sigma) \\ \mu_j = \beta_{j0} + \sum_{k \epsilon \mathrm{Pa}(j)} \beta_{jk} \mu_k \end{cases}$$

where $\mathrm{Pa}(j)$ denotes the indices of the parent nodes of X_j and for different instantiations, the $\underline{\epsilon}$ are i.i.d. Estimation of the parameters $\beta_{kj} : k \in \{0\} \cup \mathrm{Pa}(j)$ is carried out simply by maximum likelihood estimation, which is equivalent to least squares for Gaussian variables. That is, the parameters β are estimated by minimising:

$$\sum_{i=1}^{n}(x_{ij} - \beta_{0j} - \sum_{k \epsilon \mathrm{Pa}_j} x_{ik}\beta_{kj})^2.$$

We assume that the nodes have been ordered so that $\mathrm{Pa}_j \subseteq \{1, \ldots, j-1\}$. Denote the estimator by $\widehat{\beta}$. Then

$$\widehat{\mu}_j = \widehat{\beta}_{0j} + \sum_{k \epsilon \mathrm{Pa}_j} \widehat{\mu}_k \widehat{\beta}_{kj}.$$

The estimate of Σ is slightly harder than the estimate of μ, since we have to ensure that the conditional independence constraints are satisfied; the estimate $\widehat{\Sigma}$ has to correspond to the factorisation.
The estimate $\widehat{\Sigma}_{11}$ is simply the m.l.e. variance estimate derived from the univariate sample $\mathbf{x}_{\cdot 1}$ from a $N(\mu_1, \Sigma_{11})$ distribution.

For $j > 1$, assume that the components of the $j-1 \times j-1$ sub-matrix $\Sigma^{(j-1)}$, with entries $\Sigma_{ab} : 1 \le a \le j-1, 1 \le b \le j-1$, have already been estimated. Then set $(\widehat{\Sigma}^{(j)-1})_{ij} = 0$ for $i \notin \mathrm{Pa}_j$. Now let A denote the $j \times j$ symmetric matrix obtained as the maximiser of:

$$\frac{|A|^{1/2}}{(2\pi)^{j/2}} \exp\left\{-\frac{1}{2}\sum_{i=1}^{n}\sum_{a=1}^{j}\sum_{b=1}^{j}(x_{ia} - \widehat{\mu}_a)(x_{ib} - \widehat{\mu}_b)A_{ab}\right\}.$$

subject to the constraint that $A_{ij} = 0$ for $i < j, i \notin \mathrm{Pa}_j$. For $i \in \mathrm{Pa}_j \cup \{j\}$, set $\widehat{\Sigma}_{ij} = A^{-1}_{ij}$.

Conditioned Gaussian Part of Conditional Gaussian Similarly, for Conditional Gaussian, the parameters of the Gaussian variables, conditioned on the discrete, are estimated by maximum likelihood in the standard way for multivariate Gaussian.

12.3 Discrete Variables

Let $\underline{X} = (X_1, \ldots, X_d)$ be a random vector of discrete variables, with probability function $\mathbb{P}_{X_1,\ldots,X_d}$ over state space $\mathcal{X} = \times_{j=1}^{d}\mathcal{X}_j$ where $\mathcal{X}_j = (x_j^{(1)}, \ldots, x_j^{(k_j)})$ is the state space for X_j. Suppose that $\mathbb{P}$ factorises according to a DAG $\mathcal{G} = (V, D)$. For $\underline{x} \in \mathcal{X}$, let $\pi_j(\underline{x})$ denote the parent configuration for variable X_j when $\underline{X} = \underline{x}$ and let $(\pi_j^{(l)})_{l=1}^{q_j}$ denote a listing of the possible parent configurations for X_j. Let

$$\theta_{jil} = \mathbb{P}_{X_j|\mathrm{Pa}_j}(x_j^{(i)}|\pi_j^{(l)}) \quad j = 1, \ldots, d,\ i = 1, \ldots, k_j,\ l = 1, \ldots, q_j \tag{12.1}$$

Let be an $n \times d$ data matrix, representing n instantiations of $\underline{X}$. The aim is to estimate the values of $(\theta_{jil})_{j,i,l}$ based on the data matrix.

For discrete variables there are two approaches; the maximum likelihood method and the Bayesian approach.

12.4 Maximum Likelihood for Discrete Variables

We describe maximum likelihood for multinomial sampling. The crucial point about a Bayesian network is *modularity*; the components of the network are conditionally independent and each component can be treated separately, as a multinomial.

12.4.1 Maximum Likelihood for Multinomial Sampling

Let X be a random variable with state space $\mathcal{X} = (x^{(1)}, \ldots, x^{(k)})$ and $\theta_i = \mathbb{P}_X(x^{(i)})$, $i = 1, \ldots, k$, where $\theta_i \geq 0$ for $i = 1, \ldots, k$ and $\sum_{i=1}^{k} \theta_i = 1$. Let $\mathbf{X} = (X_1, \ldots, X_n)^t$ be n independent identically distributed copies of X and let $\mathbf{x} = (x_1^{(i_1)}, \ldots, x_n^{(i_n)})^t$ be an instantiation of $\mathbf{X}$.
Let

$$n_l = \text{number of times } x^{(l)} \text{ appears in } \mathbf{x}, \quad l = 1, \ldots, k$$

so that $n = n_1 + \ldots + n_k$. The probability of $\mathbf{x}$ is:

$$\mathbb{P}_{\mathbf{X}}(\mathbf{x}|\underline{\theta}) = \theta_1^{n_1} \ldots, \theta_k^{n_k}.$$

Maximum Likelihood Let

$$\Theta = \left\{ (\theta_1, \theta_2, \ldots, \theta_k) \mid \theta_j \geq 0,\ j = 1, \ldots, k,\ \sum_{j=1}^{k} \theta_j = 1 \right\}$$

denote the parameter space.

Definition 12.1 (Likelihood function, Likelihood Estimate, Log Likelihood Function)**.** *The* likelihood function of the parameters $\underline{\theta}$ *is defined as*

$$L(\underline{\theta}|\mathbf{x}) = \mathbb{P}_{\mathbf{X}}(\mathbf{x}|\underline{\theta}).$$

The maximum likelihood estimate $\widehat{\underline{\theta}}_{ME}$ *is defined as the value of* $\underline{\theta}$ *that maximises* $L(\underline{\theta}|\mathbf{x})$*. The* log likelihood function *is:*

$$\log L\left(\theta_1, \theta_2, \ldots, \theta_k\right) = \log \mathbb{P}_{\mathbf{X}}\left(\mathbf{x} \mid \underline{\theta}\right),$$

where log *is used to denote the* natural *logarithm.*

There is an elegant expression of the likelihood function in terms of the *Shannon Entropy* and *Kullback Leibler divergence* given below.

Definition 12.2 (Shannon Entropy)**.** *The* Shannon Entropy*, or* Entropy *of a probability distribution* $\underline{\theta} = (\theta_1, \ldots, \theta_k)$*, where* $\theta_j \geq 0$*,* $j = 1, \ldots, k$ *and* $\theta_1 + \ldots + \theta_k = 1$ *is defined as*

$$H(\underline{\theta}) = -\sum_{j=1}^{k} \theta_j \log \theta_j.$$

In the definition of $H(\underline{\theta})$, the definition $0 \log 0 = 0$ is used, obtained by continuous extension of the function $x \log x$, $x > 0$.

Note that $H(\underline{\theta}) \geq 0$. Recall the definition of *Kullback Leibler divergence*:

Definition 12.3 (Kullback Leibler Divergence)**.** *The Kullback Leibler Divergence between two discrete probability functions f and g with the same state space $\mathcal{X}$ is defined as*

$$D_{KL}(f|g) = \sum_{x \in \mathcal{X}} f(x) \log \frac{f(x)}{g(x)}.$$

The Kullback Leibler divergence has the property that it is non negative, and for two probability measures defined on the same finite state space, $D_{KL}(f|g) = 0$ if and only if $f = g$. This is a consequence of Jensen's inequality and is now stated.

Lemma 12.4. *For any two discrete probability distributions f and g, it holds that*

$$D_{KL}(f|g) \geq 0$$

and $D_{KL}(f|g) = 0$ if and only if $f \equiv g$.

Proof of Lemma 12.4 The proof uses Jensen's Inequality[1] namely, that for any convex function ϕ, $\mathbb{E}[\phi(X)] \geq \phi(\mathbb{E}[X])$, with equality if and only if either $\phi(x) = ax + b$ or $\mathbb{P}(X = y) = 1$ for some point y. Note that $f(x) \geq 0$ for all $x \in \mathcal{X}$ and that $\sum_{x \in \mathcal{X}} f(x) = \sum_{x \in \mathcal{X}} g(x) = 1$. Using this, together with the fact that $-\log$ is convex, yields

$$D_{KL}(f|g) = -\sum_{x \in \mathcal{X}} f(x) \log\left(\frac{g(x)}{f(x)}\right) \geq -\log\left(\sum_{x \in \mathcal{X}} f(x) \frac{g(x)}{f(x)}\right) = -\log 1 = 0$$

with equality if and only if $f = g$. □

The likelihood function may be expressed in terms of the Shannon Entropy and the Kullback Leibler divergence as follows:

Theorem 12.5. *Let*

$$L_n(\underline{\theta}|\mathbf{x}) = \mathbb{P}_{\mathbf{X}|\Theta}(\mathbf{x}|\theta) = \prod_{i=1}^{k} \theta_i^{n_i}$$

denote the likelihood function for the parameter vector $\underline{\theta} = (\theta_1, \ldots, \theta_k)$, where the n-vector $\mathbf{x}$ denotes the outcomes of n independent trials, each taking values in $\mathcal{X} = (x^{(1)}, \ldots, x^{(k)})$ and n_i denotes the number of times $x^{(i)}$ appears in the list $\mathbf{x}$. Let

$$\widehat{\underline{\theta}} = \left(\frac{n_1}{n}, \ldots, \frac{n_k}{n}\right).$$

Then

$$-\frac{1}{n} \log L_n(\underline{\theta}|\mathbf{x}) = H(\widehat{\underline{\theta}}) + D_{KL}(\widehat{\underline{\theta}}|\underline{\theta}) \tag{12.2}$$

[1] J.L. Jensen (1859 - 1925) published this in *Acta Mathematica* volume in the year 1906.

where H denotes the Shannon entropy, so that

$$H(\widehat{\underline{\theta}}) = -\sum_{i=1}^{k} \widehat{\theta}_i \log \widehat{\theta}_i = -\frac{1}{n} \log \mathbb{P}_{\mathbf{X}|\Theta}(\mathbf{x}|\widehat{\underline{\theta}})$$

and D_{KL} the Kullback Leibler divergence, so that

$$D_{KL}(\widehat{\underline{\theta}}|\underline{\theta}) = \sum_{i=1}^{k} \widehat{\theta}_i \log \frac{\widehat{\theta}_i}{\theta_i}.$$

Proof of Theorem 12.5 Since $\mathbb{P}_{\mathbf{X}}\left(\mathbf{x}^{(n)} \mid \theta\right) = \prod_{i=1}^{k} \theta_i^{n_i}$ it follows directly that

$$-\frac{1}{n} \log \mathbb{P}_{\mathbf{X}}\left(\mathbf{x} \mid \widehat{\underline{\theta}}\right) = -\frac{1}{n} \sum_{i=1}^{k} n_i \log \widehat{\theta}_i = -\sum_{i=1}^{k} \widehat{\theta}_i \log \widehat{\theta}_i = H(\widehat{\underline{\theta}}). \tag{12.3}$$

This is the *Shannon entropy* for the empirical distribution, given by Definition 12.2.

For arbitrary $\underline{\theta} \in \Theta$, it therefore follows directly that

$$\begin{aligned} -\frac{1}{n} \log L_n(\underline{\theta}) &:= -\frac{1}{n} \log \mathbb{P}_{\mathbf{X}}\left(\mathbf{x} \mid \underline{\theta}\right) = -\frac{1}{n} \log \prod_{i=1}^{k} \theta_i^{n_i} = -\frac{1}{n} \sum_{i=1}^{k} n_i \log \theta_i = -\sum_{i=1}^{k} \widehat{\theta}_i \log \theta_i \\ &= -\sum_{i=1}^{k} \widehat{\theta}_i \log \widehat{\theta}_i - \sum_{i=1}^{k} \widehat{\theta}_i \log \frac{\widehat{\theta}_i}{\theta_i} \\ &= H\left(\widehat{\underline{\theta}}\right) + D_{KL}\left(\widehat{\underline{\theta}}|\underline{\theta}\right) \end{aligned}$$

and Theorem 12.5 is proved. □

Since the Kullback Leibler distance is non-negative, it now follows directly that the maximum likelihood estimate $\widehat{\underline{\theta}}_{MLE}$ of $\underline{\theta}$ is given by

$$\widehat{\underline{\theta}}_{MLE} = \left(\frac{n_1}{n}, \ldots, \frac{n_k}{n}\right).$$

□

Recall that, for parameter estimation in statistics, the same notation $\widehat{\theta}$ is used for an estimate, estimator, and estimating function for a parameter θ.

It is important to have, at least approximately, the distribution of the estimator. Let

$$Y_j = \sum_{k=1}^{n} \mathbf{1}_{x^{(j)}}(X_k),$$

the number of times that $x^{(j)}$ appears in $\mathbf{X}$. Then $\widehat{\theta}_j = \frac{1}{n} Y_j$,

$$\mathbb{E}[Y_j] = n\theta_j$$

$$\mathbb{E}[Y_j^2] = \sum_{k_1=1}^{n} \sum_{k_2=1}^{n} \mathbb{E}[\mathbf{1}_{x^{(j)}}(X_{k_1}) \mathbf{1}_{x^{(j)}}(X_{k_2})] = n\theta_j + n(n-1)\theta_j^2$$

giving

$$\mathbf{V}(Y_j) = n\theta_j(1-\theta_j)$$

and, for $i \neq j$

$$\mathbb{E}[Y_i Y_j] = n(n-1)\theta_i\theta_j$$

so that

$$\mathrm{Cov}(Y_i, Y_j) = -n\theta_i\theta_j.$$

Since $\underline{\theta} = \frac{1}{n}\underline{Y}$, it follows that

$$\mathbb{E}[\widehat{\underline{\theta}}_{ML}] = \underline{\theta}, \qquad \mathrm{Cov}(\sqrt{n}(\widehat{\underline{\theta}}_{ML} - \underline{\theta})) = C$$

where $C_{jj} = \theta_j(1-\theta_j)$ and $C_{ij} = -\theta_i\theta_j$ for $i \neq j$. Furthermore, the following central limit is standard from multivariate analysis:

$$\sqrt{n}(\widehat{\underline{\theta}}_{ML} - \underline{\theta}) \overset{n\to+\infty}{\longrightarrow} N(\underline{0}, C).$$

12.4.2 MLE for a Probability Factorised along a DAG

Let $\underline{\theta}$ denote the entire collection of parameters (θ_{jil}) arranged as a vector and let Θ denote the parameter space. Let

$$\mathbf{X} = \begin{pmatrix} \underline{X}_{(1)} \\ \vdots \\ \underline{X}_{(n)} \end{pmatrix}$$

denote the random matrix, where each row represents an independent copy of $\underline{X}$ and let

$$\mathbf{x} = \begin{pmatrix} \underline{x}_{(1)} \\ \vdots \\ \underline{x}_{(n)} \end{pmatrix},$$

the $n \times d$ data matrix, denote an instantiation of $\mathbf{X}$.

Firstly, the probability function $\mathbb{P}_{\underline{X}}$ may be written as

$$\mathbb{P}_{\underline{X}} = \prod_{j=1}^{d} \mathbb{P}_{X_j|\mathrm{Pa}_j} \tag{12.4}$$

Setting

$$n_k(x_j^{(i)}, \pi_j^{(l)}) = \begin{cases} 1 & (\underline{x}_k)_j = x_j^{(i)},\ \pi_j(\underline{x}_{(k)}) = \pi_j^l \\ 0 & \text{otherwise} \end{cases}$$

and

$$n(x_j^{(i)}, \pi_j^{(l)}) = \sum_{k=1}^{n} n_k(x_j^{(i)}, \pi_j^{(l)}),$$

The crucial point is that, using Equation 12.4, the likelihood function has product form:

$$L(\underline{\theta}|\mathbf{x}) = \mathbb{P}_{\mathbf{X}|\Theta}(\mathbf{x}|\underline{\theta}) = \prod_{k=1}^{n} \mathbb{P}_{\underline{X}|\Theta}(\underline{x}_k|\underline{\theta}) = \prod_{j=1}^{d}\prod_{l=1}^{q_j}\prod_{i=1}^{k_j}\prod_{k=1}^{n} \theta_{jil}^{n_k(x_j^{(i)},\pi_j^{(l)})} = \prod_{j=1}^{d}\prod_{l=1}^{q_j}\left(\prod_{i=1}^{k_j} \theta_{jil}^{n(x_j^{(i)},\pi_j^{(l)})}\right). \tag{12.5}$$

This is the product of likelihood functions; each (j,l) represents the likelihood function for the parameters $(\theta_{jil})_{i=1}^{k_j}$ based on $n(\pi_j^{(l)})$ independent observations where

$$n(\pi_j^{(l)}) := \sum_{i=1}^{k_j} n(x_j^{(i)}, \pi_j^{(l)}).$$

It follows that the maximum likelihood estimate is

$$\widehat{\theta}_{ML;jil} = \frac{n(x_j^{(i)}, \pi_j^{(l)})}{n(\pi_j^{(l)})} = \frac{\text{frequency of } (x_j^{(i)}, \pi_j^{(l)}) \text{ configuration}}{\text{total frequency of } \pi_j^{(l)} \text{ configuration}}.$$

Furthermore, the *estimator* satisfies; $n(\pi_j^{(l)})\widehat{\underline{\theta}}_{j.l} \sim \text{Mult}(n; \theta_{j1l}, \ldots, \theta_{jk_jl})$, where the family of random vectors $((\theta_{j,.,l})_{i=1}^{k_j})$ are independent and for each(j,l)

$$\mathbb{E}[\widehat{\theta}_{ML;jil}] = \theta_{jil},$$

$$\text{Cov}(\widehat{\theta}_{ML;ji_1l}, \widehat{\theta}_{ML;ji_2l}) = C_{i_1i_2}^{(jl)} = \begin{cases} \frac{1}{n(\pi_j^l)}\theta_{jil}(1-\theta_{jil}) & i_1 = i_2 = i \\ -\frac{1}{n(\pi_j^l)}\theta_{ji_1l}\theta_{ji_2l} & i_1 \neq i_2, \end{cases}$$

and, asymptotically for each (j,l),

$$\sqrt{n(\pi_j^{(l)})}(\widehat{\underline{\theta}}_{ML;j.l} - \theta_{j.l}) \overset{n(\pi_j^{(l)})\to+\infty}{\longrightarrow} N(\underline{0}, C^{(jl)}).$$

12.5 The Bayesian Approach

The classical approach to statistics starts by approximating a situation by constructing a probability model with unknown parameters. Data is then obtained and the parameters estimated from the data. The estimates are then plugged into the model and the estimated probability is then used to make predictions. These predictions are considered to be approximate, where the approximation is from two sources: 1) A probability model does not give a full description of the problem; 2) the true parameter values are unknown and approximate values are used.

The Bayesian approach deals with the uncertainty in the parameter value by modelling the uncertainty as a probability distribution, so that the parameter may be regarded as the outcome of a random variable with this distribution. A probability model to approximate the situation is established that has some unknown parameters and the uncertainty in the parameter values is modelled by placing a probability distribution π_Θ over the parameter space Θ. The distribution π_Θ, which models the uncertainty in the parameters before any data is gathered, is known as the *prior* distribution. When data $\mathbf{x}$, and instantiation of $\mathbf{X}$ is gathered, this assessment is updated to:

$$\pi_{\Theta|\mathbf{X}}(\underline{\theta}|\mathbf{x}) = \frac{\mathbb{P}_{\mathbf{X}|\Theta}(\mathbf{x}|\underline{\theta})\pi_{\Theta}(\underline{\theta})}{\mathbb{P}_{\mathbf{X}}(\mathbf{x})} = \frac{\mathbb{P}_{\mathbf{X}|\Theta}(\mathbf{x}|\underline{\theta})\pi_{\Theta}(\underline{\theta})}{\int_{\Theta}\mathbb{P}_{\mathbf{X}|\mathbf{\Theta}}(\mathbf{x}|\underline{\theta})\pi_{\Theta}(\underline{\theta})d\underline{\theta}}.$$

This is then used to compute the predictive probability; of a random variable Y that is independent of $\mathbf{X}$ once the parameter value is known; $Y \perp \mathbf{X}|\Theta$. This is computed as

$$\mathbb{P}_{Y|\mathbf{X}}(y|\mathbf{x}) = \int_{\Theta}\mathbb{P}_{Y|\mathbf{X},\Theta}(y|\mathbf{x},\theta)\pi_{\Theta|\mathbf{x}}(\theta|\mathbf{x})d\theta = \int_{\Theta}\mathbb{P}_{Y|\Theta}(y|\theta)\pi_{\Theta|\mathbf{X}}(\theta|\mathbf{x})d\theta.$$

To keep computations within a reasonable framework, it is important that the prior distribution is from a *conjugate family.*

Definition 12.6 (Conjugate Prior)**.** *A prior distribution from a family that is closed under sampling is known as a* conjugate prior

12.5.1 Independent Bernoulli trials and the Beta distribution

The following discussion motivates and explains the Bayesian approach, illustrating it by a basic example.

If a thumb-tack is thrown in the air, it will come to rest either on its point (0) or on its head (1). Suppose the thumb-tack is flipped n times in identical conditions. Let $\mathbf{x}_{(n)}$ denote the sequence of outcomes

$$\underline{x}_{(n)} = (x_1, \ldots, x_n)^t.$$

Each trial is a *Bernoulli trial* with probability θ of success (obtaining a 1). This is denoted by

$$X_i \sim Be(\theta), \qquad i = 1, \ldots, n.$$

Using the Bayesian approach, the parameter θ is be regarded as the outcome of a random variable, which is denoted by Θ. The outcomes are *conditionally independent, given* θ. This is denoted by

$$X_i \perp X_j|\Theta, \quad i \neq j.$$

When $\Theta = \theta$ is given, the random variables $X_1, \ldots, X_n$ are independent. Let $\mathbf{X}_{(n)} = (X_1, \ldots, X_n)^t$ so that

$$\mathbb{P}_{\mathbf{X}_{(n)}|\Theta}(\mathbf{x}_{(n)}|\theta) = \prod_{l=1}^{n}\theta^{x_l}(1-\theta)^{1-x_l} = \theta^k(1-\theta)^{n-k}$$

where $k = \sum_{l=1}^{n} x_l$.

The problem is use $\mathbf{x}_{(n)}$ to make an assessment of θ and then use this to assess the probability function for a further outcome X_{n+1}. The Bayesian approach is, starting with a prior density $\pi_{\Theta}(.)$ over the parameter space $\widetilde{\Theta} = [0,1]$, to find the posterior density $\pi_{\Theta|\mathbf{X}_{(n)}}(.|\mathbf{x}_{(n)})$.

$$\pi_{\Theta|\mathbf{X}_{(n)}}(\theta|\mathbf{x}_{(n)}) = \frac{\mathbb{P}_{\mathbf{X}_{(n)}|\Theta}(\mathbf{x}_{(n)}|\theta)\pi_{\Theta}(\theta)}{\mathbb{P}_{\mathbf{X}_{(n)}}(\mathbf{x}_{(n)})} = \frac{\mathbb{P}_{\mathbf{X}_{(n)}|\mathbf{\Theta}}(\mathbf{x}_{(n)}|\theta)\pi_{\Theta}(\theta)}{\int\mathbb{P}_{\mathbf{X}_{(n)}|\mathbf{\Theta}}(\mathbf{x}_{(n)}|\phi)\pi_{\Theta}(\phi)d\phi}.$$

Let π_Θ be the uniform density on $[0,1]$. This represents no initial preference concerning θ; all values are equally plausible[2]. The choice of prior may seem arbitrary, but following the computations below, it should be clear that, from a large class of priors, the final answer does not depend much on the choice of prior if the thumb-tack is thrown a large number of times.

With the uniform prior,

$$\int_0^1 \mathbb{P}_{\mathbf{X}_{(n)}|\Theta}(\mathbf{x}_{(n)}|\theta)\pi_\Theta(\theta)d\theta = \int_0^1 \theta^k(1-\theta)^{n-k}d\theta = \frac{k!(n-k)!}{(n+1)!}. \tag{12.6}$$

The *posterior distribution* is a *Beta density*

$$\pi_{\Theta|\mathbf{X}_{(n)}}(\theta|\mathbf{x}^{(n)}) = \begin{cases} \frac{(n+1)!}{k!(n-k)!}\theta^k(1-\theta)^{n-k} & 0 \le \theta \le 1 \\ 0 & \text{otherwise.} \end{cases} \tag{12.7}$$

The Beta distribution is not restricted to integer values; the *Euler gamma function* is necessary to extend the definition to positive integers.

Definition 12.7 (Euler Gamma Function)**.** *The* Euler Gamma Function $\Gamma(\alpha) : (0,+\infty) \to (0,+\infty)$ *is defined as*

$$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1}e^{-x}dx. \tag{12.8}$$

The Euler Gamma function satisfies the following properties.

Lemma 12.8. *For all $\alpha > 0$, $\Gamma(\alpha+1) = \alpha\Gamma(\alpha)$. If n is an integer satisfying $n \ge 1$, then*

$$\Gamma(n) = (n-1)!$$

Proof Note that $\Gamma(1) = \int_0^\infty e^{-x}dx = 1$. For all $\alpha > 0$, integration by parts gives

$$\Gamma(\alpha+1) = \int_0^\infty x^\alpha e^{-x}dx = \alpha\Gamma(\alpha). \tag{12.9}$$

The result follows directly. □

For Bernoulli sampling, given a sequence $\underline{x} = (x_1,\ldots,x_n)$ containing k 1's and $n-k$ 0's, the likelihood function is $L(\theta) = \theta^k(1-\theta)^{n-k}$. Since

$$\pi(\theta|\underline{x}) \propto L(\theta|\underline{x})\pi(\theta)$$

[2]All statistical methods contain some ad-hoc element and in Bayesian statistics, this is contained in the choice of prior distribution. The results obtained from any statistical analysis are only reliable if there is sufficient data so that any inference will be robust under a rather general choice of prior.

There are well known difficulties with the statement that a uniform prior represents no preference concerning the value of θ. If the prior density for Θ is uniform, then the prior density of Θ^2 will *not* be uniform, so 'no preference' for values of Θ indicates that there is a *distinct* preference among possible initial values of Θ^2. If $\pi_1(x) = 1$ for $0 < x < 1$ is the density function for Θ and π_2 is the density function for Θ^2, then $\pi_2(x) = \frac{1}{2x^{1/2}}$ for $0 < x < 1$.

where π is the prior, it therefore follows that the prior should have the form

$$\pi(\theta) \propto \theta^a (1-\theta)^b$$

for some values a and b to guarantee that both prior and posterior come from the same conjugate family. In this case, the family of distributions is the family of *Beta distributions*, defined as follows:

Definition 12.9 (Beta Density)**.** *The beta density Beta*(α, β) *with parameters* $\alpha > 0$ *and* $\beta > 0$ *is defined as the function*

$$\psi(t) = \begin{cases} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} t^{\alpha-1}(1-t)^{\beta-1} & t \in [0,1] \\ 0 & t \notin [0,1] \end{cases} \tag{12.10}$$

The Beta density is a probability density function for all real $\alpha > 0$ and $\beta > 0$. It follows that, for Binomial sampling, updating may be carried out very easily for any prior distribution within the Beta family. Suppose the prior distribution π_0 is the $B(\alpha, \beta)$ density function, n trials are observed, with k taking the value 1 and $n-k$ taking the value 0. Then

$$\begin{aligned} \pi_{\Theta|\mathbf{X}_{(n)}}(\theta|\mathbf{x}_{(n)}) &= \frac{\mathbb{P}_{\mathbf{X}_{(n)}|\Theta}(\mathbf{x}_{(n)}|\theta)\pi_\Theta(\theta)}{\mathbb{P}_{\mathbf{X}_{(n)}}(\mathbf{x}_{(n)})} \\ &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)\mathbb{P}_{\mathbf{X}_{(n)}}(\mathbf{x}_{(n)})} \theta^{\alpha+k-1}(1-\theta)^{\beta+n-k-1} = c\theta^{\alpha+k-1}(1-\theta)^{\beta+n-k-1}. \end{aligned}$$

Since $\int_0^1 \pi_{\Theta|\mathbf{X}_{(n)}}(\theta|\mathbf{x}_{(n)})d\theta = 1$, therefore:

$$\pi_{\Theta|\mathbf{X}_{(n)}}(\theta|\mathbf{x}_{(n)}) = \begin{cases} \frac{\Gamma(\alpha+\beta+n)}{\Gamma(\alpha+k)\Gamma(\beta+n-k)} \theta^{\alpha+k-1}(1-\theta)^{\beta+n-k-1} & \theta \in (0,1) \\ 0 & \theta \notin (0,1). \end{cases}$$

so that $\pi_{\Theta|\mathbf{X}_{(n)}}(\theta|\mathbf{x}_{(n)})$ is a $B(\alpha+k, \beta+n-k)$ density. □

Definition 12.10 (Maximum Posterior Estimate)**.** *The* maximum posterior estimate*,* $\widehat{\theta}_{MAP}$*, is the value of* θ *which maximises the posterior density* $\pi_{\Theta|\mathbf{X}_{(n)}}(\theta|\mathbf{x}_{(n)})$*.*

When the posterior density is $B(k+\alpha, n-k+\beta)$, an easy computation gives

$$\widehat{\theta}_{MAP} = \frac{k+\alpha-1}{n+\alpha+\beta-2}.$$

Note that when the prior density is uniform, as in the case above, the MAP and MLE are exactly the same. The parameter, of course, is not an end in itself. The parameter ought to be regarded as a means to computing the predictive probability. The posterior is used to compute this.

The Predictive Probability for the Next Toss Suppose that $\pi_{\Theta|\mathbf{X}_n}(\theta|\mathbf{x}_{(n)})$ has a $B(\alpha+k, \beta+n-k)$ distribution.The *predictive probability for the next toss*, for $a = 0$ or 1, is given by

$$\mathbb{P}_{X_{n+1}|\mathbf{X}_{(n)}}(a|\mathbf{x}_{(n)}) = \int_0^1 \mathbb{P}_{X_{n+1}}(a|\theta)\pi_{\Theta|\mathbf{X}_{(n)}}(\theta|\mathbf{x}_{(n)})d\theta.$$

Since $\mathbb{P}_{X_{n+1}|\Theta}(1|\theta) = \theta$, it follows (using Equation (12.9)) that

$$\begin{aligned}\mathbb{P}_{X_{n+1}|\mathbf{X}_{(n)}}(1|\mathbf{x}_{(n)}) &= \frac{\Gamma(\alpha+\beta+n)}{\Gamma(\alpha+k)\Gamma(\beta+n-k)}\int_0^1 \theta^{(\alpha+k)}(1-\theta)^{\beta+n-k-1}d\theta \\ &= \frac{\Gamma(\alpha+\beta+n)}{\Gamma(\alpha+k)\Gamma(\beta+n-k)}\frac{\Gamma(\alpha+k+1)\Gamma(\beta+n-k)}{\Gamma(\alpha+\beta+n+1)} \\ &= \frac{\alpha+k}{\alpha+\beta+n}.\end{aligned}$$

In particular, note that the *uniform* prior, $\pi_0(\theta) = 1$ for $\theta \in (0,1)$, is the $B(1,1)$ density function, so that for binomial sampling with a uniform prior, the predictive probability is

$$\begin{aligned}\mathbb{P}_{X_{n+1}|\mathbf{X}_{(n)}}(1|\mathbf{x}_{(n)}) &= \frac{k+1}{n+2}; \\ \mathbb{P}_{X_{n+1}|\mathbf{X}_{(n)}}(0|\mathbf{x}_{(n)}) &= \frac{n+1-k}{n+2}.\end{aligned} \tag{12.11}$$

This distribution, or more precisely $\frac{k+1}{n+2}$, is known as the Laplace rule of succession.

12.5.2 Multinomial Sampling and the Dirichlet Integral

Consider the case of multinomial sampling, where there are k possible outcomes in the state space $\mathcal{X} = (x^{(1)}, \ldots, x^{(k)})$ and $\mathbb{P}_X(x^{(j)}) = \theta_j$, $j = 1, \ldots, k$ so that $\theta_1 + \ldots + \theta_k = 1$. Consider n independent trials, $\underline{X} = (X_1, \ldots, X_n)^t$ with outcomes $\underline{x} = (x_1, \ldots, x_n)$.
The likelihood function for $\underline{\theta}$, given $\underline{x}$ is:

$$L(\underline{\theta}|\underline{x}) = \theta_1^{n_1} \ldots \theta_k^{n_k}$$

where $n_j = \sum_{i=1}^n \mathbf{1}_{x^{(j)}}(x_i)$; i.e. the number of times outcome $x^{(j)}$ appears in the sequence $\underline{x}$, for $j = 1, \ldots, k$. It follows that, to ensure that the prior and posterior are within the same conjugate family, the prior has the form:

$$\pi(\theta) \propto \theta_1^{\alpha_1} \ldots \theta_k^{\alpha_k}.$$

It follows that the only possible family of distributions to use is the *Dirichlet* family, defined as follows.

Definition 12.11 (Dirichlet Density). *The Dirichlet density $Dir(a_1, \ldots, a_k)$ is the function*

$$\pi(\theta_1, \ldots, \theta_k) = \begin{cases} \frac{\Gamma(a_1+\ldots+a_k)}{\prod_{j=1}^k \Gamma(a_k)}(\prod_{j=1}^k \theta_j^{a_j-1}) & \theta_j \geq 0, \sum_{j=1}^k \theta_j = 1, \\ 0 & \textit{otherwise}, \end{cases} \tag{12.12}$$

where Γ denotes the Euler Gamma Function, given in Definition 12.7. The parameters $(a_1, \ldots, a_k)$ are all strictly positive and are known as *hyper parameters*.

This density, and integration with respect to this density function, are to be understood in the following sense. Since $\theta_k = 1 - \sum_{j=1}^{k-1} \theta_j$, it follows that π may be written as $\pi(\theta_1, \ldots, \theta_k) = \widetilde{\pi}(\theta_1, \ldots, \theta_{k-1})$, where

$$\widetilde{\pi}(\theta_1, \ldots, \theta_{k-1}) = \begin{cases} \frac{\Gamma(a_1+\ldots+a_k)}{\prod_{j=1}^{k}\Gamma(a_k)} \left(\prod_{j=1}^{k-1} \theta_j^{a_j-1}\right)\left(1 - \sum_{j=1}^{k-1}\theta_j\right)^{a_k-1} & \theta_j \geq 0, \sum_{j=1}^{k-1}\theta_j \leq 1, \\ 0 & \text{otherwise.} \end{cases} \tag{12.13}$$

Clearly, when $k = 2$, this reduces to the *Beta density*. The Dirichlet density is a probability density function.

Properties of the Dirichlet Density The family of Dirichlet densities $\mathrm{Dir}(\alpha_1, \ldots, \alpha_k) : \alpha_1 > 0, \ldots, \alpha_k > 0$ is *closed under sampling*: Consider a prior distribution $\pi_\Theta \sim \mathrm{Dir}(\alpha_1, \ldots, \alpha_k)$ and suppose that observations of n independent trials are made: $\underline{x} \coloneqq (x_1, \ldots, x_n)$ where $n_j = \sum_{i=1}^{n} \mathbf{1}_{x^{(j)}}(x_i)$, i.e. the number of appearances of $x^{(j)}$ in the sequence, for $j = 1, \ldots, n$. Let $\pi_{\Theta|\underline{X}}$ denote the posterior distribution. Then

$$\pi_{\Theta|\underline{X}}(\theta_1, \ldots, \theta_k|\underline{x}) \sim \mathrm{Dir}(\alpha_1 + n_1, \ldots, \alpha_k + n_k).$$

The Dirichlet density is usually written exclusively as a function of k variables, $\pi_\Theta(\theta_1, \ldots, \theta_k)$, where there are $k-1$ independent variables and $\theta_k = 1 - \sum_{j=1}^{k-1}\theta_j$.

Mean Posterior Estimate The *mean posterior estimate* is the *expected value* of the posterior distribution. Here,

$$\widehat{\theta}_{i,MEP} = \int \theta_i \pi(\theta_1, \ldots, \theta_k|\mathbf{x}, \alpha) d\theta_1 \ldots d\theta_k = \frac{n_i + \alpha_i}{\sum_{j=1}^{k} n_j + \sum_{j=1}^{k}\alpha_j}.$$

This computation is left as an exercise.

12.5.3 Distribution for Conditional Probabilies of a Bayesian network

The notation $\underline{\theta}_{j.l} = \left(\theta_{j1l}, \ldots, \theta_{jk_jl}\right)$ is used to denote the probability distribution over the states of X_j, given that $\pi_j^{(l)}$ is the parent configuration. The prior distribution over $\underline{\theta}_{j.l}$ is taken to be

$$\pi_{\Theta^{jl}} \sim Dir(\alpha_{j1l}, \ldots, \alpha_{jk_jl}).$$

The prior distribution over the entire collection of parameters Θ is taken to be $\pi_\Theta = \prod_{jl} \pi_{\Theta^{jl}}$. That is, the distributions over $(\underline{\theta}_{j.l})_{(j,l)}$ are mutually independent for different (j,l). Suppose an $n \times d$ data matrix is obtained, with n complete instantiations. Let $n(x_j^{(i)}, \pi_j^{(l)})$ denote the number of times that the configuration $(x_j^{(i)}, \pi_j^{(l)})$ appears in $\mathbf{x}$. It follows that

$$\pi_{\Theta|\mathbf{X}}(\underline{\theta}|\mathbf{x}) = \pi_{\Theta}(\underline{\theta})\frac{\mathbb{P}_{\mathbf{X}|\Theta}(\mathbf{x}|\underline{\theta})}{\mathbb{P}_{\mathbf{X}}(\mathbf{x})}.$$

Recall the expression for $\mathbb{P}_{\mathbf{X}|\Theta}(\mathbf{x}|\underline{\theta})$ found in Equation (12.5). It follows that

$$\pi_{\Theta|\mathbf{X}}(\underline{\theta}|\mathbf{x}) = \frac{1}{\mathbb{P}_{\mathbf{X}}(\mathbf{x})}\prod_{j=1}^{d}\prod_{l=1}^{q_j}\left(\pi_{\Theta^{jl}}(\underline{\theta}_{jl})\prod_{i=1}^{k_j}\theta_{jil}^{n(x_j^{(i)},\pi_j^{(l)})}\right).$$

From this, it follows directly that $\pi_{\Theta|\mathbf{X}} = \prod_{jl}\pi_{\Theta^{jl}|\mathbf{X}}$, where

$$\pi_{\Theta^{jl}|\mathbf{X}}(.|\mathbf{x}) \sim Dir(n(x_j^{(1)},\pi_j^{(1)}) + \alpha_{j1l},\ldots,n(x_j^{(k_j)},\pi_j^{(1)}) + \alpha_{jk_jl}).$$

The posterior distribution of $\theta_{j.l}$ depends only on counts of family configurations at node j and not on configurations at any other node.

Predictive Distribution The predictive distribution of a new case $\underline{x}_{(n+1)}$ may be computed using the posterior density; with an $n \times d$ data matrix $\mathbf{x}$ of n complete instantiations, θ_{jil}, defined in Equation (12.1), will be estimated by:

$$\widetilde{\theta}_{jil} = \mathbb{P}_{X_{n+1,j}|\mathrm{Pa}_j,\mathbf{X}}(x_j^{(i)}|\pi_j^{(l)},\mathbf{x}). \tag{12.14}$$

This is the predictive conditional probability that variable $X_{n+1,j}$ attains value $x_j^{(i)}$, given the parent configuration $\pi_j^{(l)}$ and the cases stored in $\mathbf{x}$. Let $\mathbf{X}$ denote the $n \times d$ matrix where each row $\underline{X}_{k.}$ is an independent copy of $\underline{X} = (X_1,\ldots,X_d)$. Recall that

$$\pi_{\Theta|\mathbf{X}} = \prod_{j=1}^{d}\prod_{l=1}^{q_j}\pi_{\Theta^{jl}|\mathbf{X}}.$$

Using Bayes rule,

$$\pi_{\Theta|\mathrm{Pa}_{n+1,j},\mathbf{X}} = \pi_{\Theta|\mathbf{X}}\frac{\mathbb{P}_{\mathrm{Pa}_{n+1,j}|\Theta,\mathbf{X}}}{\mathbb{P}_{\mathrm{Pa}_{n+1,j}|\mathbf{X}}} = \pi_{\Theta|\mathbf{X}}\frac{\mathbb{P}_{\mathrm{Pa}_{n+1,j}|\Theta}}{\mathbb{P}_{\mathrm{Pa}_{n+1,j}|\mathbf{X}}}.$$

Note that $\mathbb{P}_{\mathrm{Pa}_{n+1,j}|\Theta}$ is an expression containing sums and products of $(\theta_{ail})_{a=1,\ldots,j-1,i=1,\ldots,k_a,l=1,\ldots,q_a}$. It follows that $\pi_{\Theta|\mathrm{Pa}_{n+1,j},\mathbf{X}}$ may be expressed as a product

$$\pi_{\Theta|\mathrm{Pa}_{n+1,j},\mathbf{X}} = A((\underline{\theta})_{a.l})_{a=1,\ldots,j-1,l=1,\ldots,q_a})\pi_{\Theta_{jl}|\mathbf{X}}(\underline{\theta}_{j.l}|\mathbf{x})\prod_{a=j+1}^{d}\prod_{l=1}^{q_j}\pi_{\Theta_{al}|\mathbf{X}}(\underline{\theta}_{a.l}|\mathbf{x})$$

where A is a probability density over $(\theta_{ail})_{a=1,\ldots,j-1,i=1,\ldots,k_a,l=1,\ldots,q_a}$. Then, by computations as before, with $\widetilde{\theta}_{jil}$ defined by Equation (12.14),

$$
\begin{aligned}
\widetilde{\theta}_{jil} &= \mathbb{P}_{X_{n+1,j}|\mathrm{Pa}_{n+1,j},\mathbf{X}}(x_j^{(i)}|\pi_j^{(l)},\mathbf{x}) \\
&= \int_S \mathbb{P}_{X_{n+1,j}|\mathrm{Pa}_{n+1,j},\Theta,\mathbf{X}}(x_j^{(i)}|\pi_j^{(l)},\underline{\theta},\mathbf{x})\pi_{\Theta|\mathrm{Pa}_{n+1,j},\mathbf{X}}(\underline{\theta}|\pi_j^{(l)},\mathbf{x})d\underline{\theta} \\
&= \int_{S_{jl}} \theta_{jil}\pi_{\Theta_{j.l}|\mathbf{X}}(\theta_{j.l}|\mathbf{x})d\theta_{j.l} \\
&= \int_{S_{jl}} \theta_{jil}\frac{\Gamma(n(\pi_j^{(l)})+\alpha_{j.l})}{\prod_{m=1}^{k_j}\Gamma(n(x_j^{(m)},\pi_j^{(l)})+\alpha_{jml})}\prod_{i=1}^{k_j}\theta_{jil}^{n(x_j^{(i)},\pi_j^{(l)})+\alpha_{jil}}d\underline{\theta} \\
&= \frac{\Gamma(n(\pi_j^{(l)})+\alpha_{j.l})}{\prod_{m=1}^{k_j}\Gamma(n(x_j^{(m)},\pi_j^{(l)})+\alpha_{jml})}\int_{S_{jl}}\prod_{m\neq i}\theta_{jml}^{n(x_j^{(m)},\pi_j^{(l)})}\theta_{jil}^{n(x_j^{(i)},\pi_j^{(l)})+1}d\underline{\theta} \\
&= \frac{\Gamma(n(\pi_j^{(l)})+\alpha_{j.l})}{\prod_{m=1}^{k_j}\Gamma(n(x_j^{(m)},\pi_j^{(l)})+\alpha_{jml})}\times\frac{\Gamma(n(x_j^{(i)}|\pi_j^{(l)})+\alpha_{jil}+1)\prod_{m\neq i}\Gamma(n(x_j^{(i)},\pi_j^{(l)})+\alpha_{jil})}{\Gamma(n(\pi_j^{(l)})+\alpha_{j.l}+1)} \\
&= \frac{n(x_j^{(i)},\pi_j^{(l)})+\alpha_{jil}}{n(\pi_j^{(l)})+\sum_{i=1}^{k_j}\alpha_{jil}},
\end{aligned}
$$

where S_{jl} is defined as

$$
S_{jl} = \left\{(\theta_{jil})_{i=1}^{k_j}|\theta_{jil}\geq 0,\ i=1,\ldots,k_j,\ \sum_{i=1}^{k_j}\theta_{jil}=1\right\}.
$$

Comparing with $\widehat{\theta}_{i,MLE} = \frac{n_i}{n}$, note that

$$
\lim_{n\to+\infty}\frac{\widehat{\theta}_{iMEP}}{\widehat{\theta}_{iMLE}} = 1.
$$

12.6 Updating, Missing Data, Fractional Updating

Updating Suppose the cases $\underline{x}_{(1)},\ldots,\underline{x}_{(n)}$ are complete. Suppose next that $x_j^{(i)}$ and $\pi_j^{(l)}$ are observed in $\underline{x}_{(n+1)}$. Then, by Bayes rule,

$$
\theta_{j.l}|(\underline{x}_{(1)},\ldots,\underline{x}_{(n)},(x_{n+1,j}^{(i)},\pi_{n+1,j}^{(l)})) \sim Dir(n^*(x_j^{(1)}|\pi_j^{(l)})+\alpha_{j1l},\ldots,n^*(x_j^{(k_j)}|\pi_j^{(l)})+\alpha_{jk_jl}),
$$

where

$$
n^*(x_j^{(r)},\pi_j^{(l)}) = \begin{cases} n(x_j^{(r)},\pi_j^{(l)}) & r\neq i \\ n^*(x_j^{(i)},\pi_j^{(l)}) = n(x_j^{(i)},\pi_j^{(l)})+1 & r=i. \end{cases}
$$

The *virtual* sample size for $\pi_j^{(l)}$ is updated as

$$
s^* = n(\pi_j^{(l)})+1+\sum_{i=1}^{k_j}\alpha_{jil}.
$$

A Missing Instantiation Suppose the instantiation at node j is missing in the new case; the parent configuration $\pi_j^{(l)}$ is present. Let

$$\mathbf{X} = \begin{pmatrix} \underline{x}_{(1)} \\ \vdots \\ \underline{x}_{(n)} \end{pmatrix}$$

denote the complete instantiations and let $\underline{x}_{(n+1)}$ denote instantiation $n+1$ where the value $x_{n+1,j}$ is missing. The distribution of the random vector $\underline{\theta}_{j.l}|\mathbf{x}, \underline{x}_{n+1}$ is expressed as the *mixture of distributions*

$$\sum_{i=1}^{k_j} w_i \mathrm{Dir}(n(x_j^{(i)}|\pi_j^{(l)}) + \alpha_{j1l}, \ldots, n(x_j^{(i)}, \pi_j^{(l)}) + 1 + \alpha_{jil}, \ldots, n(x_j^{(k_j)}, \pi_j^{(l)}) + \alpha_{jk_jl}),$$

where

$$w_i = \mathbb{P}_{X_{j,n+1}|\mathrm{Pa}_{j,n+1},\mathbf{X}}(x_j^{(i)}|\pi_j^{(l)}, \mathbf{x}) = \int \theta_{jil} \pi_{\Theta|\mathbf{X}}(\underline{\theta}|\mathbf{x}) d\underline{\theta}.$$

Updating: Parent Configuration and the state at node j are missing Consider a new case $\underline{x}_{(n+1)}$ where both the state and the parent configuration of node j are missing. Then the distribution of $\underline{\theta}_{j.l}|\mathbf{x}, \underline{x}_{n+1}$ is given as the mixture of distributions

$$\sum_{i=1}^{k_j} v_i \mathrm{Dir}(n(x_j^{(1)}, \pi_j^{(l)}) + \alpha_{j1l}, \ldots, n(x_j^{(i)}|\pi_j^{(l)}) + 1 + \alpha_{jil}, \ldots, n(x_j^{(k_j)}, \pi_j^{(l)}) + \alpha_{jk_jl})$$
$$+\mathrm{Dir}(n(x_j^{(1)}, \pi_j^{(l)}) + \alpha_{j1l}, \ldots, n(x_j^{(k_j)}, \pi_j^{(l)}) + \alpha_{jk_jl}) v^*,$$

where

$$v_i = \mathbb{P}_{X_j,\mathrm{Pa}_j|\mathbf{X},\underline{X}_{n+1}}(x_j^{(i)}, \pi_j^{(l)}|\mathbf{x}, \underline{x}_{n+1}), \quad i = 1, \ldots, k_j$$

and

$$v^* = 1 - \mathbb{P}_{\mathrm{Pa}_j|\mathbf{X},\underline{X}_{n+1}}(\pi_j^{(l)}|\mathbf{x}, \underline{x}_{n+1}).$$

Fractional Updating The preceding shows that adding new cases with missing values results in dealing with increasingly messy mixtures, with increasing numbers of components. The standard way to deal with this is to use a Dirichlet integral that is an approximation of the true update, taking the updated distribution as:

$$\underline{\theta}_{j.l} \sim \mathrm{Dir}\left(n^*(x_j^{(1)}, \pi_j^{(l)}) + \alpha_{j1l}, \ldots, n^*(x_j^{(k_j)}, \pi_j^{(l)}) + \alpha_{jk_jl}\right)$$

where

$$n^*(x_j^{(i)}, \pi_j^{(l)}) = n(x_j^{(i)}, \pi_j^{(l)}) + \mathbb{P}_{X_j,\mathrm{Pa}_j|\mathbf{X},\underline{X}_{n+1}}(x_j^{(i)}, \pi_j^{(l)}|\mathbf{x}, \underline{x}_{n+1}), \quad i = 1, \ldots, k_j.$$

This is known as *fractional updating*.

Fading If the parameters change with time, then information learnt a long time ago may not be so useful. A way to make the old cases less relevant is to have the sample size discounted by a fading factor q_F, a positive number less than one.

The *fading update* is as follows: using $n(x_j^{(i)}, \pi_j^{(l)})$ in this section to denote $n(x_j^{(i)}, \pi_j^{(l)}) + \alpha_{jil}$ previously, if $(x_j^{(i)}, \pi_j^{(l)})$ is observed in the next instantiation, then n is updated to n^* where

$$n^*(x_j^{(r)}, \pi_j^{(l)}) = \begin{cases} q_F n(x_j^{(r)}, \pi_j^{(l)}) & r \neq i \\ 1 + q_F n(x_j^{(i)}, \pi_j^{(l)}) & r = i. \end{cases}$$

If $(\pi_j^{(l)}, x_j^{(i)})$ is observed for some $i = 1, \ldots, k_j$, the virtual sample size for parent configuration $\pi_j^{(l)}$ is updated to

$$n^*(\pi_j^{(l)}) = 1 + q_F n(\pi_j^{(l)})$$

and

$$n^*(\pi_j^{(a)}) = q_F n(\pi_j^{(a)}) \qquad a \neq l.$$

Consider the sequence of equations

$$s_n = q_F s_{n-1} + 1, \; s_0 = s.$$

This may be solved as

$$s_n = q_F^n s + \sum_{i=0}^{n} q_F^i = q_F^n s + \frac{1 - q_F^{n+1}}{1 - q_F}.$$

The limiting *effective maximal sample size* is therefore

$$s_* = \frac{1}{1 - q_F}.$$

12.7 Likelihood Function for the Graph Structure

The Bayesian approach to parameter learning leads to a straightforward and elegant approach to computing a likelihood function for the graph structure, given data $\mathbf{x}$. This was introduced by Cooper and Herskovitz (1992) [31]. Let $\mathcal{G} = (V, D)$ denote a directed acyclic graph with edge set D. Let $\mathcal{D}$ denote the collection of all possible edge sets over DAGs with node set V, where V is the collection of random variables. Suppose that, for each D, we have a prior distribution $\pi_{\Theta|\mathcal{D}}(\underline{\theta}|D)$ for the parameters $\underline{\theta}$ when the edge set is D. The likelihood for the graph structure D given data $\mathbf{x}$ is:

$$\mathbb{P}_{\mathbf{X}|\mathcal{D}}(\mathbf{x}|D) = \int \mathbb{P}_{\mathbf{X}|\underline{\Theta},\mathcal{D}}(\mathbf{x}|\underline{\theta}, D)\pi_{\Theta|\mathcal{D}}(\underline{\theta}|D)d\underline{\theta},$$

Since $\pi_{\Theta|\mathcal{D}}(\underline{\theta}|D) = \prod_{j=1}^{d} \prod_{l=1}^{q_j} \pi_{\Theta_{j.l}|\mathcal{D}}(\underline{\theta}_{j.l}|D)$, it follows that:

$$\mathbb{P}_{\mathbf{X}|\mathcal{D}}(\mathbf{x}|D) = \int \prod_{k=1}^{n} \mathbb{P}_{\underline{X}|\underline{\Theta},\mathcal{D}}(\underline{x}_{(k)}|\underline{\theta}, D) \prod_{j=1}^{d} \prod_{l=1}^{q_j} \phi(\underline{\theta}_{j.l}|\underline{\alpha}_{j.l}, D) d\underline{\theta}_{j.l}$$

where $\phi(\underline{\theta}_{j.l}|\underline{\alpha}_{j.l})$ is a compact way of referring to the Dirichlet density $\mathrm{Dir}(\alpha_{j1l}, \ldots, \alpha_{jk_jl})$.

Because $\mathbb{P}_{\underline{X}|\underline{\Theta},\mathcal{D}}(\underline{x}|\underline{\theta}, D)$ has a convenient product form, computing the Dirichlet integral is straightforward and gives

$$L(D|\mathbf{x}) := \mathbb{P}_{\mathbf{X}|\mathcal{D}}(\mathbf{x}|D) = \prod_{j=1}^{d} \prod_{l=1}^{q_j} \frac{\Gamma(\sum_{i=1}^{k_j} \alpha_{jil})}{\Gamma(n(\pi_j^l) + \sum_{i=1}^{k_j} \alpha_{jil})} \prod_{i=1}^{k_j} \frac{\Gamma(n(x_j^i|\pi_j^l) + \alpha_{jil})}{\Gamma(\alpha_{jil})}. \tag{12.15}$$

The computation is left as an exercise; this is the *Cooper Herskovitz likelihood* for the graph structure.

12.8 Bayesian Sufficient Statistics

Let $\mathbf{X}$ be an $n \times d$ random matrix, where each row is an independent copy of a discrete random vector $\underline{X} = (X_1, \ldots, X_d)$ and let $\underline{\theta}$ be a continuous random vector of unknown parameters. Suppose that, for a parameter vector $\underline{\theta}$, $\mathbf{X}$ has conditional probability function $p_{\mathbf{X}}(.|\underline{\theta})$. Suppose that there is a prior density $\pi_{\Theta}(\underline{\theta})$ over the parameter space and suppose that t is a function or a *statistic* of $\mathbf{X}$,

$$t = t(\mathbf{X}).$$

Definition 12.12 (Bayesian Sufficiency)**.** *A statistic T defined as $T = t(\mathbf{X})$ such that for every prior $\pi_{\underline{\Theta}}$ within the space of prior distributions under consideration, there is a function ϕ such that*

$$\pi_{\Theta|\mathbf{X}}(\underline{\theta}|\mathbf{x}) = \frac{p_{\mathbf{X}}(\mathbf{x}|\underline{\theta})\pi_{\Theta}(\underline{\theta})}{p_{\mathbf{X}}(\mathbf{x})} = \phi(\underline{\theta}, t(\mathbf{x})) \tag{12.16}$$

is called a Bayesian sufficient statistic for $\underline{\theta}$.

This definition states that for learning about $\underline{\theta}$ based on $\mathbf{X}$, the statistic T contains all the relevant information, since the posterior distribution depends on $\mathbf{X}$ only through T.

The following result shows that if the conditional distribution of $\mathbf{X}$ given $t(\mathbf{X})$ does not depend on $\underline{\theta}$, then $t(\mathbf{X})$ is Bayesian sufficient for $\underline{\theta}$. If the families of probability measures have finite dimensional parameter spaces, then the converse is also true. If there are an infinite number of parameters, counter examples may be obtained to the converse statement.

Proposition 12.13. *Let t denote a function and let $T = t(\mathbf{X})$. If*

$$\mathbf{X} \perp \underline{\theta}|T, \tag{12.17}$$

where Equation (12.17) *means that*

$$p_{\mathbf{X}|T}(\mathbf{x}|t, \underline{\theta}) = p_{\mathbf{X}|T}(\mathbf{x}|t) \qquad \textit{independent of } \underline{\theta} \in \Theta \tag{12.18}$$

then $T = t(\mathbf{X})$ is a Bayesian sufficient statistic for $\underline{\theta}$.

Proof of Proposition 12.13 As usual, let $T = t(\mathbf{X})$. An application of Bayes rule gives

$$\pi_{\Theta|\mathbf{X},T}(\underline{\theta}|\mathbf{x},t) = \frac{p_{\mathbf{X},T|\Theta}(\mathbf{x},t|\underline{\theta})\pi_\Theta(\underline{\theta})}{p_{\mathbf{X},T}(\mathbf{x},t)} = \frac{p_{\mathbf{X}|T,\Theta}(\mathbf{x}|t,\underline{\theta})p_{T|\Theta}(t|\underline{\theta})\pi_\Theta(\underline{\theta})}{p_{\mathbf{X},T}(\mathbf{x},t)}, \tag{12.19}$$

and Equation (12.19) with an application of (12.18) gives

$$\begin{aligned}\pi_{\Theta|\mathbf{X},T}(\underline{\theta}|\mathbf{x},t) &= \frac{p_{\mathbf{X}|T}(\mathbf{x}|t)p_{T|\Theta}(t|\underline{\theta})\pi_\Theta(\underline{\theta})}{p_{\mathbf{X}|T}(\mathbf{x}|t)p_T(t)} \\ &= \frac{p_{T|\Theta}(t|\underline{\theta})\pi_\Theta(\underline{\theta})}{p_T(t)} = \pi_{\Theta|T}(\underline{\theta}|t).\end{aligned} \tag{12.20}$$

The proposition is proved by setting $\phi(\underline{\theta}, t(\mathbf{x})) = \pi_{\underline{\Theta}|T}(\underline{\theta}|t(\mathbf{x}))$. □

Example 12.14 (Tossing a Thumb-tack)**.**

In the thumb-tack experiment described in section 12.5.1, there is a single parameter, θ. In this paragraph, a Bayesian sufficient statistic is derived for θ, for a suitable class of prior distributions. Let π_Θ denote the prior density function for θ, and let Θ denote the random variable with this density function. In this case, $\mathbf{X}$ is a $n \times 1$ matrix, a column vector, which will be written as $\underline{X} = (X_1, \ldots, X_n)^t$, a sequence of n independent Bernoulli trials, each with probability θ of success (that is $X_j \sim Be(\theta)$, $j = 1, \ldots, n$ and $X_i \perp X_j|\theta$ for $i \neq j$). The sequence of outcomes will be denoted by the vector

$$\underline{x} = (x_1, \ldots, x_n)^t.$$

That is, for each $j = 1, \ldots, n$, $x_j = 1$ or 0. The statistic t is a function of n variables, defined as

$$t(\underline{x}) = \sum_{j=1}^{n} x_j.$$

That is, when t is applied to a sequence of n 0's and 1's, it returns the number of 1's in the sequence. Here, $T = t(\underline{X}) = \sum_{j=1}^{n} X_j$ and therefore T has a binomial distribution with the parameters n and θ, since it is the sum of independent Bernoulli trials. The probability function of T is given by

$$p_{T|\Theta}(k|\theta) = \begin{cases} \binom{n}{k}\theta^k(1-\theta)^{n-k} & k = 0, 1, \ldots, n \\ 0 & \text{other} \quad k. \end{cases}$$

Since t is a function of $\underline{x}$, it follows that

$$p_{\underline{X},T|\Theta}(\underline{x}, k|\theta) = \begin{cases} \theta^k(1-\theta)^{n-k} & k = 0, 1, \ldots, n \\ 0 & \text{other} \quad k \end{cases}$$

from which

$$p_{\underline{X}|T,\Theta}(\underline{x}|k,\theta) = \frac{p_{\underline{X},T|\Theta}(\underline{x}, k|\theta)}{p_{T|\Theta}(k|\theta)} = \frac{1}{\binom{n}{k}}.$$

The right hand side does not depend on θ, from which equation (12.18) holds and hence equation (12.17) follows. Therefore, if $\underline{x} = (x_1, \ldots, x_n)$ are n independent Bernoulli trials, each with parameter θ, the function t such that $t(\underline{x}) = \sum_{l=1}^{n} x_l$ is a Bayesian sufficient statistic for the parameter θ. In the thumb-tack example, given in subsection 12.5.1, the posterior distribution, based on a uniform prior is an explicit function of the data $\underline{x}$ *only* through the function $t(\underline{x})$. □

Now consider a random vector $\underline{X}$ and suppose now that t is a generic sufficient statistic. Since t is a function of $\underline{X}$ (i.e. $t = t(\underline{X})$), it follows, using the rules of conditional probability and equation (12.18), that

$$p_{\underline{X}|\Theta}(\underline{x}|\underline{\theta}) = p_{\underline{X},T|\Theta}(\underline{x}, t(\underline{x})|\underline{\theta}) = p_{\underline{X}|T,\Theta}(\underline{x}|t(\underline{x}), \underline{\theta}) p_{T|\Theta}(t(\underline{x})|\underline{\theta}) = p_{\underline{X}|T}(\underline{x}|t(\underline{x})) p_{T|\underline{\Theta}}(t(\underline{x})|\underline{\theta}).$$

In other words, there is a *factorisation* of the form

$$p_{\underline{X}|\Theta}(\underline{x}|\underline{\theta}) = g(t(\underline{x}), \underline{\theta}) h(\underline{x}), \tag{12.21}$$

where

$$h(\underline{x}) = p_{\underline{X}|T}(\underline{x}|t(\underline{x})) = p_{\underline{X}|t(\underline{X})}(\underline{x}|t(\underline{x})).$$

In statistical literature, $t(\underline{X})$ is often *defined* to be a sufficient statistic if there is a factorisation of the type given by equation (12.21). Equation (12.21) is in fact a characterisation of sufficiency in the sense that the likelihood function for $\underline{\theta}$ depends on data only through t; the aspects of data that do not influence the value of t are not needed for inference about $\underline{\theta}$, as long as $p_{\underline{X}|\Theta}(\underline{x}|\underline{\theta})$ is the object of study. In the example above, and in many other cases, this offers a *data reduction*. That is, for any n, a sample of size n can be reduced to a quantity of fixed dimension.

12.9 Prediction Sufficiency

Let $\underline{X}$ be a discrete random vector, $\underline{Y}$ a discrete random variable or vector, t a function and let $T = t(\underline{X})$. Let $\underline{\theta}$ be a parameter vector. Suppose $\underline{X}, \underline{Y}, \underline{\Theta}$ and T satisfy

$$\underline{X} \perp (\underline{Y}, \underline{\theta}) \,|T. \tag{12.22}$$

That is, once $t(\underline{X})$ is given, there is no additional statistical information in $\underline{X}$ about $\underline{Y}$ or $\underline{\theta}$. The problem is to predict $\underline{Y}$ statistically using a function of $\underline{X}$.

Proposition 12.15. *Let t denote a function and let $T = t(\underline{X})$. If $\underline{X}, \underline{Y}, T, \underline{\theta}$ satisfy $\underline{X} \perp \underline{Y} || (T, \underline{\theta})$ and $\underline{X} \perp \underline{\theta} | T$, then*

$$\pi_{\underline{\Theta}|\underline{Y},\underline{X},T}(\underline{\theta} \mid \underline{y}, \underline{x}, t) = \pi_{\underline{\Theta}|\underline{Y},T}(\underline{\theta} \mid \underline{y}, t). \tag{12.23}$$

Proof Firstly, $\underline{X} \perp \underline{Y}|(\underline{\theta}, T)$ and $\underline{X} \perp \underline{\theta}|T$ implies that $\underline{X} \perp \underline{Y}|T$. It follows that

$$\begin{aligned} p_{\underline{X},\underline{Y}|T}(\underline{x},\underline{y}|t) &= \int_\Theta p_{\underline{X},\underline{Y}|\underline{\theta},T}(\underline{x},\underline{y}|\underline{\theta},t)\pi_{\underline{\Theta}|T}(\underline{\theta}|t)d\underline{\theta} \\ &= \int_\Theta p_{\underline{X}|\underline{\theta},T}(\underline{x},\underline{y}|\underline{\theta},t)p_{\underline{Y}|\underline{\theta},T}(\underline{y}|\underline{\theta},t)\pi_{\underline{\theta}|T}(\underline{\theta}|t)d\underline{\theta} \\ &= p_{\underline{X}|T}(\underline{x}|t)\int_\Theta p_{\underline{Y}|\underline{\theta},T}(\underline{y}|\underline{\theta},t)\pi_{\underline{\theta}|T}(\underline{\theta}|t)d\underline{\theta} \\ &= p_{\underline{X}|T}(\underline{x}|t)p_{\underline{Y}|T}(\underline{y}|t). \end{aligned}$$

It follows that

$$p_{\underline{Y}|\underline{X},T}(\underline{y}|\underline{x},t) = \frac{p_{\underline{X},\underline{Y}|T}(\underline{x},\underline{y}|t)}{p_{\underline{X}|T}(\underline{x}|t)} = p_{\underline{Y}|T}(\underline{y}|t). \tag{12.24}$$

An application of Bayes rule gives

$$\begin{aligned} \pi_{\Theta|\underline{Y},\underline{X},T}(\underline{\theta}|\underline{y},\underline{x},t) &= \frac{p_{\underline{Y},\underline{X},T|\Theta}(\underline{y},\underline{x},t|\underline{\theta})\pi_\Theta(\underline{\theta})}{p_{\underline{Y},\underline{X},T}(\underline{y},\underline{x},t)} = \frac{p_{\underline{Y},\underline{X}|T,\underline{\theta}}(\underline{y},\underline{x}|t,\underline{\theta})p_{T|\underline{\theta}}(t|\underline{\theta})\pi_\Theta(\underline{\theta})}{p_{\underline{Y},\underline{X},T}(\underline{y},\underline{x},t)} \\ &= \frac{p_{\underline{Y}|T,\Theta}(\underline{y}|t,\underline{\theta})p_{\underline{X}|T,\underline{\theta}}(\underline{x}|t,\underline{\theta})p_{T|\underline{\theta}}(t|\underline{\theta})\pi_\Theta(\underline{\theta})}{p_{\underline{Y},\underline{X},T}(\underline{y},\underline{x},t)}, \end{aligned}$$

where the conditional independence $\underline{X} \perp \underline{Y}|(\underline{\theta}, T)$ was used. Then, since $\underline{X} \perp \underline{\theta}|T$, it follows that $p_{\underline{X}|T,\Theta}(\underline{x}|t,\underline{\theta}) = p_{\underline{X}|T}(\underline{x}|t)$ and hence, using the identity (12.24), that

$$\pi_{\Theta|\underline{Y},\underline{X},T}(\underline{\theta}|\underline{y},\underline{x},t) = \frac{p_{\underline{Y}|T,\Theta}(\underline{y}|t,\underline{\theta})p_{\underline{X}|T}(\underline{x}|t)p_{T|\Theta}(t|\underline{\theta})\pi_\Theta(\underline{\theta})}{p_{\underline{Y}|\underline{X},T}(\underline{y}\mid\underline{x},t)p_{\underline{X}|T}(x\mid t)p_T(t)} = \frac{p_{\underline{Y}|T,\Theta}(\underline{y}|t,\underline{\theta})p_{T|\Theta}(t|\underline{\theta})\pi_\Theta(\underline{\theta})}{p_{\underline{Y}|T}(\underline{y}\mid t)p_T(t)}.$$

It follows that

$$\pi_{\Theta|\underline{Y},\underline{X},T}(\underline{\theta}|\underline{y},\underline{x},t) = \frac{p_{\underline{Y},T|\Theta}(\underline{y},t\mid\underline{\theta})\pi_\Theta(\underline{\theta})}{p_{\underline{Y}|T}(\underline{y},t)} = \pi_{\Theta|\underline{Y},T}(\underline{\theta}|\underline{y},t),$$

as claimed. □

12.10 Prediction Sufficiency for a Bayesian Network

Let $\mathcal{G} = (V, E)$ denote a DAG with $V = \{X_1, \ldots, X_d\}$, where the nodes are numbered, for convenience such that for each j,

$$\Pi_j \subseteq \{X_1, \ldots, X_{j-1}\},$$

where Π_j (as usual) denotes the parent set for X_j.

Using a fully Bayesian approach to the problem, the parameter vector $\underline{\theta}$ is considered as an observation on a random vector $\underline{\Theta}$ and for each $j = 1 \ldots, d$ the parameter vector $\underline{\theta}_j$ an observation on a random vector $\underline{\Theta}_j$.

Definition 12.16 (Parameter Modularity). *A set of parameters $\underline{\Theta}$ for a Bayesian Network satisfies* parameter modularity *if it may be decomposed into d distinct parameter sets $\underline{\Theta}_1, \ldots, \underline{\Theta}_d$ such that for $j = 1, \ldots, d$, the parameters in vector $\underline{\Theta}_j$ are directly linked only to node X_j.*

This definition was introduced by Heckerman, Geiger and Chickering (1995) [62].

Under the assumption of parameter modularity, the DAG may be expanded by adding the parameter nodes as parent variables in the graph, and directed links from each node in the set $\underline{\Theta}_j$ to the node X_j giving an extended graph that is directed and acyclic, where $p_{X_1,\ldots,X_d|\underline{\Theta}}$ has the decomposition

$$p_{X_1,\ldots,X_d|\underline{\Theta}} = \prod_{j=1}^{d} p_{X_j|\underline{\Theta}_j,\Pi_j}. \tag{12.25}$$

Furthermore, under the assumption of modularity, $\underline{\Theta}_1, \ldots, \underline{\Theta}_d$ are independent random vectors and the joint prior distribution is a product of individual priors; $\pi_{\underline{\Theta}} = \prod_{j=1}^{d} \pi_{\underline{\Theta}_j}$.
The following notation is useful:

$$\tilde{X}_j := \left((X_1, \underline{\Theta}_1), \ldots, (X_{j-1}, \underline{\Theta}_{j-1})\right), \qquad j = 1, \ldots, d$$

and, for $j = 1, \ldots, d$, t_j is used to denote the function such that

$$t_j(\tilde{X}_j) = \Pi_j.$$

It follows directly from equation (12.25) that

$$\tilde{X}_j \perp \left(X_j, \underline{\Theta}_j\right) | \Pi_j.$$

In other words, the parent set Π_j is a *prediction sufficient statistic* for $(X_j, \underline{\Theta}_j)$ in the sense that there is no further information in $\left((X_1, \underline{\Theta}_1), \ldots, (X_{j-1}, \underline{\Theta}_{j-1})\right)$ relevant to uncertainty about either $\underline{\Theta}_j$ or X_j.

In a Bayesian network where the parameters satisfy the modularity assumption (Definition 12.16), (Π_j, X_j) are a Bayesian sufficient statistic for Θ_j. The modularity assumption is clearly satisfied when Equation (12.25) holds.

Notes The discussion of the thumb-tack and learning for DAGs is taken from D. Heckerman [61] and [62]. Learning from incomplete data is discussed in [116]. Another treatment of learning is found in [99] (Neapolitan). The Savage distribution is due to J.L. Savage [121]. The Dickey distribution is due to J.M. Dickey [38].

12.11 Exercises

1. Suppose one has a data base C with n cases of configurations over a collection of variables V. Let $Sp(V)$ denote the set of possible configurations over V and let $\#(v)$ denote the number of cases of configuration v. Define $P^C(v) = \frac{\#(v)}{n}$. Let P^M denote a probability distribution over $Sp(V)$. Assume that $P^C(v) = 0$ if and only if $P^M(v) = 0$ and discount these configurations. Define $S^M(C) = -\sum_{c \in C} \log P^M(c)$.

   Let D_{KL} denote the Kullback Leibler distance. Show that

$$S^M(C) - S^C(C) = nD_{KL}(P^C|P^M).$$

2. (a) Consider the thumb-tack experiment and the conditional independence model for the problem and the uniform prior density for θ. Let $\mathbf{X}$ denote the vector of n i.i.d. copies of the random variable and let X_{n+1} denote an additional copy, independent of $\mathbf{X}$. Let $\mathbf{x}$ denote an outcome of $\mathbf{X}$ What is $\mathbb{P}_{X_{n+1}|\mathbf{X}}(\text{head}|\mathbf{x})$?

   (b) Prove the *Laplace Rule of Succession*. Namely, let $\{X_1, \ldots, X_{n+1}\}$ be independent, identically distributed Bernoulli random variables, where $\mathbb{P}_{X_i}(1) = 1 - \mathbb{P}_{X_i}(0) = \theta$ and $\theta \sim U(0,1)$. Then the Laplace Rule of Succession states that

$$\mathbb{P}_{X_{n+1}|X_1+\ldots+X_n}(1|s) = \frac{s+1}{n+2}.$$

3. Let $\Theta \sim Beta(\alpha, \beta)$. Compute $\mathbb{E}[\Theta]$ and $\mathbf{V}(\Theta)$. You may use the fact that if $\Theta \sim Beta(\alpha, \beta)$ then its density is given by

$$\pi(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1} \qquad \theta \in [0,1].$$

4. Let $(X_1, \ldots, X_{n+1})^t$ be a vector of independent identically distributed random variables, each with probability distribution given by $\mathbb{P}_X(x^{(i)}) = \theta_i$, $i = 1, \ldots, k$. Suppose that the prior distribution over $\underline{\theta}$ is $\text{Dir}(\alpha q_1, \ldots, \alpha q_k)$ where $\sum_{i=1}^k q_i = 1$. Let $\mathbf{X} = (X_1, \ldots, X_n)$ and let $\mathbf{x}$ be an n-vector of outcomes where $x^{(i)}$ appears n_i times, for $i = 1, \ldots, k$ and $\sum_{i=1}^k n_i = n$. Show that

$$\mathbb{P}_{X_{n+1}|\mathbf{X}}\left(x^{(i)} \mid \mathbf{x}\right) = \int_{S_L} \theta_i \pi\left(\theta_1, \ldots, \theta_L | \underline{x}; \alpha \underline{q}\right) d\theta_1 \ldots d\theta_L = \frac{n_i + \alpha q_i}{n + \alpha}. \tag{12.26}$$

5. Let $\underline{\Theta} = (\Theta_1, \ldots, \Theta_L)$ be a continuous random vector with $Dir(\alpha_1, \ldots, \alpha_L)$ distribution. Compute $\mathbf{V}(\Theta_i)$.

6. (a) Let $\underline{V} = (V_1, \ldots, V_K)$ be a continuous random vector, with

$$\underline{V} \sim \text{Dir}(a_1, \ldots, a_K),$$

   and set

$$U_i = \frac{V_i x_i^{-1}}{\sum_{i=1}^K V_i x_i^{-1}}, \quad , i = 1, \ldots, K,$$

where $\underline{x} = (x_1, \ldots, x_K)$ is a vector of positive real numbers; that is, $x_i > 0$ for each $i = 1, \ldots, K$. Show that $\underline{U} = (U_1, \ldots, U_K)$ has density function

$$\frac{\Gamma\left(\sum_{i=1}^k a_i\right)}{\prod_{i=1}^K \Gamma(a_i)} \prod_{i=1}^K u_i^{a_i - 1} \left(\frac{1}{\sum_{i=1}^K u_i x_i}\right)^{\sum_{i=1}^K a_i} \prod_{i=1}^K x_i^{a_i}.$$

This density is denoted

$$\underline{U} \sim S\left(\underline{a}, \underline{x}\right).$$

This is due to J.L. Savage [121]. Note that the Dirichlet density is obtained as a special case when $x_i = c$ for $i = 1, \ldots, K$.

The next two parts illustrate how the Savage distribution can arise in Bayesian analysis, for updating an objective distribution over the subjective assessments of a probability distribution by several different researchers, faced with a common set of data.

(b) Consider several researchers studying an unknown quantity X, where X can take values in $\{1, 2, \ldots, K\}$. Each researcher has his own initial assessment of the probability distribution $\underline{V} = (V_1, \ldots, V_K)$ for the value that X takes. That is, for a particular researcher,

$$V_i = \mathbb{P}_X(i), \quad i = 1, \ldots, K.$$

It is assumed that

$$\underline{V} \sim \mathrm{Dir}\left(a_1, \ldots, a_K\right).$$

Each researcher observes the same set of data with the common likelihood function

$$l_i = \mathbb{P}\left(\mathrm{data} | \{X = i\}\right), \quad i = 1, \ldots, K.$$

The coherent posterior probability of a researcher is

$$U_i = \mathbb{P}\left(\{X = i\} \mid \mathrm{data}\right), \quad i = 1, 2, \ldots, K.$$

Let $\underline{U} = (U_1, \ldots, U_K)$. Prove that

$$\underline{U} \sim S\left(\underline{a}, \underline{l}^{-1}\right),$$

where $\underline{a} = (a_1, \ldots, a_K)$ and $\underline{l}^{-1} = \left(l_1^{-1}, \ldots, l_K^{-1}\right)$. This is due to J.M. Dickey [38].

(c) Show that the family of distributions $S\left(\underline{a}, \underline{l}^{-1}\right)$ is closed under updating of the opinion populations. In other words, if

$$\underline{V} \sim S\left(\underline{a}, \underline{z}\right),$$

before the data is considered, then

$$\underline{U} \sim S\left(\underline{a}, \underline{z} \times \underline{l}^{-1}\right),$$

after the data update, where

$$\underline{z} \times \underline{l}^{-1} = \left(z_1 l_1^{-1}, \ldots, z_K l_K^{-1}\right).$$

7. Consider a Bayesian Network over two binary variables A and B, where the Directed Acyclic Graph is $A \to B$ and A and B each take the values 0 or 1. Let $(\Theta_a, \Theta_{b|y}, \Theta_{b|n})$ denote three independent random variables representing the unknown parameters. Let $\theta_a = \mathbb{P}_{A|\Theta_a}(1|\theta_a)$, $\theta_{b|y} = \mathbb{P}_{B|A,\Theta_{b|y}}(1|1, \theta_{b|y})$, $\theta_{b|n} = \mathbb{P}_{B|A,\Theta_{b|n}}(1|0, \theta_{b|n})$. Let the prior distributions over the parameters be

$$\pi_a(\theta) = \begin{cases} 3\theta^2 & 0 \le \theta_a \le 1 \\ 0 & \theta \notin [0,1], \end{cases}$$

$$\pi_{b|y}(\theta) = \begin{cases} 12\theta^2(1-\theta) & 0 \le \theta \le 1 \\ 0 & \theta \notin [0,1], \end{cases}$$

$$\pi_{b|n}(\theta) = \begin{cases} 12\theta(1-\theta)^2 & 0 \le \theta \le 1 \\ 0 & \theta \notin [0,1], \end{cases}$$

Suppose that there is a single instantiation, where $B = 1$ is observed, but A is unknown. Perform the approximate updating.

8. Let the likelihood for $\underline{\theta} = (\theta_1, \ldots, \theta_L)$ with data $\mathbf{x}$ be given by

$$L\left(\underline{\theta}; \mathbf{x}\right) = \prod_{j=1}^{L} \theta_j^{n_j},$$

where n_j is the number of times the symbol x_j (in a finite alphabet with L symbols) is present in $\mathbf{x}$ and $\sum_{j=1}^{L} \theta_j = 1$. For the prior distribution over θ, a finite *Dirichlet mixture* is taken, given by

$$\pi_\Theta\left(\theta\right) = \sum_{i=1}^{k} \lambda_i Dir\left(\alpha^{(i)} q_1^{(i)}, \ldots, \alpha^{(i)} q_L^{(i)}\right),$$

where $\lambda_i \ge 0$, $\sum_{i=1}^{k} \lambda_i = 1$ (the mixture distribution), $\alpha^{(i)} > 0$, $q_j^{(i)} > 0$, $\sum_{i=1}^{L} q_j^{(i)} = 1$ for every i. Compute the mean posterior estimate $\widehat{\theta}_{j;MP}$ for $j = 1, \ldots, L$.

9. Let $\phi(\underline{\theta}_{j.l}, \underline{\alpha}_{j.l})$ denote the Dirichlet density $\mathrm{Dir}(\alpha_{j1l}, \ldots, \alpha_{jk_jl})$. By performing the required integration, prove that the Likelihood function for the graph structure, defined by

$$\mathbb{P}_{\mathbf{X}|\mathcal{D}}(\mathbf{x}|D) = \int \prod_{k=1}^{n} \mathbb{P}_{\underline{X}|\Theta,\mathcal{D}}(\underline{x}_{(k)}|\underline{\theta}, D) \prod_{j=1}^{d} \prod_{l=1}^{q_j} \phi(\underline{\theta}_{j.l}, \underline{\alpha}_{j.l}) d\underline{\theta}_{j.l}$$

is given by

$$\mathbb{P}_{\mathbf{X}|\mathcal{D}}(\mathbf{x}|D) = \prod_{j=1}^{d} \prod_{l=1}^{q_j} \frac{\Gamma\left(\sum_{i=1}^{k_j} \alpha_{jil}\right)}{\Gamma\left(n(\pi_j^l) + \sum_{i=1}^{k_j} \alpha_{jil}\right)} \prod_{i=1}^{k_j} \frac{\Gamma\left(n(x_j^i, \pi_j^l) + \alpha_{jil}\right)}{\Gamma\left(\alpha_{jil}\right)}.$$

You may use the identity:

$$\int_0^1 \int_0^{1-\theta_1} \cdots \int_0^{1-(\theta_1+\ldots+\theta_{n-2})} \left(\prod_{j=1}^{n-1} \theta_j^{\alpha_j - 1}\right) \left(1 - \sum_{j=1}^{n-1} \theta_j\right)^{\alpha_n - 1} d\theta_{n-1} \ldots d\theta_1 = \frac{\prod_{j=1}^{n} \Gamma(\alpha_j)}{\Gamma(\sum_{j=1}^{n} \alpha_j)}.$$

What parameters $\alpha_{j.l}$ are used if a uniform prior is taken on every $\theta_{j.l}$? You may use $\Gamma(n) = (n-1)!$.

12.12 Short Answers

1. Firstly, note that

$$S^M(C) = -\sum_{c\in C} \log P^M(c) = -\sum_{v\in Sp(v)} (\#(v)) \log P^M(v) = -\sum_{v\in Sp(v)} nP^C(v) \log P^M(v).$$

This is true for all M; in particular, take $M = C$, so that

$$S^C(C) = -\sum_{v\in Sp(v)} nP^C(v) \log P^C(v)$$

giving

$$S^M(C) - S^C(C) = n \sum_{v\in Sp(v)} P^C(v) \log \frac{P^C(v)}{P^M(v)} = nd_K(P^C|P^M).$$

2. (a) Suppose $\mathbf{x}$ contains k heads. Then

$$\begin{aligned}
\mathbb{P}_{X_{n+1}|\mathbf{X}}(H|\mathbf{x}) &= \int_0^1 \mathbb{P}_{X_{n+1}|\Theta,\mathbf{X}}(H|\theta,\mathbf{x})\pi_{\Theta|\mathbf{X}}(\theta|\mathbf{x})d\theta \\
&= \frac{(n+1)!}{k!(n-k)!}\int_0^1 \theta^{k+1}(1-\theta)^{n-k}d\theta \\
&= \frac{(n+1)!}{k!(n-k)!}\frac{(k+1)!(n-k)!}{(n+2)!} = \frac{k+1}{n+2}.
\end{aligned}$$

(b) Using the evaluation of the Beta integral,

$$\begin{aligned}
&\mathbb{P}_{X_{n+1}|X_1+\ldots+X_n}(1|s) \\
&= \int_0^1 \mathbb{P}_{X_{n+1}|X_1+\ldots+X_n,\Theta}(1|s,\theta)\pi_{\Theta|X_1+\ldots+X_n}(\theta|s)d\theta \\
&= \int_0^1 \theta \frac{\mathbb{P}_{X_1+\ldots+X_n|\Theta}(s|\theta)\pi_\Theta(\theta)}{\int \mathbb{P}_{X_1+\ldots+X_n|\Theta}(s|\theta)\pi_\Theta(\theta)d\theta}d\theta \\
&= \frac{\binom{n}{s}\int_0^1 \theta^{s+1}(1-\theta)^{n-s}d\theta}{\binom{n}{s}\int_0^1 \theta^{s}(1-\theta)^{n-s}d\theta} \\
&= \frac{\frac{n!}{s!(n-s)!}\frac{(s+1)!(n-s)!}{(n+2)!}}{\frac{n!}{s!(n-s)!}\frac{s!(n-s)!}{(n+1)!}} \\
&= \frac{s+1}{n+2}
\end{aligned}$$

3.

$$\mathbb{E}[\Theta] = \int_0^1 \theta\pi(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\int_0^1 \theta^{\alpha}(1-\theta)^{\beta-1}d\theta = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\frac{\Gamma(\alpha+1)\Gamma(\beta)}{\Gamma(\alpha+\beta+1)} = \frac{\alpha}{\alpha+\beta}$$

using $\Gamma(x+1) = x\Gamma(x)$.

$$\mathbf{V}(\Theta) = \mathbb{E}[\Theta]^2 - \mathbb{E}[\Theta]^2.$$

$$\mathbb{E}[\Theta^2] = \int_0^1 \theta^2 \pi(\theta) d\theta = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{\Gamma(\alpha+2)\Gamma(\beta)}{\Gamma(\alpha+\beta+2)} = \frac{(\alpha+1)\alpha}{(\alpha+\beta+1)(\alpha+\beta)}$$

so

$$\begin{aligned} \mathbf{V}(\Theta) &= \frac{(\alpha+1)\alpha}{(\alpha+\beta+1)(\alpha+\beta)} - \frac{\alpha^2}{(\alpha+\beta)^2} \\ &= \frac{(\alpha+1)\alpha(\alpha+\beta) - \alpha^2(\alpha+\beta+1)}{(\alpha+\beta+1)(\alpha+\beta)^2} = \frac{\alpha\beta}{(\alpha+\beta+1)(\alpha+\beta)^2}. \end{aligned}$$

4.

$$\begin{aligned} \mathbb{P}_{X_{n+1}|\mathbf{X}}(x_i|\mathbf{x}) &= \int_S \mathbb{P}_{X_{n+1}|\Theta,\mathbf{X}}(x^{(i)}|\underline{\theta},\mathbf{x}) \pi_{\Theta|\mathbf{X}}(\underline{\theta}|\mathbf{x}) d\underline{\theta} \\ &= \frac{n+\alpha}{\prod_{j=1}^L \Gamma(\alpha q_j + n_j)} \int_S (\prod_{j\neq i} \theta_j^{n_j+\alpha q_j - 1}) \theta_i^{n_i+\alpha q_i} d\underline{\theta} \\ &= \frac{\Gamma(n+\alpha)}{\prod_{j=1}^L \Gamma(n_j+\alpha q_j)} \frac{(\prod_{j\neq i} \Gamma(n_j+\alpha q_j))\Gamma(n_i+1+\alpha q_i)}{\Gamma(n+\alpha+1)} \\ &= \frac{\Gamma(n_i+1+\alpha q_i)\Gamma(n+\alpha)}{\Gamma(n_i+\alpha q_i)\Gamma(n+\alpha+1)} \\ &= \frac{n_i+\alpha q_i}{n+\alpha}. \end{aligned}$$

5. Let $\alpha = \alpha_1 + \ldots + \alpha_L$. Then, taking $d\underline{\theta}$ in the appropriate sense and using

$$\Gamma(\alpha+1) = \alpha\Gamma(\alpha),$$

$$\begin{aligned} \mathbb{E}[\Theta_i] &= \int \theta_i \pi_\Theta(\underline{\theta}) d\underline{\theta} \\ &= \frac{\Gamma(\alpha)}{\prod_{j=1}^L \Gamma(\alpha_j)} \int \left(\prod_{j\neq i} \theta_j^{\alpha_j} \right) \theta_i^{\alpha_i+1} d\underline{\theta} \\ &= \frac{\Gamma(\alpha)}{\prod_{j=1}^L \Gamma(\alpha_j)} \frac{\left(\prod_{j\neq i} \Gamma(\alpha_j)\right)\Gamma(\alpha_i+1)}{\Gamma(\alpha+1)} = \frac{\Gamma(\alpha)\Gamma(\alpha_i+1)}{\Gamma(\alpha+1)\Gamma(\alpha_i)} \\ &= \frac{\alpha_i}{\alpha}. \end{aligned}$$

Similarly,

$$\mathbb{E}[\Theta_i^2] = \frac{\Gamma(\alpha)\Gamma(\alpha_i+2)}{\Gamma(\alpha+2)\Gamma\alpha_i)} = \frac{(\alpha_i+1)\alpha_i}{(\alpha+1)\alpha}.$$

This gives

$$\mathbf{V}(\Theta_i) = \frac{\alpha_i(\alpha_i+1)}{\alpha(\alpha+1)} - \frac{\alpha_i^2}{\alpha^2} = \frac{\alpha\alpha_i^2 + \alpha\alpha_i - \alpha\alpha_i^2 - \alpha_i^2}{\alpha^2(\alpha+1)} = \frac{\alpha_i(\alpha-\alpha_i)}{\alpha^2(\alpha+1)}.$$

6. (a) The free variables are $(v_1, \ldots, v_{K-1})$ with the constraint $v_K = 1 - \sum_{j=1}^{K-1} v_j$. Set

$$S = \sum_{j=1}^{K} \frac{v_j}{x_j} = \frac{1}{x_K} + \sum_{j=1}^{K-1} v_j \left(\frac{1}{x_j} - \frac{1}{x_K} \right)$$

then

$$u_j = \frac{v_j}{x_j S} \qquad j = 1, \ldots, K \quad \text{and} \quad u_K = 1 - \sum_{j=1}^{K-1} u_j.$$

and, since $\sum_{j=1}^{K} v_j = 1$, it follows that $1 = \sum_{j=1}^{K} v_j = S \sum_{j=1}^{K} x_j u_j$ so that

$$S = \frac{1}{x_K + \sum_{j=1}^{K-1} (x_j - x_K) u_j} = \frac{1}{\sum_{j=1}^{K} x_j u_j}$$

The original density (in terms of the free variables) is

$$\frac{\Gamma\left(\sum_{j=1}^{k} a_j\right)}{\prod_{j=1}^{K} \Gamma(a_j)} \left(\prod_{j=1}^{K-1} v_j^{a_j - 1} \right) \left(1 - \sum_{j=1}^{K-1} v_j \right)^{a_K - 1}.$$

The Jacobian determinant for $\underline{v} \to \underline{u}$ may be computed by noting that $v_j = u_j x_j S$ and using

$$\frac{\partial S}{\partial u_\alpha} = -S^2 (x_\alpha - x_K)$$

so that

$$\frac{\partial v_i}{\partial u_\alpha} = \begin{cases} -S u_i x_i (x_\alpha - x_K) & \alpha \neq i \\ S x_i - S u_i x_i (x_i - x_K) & \alpha = i \end{cases}$$

The matrix of which the determinant is to be computed is therefore $S^{K-1} \prod_{i=1}^{K-1} x_i M$, where

$$M = I - S \begin{pmatrix} u_1 \\ \vdots \\ u_{K-1} \end{pmatrix} (x_1 - x_K, \ldots, x_{K-1} - x_K).$$

Clearly 1 is an eigenvalue of multiplicity $K - 2$ for M. The remaining eigenvalue λ of M may be computed by noting that the vector $\underline{e}$ that satisfies

$$(M - \lambda)\underline{e} = 0$$

satisfies $\underline{e} = c \begin{pmatrix} u_1 \\ \vdots \\ u_{K-1} \end{pmatrix}$ and therefore λ satisfies

$$1 - \lambda = S \sum_{j=1}^{K-1} u_j (x_j - x_K) = S\left(\sum_{j=1}^{K-1} u_j x_j - x_K + x_K u_K \right) = S\left(\frac{1}{S} - x_K \right)$$

so that $\lambda = S x_K$. It follows that the density in the new coordinates is

$$\frac{\Gamma\left(\sum_{j=1}^{k} a_j\right)}{\prod_{j=1}^{K} \Gamma(a_j)} \left(\prod_{j=1}^{K-1} (S x_j u_j)^{a_j - 1} \right) \left(1 - S \sum_{j=1}^{K-1} x_j u_j \right)^{a_K - 1} S^K \prod_{j=1}^{K} x_j.$$

Since $Sx_K u_K = 1 - S\sum_{j=1}^{K-1} x_j u_j$, it follows that

$$\frac{\Gamma\left(\sum_{j=1}^{k} a_j\right)}{\prod_{j=1}^{K}\Gamma(a_j)} \prod_{j=1}^{K} x_j^{a_j} \left(\prod_{j=1}^{K} u_j^{a_j-1}\right)\left(\frac{1}{\sum_{j=1}^{K} x_j u_j}\right)^{\sum_{j=1}^{K} a_j}$$

as required.

(b) The work was in the previous part, computing the distribution. This exercise is now a straightforward application of Bayes rule.

$$U_i = \mathbb{P}(\{X = i\}|\text{data}) = \frac{\mathbb{P}_X(i) l_i}{\mathbb{P}(\text{data})} = \frac{V_i l_i}{\sum_{i=1}^{K} V_i l_i}$$

the denominator follows because $\sum_{i=1}^{K} U_i = 1$. The distribution of $\underline{U}$ now satisfies the definition of the $S(\underline{a}, \underline{l}^{-1})$ distribution of the previous exercise.

(c) Again, assume data is obtained and the likelihood is $l_i = \mathbb{P}(\text{data}|X = i)$ and the prior distribution is $S(\underline{a}, \underline{z})$. Then

$$U_i = \mathbb{P}(\{X = i\}|\text{data}) = \frac{V_i l_i}{\mathbb{P}(\text{data})} = \frac{W_i z_i^{-1} l_i}{\mathbb{P}(\text{data}) \sum_{i=1}^{K} W_i z_i^{-1}},$$

where $\underline{W} \sim \text{Dir}(a_1, \ldots, a_K)$. Since $\sum_{i=1}^{K} U_i = 1$, it follows that

$$U_i = \frac{W_i z_i^{-1} l_i}{\sum_{i=1}^{K} W_i z_i^{-1} l_i}$$

so that the distribution of $\underline{U}$ satisfies the definition of a $S(\underline{a}, \underline{z} \times \underline{l}^{-1})$ distribution.

7. With approximate updating, the independence structure of the distributions over $(\theta_{j.l})_{i=1}^{k_j}$ is retained (the distributions for each (j, l) are mutually independent). Let $n(x_j^{(i)} \pi_j^{(l)})$ denote the effective number of $(x_j^{(i)}, \pi_j^{(l)})$ configurations upon which the prior distribution is based, then

$$\Theta_{j.l} \sim \text{Dir}(n(x_j^{(1)}, \pi_j^{(l)}), \ldots, n(x_j^{(k_j)}, \pi_j^{(l)}))$$

before the update. After a partially observed instantiation, this is updated to

$$\text{Dir}(n^*(x_j^{(1)}, \pi_j^{(l)}), \ldots, n^*(x_j^{(k_j)}, \pi_j^{(l)}))$$

where

$$n^*(x_j^i, \pi_j^l) = n(x_j^{(i)}, \pi_j^{(l)}) + \mathbb{P}_{X_j, \text{Pa}_j|E}(x_j^{(i)}, \pi_j^{(l)}|e^*)$$

where $\mathbb{P}$ is the probability computed using the prior and $E = (X_{i_1}, \ldots, X_{i_m})$, those variables that are instantiated in the partial observation; e^* denotes the values that these variables take in the incomplete instantiation.

To update the distribution over Θ_a, the effective sample sizes on which the prior is based are needed. Furthermore, for $X_j = A$, $\text{Pa}_j = \phi$, so $\mathbb{P}_{X_j|\text{Pa}_j,E} = \mathbb{P}_{A|B}(.|1)$. For $X_j = B$, $\text{Pa}_j = A$, so that $\mathbb{P}_{X_j|\text{Pa}_j,E} = \mathbb{P}_{B|A,B}(.|., 1)$.

The computations of $\mathbb{P}_A$ and $\mathbb{P}_{B|A}$ are straightforward; $\mathbb{P}_{A|B}$ is obtained using Bayes rule. Note that

$$\mathbb{P}_A(1) = \int_0^1 \mathbb{P}_{A|\Theta_a}(1|\theta)\pi_{\Theta_a}(\theta)d\theta = 3\int_0^1 \theta^3 d\theta = \frac{3}{4}$$

$$\mathbb{P}_A(0) = \frac{1}{4}$$

$$\mathbb{P}_{B|A}(1|1) = \int_0^1 \mathbb{P}_{B|A,\Theta_{b|y}}(1|1,\theta)\pi_{\Theta_{b|y}}(\theta)d\theta = 12\int_0^1 \theta^3(1-\theta)d\theta = \frac{3}{5}$$

$$\mathbb{P}_{B|A}(0|1) = \frac{2}{5}$$

$$\mathbb{P}_{B|A}(1|0) = 12\int_0^1 \theta^2(1-\theta)^2 = \frac{2}{5}$$

$$\mathbb{P}_{B|A}(0|0) = \frac{3}{5}$$

so

$$\mathbb{P}_B(1) = \frac{2}{5}\times\frac{1}{4} + \frac{3}{5}\times\frac{3}{4} = \frac{11}{20}$$

$$\mathbb{P}_B(0) = \frac{9}{20}$$

$$\mathbb{P}_{A|E^*}(1|e^*) = \mathbb{P}_{A|B}(1|1) = \frac{\mathbb{P}_A(1)\mathbb{P}_{B|A}(1|1)}{\mathbb{P}_B(1)} = \frac{9}{11}$$

$$\mathbb{P}_{A|E^*}(0|e^*) = \frac{2}{11}$$

$$\mathbb{P}_{A,B|B}((0,1)|1) = \mathbb{P}_{A|B}(0|1) = \frac{2}{11}$$

$$\mathbb{P}_{A,B|B}((1,1)|1) = \mathbb{P}_{A|B}(1|1) = \frac{9}{11}$$

$$\mathbb{P}_{A,B|B}((0,0)|1) = \mathbb{P}_{A,B|B}((1,0)|1) = 0.$$

So updating is

$$\pi_{a|e^*}(\theta) = \frac{\Gamma(5)}{\Gamma(3+\frac{9}{11})\Gamma(1+\frac{2}{11})}\theta^{2+\frac{9}{11}}(1-\theta)^{\frac{2}{11}}, \quad \theta\in[0,1]$$

$$\pi_{b|y,e^*}(\theta) = \frac{\Gamma(5+\frac{9}{11}))}{\Gamma(3+\frac{9}{11})\Gamma(2)}\theta^{2+\frac{9}{11}}(1-\theta), \quad \theta\in[0,1]$$

$$\pi_{b|n,e^*}(\theta) = \frac{\Gamma(5+\frac{2}{11}))}{\Gamma(2+\frac{2}{11})\Gamma(3)}\theta^{1+\frac{2}{11}}(1-\theta)^2, \quad \theta\in[0,1]$$

8. First note that

$$\begin{aligned}
\pi_{\Theta|\mathbf{X}}(\theta|\mathbf{x}) &= (\text{const})\pi_\Theta(\theta)\mathbb{P}_{\mathbf{X}|\Theta}(\mathbf{x}|\theta) \\
&= (\text{const})\sum_i \lambda_i \text{Dir}(\alpha^{(i)}q_1^{(i)} + n_1, \ldots, \alpha^{(i)}q_L^{(i)} + n_L) \\
&= \sum \lambda_i \frac{\Gamma(\sum_j \alpha^{(i)}q_j^{(i)} + n_j)}{\prod_j \Gamma(\alpha^{(i)}q_j^{(i)} + n_j)} \prod_{j=1}^{L} \theta_j^{\alpha^{(i)}q_j^{(i)}+n_j-1}.
\end{aligned}$$

By standard Dirichlet integral calculations,

$$E[\theta_j] = \sum_i \lambda_i \frac{\alpha^{(i)}q_j^{(i)} + n_j + 1}{n + \alpha^{(i)} + 1}$$

where n is the total sample size.

9. (straightforward application of Dirichlet integrals)

Chapter 13

Parameters and Sensitivity

Notations As usual, for a variable X_j, let Pa_j denote the set of parent variables and let $(\pi_j^{(i)})_{i=1}^{q_j}$ denote the possible configurations for the parent set. Set

$$\theta_{jil} = \mathbb{P}_{X_j|\mathrm{Pa}_j}(x_j^{(i)}|\pi_j^{(l)}),$$

so that $\sum_{i=1}^{k_j} \theta_{jil} = 1$ for each (j,l). The collection of $\theta_{jil} : j = 1,\ldots,d,\ i = 1,\ldots,k_j,\ l = 1,\ldots,q_j$ with the constraint given above denotes the entire set of parameters for the network.

The functions $\mathbb{P}_{X_j|\mathrm{Pa}_j}$ will be referred to as *potentials* or CPPs (conditional probability potentials).

13.1 Parameter Changes to Satisfy Query Constraints

Definition 13.1 (Query, Query Constraint)**.** *A* query *in probabilistic inference is simply a conditional probability distribution, over the variables of interest (the* query *variables) conditioned on information received. A* query constraint *is a restriction; for example, if it is known that two conditional probabilities differ by a certain amount, or if there is a restriction on the ratio between two conditional probabilities.*

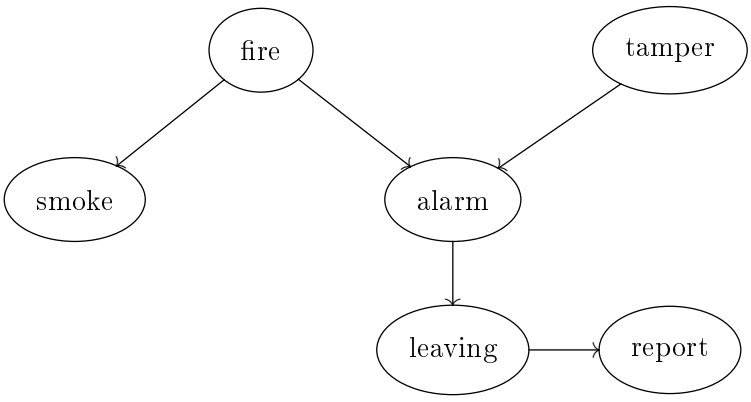

Figure 13.1: The DAG for the Bayesian Network 'Fire'

The problem considered in this section is to decide whether an individual parameter is relevant to a given query constraint and, if it is, to compute the minimum amount of change needed to that parameter to enforce the constraint. The constraints considered are in the form of *hard evidence* where the collection $\underline{E}$ is instantiated as $\underline{e}$, where $\underline{E} = (X_{e_1}, \ldots, X_{e_m})$ is a subset of $(X_1, \ldots, X_d)$.

Example 13.2 (Fire)**.**

Consider the Bayesian network called *Fire*.[1] The model is shown in Figure 13.1. The network models the scenario of whether or not there is a fire in the building. Let F denote 'fire', T denote 'tampering', S 'smoke', A 'alarm', L 'leaving' and R 'report'. A fire may causes smoke to be seen; it may also cause the alarm to go off. Equally, if somebody tampers with the alarm, this could also cause it to go off, even without a fire. When people hear the alarm, they may leave the building and when a large number of people leave the building at an unscheduled time, this may be reported to the fire department.

Now consider the following evidence: $\{\text{report} = \textit{true}, \text{smoke} = \textit{false}\}$. That is, the fire department receives a report that people are evacuating the building, but no smoke is observed. This evidence should make it more likely that the fire alarm has been tampered with than that there is a real fire. Let t denote 'true' and f denote 'false'. Suppose that the conditional probability values for this network derived, perhaps, from experience, are

$$\mathbb{P}_F = \begin{array}{cc} t & f \\ \hline 0.01 & 0.99 \end{array}, \qquad \mathbb{P}_T = \begin{array}{cc} t & f \\ \hline 0.02 & 0.98 \end{array},$$

$$\mathbb{P}_{R|L} = \begin{array}{c|cc} L\backslash R & t & f \\ \hline t & 0.75 & 0.25 \\ f & 0.01 & 0.99 \end{array}$$

$$\mathbb{P}_{S|F} = \begin{array}{c|cc} F\backslash S & t & f \\ \hline t & 0.9 & 0.1 \\ f & 0.01 & 0.99 \end{array}, \qquad \mathbb{P}_{L|A} = \begin{array}{c|cc} A\backslash L & t & f \\ \hline T & 0.88 & 0.12 \\ f & 0.001 & 0.999 \end{array}$$

$$\mathbb{P}_{A|F,T}(t|.,.) = \begin{array}{c|cc} F\backslash T & t & f \\ \hline t & 0.5 & 0.99 \\ f & 0.85 & 0.0001. \end{array}$$

The evidence is $(R, S) = (t, f)$. The probability that someone has tampered with the alarm given this evidence is

$$\mathbb{P}_{T|R,S}(t|t, f) = \frac{\mathbb{P}_{T,R,S}(t, t, f)}{\mathbb{P}_{R,S}(t, f)}.$$

Using the notation $\mathcal{X}_Z$ to denote the state space of a variable Z,

[1] This Bayesian network is distributed with the evaluation version of the commercial HUGIN Graphical User Interface, by HUGIN Expert.

$$\mathbb{P}_{T,R,S}(t,t,f) = \mathbb{P}_T(t) \sum_{\mathcal{X}_L} p_{R|L}(t|.) \sum_{\mathcal{X}_A} \mathbb{P}_{L|A} \sum_{\mathcal{X}_F} \mathbb{P}_{A|T,F}(.|t,.) \mathbb{P}_{S|F}(f|.) \mathbb{P}_F$$

and

$$\mathbb{P}_{R,S}(t,f) = \sum_{\mathcal{X}_T} \mathbb{P}_T \sum_{\mathcal{X}_L} \mathbb{P}_{R|L}(t|.) \sum_{\mathcal{X}_A} \mathbb{P}_{L|A} \sum_{\mathcal{X}_F} \mathbb{P}_{A|T,F} \mathbb{P}_{S|F}(f|.) \mathbb{P}_F.$$

Similarly,

$$\mathbb{P}_{F|R,S}(t|t,f) = \frac{\mathbb{P}_{F,R,S}(t,t,f)}{\mathbb{P}_{R,S}(t,f)},$$

and

$$\mathbb{P}_{F,R,S}(t,t,f) = \mathbb{P}_F(t) \mathbb{P}_{S|F}(f|t) \sum_{\mathcal{X}_T} \mathbb{P}_T \sum_{\mathcal{X}_A} \mathbb{P}_{A|T,F}(.|.,f) \sum_{\mathcal{X}_L} \mathbb{P}_{L|A} \mathbb{P}_{R|L}(t|.)$$

The computations are straightforward and give

$$\mathbb{P}_{T|R,S}(t|t,f) = 0.501, \qquad \mathbb{P}_{F|R,S}(t|t,f) = 0.0294.$$

Suppose that it is known from experience that the probability that the alarm has been tampered with should be no less than 0.65 given this evidence. The network should therefore be adjusted to accommodate. It is simplest to try changing only one network parameter. Suppose that the probability function $\mathbb{P}_T$ is to be adjusted. Let $\theta = \mathbb{P}_T(t)$. Let

$$\alpha = \sum_{\mathcal{X}_L} \mathbb{P}_{R|L}(t|.) \sum_{\mathcal{X}_A} \mathbb{P}_{L|A} \sum_{F} \mathbb{P}_{A|T,F}(.|t,.) \mathbb{P}_{S|F}(f|.) \mathbb{P}_F$$

so that

$$\mathbb{P}_{T,R,S}(t,t,f) = \theta\alpha$$

and

$$\beta = \sum_{\mathcal{X}_L} \mathbb{P}_{R|L}(t|.) \sum_{\mathcal{X}_A} \mathbb{P}_{L|A} \sum_{\mathcal{X}_F} \mathbb{P}_{A|T,F}(.|f,.) \mathbb{P}_{S|F}(f|.) \mathbb{P}(.),$$

so that

$$\mathbb{P}_{R,S}(t,f) = \theta\alpha + (1-\theta)\beta.$$

Then the computation of α and β is straightforward arithmetic and

$$\mathbb{P}_{T|R,S}(t|t,f) = \frac{\mathbb{P}_{T,R,S}(t,t,f)}{\mathbb{P}_{R,S}(t,f)} = \frac{\alpha\theta}{(\alpha-\beta)\theta+\beta}.$$

The solution to the equation

$$\frac{\alpha\theta}{(\alpha-\beta)\theta} = 0.65$$

is $\theta = 0.0364$.

Similarly, let $\psi = \mathbb{P}_{R|L}(t|f)$. Keeping all other potentials fixed, $\mathbb{P}_{T|R,S}(t|t,f)$ may be computed as a function of ψ and the equation $\mathbb{P}_{T|R,S}(t|t,f)(\psi) = 0.65$ has solution $\psi = 0.00471$.

For all other single parameter adjustments, the equation does not have a solution in the interval $[0,1]$. Therefore, if only one parameter is to be adjusted, the constraint $\mathbb{P}_{T|R,S}(t|t,f) = 0.65$ can be dealt with in either of the following two ways:

1. Increase $\mathbb{P}_T(t)$ from 0.02 to greater than 0.0364, or
2. Decrease the probability of a *false* report, given that there is an evacuation, from 0.01 to less than 0.00471.

It turns out for this example that it is not possible to enforce the desired constraint by adjusting a *single* parameter in any of the CPPs of the variables *fire*, *smoke*, *alarm* and *leaving*. □

13.2 Proportional Scaling

A network where each conditional probability distribution $(\theta_{jil})_{i=1}^{k_j}$ has at most one variable parameter $t^{(jl)}$ is said to satisfy the *proportional scaling* property.

Definition 13.3 (Proportional Scaling Property)**.** *A Bayesian network satisfies the* proportional scaling property *if for each conditional probability distribution $\theta_{j.l}$, where $\theta_{jil} = p_{X_j|Pa_j}(x_j^{(i)}|\pi_j^{(l)})$, there is a parameter $t^{(jl)}$ such that*

$$\mathbb{P}_{X_j|Pa_j}(.|\pi_j^{(l)}) = (\alpha_{j1l} + \beta_{j1l}t^{(jl)}, \ldots, \alpha_{jk_jl} + \beta_{jk_jl}t^{(jl)}),$$

where $\sum_{m=1}^{k_j} \alpha_{jml} = 1$ and $\sum_{m=1}^{k_j} \beta_{jml} = 0$.

Theorem 13.4. *Consider a Bayesian network over a collection of variables $V = \{X_1, \ldots, X_d\}$. Suppose that the network satisfies proportional scaling, where there is a single variable parameter t in a conditional probability distribution $\theta_{j.l}$. Then for any $E = (X_{i_1}, \ldots, X_{i_m})$ and $\underline{e} = (x_{i_1}^{(c_1)}, \ldots, x_{i_m}^{(c_m)})$,*

$$\mathbb{P}_E(\underline{e})(t) = at + b$$

for two constants a and b that depend on $\underline{e}$.

Proof of Theorem 13.4 Let $\theta_{jil} = \alpha_{jil} + \beta_{jil}t$ for $i = 1, \ldots, k_j$. Then

$$\begin{aligned}
\mathbb{P}_E(\underline{e}) &= \sum_{\underline{y} \in \mathcal{X} | (y_{i_1}, \ldots, y_{i_m}) = (x_{i_1}^{(c_1)}, \ldots, x_{i_m}^{(c_m)})} \mathbb{P}_V(y_1, \ldots, y_d) \\
&= \sum_{\underline{y} \in \mathcal{X} | (y_{i_1}, \ldots, y_{i_m}) = (x_{i_1}^{(c_1)}, \ldots, x_{i_m}^{(c_m)})} \mathbb{P}_{X_j|\mathrm{Pa}_j}(y_j|\pi_j(\underline{y})) \prod_{k \neq j} \mathbb{P}_{X_k|\mathrm{Pa}_k}(y_k|\pi_k(\underline{y})) \qquad (13.1)
\end{aligned}$$

and it is clear from the definition of proportional scaling, and from Equation (13.1), that t enters linearly. It therefore follows that

$$\mathbb{P}_E(\underline{e})(t) = at + b.$$

□

It follows that for two disjoint sets of variables A and E, there are numbers $a(\underline{e})$, $b(\underline{e})$, $c(\underline{x},\underline{e})$, $d(\underline{x},\underline{e})$ such that

$$\mathbb{P}_{A|E}(\underline{x}|\underline{e})(t) = \frac{\mathbb{P}_{A,E}(\underline{x},\underline{e})}{\mathbb{P}_E(\underline{e})} = \frac{ct+d}{at+b}.$$

The Optimality of Proportional Scaling Consider one of the conditional probability distributions $(\theta_{j1l},\ldots,\theta_{jk_jl})$ and suppose that θ_{j1l} is to be altered to a different value, denoted by $\widetilde{\theta}_{jl1}$. Under *proportional scaling*, the probabilities of the other states are given by

$$\widetilde{\theta}_{jil} = \frac{1-\widetilde{\theta}_{j1l}}{1-\theta_{j1l}}\theta_{jil} \qquad i = 2,\ldots,k_j.$$

This is clearly proportional scaling with

$$t^{(jl)} = \frac{1-\widetilde{\theta}_{j1l}}{1-\theta_{j1l}}$$

$$\beta_{jil} = \theta_{jil} \qquad i = 2,\ldots,k_j, \qquad \beta_{j1l} = \theta_{j1l} - 1$$

$$\alpha_{jil} = 0 \qquad i = 2,\ldots,k_j, \qquad \alpha_{j1l} = 1.$$

Proportional scaling turns out to be *optimal* under the Chan - Darwiche distance measure.

Theorem 13.5. *Consider a probability distribution $\mathbb{P}$ factorised according to a DAG $\mathcal{G}$. Suppose the value θ_{j1l} is changed to $\widetilde{\theta}_{j1l}$. Among the class of probability distributions $\mathcal{Q}$ factorised along $\mathcal{G}$ with $\mathbb{Q}_{X_j|Pa_j}(x_j^{(1)}|\pi_j^{(l)}) = \widetilde{\theta}_{j1l}$, $\min_{\mathbb{Q}\in\mathcal{Q}} D_{CD}(\mathbb{P},\mathbb{Q})$ is obtained for $\mathbb{Q}$ such that $\widetilde{\theta}_{a.b} = \theta_{a.b}$ for all $(a,b) \neq (j,l)$ and*

$$\widetilde{\theta}_{jil} = \frac{1-\widetilde{\theta}_{j1l}}{1-\theta_{j1l}}\theta_{jil}.$$

Under proportional scaling, the Chan - Darwiche distance is then given by

$$D_{CD}(\mathbb{P},\mathbb{Q}) = |\ln\widetilde{\theta}_{j1l} - \ln\theta_{j1l}| + |\ln(1-\widetilde{\theta}_{j1l}) - \ln(1-\theta_{j1l})|.$$

Proof Let $\mathbb{P}$ be a distribution that factorises along a DAG $\mathcal{G}$, with conditional probabilities $\theta_{aib} = \mathbb{P}_{X_a|\mathrm{Pa}_a}(x_a^{(i)}|\pi_a^{(b)})$. Let $\mathbb{Q}$ denote the distribution that factorises along $\mathcal{G}$, with conditional probabilities

$$\widetilde{\theta}_{aib} = \mathbb{Q}_{X_a|\mathrm{Pa}_a}(x_a^{(i)}|\pi_a^{(b)}),$$

where $\widetilde{\theta}_{j1b}$ is given,

$$\widetilde{\theta}_{aib} = \theta_{aib} \qquad (a,b) \neq (j,l)$$

and

$$\widetilde{\theta}_{jil} = \frac{1-\widetilde{\theta}_{j1l}}{1-\theta_{j1l}}\theta_{jil} \qquad i = 2,\ldots,k_j.$$

This is the distribution generated by the proportional scheme. Let $\mathbb{R}$ denote any other probability belonging to class $\mathcal{Q}$.

If $\theta_{j1l} = 1$ and $\widetilde{\theta}_{j1l} < 1$, then there is a $\widetilde{\theta}_{jkl} > 0$ with $\theta_{jkl} = 0$ and it follows that $D_{CD}(\mathbb{P},\mathbb{Q}) = D_{CD}(\mathbb{P},\mathbb{R}) = +\infty$.

If $\theta_{j1l} = 0$ and $\widetilde{\theta}_{j1l} > 0$ then, similarly, it follows directly that $D_{CD}(\mathbb{P},\mathbb{Q}) = D_{CD}(\mathbb{P},\mathbb{R}) = +\infty$.

Consider $0 < \theta_{j1l} < 1$. Firstly, consider $\widetilde{\theta}_{j1l} > \theta_{j1l}$. Then

$$\max_{x\in\mathcal{X}} \frac{\mathbb{Q}(x)}{\mathbb{P}(x)} = \max\left(\frac{\widetilde{\theta}_{j1l}}{\theta_{j1l}}, \frac{1-\widetilde{\theta}_{j1l}}{1-\theta_{j1l}}\right) = \frac{\widetilde{\theta}_{j1l}}{\theta_{j1l}}$$

and

$$\min_{x\in\mathcal{X}} \frac{\mathbb{Q}(x)}{\mathbb{P}(x)} = \frac{1-\widetilde{\theta}_{j1l}}{1-\theta_{j1l}}.$$

Similarly, if $\widetilde{\theta}_{j1l} < \theta_{j1l}$, then $\max_{x\in\mathcal{X}} \frac{\mathbb{Q}(x)}{\mathbb{P}(x)} = \frac{1-\widetilde{\theta}_{j1b}}{1-\theta_{j1b}}$ and $\min_{x\in\mathcal{X}} \frac{\mathbb{Q}(x)}{\mathbb{P}(x)} = \frac{\widetilde{\theta}_{j1l}}{\theta_{j1l}}$, so

$$D_{CD}(\mathbb{P},\mathbb{Q}) = |\ln\widetilde{\theta}_{j1l} - \ln\theta_{j1l}| + |\ln(1-\widetilde{\theta}_{j1l}) - \ln(1-\theta_{j1l})|.$$

Let $\mathbb{R}$ denote any other distribution that factorises along $\mathcal{G}$ with $\mathbb{R}_{X_j|\mathrm{Pa}_j}(x_j^{(1)}|\pi_j^{(l)}) = \widetilde{\theta}_{j1l}$. The next task is to prove that $D_{CD}(\mathbb{P},\mathbb{R}) \geq D_{CD}(\mathbb{P},\mathbb{Q})$.

$\mathbb{P}$ and $\mathbb{R}$ may be expressed as $\mathbb{P}_{X,Y}$ and $\mathbb{R}_{X,Y}$ where (X,Y) are two sets of variables. Using $\mathbb{P}_{X,Y} = \mathbb{P}_X\mathbb{P}_{Y|X}$ and $\mathbb{R}_{X,Y} = \mathbb{R}_X\mathbb{R}_{Y|X}$ and (x^*,y^*) and (x_*,y_*) to denote the points where the maxima and minima of the ratios are achieved, it follows that

$$\begin{aligned}
D_{CD}(\mathbb{P}_{X,Y},\mathbb{R}_{X,Y}) &= \ln\frac{\mathbb{P}_{X,Y}(x^*,y^*)}{\mathbb{R}_{X,Y}(x^*,y^*)} - \ln\frac{\mathbb{P}_{X,Y}(x_*,y_*)}{\mathbb{R}_{X,Y}(x_*,y_*)} \\
&= \ln\frac{\mathbb{P}_X(x^*)}{\mathbb{R}_X(x^*)} - \ln\frac{\mathbb{P}_X(x_*)}{\mathbb{R}_X(x_*)} + \ln\frac{\mathbb{P}_{Y|X}(y^*|x^*)}{\mathbb{R}_{Y|X}(y^*|x^*)} - \ln\frac{\mathbb{P}_{Y|X}(y_*|x_*)}{\mathbb{R}_{Y|X}(y_*|x_*)} \\
&\geq D_{CD}(\mathbb{P}_X,\mathbb{R}_X) + \ln\frac{\mathbb{P}_{Y|X}(y^*|x^*)}{\mathbb{R}_{Y|X}(y^*|x^*)} - \ln\frac{\mathbb{P}_{Y|X}(y_*|x_*)}{\mathbb{R}_{Y|X}(y_*|x_*)}.
\end{aligned}$$

Now, because (x^*,y^*) maximises the ratio, it follows that y^* maximises the ratio $\frac{\mathbb{P}_{Y|X}(.|x^*)}{\mathbb{R}_{Y|X}(.|x^*)}$ and hence that $\ln\frac{\mathbb{P}_{Y|X}(y^*|x^*)}{\mathbb{R}_{Y|X}(y^*|x^*)} \geq 1$; similarly, $\ln\frac{\mathbb{P}_{Y|X}(y^*|x^*)}{\mathbb{R}_{Y|X}(y^*|x^*)} \leq 1$, so that

$$D_{CD}(\mathbb{P}_{X,Y}, \mathbb{R}_{X,Y}) \geq D_{CD}(\mathbb{P}_X, \mathbb{R}_X).$$

Now let X denote the set of variables $(X_1, \ldots, X_j)$ and Y the set of variables $(X_{j+1}, \ldots, X_d)$. It follows that

$$D_{CD}(\mathbb{P}, \mathbb{R}) \geq D_{CD}(\mathbb{P}_{X_1,\ldots,X_j}, \mathbb{R}_{X_1,\ldots,X_j})$$

where the notation is clear. Finally, for any $\underline{z}$ corresponding to parent configuration $\pi_j^{(l)}$,

$$\begin{aligned} D_{CD}(\mathbb{P}_{X_1,\ldots,X_j}, \mathbb{R}_{X_1,\ldots,X_j}) &\geq \ln \max_{\underline{x}|(x_1,\ldots,x_{j-1})=\underline{z}} \frac{\mathbb{P}_{X_1,\ldots,X_j}}{\mathbb{R}_{X_1,\ldots,X_j}} - \ln \min_{\underline{x}|(x_1,\ldots,x_{j-1})=\underline{z}} \frac{\mathbb{P}_{X_1,\ldots,X_j}}{\mathbb{R}_{X_1,\ldots,X_j}} \\ &= \ln \max_{\underline{x}|(x_1,\ldots,x_{j-1})=\underline{z}} \frac{\mathbb{P}_{X_j|\mathrm{Pa}_j}}{\mathbb{R}_{X_j|\mathrm{Pa}_j}} - \ln \min_{\underline{x}|(x_1,\ldots,x_{j-1})=\underline{z}} \frac{\mathbb{P}_{X_j|\mathrm{Pa}_j}}{\mathbb{R}_{X_j|\mathrm{Pa}_j}} \\ &= D_{CD}(\theta_{j.l}, \widetilde{\theta}_{j.l}). \end{aligned}$$

□

13.2.1 Query Constraints

Let Y, Z denote two random variables such that $Y \notin \underline{E}$ and $Z \notin \underline{E}$. The query constraints considered in this section are of the following type:

- $$\mathbb{P}_{Y|\underline{E}}(y|\underline{e}) - \mathbb{P}_{Z|\underline{E}}(z|\underline{e}) \geq \epsilon, \tag{13.2}$$

- $$\frac{\mathbb{P}_{Y|\underline{E}}(y|\underline{e})}{\mathbb{P}_{Z|\underline{E}}(z|\underline{e})} \geq \epsilon. \tag{13.3}$$

The notation will be abbreviated by writing: $\mathbb{P}(y|\underline{e})$ when the abbreviation is clear from the context.

Let $\mathbb{P}_{\underline{X}}$ denote the probability function for a collection of variables $\underline{X} = (X_1, \ldots, X_d)$, which may be factorised along a graph $\mathcal{G} = (V, E)$ (where $V = \{X_1, \ldots, X_d\}$), with given conditional probability potentials, $\theta_{jil} = \mathbb{P}_{X_j|\mathrm{Pa}_j}(x_j^{(i)}|\pi_j^{(l)})$. Then

$$\mathbb{P}_{\underline{X}}(\underline{x}) = \prod_{j=1}^{d} \prod_{l=1}^{q_j} \prod_{i=1}^{k_j} \theta_{jil}^{n_j(i,l)},$$

where $n_j(i,l) = 1$ if the child parent configuration $(x_j^{(i)}, \pi_j^{(l)})$ appears in $\underline{x}$ and 0 otherwise. Suppose that the probabilities $(\theta_{j1l}, \ldots, \theta_{j,k_j,l})$ are parametrised by $(t_1^{(jl)}, \ldots, t_{m_j}^{(jl)})$, where $m_j \leq k_j - 1$. The following result holds.

Theorem 13.6. *Let $\underline{X} = (X_1, \ldots, X_d)$ denote a set of variables and let $\mathbb{P}$ be a probability distribution that factorises along a DAG $\mathcal{G}$ with node set $V = \{X_1, \ldots, X_d\}$. Let $\theta_{jil} = \mathbb{P}_{X_j|Pa_j}(x_j^{(i)}|\pi_j^{(l)})$. Suppose that for each (j,l) the probabilities $(\theta_{j1l}, \ldots, \theta_{j,k_j,l})$ are parametrised by $(t_1^{(jl)}, \ldots, t_{m_{jl}}^{(jl)})$ where $m_{jl} \leq k_j - 1$. Let $\underline{E} = (X_{e_1}, \ldots X_{e_m})$ denote a subset of $\underline{X}$ and let $\underline{e} = (x_{e_1}^{(i_1)}, \ldots, x_{e_m}^{(i_m)})$ denote an instantiation of $\underline{E}$. Then for all $1 \leq k \leq m_{jl}$,*

$$\frac{\partial}{\partial t_k^{(jl)}} \mathbb{P}_{\underline{E}}(\underline{e}) = \sum_{i=1}^{k_j} \frac{\mathbb{P}_{\underline{E}, X_j, Pa_j}(\underline{e}, x_j^{(i)}, \pi_j^{(l)})}{\theta_{jil}} \frac{\partial}{\partial t_k^{(jl)}} \theta_{jil}.$$

Proof Firstly,

$$\begin{aligned}
\mathbb{P}_{\underline{E}}(\underline{e}) &= \sum_{il} \mathbb{P}_{\underline{E}|X_j, \mathrm{Pa}_j}(\underline{e}|x_j^{(i)}, \pi_j^{(l)}) p_{X_j|\mathrm{Pa}_j}(x_j^{(i)}|\pi_j^{(l)}) \mathbb{P}_{\mathrm{Pa}_j}(\pi_j^{(l)}) \\
&= \sum_{il} \mathbb{P}_{\underline{E}|X_j, \mathrm{Pa}_j}(\underline{e}|x_j^{(i)}, \pi_j^{(l)}) \theta_{jil} \mathbb{P}_{\mathrm{Pa}_j}(\pi_j^{(l)}).
\end{aligned}$$

It follows that

$$\begin{aligned}
\frac{\partial}{\partial t_k^{(jl)}} \mathbb{P}_{\underline{E}}(\underline{e}) &= \sum_{i=1}^{k_j} \mathbb{P}_{\underline{E}|X_j, \mathrm{Pa}_j}(\underline{e}|x_j^{(i)}, \pi_j^{(l)}) \mathbb{P}_{\mathrm{Pa}_j}(\pi_j^{(l)}) \frac{\partial \theta_{jil}}{\partial t_k^{(jl)}} \\
&= \sum_{i=1}^{k_j} \frac{\mathbb{P}_{X_j, \mathrm{Pa}_j|\underline{E}}(x_j^{(i)}, \pi_j^{(l)}|\underline{e}) \mathbb{P}_{\underline{E}}(\underline{e}) \mathbb{P}_{\mathrm{Pa}_j}(\pi_j^{(l)})}{\mathbb{P}_{X_j, \mathrm{Pa}_j}(x_j^{(i)}, \pi_j^{(l)})} \frac{\partial \theta_{jil}}{\partial t_k^{(jl)}} \\
&= \sum_{i=1}^{k_j} \frac{\mathbb{P}_{X_j, \mathrm{Pa}_j, \underline{E}}(x_i^{(j)}, \pi_l^{(j)}, \underline{e})}{\mathbb{P}_{X_j|\mathrm{Pa}_j}(x_j^{(i)}|\pi_j^{(l)})} \frac{\partial \theta_{jil}}{\partial t_k^{(jl)}} \\
&= \sum_{i=1}^{k_j} \frac{\mathbb{P}_{X_j, \mathrm{Pa}_j, \underline{E}}(x_j^{(i)}, \pi_j^{(l)}, \underline{e})}{\theta_{jil}} \frac{\partial \theta_{jil}}{\partial t_k^{(jl)}}
\end{aligned}$$

as required. □

Proportional Scaling Again, the complete set of variables is $\underline{X} = (X_1, \ldots, X_d)$, with a joint probability distribution $\mathbb{P}$ that may be factorised along a Directed Acyclic Graph $\mathcal{G}$. Evidence is received on a subset of the variables $\underline{E} = (X_{e_1}, \ldots, X_{e_m})$. Consider a *proportional scaling* scheme, where each conditional probability distribution $(\theta_{j1l}, \ldots, \theta_{jk_jl})$ has exactly one parameter. Under proportional scaling, this may be represented as $\theta_{j1l} = t^{(jl)}$ and there are non negative numbers $a_2^{(jl)}, \ldots, a_{k_j}^{(jl)}$ satisfying $\sum_{\alpha=2}^{k_j} a_\alpha^{(jl)} = 1$, such that

$$\begin{aligned}
\theta_{j1l} &= t^{(jl)} \\
\theta_{j\alpha l} &= a_\alpha^{(jl)}(1 - t^{(jl)}), \qquad \alpha = 2, \ldots, k_j.
\end{aligned}$$

Then, an application of Theorem 13.6 in the simplified setting of proportional scaling immediately gives

$$\frac{\partial}{\partial t^{(jl)}}\mathbb{P}_{\underline{E}}(\underline{e}) = \frac{\mathbb{P}_{\underline{E},X_j,\mathrm{Pa}_j}(\underline{e},x_j^{(1)},\pi_j^{(l)})}{\theta_{j1l}} - \sum_{\alpha=2}^{k_j}\frac{\mathbb{P}_{\underline{E},X_j,\mathrm{Pa}_j}(\underline{e},x_j^{(\alpha)},\pi_j^{(l)})}{\theta_{j\alpha l}}a_\alpha^{(jl)}. \tag{13.4}$$

When a proportional scaling scheme is used, Theorem 13.4 gives

$$\mathbb{P}_{\underline{E}}(\underline{e}) = \alpha + \beta t^{(jl)},$$

where α and β do not depend on $t^{(jl)}$. It follows that for any $t^{(jl)}$, $\frac{\partial}{\partial t^{(jl)}}\mathbb{P}_{\underline{E}}(\underline{e}) = \beta$, where β is constant (i.e. it does not depend on $t^{(jl)}$). This observation makes it straight forward, under proportional scaling, to find the necessary change in a single parameter $t^{(jl)}$ (if such a parameter change is possible) to enforce a query constraint.

13.2.2 Binary Variables

Assume that variable X_j is *binary*, with $\mathbb{P}_{X_j|\mathrm{Pa}_j}(x_j^{(1)}|\pi_j^{(l)}) = t^{(jl)}$ and $\mathbb{P}_{X_j|\mathrm{Pa}_j}(x_j^{(0)}|\pi_j^{(l)}) = 1 - t^{(jl)}$. Then Equation (13.4) reduces to:

$$\frac{\partial}{\partial t^{(jl)}}\mathbb{P}_{\underline{E}}(\underline{e}) = \frac{\mathbb{P}_{\underline{E},X_j,\mathrm{Pa}_j}(\underline{e},x_j^{(1)},\pi_j^{(l)})}{t^{(jl)}} - \frac{\mathbb{P}_{\underline{E},X_j,\mathrm{Pa}_j}(\underline{e},x_j^{(0)},\pi_j^{(l)})}{1-t^{(jl)}}. \tag{13.5}$$

The statement $Y = y, \underline{E} = \underline{e}$ may be treated as hard evidence. By Theorem 13.4, it follows that there are real numbers λ, λ_y and λ_z such that

$$\lambda = \frac{\partial}{\partial t^{(jl)}}\mathbb{P}_{\underline{E}}(\underline{e}) = \frac{\mathbb{P}_{\underline{E},X_j,\mathrm{Pa}_j}(\underline{e},x_j^{(1)},\pi_j^{(l)})}{t^{(jl)}} - \frac{\mathbb{P}_{\underline{E},X_j,\mathrm{Pa}_j}(\underline{e},x_j^{(0)},\pi_j^{(l)})}{1-t^{(jl)}},$$

$$\lambda_y = \frac{\partial}{\partial t^{(jl)}}\mathbb{P}_{Y,\underline{E}}(y,\underline{e}) = \frac{\mathbb{P}_{Y,\underline{E},X_j,\mathrm{Pa}_j}(y,\underline{e},x_j^{(1)},\pi_j^{(l)})}{t^{(jl)}} - \frac{\mathbb{P}_{Y,\underline{E},X_j,\mathrm{Pa}_j}(y,\underline{e},x_j^{(0)},\pi_j^{(l)})}{1-t^{(jl)}}$$

and

$$\lambda_z = \frac{\partial}{\partial t^{(jl)}}\mathbb{P}_{Z,\underline{E}}(z,\underline{e}) = \frac{\mathbb{P}_{Z,\underline{E},X_j,\mathrm{Pa}_j}(z,\underline{e},x_j^{(1)},\pi_j^{(l)})}{t^{(jl)}} - \frac{\mathbb{P}_{Z,\underline{E},X_j,\mathrm{Pa}_j}(z,\underline{e},x_j^{(0)},\pi_j^{(l)})}{1-t^{(jl)}}.$$

The following is a corollary of Theorem 13.6, which reduces to Equation (13.5) for the binary case.

Corollary 13.7. *To satisfy the constraint given by Equation* (13.2)*, the parameter* $t^{(jl)}$ *has to be changed to* $t^{(jl)} + \delta$*,where* δ *satisfies*

$$\mathbb{P}_{Y,\underline{E}}(y,\underline{e}) - \mathbb{P}_{Z,\underline{E}}(z,\underline{e}) - \epsilon\mathbb{P}_{\underline{E}}(\underline{e}) \geq \delta(-\lambda_y + \lambda_z + \epsilon\lambda). \tag{13.6}$$

To satisfy the constraint given by Equation (13.3)*, the parameter* $t^{(jl)}$ *has to be changed to* $t^{(jl)} + \delta$*, where*

$$\mathbb{P}_{Y,\underline{E}}(y,\underline{e}) - \epsilon\mathbb{P}_{Z,\underline{E}}(z,\underline{e}) \geq \delta(-\lambda_y + \epsilon\lambda_z). \tag{13.7}$$

Proof Since $\mathbb{P}_{Y|\underline{E}}(y|\underline{e}) = \frac{\mathbb{P}_{Y,\underline{E}}(y,\underline{e})}{\mathbb{P}_{\underline{E}}(\underline{e})}$, it follows that $\mathbb{P}_{Y|\underline{E}}(y|\underline{e}) - \mathbb{P}_{Z|\underline{E}}(z|\underline{e}) \geq \epsilon$ is equivalent to $\mathbb{P}_{Y,\underline{E}}(y,\underline{e}) - \mathbb{P}_{Z,\underline{E}}(z,\underline{e}) \geq \epsilon\mathbb{P}_{\underline{E}}(\underline{e})$. A change in the constraint changes $\mathbb{P}_{Y,\underline{E}}(y,\underline{e})$, $\mathbb{P}_{Z,\underline{E}}(z,\underline{e})$ and $\mathbb{P}_{\underline{E}}(\underline{e})$ to $\mathbb{P}_{Y,\underline{E}}(y,\underline{e}) + \delta\lambda_y$, $\mathbb{P}_{Z,\underline{E}}(z,\underline{e}) + \delta\lambda_z$ and $\mathbb{P}_{\underline{E}}(\underline{e}) + \delta\lambda$ respectively. To enforce the difference constraint, it follows that δ satisfies

$$(\mathbb{P}_{Y,\underline{E}}(y,\underline{e}) + \lambda_y\delta) - (\mathbb{P}_{Z,\underline{E}}(z,\underline{e}) + \lambda_z\delta) \geq \epsilon(\mathbb{P}_{\underline{E}}(\underline{e}) + \lambda\delta).$$

Equation (13.6) follows directly.

Similarly, to enforce the ratio constraint, the following inequality is required:

$$\frac{\mathbb{P}_{Y,\underline{E}}(y,\underline{e}) + \lambda_y\delta}{\mathbb{P}_{Z,\underline{E}}(z,\underline{e}) + \lambda_z\delta} \geq \epsilon.$$

Equation (13.7) now follows directly and the proof is complete. □

13.3 The Sensitivity of Queries to Parameter Changes

In line with the Chan - Darwiche distance measure, sensitivity is defined in the following way.

Definition 13.8 (Sensitivity). *Let $\mathbb{P}$ denote a parametrised family of probability distributions, over a finite, discrete state space $\mathcal{X}$, parametrised by k parameters $(\theta_1, \ldots, \theta_k) \in \widetilde{\Theta}$, where $\widetilde{\Theta} \subseteq \mathbf{R}^k$ denotes the parameter space. Let $\mathbb{P}^{(\theta_1,\ldots,\theta_k)}(.)$ denote the probability function over $\mathcal{X}$ when the parameters are fixed at $\theta_1, \ldots, \theta_k$. Then the sensitivity of $\mathbb{P}$ to parameter θ_j is defined as*

$$S_j(\mathbb{P})(\theta_1, \ldots, \theta_k) = \max_{\underline{x}\in\mathcal{X}} \frac{\partial}{\partial\theta_j} \ln \mathbb{P}^{(\theta_1,\ldots,\theta_k)}(\underline{x}) - \min_{\underline{x}\in\mathcal{X}} \frac{\partial}{\partial\theta_j} \ln \mathbb{P}^{(\theta_1,\ldots,\theta_k)}(\underline{x}).$$

Example 13.9.

If $\mathbb{P}$ is a family of binary variables, with state space $\mathcal{X} = \{x_0, x_1\}$ and a single parameter θ, then

$$S(\mathbb{P})(\theta) = \left| \frac{\partial}{\partial\theta} \ln \frac{\mathbb{P}^{(\theta)}(x_1)}{\mathbb{P}^{(\theta)}(x_0)} \right|.$$

□

This section restricts attention to a single parameter model. Consider a network with d variables, $\underline{X} = (X_1, \ldots, X_d)$ where one particular variable X_j is a binary variable. The other variables may be multivalued. Let

$$t^{(jl)} = \mathbb{P}_{X_j|\mathrm{Pa}_j}(x_j^{(1)}|\pi_j^{(l)}).$$

Let $\underline{Y}$ denote a collection of variables, taken from $(X_1, \ldots, X_n)$ and let $\underline{Y} = \underline{y}$ denote an instantiation of these variables. Let $\underline{y}$ denote the event $\{\underline{Y} = \underline{y}\}$ and let $\underline{y}^c$ denote the event $\{\underline{Y} \neq \underline{y}\}$. Similarly, let $\underline{e}$ denote the event $\{\underline{E} = \underline{e}\}$, where $\underline{E}$ is a different sub-collection of variables from $\underline{X}$. From Definition 13.8, the sensitivity of a query $\mathbb{P}(\underline{y}|\underline{e})$ to the parameter $t^{(jl)}$ is defined as

$$\left| \frac{\partial}{\partial t^{(jl)}} \ln \frac{\mathbb{P}(\underline{y}|\underline{e})}{\mathbb{P}(\underline{y}^c|\underline{e})} \right|.$$

The following theorem provides a simple bound on the derivative in terms of $\mathbb{P}(y|\underline{e})$ and $t^{(jl)}$ only.

Theorem 13.10. *Suppose X_j is a binary variable taking values $x_j^{(1)}$ or $x_j^{(0)}$. Set*

$$t^{(jl)} = \mathbb{P}_{X_j|Pa_j}(x_j^{(1)}|\pi_j^{(l)}).$$

Then

$$\left|\frac{\partial}{\partial t^{(jl)}}\mathbb{P}(\underline{y}|\underline{e})\right| \leq \frac{\mathbb{P}(\underline{y}|\underline{e})(1-\mathbb{P}(\underline{y}|\underline{e}))}{t^{(jl)}(1-t^{(jl)})}. \tag{13.8}$$

The example given after the proof shows that this bound is *sharp*; there are situations where the derivative assumes the bound *exactly*.

Proof of Theorem 13.10 Firstly, $\mathbb{P}(\underline{y}|\underline{e}) = \frac{\mathbb{P}(\underline{y},\underline{e})}{\mathbb{P}(\underline{e})}$, so that

$$\frac{\partial}{\partial t^{(jl)}}\mathbb{P}(\underline{y}|\underline{e}) = \frac{1}{\mathbb{P}(\underline{e})}\frac{\partial}{\partial t^{(jl)}}\mathbb{P}(\underline{y},\underline{e}) - \frac{\mathbb{P}(\underline{y},\underline{e})}{\mathbb{P}^2(\underline{e})}\frac{\partial}{\partial t^{(jl)}}\mathbb{P}(\underline{e}).$$

Using this, Equation (13.5) gives

$$\begin{aligned}
&\frac{\partial}{\partial t^{(jl)}}\mathbb{P}(\underline{y}|\underline{e}) \\
&\quad = \frac{\left\{(1-t^{(jl)})\mathbb{P}(\underline{y},x_j^{(1)},\pi_j^{(l)}|\underline{e}) - t^{(jl)}\mathbb{P}(\underline{y},x_j^{(0)},\pi_j^{(l)}|\underline{e})\right\}}{t^{(jl)}(1-t^{(jl)})} \qquad (13.9)\\
&\qquad - \frac{\left\{(1-t^{(jl)})\mathbb{P}(\underline{y}|\underline{e})\mathbb{P}_{X_j,\mathrm{Pa}_j|\underline{E}}(x_j^{(1)}\pi_j^{(l)}|\underline{e}) - t^{(jl)}\mathbb{P}(\underline{y}|\underline{e})\mathbb{P}_{X_j,\mathrm{Pa}_j|\underline{E}}(x_j^{(0)},\pi_j^{(l)}|\underline{e})\right\}}{t^{(jl)}(t-t^{(jl)})} \\
&\quad = \frac{(1-t^{(jl)})(\mathbb{P}_{\underline{Y},X_j,\mathrm{Pa}_j|\underline{E}}(\underline{y},x_j^{(1)},\pi_j^{(l)}|\underline{e}) - \mathbb{P}(\underline{y}|\underline{e})\mathbb{P}_{X_j,\mathrm{Pa}_j|\underline{E}}(x_j^{(1)}\pi_j^{(l)}|\underline{e}))}{t^{(jl)}(1-t^{(jl)})} \\
&\qquad - \frac{t^{(jl)}(\mathbb{P}_{\underline{Y},X_j,\mathrm{Pa}_j|\underline{E}}(\underline{y},x_j^{(0)},\pi_j^{(l)}|\underline{e}) - \mathbb{P}(\underline{y}|\underline{e})\mathbb{P}_{X_j,\mathrm{Pa}_j|\underline{E}}(x_j^{(0)}\pi_j^{(l)}|\underline{e}))}{t^{(jl)}(t-t^{(jl)})}. \qquad (13.10)
\end{aligned}$$

With the shorthand notation $\underline{y}^c$ to denote the event $\{\underline{Y} \neq \underline{y}\}$,

$$\begin{aligned}
&\mathbb{P}_{X_j,\mathrm{Pa}_j,\underline{Y}|\underline{E}}(x_j^{(1)},\pi_j^{(l)},\underline{y}|\underline{e}) - \mathbb{P}(\underline{y}|\underline{e})\mathbb{P}_{X_j,\mathrm{Pa}_j|\underline{E}}(x_j^{(1)},\pi_j^{(l)}|\underline{e}) \\
&\quad \leq \mathbb{P}_{X_j,\mathrm{Pa}_j,\underline{Y}|\underline{E}}(x_j^{(1)},\pi_j^{(l)},\underline{y}|\underline{e}) - \mathbb{P}_{\underline{Y}|\underline{E}}(\underline{y}|\underline{e})\mathbb{P}_{X_j,\mathrm{Pa}_j,\underline{Y}|\underline{E}}(x_j^{(1)},\pi_j^{(l)},\underline{y}|\underline{e}) \\
&\quad = \mathbb{P}_{X_j,\mathrm{Pa}_j,\underline{Y}|\underline{E}}(x_j^{(1)},\pi_j^{(l)},\underline{y}|\underline{e})(1-\mathbb{P}(\underline{y}|\underline{e}) \\
&\quad \leq \mathbb{P}(\underline{y}|\underline{e})(1-\mathbb{P}(\underline{y}|\underline{e}))
\end{aligned}$$

and

$$\begin{aligned}
&\mathbb{P}(\underline{y}|\underline{e})\mathbb{P}_{X_j,\mathrm{Pa}_j|\underline{E}}(x_j^{(1)},\pi_j^{(l)}|\underline{e}) - \mathbb{P}_{X_j,\mathrm{Pa}_j,\underline{Y}|\underline{E}}(x_j^{(1)},\pi_j^{(l)},\underline{y}|\underline{e}) \\
&\quad = (1-\mathbb{P}(\underline{y}^c|\underline{e}))\mathbb{P}_{X_j,\mathrm{Pa}_j|\underline{E}}(x_j^{(1)},\pi_j^{(l)}|\underline{e}) \\
&\qquad -\mathbb{P}_{X_j,\mathrm{Pa}_j|\underline{E}}(x_j^{(1)},\pi_j^{(l)}|\underline{e}) + \mathbb{P}_{X_j,\mathrm{Pa}_j,\underline{Y}|\underline{E}}(x_j^{(1)},\pi_j^{(l)},\underline{y}^c|\underline{e}) \\
&\quad = \mathbb{P}_{X_j,\mathrm{Pa}_j,\underline{Y}|\underline{E}}(x_j^{(1)},\pi_j^{(l)},\underline{y}^c|\underline{e}) - \mathbb{P}(\underline{y}^c|\underline{e})\mathbb{P}_{X_j,\mathrm{Pa}_j|\underline{E}}(x_j^{(1)},\pi_j^{(l)}|\underline{e}) \\
&\quad = \mathbb{P}_{X_j,\mathrm{Pa}_j,\underline{Y}|\underline{E}}(x_j^{(1)},\pi_j^{(l)},\underline{y}^c|\underline{e})(1-\mathbb{P}(\underline{y}^c|\underline{e})) \\
&\quad \leq \mathbb{P}(\underline{y}^c|\underline{e})(1-\mathbb{P}(\underline{y}^c|\underline{e})) \\
&\quad = (1-\mathbb{P}(\underline{y}|\underline{e}))\mathbb{P}(\underline{y}|\underline{e})
\end{aligned}$$

From this, it follows directly from Equation (13.10) that

$$\left|\frac{\partial}{\partial t^{(jl)}}\mathbb{P}(\underline{y}|\underline{e})\right| \leq \frac{\mathbb{P}(\underline{y}|\underline{e})(1-\mathbb{P}(\underline{y}|\underline{e}))}{t^{(jl)}(1-t^{(jl)})}.$$

The proof of Theorem 13.10 is complete. □

Corollary 13.11. *The sensitivity of* $\mathbb{P}(\underline{y}|\underline{e})$ *to the parameter* $t^{(jl)}$ *is bounded by*

$$\left|\frac{\partial}{\partial t^{(jl)}}\ln\frac{\mathbb{P}(\underline{y}|\underline{e})}{\mathbb{P}(\underline{y}^c|\underline{e})}\right| \leq \frac{1}{t^{(jl)}(1-t^{(jl)})}. \tag{13.11}$$

Proof Immediate. □

It is clear that the *worst* situation from a robustness point of view arises when the parameter value $t^{(jl)}$ is close to either 0 or 1, while the query takes values that are close to neither 0 nor 1.

Example 13.12.

This example shows that the bounds given by inequalities (13.8) and (13.11) are sharp, in the sense that there are examples where the bounds are attained. Consider the network given in Figure 13.2, where X and Y are binary variables taking values from (x_0, x_1) and (y_0, y_1) respectively. $\mathbb{P}_X(x_0) = \theta_x$ and $\mathbb{P}_Y(y_0) = \theta_y$. Suppose that E is a *deterministic* binary variable; that is, $\mathbb{P}(\{E = e\}|\{X = Y\}) = 1$ and $\mathbb{P}(\{E = e\}|\{X \neq Y\}) = 0$.

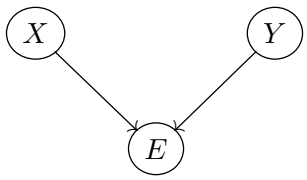

Figure 13.2: The Network Used in Example 13.12

The probability potentials are

$$\mathbb{P}_X = \begin{array}{cc} x_0 & x_1 \\ \hline \theta_x & 1-\theta_x \end{array} \qquad \mathbb{P}_Y = \begin{array}{cc} y_0 & y_1 \\ \hline \theta_y & 1-\theta_y \end{array}$$

$$\mathbb{P}_{E|X,Y}(e|.,.) = \begin{array}{c|cc} X\backslash Y & y_0 & y_1 \\ \hline x_0 & 1 & 0 \\ x_1 & 0 & 1 \end{array}$$

from which it follows that

$$\mathbb{P}_{Y|E}(y_0|e) = \frac{\mathbb{P}_{Y,E}(y_0,e)}{\mathbb{P}_E(e)} = \frac{\mathbb{P}_Y(y_0)\sum_x \mathbb{P}_X(x)\mathbb{P}_{E|X,Y}(e|x,y_0)}{\sum_{x,y}\mathbb{P}_X(x)\mathbb{P}_Y(y)\mathbb{P}_{E|X,Y}(e|x,y)} = \frac{\theta_y\theta_x}{\theta_y\theta_x + (1-\theta_y)(1-\theta_x)}$$

and

$$\frac{\partial}{\partial\theta_x}\mathbb{P}_{Y|E}(y_0|e) = \frac{\theta_y(1-\theta_y)}{(\theta_x\theta_y + (1-\theta_x)(1-\theta_y))^2}$$

while

$$\frac{\mathbb{P}_{Y|E}(y_0|e)(1-\mathbb{P}_{Y|E}(y_0|e))}{\theta_x(1-\theta_x)} = \frac{\theta_y\theta_x(1-\theta_y)(1-\theta_x)}{(\theta_x\theta_y + (1-\theta_x)(1-\theta_y))^2\theta_x(1-\theta_x)} = \frac{\theta_y(1-\theta_y)}{(\theta_x\theta_y + (1-\theta_x)(1-\theta_y))^2},$$

so that

$$\frac{\partial}{\partial\theta_x}\mathbb{P}_{Y|E}(y_0|e) = \frac{\theta_y(1-\theta_y)}{(\theta_x\theta_y + (1-\theta_x)(1-\theta_y))^2}$$

showing that the bound (13.8) is achieved. □

For the bound (13.11), note from the above that

$$\frac{\partial}{\partial\theta_x}\mathbb{P}_{Y|E}(y_0|e) = \frac{\mathbb{P}_{Y|E}(y_0|e)\mathbb{P}_{Y|E}(y_1|e)}{\theta_x(1-\theta_x)}$$

so that

$$\frac{\partial}{\partial\theta_x}\ln\mathbb{P}_{Y|E}(y_0|e) = \frac{\mathbb{P}_{Y|E}(y_1|e)}{\theta_x(1-\theta_x)}$$

and, because $\mathbb{P}_{Y|E}(y_0|e) + \mathbb{P}_{Y|E}(y_1|e) = 1$,

$$\frac{\partial}{\partial\theta_x}\mathbb{P}_{Y|E}(y_1|e) = -\frac{\partial}{\partial\theta_x}\mathbb{P}_{Y|E}(y_0|e) = -\frac{\mathbb{P}_{Y|E}(y_0|e)\mathbb{P}_{Y|E}(y_1|e)}{\theta_x(1-\theta_x)}$$

so that

$$\frac{\partial}{\partial\theta_x}\ln\frac{\mathbb{P}_{Y|E}(y_0|e)}{\mathbb{P}_{Y|E}(y_1|e)} = \frac{1}{\theta_x(1-\theta_x)},$$

so that equality is achieved in bound (13.11). □

The following results bound the *odds*.

Theorem 13.13. *Let $\mathbb{P}$ be a parametrised family of probability distributions, factorised along the same DAG, with a single parameter θ. Let X_j be a binary variable and let $\theta = \mathbb{P}^{(\theta)}_{X_j|Pa_j}(x_j^{(0)}|\pi_j^{(l)})$; all the other CPPs remain fixed and let $O_\theta = \frac{\theta}{1-\theta}$. Consider a parameter change from $\theta = t$ to $\theta = s$. Note that $O_t = \frac{t}{1-t}$ and $O_s = \frac{s}{1-s}$. Let $\mathbb{P}^{(\theta)}(\underline{y}|\underline{e})$ denote the probability value of a query when θ is the parameter value. Let $\widetilde{O}_\theta(\underline{y}|\underline{e}) = \frac{\mathbb{P}^{(\theta)}(\underline{y}|\underline{e})}{1-\mathbb{P}^{(\theta)}(\underline{y}|\underline{e})}$. Then*

$$\frac{O_t}{O_s} \leq \frac{\widetilde{O}_s(\underline{y}|\underline{e})}{\widetilde{O}_t(\underline{y}|\underline{e})} \leq \frac{O_s}{O_t} \qquad s \geq t$$

$$\frac{O_s}{O_t} \leq \frac{\widetilde{O}_s(\underline{y}|\underline{e})}{\widetilde{O}_t(\underline{y}|\underline{e})} \leq \frac{O_t}{O_s} \qquad t \leq s.$$

This gives the bound

$$\left|\ln \widetilde{O}_s(\underline{y}|\underline{e}) - \ln \widetilde{O}_t(\underline{y}|\underline{e})\right| \leq |\ln O_s - \ln O_t|.$$

Proof Let x denote the probability of the query $\mathbb{P}(\underline{y}|\underline{e})$ when the value of the parameter $t^{(jl)}$ is z. Note that, for $0 < a \leq b < 1$,

$$\int_a^b \frac{dx}{x(1-x)} = \int_a^b \frac{dx}{x} + \int_a^b \frac{dx}{1-x} = \ln \frac{b}{a}\frac{1-a}{1-b}.$$

Then, for $t^{(jl)} \leq s^{(jl)}$, Equation (13.8) gives

$$-\int_t^s \frac{dz}{z(1-z)} \leq \int_{\mathbb{P}_t(\underline{y}|\underline{e})}^{\mathbb{P}_s(\underline{y}|\underline{e})} \frac{dx}{x(1-x)} \leq \int_t^s \frac{dz}{z(1-z)},$$

so that

$$-\ln \frac{s}{t}\frac{1-t}{1-s} \leq \ln \frac{\mathbb{P}_s(\underline{y}|\underline{e})}{\mathbb{P}_t(\underline{y}|\underline{e})}\frac{1-\mathbb{P}_s(\underline{y}|\underline{e})}{1-\mathbb{P}_t(\underline{y}|\underline{e})} \leq \ln \frac{s}{t}\frac{1-t}{1-s}$$

giving immediately that

$$\frac{O_t}{O_s} \leq \frac{\widetilde{O}_s(\underline{y}|\underline{e})}{\widetilde{O}_t(\underline{y}|\underline{e})} \leq \frac{O_s}{O_t}.$$

For $s \leq t$ the argument is similar and gives

$$\frac{O_s}{O_t} \leq \frac{\widetilde{O}_s(\underline{y}|\underline{e})}{\widetilde{O}_t(\underline{y}|\underline{e})} \leq \frac{O_t}{O_s}.$$

In both cases

$$\left|\ln \widetilde{O}_s(\underline{y}|\underline{e}) - \ln \widetilde{O}_t(\underline{y}|\underline{e})\right| \leq |\ln O_s - \ln O_t|$$

and the result follows. □

Notes The observation that the probability of evidence is a linear function of any single parameter in the model and hence that the conditional probability is the ratio of two linear functions is due to Castillo, Gutiérrez and Hadi (1997) [12] and [13]. The most significant developments in sensitivity analysis, which comprise practically the whole chapter, were introduced by Chan and Darwiche in the article [21] (2002) and developed in the articles [20] (2005) and article [22].

13.4 Exercises

1. Consider a Bernoulli trial, with probability function $\mathbb{P}_X(.|t)$ defined by

$$\mathbb{P}_X(x|t) = t^x(1-t)^{1-x}, \qquad x = 0, 1, \qquad t \in [0, 1].$$

Recall the definition of sensitivity, Definition 13.8. Compute the sensitivity with respect to the parameter t.

2. Consider the 'fire' example given in the text. Suppose that the evidence is $(R, S) = (t, f)$. Let $\mathbb{P}_T(t) = \theta$ be a variable parameter so that $\mathbb{P}_T(f) = 1-\theta$ and suppose that all the other probabilities are fixed, according to the values given. From an initial value $\theta_0 = 0.02$, compute the lower bound for the change δ required to satisfy the query constraint

$$\frac{\mathbb{P}_{T|R,S}(t|t, f)}{\mathbb{P}_{F|R,S}(t|t, f)} \geq 10$$

corresponding to Corollary 13.7 and express the probabilities needed in terms of the conditional probabilities given. This represents the constraint that, given the report without smoke, it is 10 times more likely that the alarm has been tampered with than that there is a real fire.

3. Consider a probability distribution

$$\mathbb{P}_{X,Y,E} = \mathbb{P}_X \mathbb{P}_Y \mathbb{P}_{E|X,Y}$$

where X, Y, E are all binary variables and

$$\mathbb{P}_X = \begin{array}{cc} x_0 & x_1 \\ \hline \theta_x & 1-\theta_x \end{array} \qquad \mathbb{P}_Y = \begin{array}{cc} y_0 & y_1 \\ \hline \theta_y & 1-\theta_y \end{array}$$

$$\mathbb{P}_{E|X,Y}(e|.,.) = \begin{array}{c|cc} X \backslash Y & y_0 & y_1 \\ \hline x_0 & \alpha & \beta \\ x_1 & \beta & \alpha \end{array}$$

and $\beta < \alpha$.

   (a) Compute $\frac{\partial}{\partial \theta_x} \ln \frac{\mathbb{P}_{Y|E}(y_0|e)}{\mathbb{P}_{Y|E}(y_1|e)}$ and compare the result with the bound from Corollary 13.11.

   (b) Let $O_s(y_0|e) = \frac{\mathbb{P}_{Y|E}(y_0|e)}{\mathbb{P}_{Y|E}(y_1|e)}$ when $\theta_x = s$. Compute $\frac{O_s(y_0|e)}{O_t(y_0|e)}$ and compare with the bounds given by Theorem 13.13.

4. (a) **On Odds and the Weight of Evidence** Let $\mathbb{P}$ be a probability distribution over a space $\mathcal{X}$. The *odds* of an event $A \subseteq \mathcal{X}$ given $B \subseteq \mathcal{X}$ under $\mathbb{P}$, denoted by $O_{\mathbb{P}}(A \mid B)$, is defined as

$$O_{\mathbb{P}}(A \mid B) = \frac{\mathbb{P}(A \mid B)}{\mathbb{P}(A^c \mid B)}. \tag{13.12}$$

The *weight of evidence* E in favour of an event A given B, denoted by $W(A:E \mid B)$, is defined as

$$W(A:E \mid B) = \ln \frac{O_{\mathbb{P}}(A \mid B \cap E)}{O_{\mathbb{P}}(A \mid B)}. \tag{13.13}$$

Show that if $\mathbb{P}(E \cap A^c \cap B) > 0$, then

$$W(A:E \mid B) = \ln \frac{\mathbb{P}(E \mid A \cap B)}{\mathbb{P}(E \mid A^c \cap B)}. \tag{13.14}$$

(b) **On a generalised Odds and the Weight of Evidence** Let $\mathbb{P}$ denote a probability distribution over a space $\mathcal{X}$ and let $H_1 \subseteq \mathcal{X}$, $H_2 \subseteq \mathcal{X}$, $G \subseteq \mathcal{X}$ and $E \subseteq \mathcal{X}$. The *odds of* H_1 *compared to* H_2 given G, denoted by $O_{\mathbb{P}}(H_1/H_2 \mid G)$, is defined as

$$O_{\mathbb{P}}(H_1/H_2 \mid G) = \frac{\mathbb{P}(H_1 \mid G)}{\mathbb{P}(H_2 \mid G)}. \tag{13.15}$$

The *generalised weight of evidence* is defined by

$$W(H_1/H_2 : E \mid G) = \ln \frac{O_{\mathbb{P}}(H_1/H_2 \mid G \cap E)}{O_{\mathbb{P}}(H_1/H_2 \mid G)}. \tag{13.16}$$

Show that if $\mathbb{P}(H_1 \cap G \cap E) > 0$ and $\mathbb{P}(H_2 \cap G \cap E) > 0$ then

$$W(H_1/H_2 : E \mid G) = \ln \frac{\mathbb{P}(E \mid H_1 \cap G)}{\mathbb{P}(E \mid H_2 \cap G)}. \tag{13.17}$$

This is clearly a loglikelihood ratio and these notions are another expression for

$$\text{posterior odds} \;=\; \text{likelihood ratio} \;\times\; \text{prior odds}.$$

13.5 Answers

1.

$$\begin{aligned} S(\mathbb{P})(t) &= \max_{x\in\{0,1\}} \frac{d}{dt}\ln \mathbb{P}_X(x|t) - \min_{x\in\{0,1\}} \frac{d}{dt}\ln \mathbb{P}_X(x|t) \\ &= \max_{x\in\{0,1\}} \left(x\frac{d}{dt}\ln t + (1-x)\frac{d}{dt}\ln(1-t)\right) \\ &\quad - \min_{x\in\{0,1\}} \left(x\frac{d}{dt}\ln t + (1-x)\frac{d}{dt}\ln(1-t)\right) \\ &= \max_{x\in\{0,1\}} \left(\frac{x}{t} - \frac{1-x}{1-t}\right) - \min_{x\in\{0,1\}} \left(\frac{x}{t} - \frac{1-x}{1-t}\right) \\ &= \frac{1}{t} + \frac{1}{1-t} = \frac{1}{t(1-t)}. \end{aligned}$$

2. The parameter is in the variable T, which has no parents; $\mathrm{Pa}_T = \phi$. According to the corollary, it is required to choose δ such that

$$\mathbb{P}_{T,R,S}(t,t,f) - 10\mathbb{P}_{F,R,S}(t,t,f) \geq \delta(-\lambda_T + 10\lambda_F)$$

is required, where

$$\lambda_T = \frac{\mathbb{P}_{R,S,T}(t,f,t)}{\theta_0}$$

since $\mathbb{P}_{R,S,T,R}(t,f,t,f) = 0$,

$$\lambda_F = \frac{\mathbb{P}_{F,R,S,T}(t,t,f,t)}{\theta_0} - \frac{\mathbb{P}_{F,R,S,T}(t,t,f,f)}{1-\theta_0}.$$

The probabilities are obtained by summation:

$$\mathbb{P}_{R,S,T}(t,f,t) = \mathbb{P}_T(t)\sum_{x_f}\mathbb{P}_F(x_f)\mathbb{P}_{S|F}(f|x_f)\sum_{x_a}\mathbb{P}_{A|T,F}(x_a|t,x_f)\sum_{x_l}\mathbb{P}_{L|A}(x_l|x_a)\mathbb{P}_{R|L}(t|x_l).$$

$$\mathbb{P}_{F,R,S}(t,t,f) = \mathbb{P}_F(t)\mathbb{P}_{S|F}(f|t)\sum_{x_t}\mathbb{P}_T(x_t)\sum_{x_a}\mathbb{P}_{A|T,F}(x_a|x_t,t)\sum_{x_l}\mathbb{P}_{L|A}(x_l|x_a)\mathbb{P}_{R|L}(t|x_l)$$

$$\mathbb{P}_{F,R,S,T}(t,t,f,t) = \theta_0\mathbb{P}_F(t)\mathbb{P}_{S|F}(f|t)\sum_{x_a}\mathbb{P}_{A|F,T}(x_a|t,t)\sum_{x_l}\mathbb{P}_{L|A}(x_l|x_a)\mathbb{P}_{R|L}(t|x_l)$$

$$\mathbb{P}_{F,R,S,T}(t,t,f,f) = (1-\theta_0)\mathbb{P}_F(t)\mathbb{P}_{S|F}(f|t)\sum_{x_a}\mathbb{P}_{A|F,T}(x_a|t,f)\sum_{x_l}\mathbb{P}_{L|A}(x_l|x_a)\mathbb{P}_{R|L}(t|x_l)$$

3. (a)

$$\mathbb{P}_{Y|E}(y_0|e) = \frac{\mathbb{P}_{Y,E}(y_0,e)}{\mathbb{P}_E(e)} = \mathbb{P}_Y(y_0)\frac{\sum_{i=0}^1 \mathbb{P}_X(x_i)\mathbb{P}_{X,Y|E}(x_i,y_0|e)}{\sum_{i,j=0}^1 \mathbb{P}_Y(y_j)\mathbb{P}_X(x_i)\mathbb{P}_{X,Y|E}(x_i,y_j|e)}$$

$$\mathbb{P}_{Y|E}(y_0|e) = \frac{(\alpha-\beta)\theta_x\theta_y + \theta_y\beta}{2\theta_x\theta_y(\alpha-\beta) + (\beta-\alpha)(\theta_x+\theta_y) + \alpha}.$$

$$\mathbb{P}_{Y|E}(y_1|e) = \frac{(\alpha-\beta)\theta_x\theta_y + \alpha + \beta\theta_x - \alpha(\theta_x+\theta_y)}{2\theta_x\theta_y(\alpha-\beta) + (\beta-\alpha)(\theta_x+\theta_y) + \alpha}.$$

$$\ln \frac{\mathbb{P}_{Y|E}(y_0|e)}{\mathbb{P}_{Y|e}(y_1|e)} = \ln\left((\alpha-\beta)\theta_x\theta_y + \theta_y\beta\right) - \ln\left((\alpha-\beta)\theta_x\theta_y + \alpha + \beta\theta_x - \alpha(\theta_x+\theta_y)\right)$$

$$\begin{aligned}\frac{\partial}{\partial\theta_x}\ln\frac{\mathbb{P}_{Y|E}(y_0|e)}{\mathbb{P}_{Y|E}(y_1|e)} &= \frac{(\alpha-\beta)}{(\alpha-\beta)\theta_x+\beta} + \frac{(\alpha-\beta)}{\alpha-(\alpha-\beta)\theta_x} \\ &= \frac{1}{\theta_x + \frac{\beta}{\alpha-\beta}} + \frac{1}{\frac{\alpha}{\alpha-\beta}-\theta_x}.\end{aligned}$$

Set $\widetilde{\theta}_x = \frac{\beta}{\alpha-\beta} + \theta_x$, then

$$\frac{\partial}{\partial\theta_x}\ln\frac{\mathbb{P}_{Y|E}(y_0|e)}{\mathbb{P}_{Y|E}(y_1|e)} = \frac{1}{\widetilde{\theta}_x} + \frac{1}{1+\frac{2\beta}{\alpha-\beta}-\widetilde{\theta}}.$$

Clearly, if $\alpha < 1$ or $\beta > 0$,

$$\left|\frac{\partial}{\partial\theta_x}\ln\frac{\mathbb{P}_{Y|E}(y_0|e)}{\mathbb{P}_{Y|E}(y_1|e)}\right| < \frac{1}{\theta_x(1-\theta_x)}.$$

(b)

$$O_s(y_0|e) = \frac{(\alpha-\beta)s\theta_y + \theta_y\beta}{(\alpha-\beta)s\theta_y + \alpha + \beta s - \alpha(s+\theta_y)}$$

so that

$$\begin{aligned}\frac{O_s(y_0|e)}{O_t(y_0|e)} &= \left(\frac{(\alpha-\beta)s+\beta}{(\alpha-\beta)t+\beta}\right)\left(\frac{(\alpha-(\alpha-\beta)t}{(\alpha-(\alpha-\beta)s}\right) \\ &= \left(\frac{s+\frac{\beta}{\alpha-\beta}}{t+\frac{\beta}{\alpha+\beta}}\right)\left(\frac{\frac{\alpha}{\alpha-\beta}-t}{\frac{\alpha}{\alpha-\beta}-s}\right).\end{aligned}$$

For $s < t$, clearly

$$1 \geq \frac{O_s(y_0|e)}{O_t(y_0|e)} \geq \left(\frac{s}{t}\right)\left(\frac{1-t}{1-s}\right)$$

as required. □

4. (a)

$$\begin{aligned}W(A:E|B) &= \ln\frac{O_{\mathbb{P}}(A|BE)}{O_{\mathbb{P}}(A|B)} = \ln\frac{\mathbb{P}(A|BE)}{\mathbb{P}(A^c|BE)}\frac{\mathbb{P}(A^c|B)}{\mathbb{P}(A|B)} \\ &= \ln\frac{\mathbb{P}(ABE)\mathbb{P}(BE)}{\mathbb{P}(BE)\mathbb{P}(A^cBE)}\frac{\mathbb{P}(A^cB)\mathbb{P}(B)}{\mathbb{P}(B)\mathbb{P}(AB)} = \ln\frac{\mathbb{P}(ABE)\mathbb{P}(A^cB)}{\mathbb{P}(A^cBE)\mathbb{P}(AB)} = \ln\frac{\mathbb{P}(E|AB)}{\mathbb{P}(E|A^cB)}\end{aligned}$$

(b)

$$\begin{aligned}W(H_1/H_2:E|G) &= \ln\frac{O_{\mathbb{P}}(H_1/H_2|GE)}{O_{\mathbb{P}}(H_1/H_2|G)} = \ln\frac{\mathbb{P}(H_1|GE)\mathbb{P}(H_2|G)}{\mathbb{P}(H_2|GE)\mathbb{P}(H_1|G)} \\ &= \ln\frac{\mathbb{P}(H_1GE)\mathbb{P}(GE)\mathbb{P}(H_2G)\mathbb{P}(G)}{\mathbb{P}(GE)\mathbb{P}(H_2GE)\mathbb{P}(G)\mathbb{P}(H_1G)} = \ln\frac{\mathbb{P}(E|H_1G)}{\mathbb{P}(E|H_2G)}.\end{aligned}$$

Chapter 14

Structure Learning

14.1 Introduction

This chapter considers the problem of learning the structure of a DAG corresponding to a Bayesian network for a random (row) vector $X = (X_1, \ldots, X_d)$ when presented with an $n \times d$ data matrix $\mathbf{x}$, considered as an instantiation of a random matrix

$$\mathbf{X} = \begin{pmatrix} X_{1\cdot} \\ \vdots \\ X_{n\cdot} \end{pmatrix}$$

where $X_{1\cdot}, \ldots, X_{n\cdot}$ is a collection of independent identically distributed random vectors, each with the same distribution as X. The notation $X_{j\cdot}$ means $(X_{j1}, \ldots, X_{jd})$ for $j = 1, \ldots, n$.

Methods available fall into two categories; *search and score* techniques, where a score function is used and the algorithm attempts to find the structure that maximises the score function and *constraint based* methods, where conditional independence tests are carried out and the independence relations thus established provide constraints, limiting the edges that can be added.

Algorithms can, broadly speaking, be placed in one of three different categories; *search-and-score*, *constraint based* and *hybrid*. Hybrid algorithms use features from both constraint based and search and score methods.

The aim of this chapter is to give a broad introduction and describe some of the search-and-score algorithms. Constraint based algorithms will be dealt with in considerably more detail in Chapter 16, while Markov chain Monte Carlo (MCMC), the most popular search-and-score approach, will be dealt with in Chapter 18.

The straightforward approach of maximising the likelihood, or a posterior distribution, over graph structures leads to a problem that, a first glance, may appear fairly straightforward. There is a finite number of different possible DAGs $\mathcal{G} = (V, D)$ with d nodes. In general, though, testing all possible structures is not computationally feasible. This is because the number of possible DAGs grows super exponentially in the number of nodes. In [118], Robinson gave the following recursive function for computing the number $N(d)$ of acyclic directed graphs with d nodes:

$$N(d) = \sum_{i=1}^{d} (-1)^{i+1} \begin{pmatrix} d \\ i \end{pmatrix} 2^{i(d-1)} N(d-i). \tag{14.1}$$

For $d = 5$ it is 29000 and for $d = 10$ it is approximately 4.2×10^{18}. Here $N(d)$ is a very large number, even for small values of d. Therefore, it is clearly not feasible to compute this sum, even for modest values of d.

14.2 Distance Measures

When measuring distance, there are two criteria of interest: firstly, the graph alone can be considered. A distance measure between graphs will simply compare the numbers of edges and their orientations between graphs. Secondly, the differences between the probability distributions, estimated from data, factorised along the graph may be considered.

14.2.1 Structural Hamming Distance

This is a distance measure that simply measures the distance between graphs. In the context of fitting a Bayesian network, structures that are Markov equivalent should be considered equal, since only the Markov equivalence class can be obtained from data. The Structural Hamming distance between two DAGs graphs is defined as follows

Definition 14.1 (Structural Hamming Distance)**.** *The* Structural Hamming Distance *between two DAGs graphs* $\mathcal{G}_1 = (V, D_1)$ *and* $\mathcal{G}_2 = (V, D_2)$ *is defined as*

$$\begin{aligned} SHD(D_1, D_2) \quad = \quad & (\textit{number of edges that have to be added to } D_1) \\ & + (\textit{number of edges that have to be deleted from } D_1) \\ & + (\textit{number of edges in } D_1 \textit{ that have to have their direction changed }) \\ & \textit{to obtain } D_2 \end{aligned}$$

The structural Hamming Distance between two essential graphs $\mathcal{G}_1 = (V, E_1)$ *and* $\mathcal{G}_2 = (V, E_2)$ *is defined as*

$$SHD_{ess}(E_1, E_2) \quad = \quad \min_{D_1 \in \mathcal{E}_1, D_2 \in \mathcal{E}_2} SHD(D_1, D_2)$$

where $\mathcal{E}_1$ *is the set of DAGs within the Markov equivalence class of* E_1 *and* $\mathcal{E}_2$ *is the set of DAGs within the Markov equivalence class of* E_2*.* D_1 *and* D_2 *are the edge sets for directed acyclic graphs chosen from the equivalence classes* $\mathcal{E}_1$ *and* $\mathcal{E}_2$ *respectively.*

The SHD is a distance measure, or metric, in the sense that it satisfies the definition of a distance or metric. That is, it satisfies:

- $SHD(E_1, E_2) \geq 0 \quad \forall E_1, E_2$

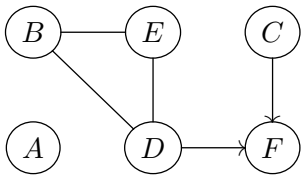

Figure 14.1: Essential graph $\mathcal{G}_1$

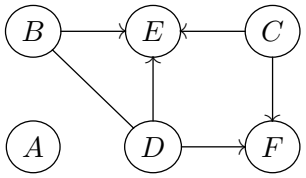

Figure 14.2: Essential graph $\mathcal{G}_2$

- $SHD(E_1, E_2) = 0 \Leftrightarrow E_1 = E_2$
- $SHD(E_1, E_2) = SHD(E_2, E_1)\ \forall E_1, E_2,$
- $SHD(E_1, E_3) \leq SHD(E_1, E_2) + SHD(E_2, E_3)\ \forall E_1, E_2, E_3.$

The Structural Hamming Distance measures the distance between two essential graphs, but if comparison is being made between a 'fitted' graph and a 'true' graph, the SHD distance measure does not distinguish between 'false positives' (edges in the fitted graph that are not in the true graph) and 'false negatives' (edges not present in the fitted graph that are present in the true graph).

The distance thus defined between the two graphs in Figures 14.1 and 14.2 is 1, since there is a valid orientation of the edges in 14.1 where all except $C - E$ (which is not present) have the same orientation as the edges in Figure 14.2.

14.2.2 Sensitivity and Specificity

Rough measures of 'goodness of fit', when comparing the skeletons of a fitted graph with a true graph, are the *sensitivity* and *specificity*. Sensitivity, *True Positive Rate*, is defined as follows:

$$\text{TPR} = \frac{\text{number of edges correctly identified}}{\text{number of edges correctly identified} + \text{number of edges falsely rejected}} \tag{14.2}$$

The specificity, SPC is defined as

$$\text{SPC} = \frac{\text{number of edges correctly rejected}}{\text{number of edges correctly rejected} + \text{number of edges wrongly included}}. \tag{14.3}$$

The TPR measure is useful, but for the relatively sparse graphs in view for genetics data, where the parent / child sets are limited, the SPC measure is not so useful, unless it is modified. For d variables, there are 2^d possible edges to consider for the skeleton, exponential in the number of variables. For

sparse graphs, with a large number of nodes, the specificity measure will always be approximately 1 for an algorithm with a tendency to wrongly reject edges rather than wrongly include edges. The following definitions for sensitivity and specificity are therefore more convenient; the specificity corresponding to the usual definition, the sensitivity modified. The following definitions are proposed for sparse graphs:

Definition 14.2 (Sensitivity and Specificity)**.** *For the construction of the skeleton, the sensitivity is defined as*

$$TPR = \frac{number\ of\ edges\ correctly\ identified}{number\ of\ edges\ correctly\ identified + number\ falsely\ rejected} \tag{14.4}$$

while the proposed definition for specificity is

$$SPC = \frac{total\ number\ of\ edges\ in\ the\ skeleton}{total\ number\ of\ edges\ in\ the\ skeleton + number\ of\ edges\ wrongly\ included}. \tag{14.5}$$

Equation (14.5) is not the standard definition of specificity, but if the value is close to 1, it implies that the rate of wrong inclusion is insignificant, rather than that the graph is large and sparse, which would lead to a value close to 1 using the definition in Equation (14.3) even if the number of edges wrongly included is large compared with the total number of edges in the true graph.

14.2.3 The Kullback Leibler Divergence

The Kullback Leibler Divergence may be used as the basis of measuring the distance between two DAGs over d variables, with respect to a data set. Recall the definition of the Kullback Leibler Divergence between two probability functions p and q each defined over the same state space $\mathcal{X}$:

$$D_{KL}(p\|q) = \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)}.$$

In view here is the divergence between the factorisation over a true directed acyclic graph $\mathcal{G}_1 = (V, D_1)$ and a fitted directed acyclic graph $\mathcal{G}_2 = (V, D_2)$. Let $\widehat{p}_1$ and $\widehat{p}_2$ denote the fitted probability distributions from the data, according to the factorisations along $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively. The fitted distribution $\widehat{p}$ is the same for each directed acyclic graph within the Markov equivalence class of an essential graph.

14.3 Search and Score Algorithms

For a Bayesian network with a directed acyclic graph $\mathcal{G} = (V, D)$, the edge set D is often referred to as the *structure* of the network. Let $\tilde{\mathcal{D}}$ denote the set of all possible edge sets that give a directed acyclic graph with node set V and suppose that D is unknown and has to be inferred only from the $n \times d$ data matrix $\mathbf{x}$.

For a given structure, the prior distribution over the parameter vectors $\underline{\theta}_{j.l}$ are taken from the family $Dir(\alpha_{j1l}, \ldots, \alpha_{jk_jl})$ for all nodes and parent configurations (j, l). There is a prior distribution $p_{\mathcal{D}}$ over the collection of possible structures $\tilde{\mathcal{D}}$,which is the probability function for a random variable $\mathcal{D}$ taking values in $\tilde{\mathcal{D}}$.

The Prior Distribution for the Graph Structure There are several possible ways of constructing a prior distribution $p_{\mathcal{D}}$. If it is known a priori that the graph structure lies within a subset $A \subseteq \tilde{\mathcal{D}}$, then an obvious choice is the uniform prior over A;

$$\mathbb{P}_{\mathcal{D}}(D) = \begin{cases} \frac{1}{|A|} & \text{if } D \in A \\ 0 & \text{otherwise} \end{cases}$$

where $|A|$ is the number of elements in a subset $A \subseteq \tilde{\mathcal{D}}$.
The Bayesian selection rule for a graph $\mathcal{G} = (V, D)$ uses the graph which maximises the posterior probability

$$\mathbb{P}_{\mathcal{D}|\mathbf{X}}(D|\mathbf{x}) = \frac{\mathbb{P}_{\mathbf{X}|\mathcal{D}}(\mathbf{x}|D)\mathbb{P}_{\mathcal{D}}(D)}{\mathbb{P}_{\mathbf{X}}(\mathbf{x})}, \tag{14.6}$$

The task is then, for a given $\mathbf{x}$, to find the D that maximises the *Bayesian Dirichlet* score function

$$S(D) = \mathbb{P}_{\mathbf{X}|\mathcal{D}}(\mathbf{x}|D)\mathbb{P}_{\mathcal{D}}(D). \tag{14.7}$$

where $\mathbb{P}_{\mathcal{D}}$ is the prior probability over the space of edge sets. The *prior odds ratio* for two different edge sets D_1 and D_2 is defined as $\frac{\mathbb{P}_{\mathcal{D}}(D_1)}{\mathbb{P}_{\mathcal{D}}(D_2)}$ and the *posterior odds ratio* is defined as $\frac{\mathbb{P}_{\mathcal{D}|\mathbf{X}}(D_1|\mathbf{x})}{\mathbb{P}_{\mathcal{D}|\mathbf{X}}(D_2|\mathbf{x})}$. Equation (14.6) may then be expressed as

$$\text{Posterior odds} = \text{Likelihood ratio} \times \text{Prior odds} = \frac{S(D_1)}{S(D_2)}.$$

Using factorisations along the relevant graphs, the computation of a ratio, rather than simply computing each score function, is sometimes easier if the two graphs have some part of the structure in common.

Computing the posterior distribution is an *NP hard problem*; Cooper [30] proves that the inference problem is NP hard. That means, worse than an NP problem. This discussed in [26]. Koivisto and Sood [75] (2004) constructed the first algorithm that had a complexity less than super exponential for finding the posterior probability of a network, at the expense of limiting the maximum number of parents for each variable; the run time is $O(d2^d + d^{k+1}C(n))$ where d is the number of notes, k is the maximum in-degree permitted and $C(n)$ the cost of computing a single local marginal conditional marginal likelihood for n instantiations.

Aside: P, NP and NP Hard Problems A problem is assigned to the NP (non-deterministic polynomial time) class if it is verifiable in polynomial time by a non-deterministic Turing machine. (A non-deterministic Turing machine is a 'parallel' Turing machine which can take many computational paths simultaneously, with the restriction that the parallel Turing machines cannot communicate.) A P-problem (whose solution time is bounded by a polynomial) is always also NP. If a problem is known to be NP, and a solution to the problem is somehow known, then demonstrating the correctness of the solution can always be reduced to a single P (polynomial time) verification. A problem is *NP-hard* if an algorithm for solving it can be translated into one for solving any other NP-problem (non-deterministic

polynomial time) problem. NP-hard therefore means 'at least as hard as any NP-problem' although it might, in fact, be harder.

14.3.1 Score Functions

For a given data matrix $\mathbf{x}$, one example of a score function is simply a function proportional to the posterior probability given, for example, by Equation (14.7). It is often considered that this score function gives too much preference to graphs with large number of edges.

AIC and BIC Score Functions One standard score function is the Akaike Information Criterion (AIC) defined as:

$$\mathrm{AIC}(D) = -2\log L(D|\mathbf{x}) + 2|\underline{\theta}| \tag{14.8}$$

where $|\underline{\theta}| := \sum_{j=1}^{d} q_j(k_j - 1)$ denotes the number of parameters required to define the network and $L(D|\mathbf{x})$ is the Cooper Herskovitz likelihood given by Equation (12.15). The Bayesian Information Criterion is similar, but uses $\log|\underline{\theta}|$;

$$\mathrm{BIC}(D) = -2\log L(D|\mathbf{x}) + (\log n)|\underline{\theta}|. \tag{14.9}$$

The BDeu Score The *BDeu score* was introduced by Heckerman, Geiger and Chickering [62]. The BD score is simply the score function given by Equation (14.7), the posterior probability over directed acyclic graphs, assuming that the variables each have multinomial distribution. The BDeu score uses a uniform prior over graph structures, so that the posterior distribution is proportional to the likelihood, and then multiplies by a factor that penalises according to the number of edges where the graph differs from some 'target' graph, based on prior information. The BDeu score function for a directed acyclic graph, based on the data is defined as follows

Definition 14.3 (BD, BDeu Score Function)**.**

$$S(D;\mathbf{x}) = \kappa^{\delta(D)} \prod_{j=1}^{d} \prod_{l=1}^{q_j} \frac{\Gamma(\sum_{i=1}^{k_j} \alpha_{jil})}{\Gamma\left(n(\pi_j^l) + \sum_{i=1}^{k_j} \alpha_{jil}\right)} \prod_{i=1}^{k_j} \frac{\Gamma(n(x_j^i|\pi_j^l) + \alpha_{jil})}{\Gamma(\alpha_{jil})}, \tag{14.10}$$

where $\mathbf{x}$ *denotes the* $n \times d$ *data matrix of* n *independent instantiations of the* d *variables in the variable set* V*,* D *denotes the edge set for the directed acyclic graph* $\mathcal{G} = (V, D)$*,* κ *is a number* $0 < \kappa \leq 1$*,* $\delta(D)$ *denotes the number of edges in* D *that differ from those in a 'target' graph, a graph that is a priori considered most likely, based on prior information.*

The BD score function is the BDeu score function with $\kappa = 1$*.*

When the aim is to construct a graph representing the dependence relations in the data, with as few edges as possible, $\delta(D)$ simply counts the number of edges in the edge set D.

Definition 14.4 (Prior Sample Size)**.** *The* prior sample size *is defined as the quantity*

$$\tilde{n} = \sum_{jil} \alpha_{jil}.$$

The quantity $\tilde{n} = \sum_{jil} \alpha_{jil}$ is considered to be the weight attached to the prior assessment. Loosely speaking it is the 'number' of observations on which the prior is based.

For the BDeu score function, the value $\kappa = \frac{1}{1+n+\tilde{n}}$ is often chosen. Note that, with this choice of κ, the BDeu and BIC are similar; the BIC penalty is the number of parameters, while the BDeu penalty is the number of edges.

14.3.2 Sparse Candidate Algorithm

The discussion now moves onto a selection of search and score algorithms. The first of these is *the sparse candidate algorithm*, which was developed by Friedman, Nachman and Pe'er (1999) [45] and used for analysis of genetic expression data in Friedman et. al. (2000) [46]. The main idea of the technique is to identify a relatively small number of *candidate* parents for each variable. This is based on simple local statistics, such as correlation. Attention is then restricted to networks in which the parent set is a subset of the candidate parent set.

The algorithm proceeds as follows: let D_n denote the DAG chosen at iteration n, let $\mathrm{Pa}_i^{(n)}$ denote the parent set for variable X_i in D_n.

- For $i = 1, \ldots, d$, choose the candidate set $C_i^{(n)} = \{Y_1, \ldots, Y_k\}$ of candidate variables for Pa_i, the parent set for variable X_i. The set $C_i^{(n)}$ is chosen as $\mathrm{Pa}_i^{(n-1)}$ together with children and parents of children of X_i in D_n, and all those variables $Y \notin MB(X_i)$ such that the score

$$\sum_{(x,y,z)\in\mathcal{X}_{X_i}\times\mathcal{X}_Y\times\mathcal{X}_{\mathrm{Pa}_i^{(n-1)}}} n_{X_i,Y,\mathrm{Pa}_i^{(n-1)}}(x,y,z) \ln \frac{n_{X_i,Y,\mathrm{Pa}_i^{(n-1)}}(x,y,z) n_{\mathrm{Pa}_i^{(n-1)}}(z)}{n_{Y,\mathrm{Pa}_i^{(n-1)}}(y,z) n_{X_i,\mathrm{Pa}_i^{(n-1)}}(x,z)}$$

  is sufficiently high. Here MB denotes *Markov blanket* (parents, children and parents of children). Also, for a set W, $n_W(w)$ denotes the number of appearances of configuration w in the data matrix $\mathbf{x}$. If the test statistic is low, it supports $X_i \perp Y | \mathrm{Pa}_i^{(n-1)}$ and hence Y is not a candidate parent.

  There are other ways of determining the candidate parents; anything in the current Markov blanket not d-separated from the variable by the Markov blanket should be included as a candidate parent.

- Find a high scoring network D_n where $\mathrm{Pa}_i^{D_n} \subset C_i^{(n)}$ for $i = 1, \ldots, d$.

Optimal Reinsertion The *optimal reinsertion* algorithm, introduced by A. Moore and W-K. Wong (2003) [96], is a search - and -score algorithm that works along the following lines: at each step a *target node* is chosen, all edges entering or leaving the target are deleted, and the optimal combination of in-edges and out-edges is found, the node is re-inserted with these edges. This involves searching through the legal candidate parent sets and, for each candidate parent set, the legal child sets. The optimal reinsertion may be combined with sparse candidate.

14.3.3 Greedy Search and Greedy Equivalence Search

The *Greedy Search* was introduced by Meek (1997) in his Ph.D. thesis and correctness was proved by Chickering (2002) [25] under the assumption that there was a DAG faithful to the probability distribution. It works along the following lines to produce a DAG, along which the probability distribution factorises, starting from the graph with no edges:

- *Forward phase* Let E_0 denote the graph with no edges. Let E_n denote the essential graph from stage n of the forward phase. Consider all possible DAGs within the Markov equivalence class, all possible DAGs obtained by adding exactly one edge to a DAG from this equivalence class and consider the set of essential graphs corresponding to this collection of DAGs. Let E_{n+1} denote the essential graph with the highest score if it has a higher score than E_n and continue to forward phase stage $n+1$. Otherwise, terminate the forward phase, with output E_n.

- *Backward phase* Let $\tilde{E}_0$ denote the output graph from the forward phase. Let $\tilde{E}_n$ denote the output graph from stage n of the backward phase. Consider all possible DAGs corresponding to the equivalence class $\tilde{E}_n$, all possible DAGs formed by an edge deletion from these DAGs and consider the set of essential graphs corresponding to this collection of DAGs. Let $\tilde{E}_{n+1}$ denote the essential graph with the highest score if it is higher than that for $\tilde{E}_n$ and continue to backward phase stage $n+1$. Otherwise terminate; $\tilde{E}_n$ is the output of the backward phase and of the greedy equivalence search algorithm.

After the forward and backward phase, this algorithm is guaranteed to return an optimal structure *provided there exists a faithful DAG*. The faithfulness assumption may be relaxed; the algorithm returns a suitable structure provided the weaker *composition* condition holds (compositional graphoid, Equation (2.1.1)). The compositional axiom is essential for the algorithm to return the correct graph.

The necessity of composition is clear from the three variable example, where Y_1, Y_2, Y_3 are independent binary variables $\mathbb{P}(Y_i = 1) = \mathbb{P}(Y_i = 0) = \frac{1}{2}$, $X_1 = \mathbf{1}(Y_2 = Y_3)$, $X_2 = \mathbf{1}(Y_1 = Y_3)$, $X_3 = \mathbf{1}(Y_1 = Y_2)$. Since $X_1 \perp X_2$, $X_1 \perp X_3$ and $X_2 \perp X_3$, adding a *single* edge to the empty graph will not increase the score. The algorithm will therefore terminate after the first step of the forward phase and return the empty graph.

Notes The Cooper Herskovitz likelihood was introduced by Cooper and Herskovitz in [31]. In [30], Cooper proves that the inference problem for structure learning is NP hard. In [26], Chickering Heckerman and Meek prove, under some assumptions, that identifying high scoring structures in search - and - score algorithms is NP - hard. Koivisto and Sood [75] [2004] constructed the first algorithm that had a complexity less than super - exponential for finding the posterior probability of a network. The Chow - Liu tree is taken from [28] [1969]. The K2 algorithm is by Cooper and Herskovitz [31] [1992]. The robotics example is due to E. Lazkano, B. Sierra, A. Astigarraga, and J.M. Martínez - Otzeta [79] [2007] The maximum minimum hill climbing algorithm is found in [137]. The Markov chain Monte Carlo model composition algorithm, known as MC^3, and the augmented Markov chain Monte

Carlo model composition (AMC^3) algorithm were introduced by Madigan and York [89] in 1995 and Madigan, Andersson, Perlman and Volinsky [88] in 1997.

14.4 Exercises

These exercises should be carried out using R. The `bnlearn` package may be useful.

1. **Chow - Liu Tree** Generate three columns, $c1$, $c2$ and $c3$, each containing independent random samples of 50 $Be(1/2)$ observations. Here $Be(1/2)$ means Bernoulli trials, returning 0 with probability 1/2 and 1 with probability 1/2. Let $c4 = c1 + c2$ and let $c5 = c3 + c4$. Implement the Kruskal algorithm on the variables $c1, c2, c3, c4, c5$ and see which edges are chosen.

2. **Chow - Liu Tree** Download the data set from the URL address
   `http://archive.ics.uci.edu/ml/machine-learning-databases/zoo/zoo.data`
   A description of the data is found at the address
   `http://archive.ics.uci.edu/ml/datasets/Zoo`
   The data set presents attributes of various animals; hair type, feather type, egg type, milk type, whether it is airborne, aquatic, a predator, whether or not it has teeth, a backbone, whether it breathes, or is venomous, has fins, legs, tail, or domestic, or catsize. The last variable is a classification of the type of animal.

   (a) Compute the estimated probability distribution for all the variables except for the 'class' variable, assuming that they are independent. What is the Kullback Leibler distance between the empirical distribution and the estimate using the independence model?

   (b) Perform Kruskal's algorithm, to determine the optimal Chow - Liu tree. Use the data from all the variables, except for the class variable, to construct a single Chow - Liu tree. Calculate the estimated probability distribution, assuming that the distribution factorises according to the Chow - Liu tree. Calculate the Kullback Leibler distance between this estimate and the empirical probability distribution.

   Note You have to specify a root for the Chow-Liu tree. This determines the directions of the arrows. All possible Chow-Liu trees from the same skeleton are Markov equivalent.

   (c) **Classification** See how the Chow - Liu tree performs for classification. Compute the classifier using the data and then use the classifier to predict the classes of the same data set. Such a procedure is not so satisfactory; different data should be used for training and classification.

   (d) Perform an MMPC algorithm on the `zoo` data, using a nominal significance level of 0.05. Compare it with the Chow-Liu tree with those edges that fail the significance test at 0.05 level are removed.

The following R code solves the Chow-Liu tree problem. The '50 warnings' basically come from zero divided by zero problems. The code should be modified (by adding on a small value such as 0.01 to each cell) to prevent this. The classifier works reasonably well in any case.

```
> library("bnlearn")
> zoo <- read.csv("~/data/zoo.data", header=F)
```

```
> colnames(zoo) <-    c("animal name", "hair", "feathers", "eggs","milk","airborne","aquatic",
+                        "predator","toothed","backbone","breathes","venomous","fins", "legs","tai
> for(i in 2:ncol(zoo)) zoo[,i] <- factor(zoo[,i]) # conversion to factors
> s <- sample(100,70)
> trainingdata <- zoo[s,]
> testdata <- zoo[-s,]
> res <- chow.liu(trainingdata[,-c(1,18)]) # learning the structure
> print(res)

  Bayesian network learned via Pairwise Mutual Information methods

  model:
    [undirected graph]
  nodes:                                16
  arcs:                                 15
    undirected arcs:                    15
    directed arcs:                      0
  average markov blanket size:          1.88
  average neighbourhood size:           1.88
  average branching factor:             0.00

  learning algorithm:                   Chow-Liu
  mutual information estimator:         Maximum Likelihood (disc.)
  training node:
  tests used in the learning procedure: 120
  > print(res$arcs)
       from       to
 [1,] "hair"     "milk"
 [2,] "milk"     "hair"
 [3,] "feathers" "legs"
 [4,] "legs"     "feathers"
 [5,] "eggs"     "milk"
 [6,] "milk"     "eggs"
 [7,] "eggs"     "toothed"
 [8,] "toothed"  "eggs"
 [9,] "milk"     "catsize"
[10,] "catsize"  "milk"
[11,] "airborne" "legs"
[12,] "legs"     "airborne"
[13,] "aquatic"  "breathes"
```

```
[14,] "breathes" "aquatic"
[15,] "predator" "legs"
[16,] "legs"     "predator"
[17,] "predator" "domestic"
[18,] "domestic" "predator"
[19,] "toothed"  "legs"
[20,] "legs"     "toothed"
[21,] "backbone" "legs"
[22,] "legs"     "backbone"
[23,] "backbone" "tail"
[24,] "tail"     "backbone"
[25,] "breathes" "legs"
[26,] "legs"     "breathes"
[27,] "venomous" "legs"
[28,] "legs"     "venomous"
[29,] "fins"     "legs"
[30,] "legs"     "fins"
> plot(res)
> res2 <- pdag2dag(res, colnames(zoo)[c(2,5,4,14,17,3,6,9,10,11,12,15,7,8,13,16)]) # directing t
> plot(res2)
> # parameters estimation for every class
> res3 <- list()
> for(i in 1:7){
+     res3[[i]] <- bn.fit(res2, trainingdata[trainingdata$type==i,-c(1,18)],method="bayes")
+ }
>
> lik <- array(dim=c(7,nrow(testdata))) # matrix of likelihoods
> for(i in 1:7) for (j in 1:nrow(testdata)){
+     lik[i,j] <- logLik(res3[[i]],testdata[j,-c(1,18)])
+ }
>
> pred <- apply(lik,2,which.max)
> table (pred, testdata$type)

pred  1  2  3  4  5  6  7
   1 11  0  0  0  0  0  0
   2  0  7  0  0  0  0  0
   3  0  0  0  0  4  0  0
   4  0  0  0  6  0  0  0
   6  0  0  0  0  0  1  0
```

```
   7  0  0  0  0  0  0  2
> > resmmpc <- mmpc(zoo[,-c(1,18)])
> print(resmmpc$arcs)
      from       to
 [1,] "hair"     "aquatic"
 [2,] "hair"     "milk"
 [3,] "eggs"     "milk"
 [4,] "milk"     "catsize"
 [5,] "milk"     "eggs"
 [6,] "milk"     "hair"
 [7,] "aquatic"  "fins"
 [8,] "aquatic"  "predator"
 [9,] "aquatic"  "hair"
[10,] "predator" "domestic"
[11,] "predator" "aquatic"
[12,] "toothed"  "backbone"
[13,] "backbone" "tail"
[14,] "backbone" "toothed"
[15,] "fins"     "aquatic"
[16,] "tail"     "backbone"
[17,] "domestic" "predator"
[18,] "catsize"  "milk"
```

Chapter 15

Data Storage, Product Approximations, Chow Liu Trees

15.1 Introduction

Let $\underline{X} = (X_1, \ldots, X_d)$ denote a random vector with probability function $\mathbb{P}_{X_1,\ldots,X_d}$. Let

$$\mathcal{X}_j = (x_j^{(1)}, \ldots, x_j^{(k_j)})$$

denote the state space of X_j, $j = 1, \ldots, d$ and let

$$\mathcal{X} = \times_{j=1}^{d} \mathcal{X}_j.$$

The number of elements in the state space is $|\mathcal{X}| = \left(\prod_{j=1}^{d} k_j\right)$ and, without further assumptions on $\mathbb{P}$, $|\mathcal{X}| - 1$ elements are required to store the entire distribution.

The problem of storing the entire probability distribution is one of many expressions of the 'curse of dimensionality'. The size of the problem is reduced if one instead stores lower dimensional marginals and approximates the distribution by an appropriate product of lower dimensional marginals.

The topic of storing a high dimensional discrete probability distribution in a digital medium appeared in the journal literature, probably for the first time, by J. Hartmanis (1959) [59] and P.M. Lewis II (1959) [85]. The Chow - Liu tree by Chow and Liu (1969) [28], approximately 10 years later, provides an influential and effective solution to the problem. Chow and Liu gave an algorithm for selecting first order factors for the product approximation so that among all such first order approximations, the constructed approximation has the minimum Kullback-Leibler distance to the actual distribution to be stored.

15.2 Product Approximations

15.2.1 Existence of Extensions with Given Marginals

For a probability distribution $\mathbb{P}_{X_1,\ldots,X_d}$ over a set of random variables $\underline{X} = (X_1, \ldots, X_d)$, there are $\sum_{j=1}^{d-1} \binom{d}{j} = 2^d - 2$ lower dimensional marginal distributions, which may be obtained by marginalising

the distribution. The *classical marginal problem* considers an 'inverse problem'; given a family $(\mathbb{P}_{W_i})_{i=1}^s$ of probability distributions, for $s < 2^d - 2$ and $W_i \subset V = \{X_1, \ldots, X_d\}$, the question is whether there exists a probability distribution $\mathbb{P}_V$ that satisfies the so-called *collective compatibility condition* given by Equation (15.1).

$$\mathbb{P}_{W_i} = (\mathbb{P}_V)^{\downarrow W_i} \qquad \forall i = 1, \ldots, s \tag{15.1}$$

Here the notation $\downarrow A$ means the marginalisation down to a set of variables A. This problem is of importance in the following setting: if the full probability distribution cannot be estimated and stored, it may be possible to estimate and store probability distributions over selected subsets of the variables. These subsets should be chosen so that, formally, the collection of distributions over the subsets are compatible. Some fundamental contributions to this problem are due to H.G. Kellerer [73] and others.

If the sets $(W_i)_{i=1}^s$ are disjoint, satisfying $\cup_i W_i = V$, then the problem has an obvious trivial solution:

$$\mathbb{P}(\underline{x}) = \prod_{i=1}^{s} \mathbb{P}_{W_i}(\underline{x}_{W_i})$$

where the product operation means first extending the probabilities $\mathbb{P}_{W_i}$ as functions, to functions $\tilde{\mathbb{P}}_{W_i}$ over the domain V where (using obvious notation) $\tilde{\mathbb{P}}_{W_i}(\underline{x}_{W_i}.\underline{x}_{V \setminus W_i}) = \mathbb{P}_{W_i}(\underline{x}_{W_i})$ for each $\underline{x}_{W_i} \in \mathcal{X}_{W_i}$ and then multiplying. With the appropriate projections of $\underline{x}$,

$$\mathbb{P}(\underline{x}) = \prod_{j=1}^{s} \mathbb{P}_{W_j}(\underline{x}_{W_j}).$$

If $(W_j)_{j=1}^s$ are not disjoint, then clearly the collection of probabilities $(\mathbb{P}_{W_j})_{j=1}^s$ should satisfy a *pairwise compatibility* condition:

$$\mathbb{P}_{C_{ij}} = \mathbb{P}_{W_i}^{\downarrow C_{ij}} = \mathbb{P}_{W_j}^{\downarrow C_{ij}} \qquad \forall i, j \in \{1, \ldots, s\}^2.$$

The following example due to Vorobev (1962) [141] shows that pairwise compatibility does not imply collective compatibility.

Example 15.1 (Vorobev's example)**.**

Let $V = \{1, 2, 3\}$, $W_1 = \{2, 3\}$, $W_2 = \{1, 3\}$, $W_3 = \{1, 2\}$. Suppose that the following three pairwise joint distributions are specified:

$$\mathbb{P}_{W_1}(x_2, x_3) = \begin{array}{c|cc} x_2 \backslash x_3 & 0 & 1 \\ \hline 0 & \frac{1}{2} & 0 \\ 1 & 0 & \frac{1}{2} \end{array} \qquad \mathbb{P}_{W_2}(x_3, x_1) = \begin{array}{c|cc} x_1 \backslash x_3 & 0 & 1 \\ \hline 0 & 0 & \frac{1}{2} \\ 1 & \frac{1}{2} & 0 \end{array} \qquad \mathbb{P}_{W_3}(x_1, x_2) = \begin{array}{c|cc} x_1 \backslash x_2 & 0 & 1 \\ \hline 0 & \frac{1}{2} & 0 \\ 1 & 0 & \frac{1}{2} \end{array}$$

These are pairwise compatible; $W_1 \cap W_2 = \{3\}$ and

$$\mathbb{P}_{W_1}^{\downarrow\{3\}}(x_3) = \mathbb{P}_{W_2}^{\downarrow\{3\}}(x_3) = \begin{array}{cc} 0 & 1 \\ \hline \frac{1}{2} & \frac{1}{2} \end{array}.$$

$W_1 \cap W_3 = \{2\}$ and

$$\mathbb{P}_{W_1}^{\downarrow\{2\}}(x_2) = \mathbb{P}_{W_3}^{\downarrow\{2\}}(x_2) = \begin{array}{cc} 0 & 1 \\ \hline \frac{1}{2} & \frac{1}{2} \end{array}.$$

$W_2 \cap W_3 = \{1\}$ and

$$\mathbb{P}_{W_2}^{\downarrow\{1\}}(x_1) = \mathbb{P}_{W_3}^{\downarrow\{1\}}(x_1) = \begin{array}{cc} 0 & 1 \\ \hline \frac{1}{2} & \frac{1}{2} \end{array}.$$

If a common extension $\mathbb{P}^*$ existed, it would follow that (for example)

$$\frac{1}{2} = \mathbb{P}_{W_1}(0,0) = \mathbb{P}^*(0,0,0) + \mathbb{P}^*(1,0,0) \leq \mathbb{P}_{W_2}(0,0) + \mathbb{P}_{W_3}(1,0) = 0,$$

which is a contradiction. The three marginals satisfy a pairwise compatibility condition, but not a collective compatibility condition. □

Without loss of generality, let $V = \cup_{j=1}^{s} W_j$. The condition to ensure that pairwise compatibility implies collective compatibility is known as the *acyclic condition.*

Definition 15.2 (Acyclic, Running Intersection Property)**.** *Suppose that there is an ordering of the sets $W_1, \ldots, W_s$ such that for each j there is an $l < j$ such that*

$$B_j = W_j \cap \left(\cup_{k=1}^{j-1} W_k \right) \subseteq W_j \cap W_l \tag{15.2}$$

This property is known as the running intersection property. *A set of subsets of $W_1, \ldots, W_s$ having the running intersection property, given some ordering, is known as* acyclic.

Remark In Example 15.1, if the ordering W_1, W_2, W_3 is chosen, then

$$W_3 \cap (W_1 \cup W_2) = \{1, 2\},$$

but $\{1, 2\}$ is not a subset of W_1 or W_2. It follows that Equation (15.2) does not hold for this ordering. It is easy to check that there is no ordering that satisfies Equation (15.2), hence acyclicity does not hold for Example 15.1.

The following important result is due to Beeri et. al. (1983) [5]

Theorem 15.3. *Acyclicity is* equivalent *to 'pairwise compatibility for all (i, j) implies collective compatibility'. Furthermore, under acyclicity, there is a unique product form extension,*

$$\mathbb{P}^*(\underline{x}) = \frac{\prod_{l=1}^{s} \mathbb{P}(\underline{x}_{W_l})}{\prod_{h=1}^{s-1} \mathbb{P}(\underline{x}_{V_h})} \tag{15.3}$$

where each V_h is the intersection of two or more W_l.

Proof Assume we have the acyclic / running intersection property. Consider the variables as nodes of a graph, where the sets $(W_i)_{i=1}^s$ are maximal cliques. The running intersection property is equivalent to a perfect order of the maximal cliques, which implies that the maximal cliques $W_1, \ldots, W_s$ can be arranged as a *junction tree*, where for each $j \in \{2, \ldots, s\}$, we choose an $l(j) < j$ from the set $\{l : W_j \cap \left(\cup_{k=1}^{j-1} W_k\right) = W_j \cap W_l\}$ and insert an edge $j - l(j)$. In this way, we have a tree with $s-1$ edges. For each edge $\langle j, l\rangle$, let $V_{\langle j,l\rangle} = W_j \cap W_l$ and let U denote the (undirected) edge set. Now consider any collection $(\mathbb{P}_{W_j})_{j=1}^s$ which is pairwise compatible. Then we can define a distribution $\mathbb{P}^*$ by:

$$\mathbb{P}^*(\underline{x}) = \frac{\prod_{l=1}^s \mathbb{P}_{W_l}(\underline{x}_{W_l})}{\prod_{\langle j,l\rangle \in U} \mathbb{P}_{V_{\langle j,l\rangle}}(\underline{x}_{V_{\langle j,l\rangle}})}$$

where $\mathbb{P}_{V_{\langle j,l}} = (\mathbb{P}_{W_l})^{\downarrow V_{\langle j,l\rangle}} = (\mathbb{P}_{W_j})^{\downarrow V_{\langle j,l\rangle}}$. Since the sets $(W_i)_{i=1}^s$ are arranged on a junction tree, hence any intersection $W_\alpha \cap W_\beta$ is contained in $W_\gamma \cap W_\delta$ for any edge $\gamma - \delta$ on the unique path $\alpha \leftrightarrow \beta$ in the tree. Hence the acyclic property gives (pairwise compatibility implies collective compatibility).

Now suppose that pairwise compatibility implies collective compatibility for $W_1, \ldots, W_s$ and assume that acyclicity is not possible. Taking $W_1, \ldots, W_s$ as the maximal cliques of an undirected graph, lack-of-acyclicity is equivalent to existence of a cycle of length ≥ 4 in the graph without a chord. Let the cycle be $\alpha_1, \ldots, \alpha_m$. Then there are $W_{j_1}, \ldots, W_{j_m}$ such that $\{\alpha_i, \alpha_{i+1}\} \subseteq W_{j_i}$ for $i = 1, \ldots, m$, using $\alpha_{m+1} = \alpha_1$. Furthermore, the lack-of-chord implies that $W_{j_a} \cap W_{j_b} = \varnothing$ for $|a-b| \geq 2$, where we take a and b mod m. We may therefore find (similar to Vorobev's example) distributions $\mathbb{P}_{W_{j_1}}, \ldots, \mathbb{P}_{W_{j_m}}$ which are pairwise compatible, but where the distributions $\mathbb{P}_{W_{j_i}}^{\downarrow\{\alpha_i, \alpha_{i+1}\}}$ (using $\alpha_{m+1} \equiv \alpha_1$) are not collectively compatible. □

Uniqueness of representation (15.3) requires that $\mathbb{P}_{W_i}(\underline{x}_{W_i}) > 0$ for all $\underline{x}_{W_i}$ and all $i = 1, \ldots, s$.

15.2.2 Dependence Structures

Let $W_1, \ldots, W_s$ be sets of random variables, $V = \cup_{j=1}^s W_j$ and suppose that $W_1, \ldots, W_s$ satisfy the running intersection property of Equation (15.2). With this ordering, set $B_1 = \phi$ and

$$B_j = W_j \cap \left(\cup_{k=1}^{j-1} W_k\right), \qquad j = 2, \ldots, k.$$

Let $A_j = W_j \backslash B_j$ so that $W_j = A_j \cup B_j$. It follows that $A_1, \ldots, A_s$ is a *partition* of V and that the sets $(B_j)_{j=1}^s$ satisfy

$$B_j \subset A_i \cup B_i \qquad \text{some} \qquad i \in \{1, \ldots, j-1\}.$$

This leads to the definition of a *dependence structure*, the term used to describe collections $(A_j, B_j)_{j=1}^s$ which satisfy this property.

Definition 15.4 (Dependence Structure)**.** *Let $(A_i)_{i=1}^k$ be a partition of a set V and let S be a sequence of pairs of subsets of V, $S = (A_i, B_i)_{i=1}^k$ satisfying*

$$B_1 = \phi, \qquad B_r \subset A_i \cup B_i \qquad 1 \leq i \leq r-1 \qquad r = 2, \ldots, k$$

Then S is a dependence structure.

Definition 15.5 (Product Approximation)**.** *Let S be a dependence structure. Then the probability distribution defined by*

$$\mathbb{P}^{(S)}(\underline{x}) = \mathbb{P}_{A_1}(\underline{x}_{A_1}) \prod_{j=2}^{k} \mathbb{P}_{A_j|B_j}(\underline{x}_{A_j}|\underline{x}_{B_j})$$

is called the product approximation *of the probability distribution $\mathbb{P}$ determined by S.*

A *product approximation* is clearly a well defined *probability* distribution. Furthermore, it satisfied the following compatibility condition:

Lemma 15.6.

$$\mathbb{P}^{(S)\downarrow A_j \cup B_j}(\underline{x}_{A_j \cup B_j}) = \mathbb{P}_{A_j \cup B_j}(\underline{x}_{A_j \cup B_j}) \qquad \forall \underline{x} \in \mathcal{X}, \qquad j = 1, \ldots, s.$$

Proof By marginalising over $A_{j+1} \cup \ldots \cup A_s$,

$$\mathbb{P}^{(S)\downarrow A_1 \cup \ldots \cup A_j}(\underline{x}_{A_1 \cup \ldots \cup A_j}) = \mathbb{P}_{A_1}(\underline{x}_{A_1}) \prod_{k=2}^{j} \mathbb{P}_{A_k|B_k}(\underline{x}_{A_k}|\underline{x}_{B_k}) \qquad j = 1, \ldots, s$$

so that

$$\begin{aligned}
\mathbb{P}^{(S)\downarrow A_j \cup B_j}(\underline{x}_{A_j \cup B_j}) &= \mathbb{P}_{A_j|B_j}(\underline{x}_{A_j}|\underline{x}_{B_j}) \sum_{A_1 \cup \ldots \cup A_{j-1} \setminus B_j} \mathbb{P}_{A_1}(\underline{x}_{A_1}) \prod_{k=2}^{j-1} \mathbb{P}_{A_k|B_k}(\underline{x}_{A_k}|\underline{x}_{B_k}) \\
&= \mathbb{P}_{A_j|B_j}(\underline{x}_{A_j}|\underline{x}_{B_j}) \sum_{A_1 \cup \ldots \cup A_{j-1} \setminus B_j} \mathbb{P}^{(S)\downarrow A_1 \cup \ldots \cup A_{j-1}}(\underline{x}_{A_1 \cup \ldots \cup A_{j-1}}) \\
&= \mathbb{P}_{A_j|B_j}(\underline{x}_{A_j}|\underline{x}_{B_j}) \mathbb{P}^{(S)\downarrow B_j}(\underline{x}_{B_j}).
\end{aligned}$$

It remains to show that $\mathbb{P}^{(S)\downarrow B_j}(\underline{x}_{B_j}) = \mathbb{P}_{B_j}(\underline{x}_{B_j})$. This follows inductively; $B_1 = \phi$. Assume true for all $i = 1, \ldots, j-1$. Then $B_j \subset A_i \cup B_i$ for some $i \in 1, \ldots, j-1$. Assume that $\mathbb{P}_{A_i \cup B_i} = \mathbb{P}^{(S)\downarrow (A_i \cup B_i)}$ for $1 = 1, \ldots, j-1$, then $\mathbb{P}^{(S)\downarrow B_j} = \mathbb{P}_{B_j}$ and the result follows by induction. □

Note that if $W_i = A_i \cup B_i$ for $i = 1, \ldots, s$ and $\mathbb{P}_{W_i}$ are given, then

$$\mathbb{P}^{(S)} = \frac{\prod_{i=1}^{s} \mathbb{P}_{W_i}}{\prod_{i=2}^{s} \mathbb{P}_{B_i}}$$

where $B_i = W_i \cap \cup_{j=1}^{i-1} W_j$ and the convention $\mathbb{P}_\phi \equiv 1$ is used. In this situation, clearly

$$\mathbb{P}^{(S)W_j} = \mathbb{P}_{W_j}.$$

It follows directly from this factorisation that

$$A_j \perp \cup_{k=1}^{j-1} A_k \setminus B_j |_{\mathbb{P}^{(S)}} B_j.$$

15.3 Reverse I-Projection and the Optimal Product Approximation

The *relative entropy*, or *information divergence*, or *I-divergence*, or *Kullback Leibler distance*, written $D_{KL}(\mathbb{P}\|\mathbb{P}^{(S)})$, is defined by

$$D_{KL}(\mathbb{P}\|\mathbb{P}^{(S)}) = \sum_{\underline{x}\in\mathcal{X}} \mathbb{P}(\underline{x}) \ln \frac{\mathbb{P}(\underline{x})}{\mathbb{P}^{(S)}(\underline{x})}. \tag{15.4}$$

The task of *Optimal Product Representation* of $\mathbb{P}$ is, for a given dependence structure S, to find a $\mathbb{P}^*_S$ such that $D(\mathbb{P}\|\mathbb{P}_S)$ is minimised. The solution $\mathbb{P}^*_S$ is called a *Reverse I-Projection* of $\mathbb{P}$ onto the set of all probability measures with S as a dependence structure.

Definition 15.7 (Shannon Entropy)**.** *Let $A \subseteq V$. The* Shannon entropy *of the set of variables A for a probability distribution $\mathbb{P}$ is defined as*

$$H_{\mathbb{P}}(A) := -\sum_{\underline{x}_A\in\mathcal{X}_A} \mathbb{P}_A(\underline{x}_A) \ln \mathbb{P}_A(\underline{x}_A)$$

where $\mathbb{P}_A = \mathbb{P}^{\downarrow A}$.

With $A = V$, it follows that, for a probability distribution $\mathbb{Q}$,

$$D_{KL}(\mathbb{P}\|\mathbb{Q}) = \sum_{\underline{x}\in\mathcal{X}} \mathbb{P}(\underline{x}) \ln \mathbb{P}(\underline{x}) - \sum_{\underline{x}\in\mathcal{X}} \mathbb{P}(\underline{x}) \ln \mathbb{Q}(\underline{x}) = -H_{\mathbb{P}}(V) - \sum_{\underline{x}\in\mathcal{X}} \mathbb{P}(\underline{x}) \ln \mathbb{Q}(\underline{x}).$$

For a dependence structure $S = (A_i, B_i)_{i=1}^s$ and a probability distribution $\mathbb{Q}$ that factorises according to: $\mathbb{Q} = \prod_{i=1}^s \mathbb{Q}_{A_i|B_i}$, it is straightforward to compute that

$$\begin{aligned}
D_{KL}(\mathbb{P}\|\mathbb{Q}) &= -H_{\mathbb{P}}(V) - \sum_{\underline{x}\in\mathcal{X}} \mathbb{P}(\underline{x}) \sum_{i=1}^s \ln \mathbb{Q}_{A_i|B_i}(\underline{x}_{A_i}|\underline{x}_{B_i}) \\
&= -H_{\mathbb{P}}(V) - \sum_{i=1}^s \sum_{\underline{x}_{A_i\cup B_i}} \mathbb{P}_{A_i\cup B_i}(\underline{x}_{A_i\cup B_i}) \ln \mathbb{Q}_{A_i|B_i}(\underline{x}_{A_i}|\underline{x}_{B_i}) \\
&= -H_{\mathbb{P}}(V) - \sum_{i=1}^s \sum_{\underline{x}_{B_i}} \mathbb{P}_{B_i}(\underline{x}_{B_i}) \sum_{\underline{x}_{A_i}} \mathbb{P}_{A_i|B_i}(\underline{x}_{A_i}|\underline{x}_{B_i}) \ln \mathbb{Q}_{A_i|B_i}(\underline{x}_{A_i}|\underline{x}_{B_i}).
\end{aligned}$$

Now use *Gibb's inequality*; for any two probability distributions $\underline{f}$ and $\underline{g}$ over the same state space,

$$\sum_{j=1}^L f_j \ln f_j \geq \sum_{j=1}^L f_j \ln g_j. \tag{15.5}$$

This follows from the fact that

$$D_{KL}(\underline{f}\|\underline{g}) = \sum_{j=1}^L f_j \ln \frac{f_j}{g_j} \geq 0$$

with equality if and only if $\underline{f} = \underline{g}$. It follows that the *reverse I-projection of $\mathbb{P}$ onto a dependency structure $S = (A_i, B_i)_{i=1}^s$* is

$$\mathbb{P}^{(S)} = \prod_{i=1}^s \mathbb{P}_{A_i|B_i}$$

and

$$D_{KL}(\mathbb{P}\|\mathbb{P}^{(S)}) = -H_{\mathbb{P}}(V) + \sum_{i=1}^{k} \left(H_{\mathbb{P}}(A_i \cup B_i) - H_{\mathbb{P}}(B_i)\right).$$

Definition 15.8 (Mutual Information). *The* mutual information $I(A,B)$ *between two disjoint sets of variables* A *and* B *is defined as*

$$I(A,B) = H(A) + H(B) - H(A \cup B).$$

This may be written as

$$I(A,B) = \sum \mathbb{P}_{A\cup B}(\underline{x}_{A\cup B}) \ln \frac{\mathbb{P}_{A\cup B}(\underline{x}_{A\cup B})}{\mathbb{P}_A(\underline{x}_A)\mathbb{P}_B(\underline{x}_B)} = D_{KL}(\mathbb{P}_{A\cup B}\|\mathbb{P}_A\mathbb{P}_B).$$

Note that $I(A,B) = 0 \Leftrightarrow \underline{X}_A \perp \underline{X}_B$.

If one is choosing a dependence structure $S = (A_i, B_i)_{i=1}^s$, from within a class $\mathcal{S}$ of dependence structures with the same storage properties, it follows that the dependence structure $S = (A_i, B_i)_{i=1}^s$ is chosen to maximise

$$Q(S) = -\sum_{i=1}^{k} H(A_i) + \sum_{i=1}^{k} I(A_i, B_i).$$

15.4 The Optimal Chow-Liu Product Approximation

For a Chow Liu tree, the dependence structure $(A_i, B_i)_{i=1}^k$ satisfies

$$|A_i \cup B_i| \leq 2 \qquad i = 1, \ldots, k.$$

Let $\mathcal{G} = (V,U)$ denote an undirected graph, where $V = \{1, \ldots, d\}$ is the indexing set for the nodes and U is the undirected edge set. An undirected graph $\mathcal{G}$ is *complete* if $U = \{\langle i,j\rangle : 1 \leq i < j \leq d\}$. The *degree of a node* i is defined as the number of distinct edges containing the node i.

A *subgraph* $\mathcal{H}$ of $\mathcal{G}$ is a graph (V_1, U_1) where $V_1 \subseteq V$ and $U_1 \subseteq U$. A subgraph V_1 is *induced* by $A \subset V$ if $V_1 = A$ and $U_1 = U \cap A \times A$. A subgraph $\mathcal{H}$ is a *spanning subgraph* of $\mathcal{G}$ if it is connected and $V_1 = V$.

An undirected tree $\mathcal{T}$ is a connected undirected graph that has no cycles. It follows that there is a unique path between any two nodes. A *spanning tree of a graph* is a spanning graph of $\mathcal{G}$ which is a tree.

A *labelled tree* is a tree on d nodes where each node is labelled by one of the integers $\{1, \ldots, d\}$. In the sequel, labelled trees will be referred to as trees.

A *weighted undirected graph* is

$$\mathcal{G} = ((V,U)|\mathbf{w})$$

where $\mathbf{w} : U \to \mathbb{R}_+$ (non negative real numbers). The weight of a tree is the sum of its edge weights. The weight to be used by the Chow-Liu algorithm will be defined via the mutual information

$$\mathbf{w}(i,j) := I(j,k) = H(j) + H(k) - H(j \cup k).$$

15.4.1 Chow Liu Tree with known $\mathbb{P}$

Definition 15.9 (Chow-Liu Dependence Structure). *Let $(i_r)_{r=1}^d$ be an arbitrary permutation of $V = \{1,\ldots,d\}$. The singleton sets $A_r = \{i_r\}$ $r = 1,\ldots,d$ are a partition of V. Let σ be a sequence of pairs of singletons of V, $\sigma = (i_r, j_r)_{r=1}^d$, where*

$$j_1 = \phi, \qquad j_r \in \{i_1,\ldots,i_{r-1}\} \subseteq V \qquad r = 2,\ldots,d.$$

Then σ is a Chow-Liu dependence structure.

A Chow-Liu dependence structure will give a tree. Since the tree connects all the nodes, it is a *spanning tree*. Arrows are directed *from j_r to i_r*. If $j_r = \phi$, there is no arrow pointing to the node i_r. Any node in a *directed* tree with $j_r = \phi$ is called a root. By construction, i_1 is the only root. A tree with exactly one root is said to be *proper*.

Note that

$$i_r \perp_{\mathbb{P}(S)} \{i_1,\ldots,i_{r-1}\}\backslash\{j_r\}|j_r.$$

For σ thus defined,

$$Q(\sigma) = -\sum_{r=1}^{d} H(i_r) + \sum_{r=1}^{d} I(i_r, j_r).$$

The Chow-Liu dependence structure defines a product approximation of a known probability distribution $\mathbb{P}$ by

$$\mathbb{P}^{(\sigma)}(\underline{x}) = \mathbb{P}_{i_1}(x_{i_1}) \prod_{r=2}^{d} \mathbb{P}(x_{i_r}|x_{j_r}).$$

The following theorem is the first main result in Chow and Liu [28] (1968).

Theorem 15.10. *Let $\mathbb{P}$ be a probability distribution over $\mathcal{X}$. Let $\mathcal{G} = ((V,U)|\mathbf{w})$ be a complete weighted graph with $\mathbf{w}$ given by*

$$\mathbf{w}(j,k) = I(j,k) \qquad \langle j,k\rangle \in U$$

where the $I(j,k)$s are computed using the $\mathbb{P}_{j,k}$s. Then the maximum weight spanning tree of $\mathcal{G}$ defines a Chow-Liu dependence structure σ, which maximises

$$Q(\sigma) = -\sum_{r=1}^{d} H(i_r) + \sum_{r=2}^{d} I(i_r, j_r).$$

Proof Firstly, $\sum_{r=1}^{d} H(i_r) = \sum_{i=1}^{d} H(i)$ so that the first term in $Q(\sigma)$ is independent of σ, hence the problem is equivalent to the maximisation of $\sum_{r=1}^{d} I(i_r, j_r)$. □

15.4.2 Chow-Liu Algorithm with Unknown $\mathbb{P}$

For $\mathbb{P}$ unknown, suppose there is an $n \times d$ data matrix $\mathbf{x}$, where $\mathbf{x} = \begin{pmatrix} \underline{x}_{(1)} \\ \vdots \\ \underline{x}_{(n)} \end{pmatrix}$. Each $\underline{x}_{(j)} \in \mathcal{X}$ for $j = 1, \ldots, n$.

Let $\mathcal{P}(\mathcal{X})$ denote the space of all probability distributions over $\mathcal{X}$; that is

$$\mathcal{P}(\mathcal{X}) = \{\mathbb{P} | \mathbb{P} = \{\mathbb{P}(\underline{x})\}_{\underline{x} \in \mathcal{X}}\}$$

Let $\mathcal{T}_d = (V, \sigma)$ be a spanning tree on V, where σ is a Chow-Liu dependence structure. Let $\mathbb{T}_d$ denote the set of all spanning trees, then

$$\mathcal{P}(\mathcal{X}, \mathbb{T}_d) = \{\mathbb{P}^{(\sigma)}\}$$

is the set of all tree dependent probability distributions on $\mathcal{X}$ and $\mathcal{P}(\mathcal{X}, \mathbb{T}_d) \subset \mathcal{P}(\mathcal{X})$. The empirical probability is defined as

$$\widehat{\mathbb{P}}_n(\underline{x}) = \frac{1}{n} \sum_{k=1}^{n} \mathbf{1}_{\underline{x}}(\underline{x}_{(k)}).$$

Lemma 15.11. *The maximum likelihood estimate $\widehat{\mathbb{P}}^{(ML)}$ is given by*

$$\widehat{\mathbb{P}}^{(ML)} = \arg \min_{\mathbb{P} \in \mathcal{P}(\mathcal{X}, \mathbb{T}_d)} D_{KL}(\widehat{\mathbb{P}}_n \| \mathbb{P}).$$

Proof

$$D_{KL}(\widehat{\mathbb{P}}_n \| \mathbb{P}) = -\widehat{H}_n(V) - \sum_{\underline{x} \in \mathcal{X}} \widehat{\mathbb{P}}_n(\underline{x}) \ln \mathbb{P}(\underline{x}),$$

where $\widehat{H}_n(V) = -\sum_{\underline{x} \in \mathcal{X}} \widehat{\mathbb{P}}_n(\underline{x}) \ln \widehat{\mathbb{P}}_n(\underline{x})$. Note that this does not depend on the tree structure. For the other part,

$$\sum_{\underline{x} \in \mathcal{X}} \widehat{\mathbb{P}}_n(\underline{x}) \ln \mathbb{P}(\underline{x}) = \frac{1}{n} \sum_{j=1}^{n} \ln \mathbb{P}(\underline{x}_{(j)}),$$

which is the log likelihood function. Hence, the maximum likelihood estimate is equivalent to the *reverse I-projection* of $\widehat{\mathbb{P}}_n$ onto the set of tree dependent distributions $\mathcal{P}(\mathcal{X}, \mathbb{T}_d)$. □

15.4.3 The Log Likelihood Function

When $\sigma = (i_r, j_r)_{r=1}^{d}$ is a Chow-Liu dependence structure, the parameter set $\mathcal{P}$ is the set of two dimensional distributions given by

$$\mathcal{P} = \{\mathbb{P}_{i,j} | (i, j) \in V \times V, i \neq j\}.$$

The corresponding parametric probability is

$$\mathbb{P}^{(\sigma)}(\underline{x}) = \mathbb{P}_{i_1}(x_{i_1}) \prod_{r=2}^{d} \mathbb{P}_{i_r|j_r}(x_{i_r}|x_{j_r}) = \prod_{r=1}^{d} \mathbb{P}_{i_r}(x_{i_r}) \prod_{r=2}^{d} \frac{\mathbb{P}_{i_r,j_r}(x_{i_r}, x_{j_r})}{\mathbb{P}_{i_r}(x_{i_r}) \mathbb{P}_{j_r}(x_{j_r})} \qquad \underline{x} = (x_j)_{j=1}^{d} \in \mathcal{X}.$$

The likelihood function is therefore

$$L(\sigma, \mathcal{P}) = \prod_{j=1}^{n} \mathbb{P}^{(\sigma)}(\underline{x}_{(j)}|\mathcal{P})$$

and the log likelihood function, divided by n, is

$$\mathcal{L}(\sigma, \mathcal{P}) = \frac{1}{n} \sum_{j=1}^{n} \ln \mathbb{P}(\underline{x}_{(j)}|\sigma, \mathcal{P}).$$

which may be re-written as

$$\mathcal{L}(\sigma, \mathcal{P}) = \frac{1}{n} \sum_{x \in \mathcal{X}_{i_1}} N(x) \ln \mathbb{P}_{i_1}(x) + \frac{1}{n} \sum_{r=2}^{d} \sum_{(x,y) \in \mathcal{X}_{i_r} \times \mathcal{X}_{j_r}} N(x,y) \ln \mathbb{P}_{i_r,j_r}(x,y) - \frac{1}{n} \sum_{j=2}^{d} \sum_{x \in \mathcal{X}_{j_r}} N(x) \ln \mathbb{P}_{j_r}(x)$$

where $N(x)$ denotes number of appearances of the appropriate configuration x in the data matrix $\mathbf{x}$. This reduces to

$$\mathcal{L}(\sigma, \mathcal{P}) = \sum_{x \in \mathcal{X}_{i_1}} \widehat{\mathbb{P}}_{n;i_1}(x) \ln \mathbb{P}_{i_1}(x) + \sum_{r=2}^{d} \sum_{(x,y) \in \mathcal{X}_{i_r} \times \mathcal{X}_{j_r}} \widehat{\mathbb{P}}_{n;i_r,j_r}(x,y) \ln \mathbb{P}_{i_r,j_r}(x,y) - \sum_{j=2}^{d} \sum_{x \in \mathcal{X}_{j_r}} \widehat{\mathbb{P}}_{n;j_r}(x) \ln \mathbb{P}_{j_r}(x).$$

Using the notation $\mathbb{P}_{i_r|j_r} = \frac{\mathbb{P}_{i_r,j_r}}{\mathbb{P}_{j_r}}$, this may be written as

$$\mathcal{L}(\sigma, \mathcal{P}) = \sum_{x \in \mathcal{X}_{i_1}} \widehat{\mathbb{P}}_{n;i_1}(x) \ln \mathbb{P}_{i_1}(x) + \sum_{r=2}^{d} \sum_{y \in \mathcal{X}_{j_r}} \widehat{\mathbb{P}}_{n;j_r}(y) \sum_{x \in \mathcal{X}_{i_r}} \widehat{\mathbb{P}}_{n;i_r|j_r}(x,y) \ln \mathbb{P}_{i_r|j_r}(x|y). \tag{15.6}$$

The log likelihood $\mathcal{L}(\sigma, \mathcal{P})$ is to be maximised. For a fixed structure σ, it therefore follows from Gibb's inequality that the maximum likelihood estimates are:

$$\mathbb{P}_{i_1}^{(ML)} = \widehat{\mathbb{P}}_{n;i_1} \qquad \mathbb{P}_{i_r|j_r}^{(ML)} = \frac{\widehat{\mathbb{P}}_{n;i_r,j_r}}{\widehat{\mathbb{P}}_{n;j_r}} \qquad r = 2, \ldots, d.$$

from which

$$\begin{aligned}
\mathcal{L}(\sigma, \mathcal{P}^{(ML)}) &= \sum_{x \in \mathcal{X}_{i_1}} \widehat{\mathbb{P}}_{i_1}(x) \ln \widehat{\mathbb{P}}_{n;i_1}(x) + \sum_{r=2}^{d} \sum_{(x,y) \in \mathcal{X}_{i_r} \times \mathcal{X}_{j_r}} \widehat{\mathbb{P}}_{n;i_r,j_r}(x,y) \ln \frac{\widehat{\mathbb{P}}_{n;i_r,j_r}(x,y)}{\widehat{\mathbb{P}}_{n;j_r}(y)} \\
&= \sum_{r=1}^{d} \sum_{x \in \mathcal{X}_{i_r}} \widehat{\mathbb{P}}_{n;i_r}(x) \ln \widehat{\mathbb{P}}_{n;i_r}(x) + \sum_{r=2}^{d} \sum_{(x,y) \in \mathcal{X}_{i_r} \times \mathcal{X}_{j_r}} \widehat{\mathbb{P}}_{n;i_r,j_r}(x,y) \ln \frac{\widehat{\mathbb{P}}_{n;i_r,j_r}(x,y)}{\widehat{\mathbb{P}}_{n;i_r}(x) \widehat{\mathbb{P}}_{n;j_r}(y)} \\
&= \sum_{r=1}^{d} \sum_{x \in \mathcal{X}_{i_r}} \widehat{\mathbb{P}}_{n;i_r}(x) \ln \widehat{\mathbb{P}}_{n;i_r}(x) + \sum_{r=2}^{d} \widehat{I}(i_r, j_r)
\end{aligned}$$

where

$$\widehat{I}(i_r, j_r) = \sum_{(x,y) \in \mathcal{X}_{i_r} \times \mathcal{X}_{j_r}} \widehat{\mathbb{P}}_{n;i_r,j_r}(x,y) \ln \frac{\widehat{\mathbb{P}}_{n;i_r,j_r}(x,y)}{\widehat{\mathbb{P}}_{n;i_r}(x) \widehat{\mathbb{P}}_{n;j_r}(y)}$$

is the plug in estimate of the mutual information. Clearly, the first term in the expression for $\mathcal{L}(\sigma, \mathcal{P}^{(ML)})$ does not depend on σ and hence the maximum likelihood estimate $\sigma^{(ML)}$ is given by

$$\sigma^{(ML)} = \text{argmax}_\sigma \left\{ \sum_{r=2}^{d} \widehat{I}(i_r, j_r) \right\}.$$

The number of spanning trees on d nodes is d^{d-2}. This is Cayley's formula. An exhaustive search is not feasable in practise. Besides, as pointed out by Chow - Liu [28], a *greedy approach* finds the maximal spanning tree.

There are several well known standard algorithms for finding the spanning tree of maximum weight, for example *Kruskal's algorithm* and *Prim's algorithm*. These algorithms are almost identical and find the maximum weight spanning tree in $O(d^2 \ln d)$ time.

Kruskal's algorithm Kruskal's Algorithm runs as follows:

1. The d variables yield $d(d-1)/2$ edges. The edges are indexed in decreasing order, according to their weights $b_1, b_2, b_3, \ldots, b_{d(d-1)/2}$.
2. The edges b_1 and b_2 are selected. Then the edge b_3 is added, *if it does not form a cycle*.
3. This is repeated, through $b_4, \ldots b_{d(d-1)/2}$, in that order, adding edges if they do not form a cycle and discarding them if they form a cycle.

This procedure returns a unique tree if the weights are different. If two weights are equal, one may impose an arbitrary ordering. From the $d(d-1)/2$ edges, exactly $d-1$ will be chosen.

Lemma 15.12. *Kruskal's algorithm returns the tree with the maximum weight.*

Proof The result may be proved by induction. It is clearly true for 2 nodes. Assume that it is true for d nodes and consider a collection of $d+1$ nodes, labelled $(X_1, X_2, \ldots, X_{d+1})$, where they are ordered so that for each $j = 1, \ldots, d+1$, the maximal tree from $(X_1, \ldots, X_j)$ gives the maximal tree from any selection of j nodes from the full set of $d+1$ nodes. Let $b_{(i,j)}$ denote the weight of edge (i,j) for $1 \le i < j \le d+1$. Edges will be considered to be undirected. Let $\mathcal{T}_j^{(d+1)}$ denote the maximal tree obtained by selecting j nodes from the $d+1$ and consider $\mathcal{T}_{d+1}^{(d+1)}$.

Let Z denote the leaf node in $\mathcal{T}_{d+1}^{(d+1)}$ such that among all leaf nodes in $\mathcal{T}_{d+1}^{(d+1)}$ the edge (Z,Y) in $\mathcal{T}_{d+1}^{(d+1)}$ has the smallest weight. Removing the node Z gives the maximal tree on d nodes from the set of $d+1$ nodes. This is seen as follows. Clearly, there is no tree with larger weight that can be formed with these d nodes, otherwise the tree on d nodes with larger weight, with the addition of the leaf (Z,Y) would be a tree on $d+1$ nodes with greater weight than $\mathcal{T}_{d+1}^{d+1}$. It follows that $Z = X_{d+1}$ and hence that X_{d+1} is a leaf node of $\mathcal{T}_{d+1}^{(d+1)}$.

By the inductive hypothesis, $\mathcal{T}_d^{(d+1)}$ may be obtained by applying Kruskal's algorithm to the weights $(b_{(i,j)})_{1 \le i < j \le d}$. Now consider an application of Kruskal's algorithm to the weights $(b_{(i,j)})_{1 \le i < j \le d+1}$ and note that for any (i,j) with $i < j$ such that the undirected edge (X_i, X_j) forms part of the tree $\mathcal{T}_d^{(d+1)}$, $b_{(i,d+1)} < b_{(i,j)}$ and $b_{(j,d+1)} < b_{(i,j)}$. Therefore, if the edges $(b_{(i,j)})_{1 \le i < j \le d+1}$ are listed according to their

weight and the Kruskal algorithm applied, then all the edges used in $\mathcal{T}_d^{(d+1)}$ will appear further up the list than any edge $(b_{(k,d+1)})_{k=1}^d$ and therefore all the edges of $\mathcal{T}_d^{(d+1)}$ will be included by the algorithm before the edges $(b_{(k,d+1)})_{k=1}^d$ are considered. It follows that $\mathcal{T}_{d+1}^{(d+1)}$ is the graph obtained by applying Kruskal's algorithm to the nodes $(X_1, \ldots, X_{d+1})$. □

Corollary 15.13 (Prim's Algorithm). *The tree of maximal weight may be chosen by choosing any initial node C_i, adding a link $C_i - C_j$ where j is chosen such that $b_{ij} = \max_k b_{ik}$ and at each stage, adding the node C_a to the tree that maximises b_{ak} over nodes C_k already in the tree.*

Proof It is clear that, with the same ordering of the weight, Prim's algorithm returns the same tree as Kruskal's algorithm. □

15.4.4 The Chow-Liu Algorithm and Polytrees

A probability distribution $\mathbb{P}_{X_1,\ldots,X_d}$ factorises according to a *polytree* if there is an ordering of the variables σ such that

$$\mathbb{P}_{X_1,\ldots,X_d} = \prod_{j=1}^{d} \mathbb{P}_{X_{\sigma(j)}|\Pi_j^{(\sigma)}},$$

$$\Pi_j^{(\sigma)} \subseteq \{X_{\sigma(1)}, \ldots X_{\sigma(j-1)}\}$$

and where the directed graph, formed by placing directed edges from each variable in $\Pi_j^{(\sigma)}$ to $X_{\sigma(j)}$ for $j = 1, \ldots, d$ is a *tree*.

A distribution that factorises along a polytree satisfies the condition: if $\Pi_j^{(\sigma)} = \{Y_1^{(\sigma,j)}, \ldots, Y_m^{(\sigma,j)}\}$, then

$$\mathbb{P}_{\Pi_j^{(\sigma)}} = \prod_{k=1}^{m} \mathbb{P}_{Y_k^{(\sigma,j)}}.$$

To extend the Chow-Liu algorithm to polytrees, the *conditional* mutual information is required;

$$I(A, C|B) = \sum_{\underline{x}_{A\cup B\cup C}} \mathbb{P}_{A\cup B\cup C}(\underline{x}_{A\cup B\cup C}) \ln \frac{\mathbb{P}_{A\cup C|B}(\underline{x}_A, \underline{x}_C|\underline{x}_B)}{\mathbb{P}_{A|B}(\underline{x}_A|\underline{x}_B)\mathbb{P}_{C|B}(\underline{x}_C|\underline{x}_B)}.$$

The following lemma is required.

Lemma 15.14. *If $A \perp_{\mathbb{P}} B|C$, then*

$$\min(I(A, C), I(B, C)) \geq I(A, B).$$

Proof This follows from observing that if $A \perp_{\mathbb{P}} B|C$, then

$$I(A, B) + I(A, C|B) = I(A, C), \qquad I(A, B) + I(C, B|A) = I(B, C),$$

which follows from:

$$A \perp_{\mathbb{P}} B|C \Leftrightarrow \mathbb{P}_{A\cup C|B} = \frac{\mathbb{P}_{A\cup B\cup C}}{\mathbb{P}_B} = \frac{\mathbb{P}_{A\cup C}\mathbb{P}_{B\cup C}}{\mathbb{P}_B\mathbb{P}_C}$$

and hence

$$I(A, C|B) = \sum \mathbb{P}_{A\cup B\cup C} \ln \frac{\mathbb{P}_{A\cup C}\mathbb{P}_B}{\mathbb{P}_{A\cup B}\mathbb{P}_C} = I(A, C) - I(A, B).$$

The other is similar. □

Theorem 15.15. *Suppose that $\mathbb{P}$ factorises according to a polytree. Kruskal's algorithm will locate the skeleton of the polytree.*

Proof Let $A = \{i\}$, $B = \{j\}$ and $D = \{k\}$ be three distinct nodes. Assume that $i \perp_{\mathbb{P}} j|k$. This can happen in the following cases:

$$i \to k \to j, \qquad i \leftarrow k \leftarrow j, \qquad i \leftarrow k \to j,$$

where $m \to n$ indicates a directed path. In all cases, $I(i,k|j) > 0$ and $I(k,j|i) > 0$. It follows that

$$\min(I(i,k), I(j,k)) > I(i,j).$$

Kruskal's algorithm takes the edge of largest weight that does not form a cycle. The algorithm will therefore not choose the edge (i,j) if there is a node k between i and j in σ.

For $i \to k \leftarrow j$, $i \perp_{\mathbb{P}} j$ and hence the edge $i - j$ will not be chosen by Kruskal's algorithm. □

It is straightforward to find appropriate directions for the edges; if there are edges $i - j - k$, then the edges take directions $i \to j \leftarrow k$ if and only if $I(i,k) = 0$.

15.5 Asymptotic Consistency of the Maximum Likelihood Estimate

Let $\mathbb{W}(\mathcal{T}_d^{(ML)}(n)) = \sum_{r=2}^{d} \widehat{I}(i_r, j_r)$ denote the weight of the tree computed by the Chow-Liu algorithm. Suppose that there is a distribution $\mathbb{P}^0$ which is the true, but unknown distribution. Let

$$\mathbb{P}^{(\sigma)0} = \text{argmin}_{\mathbb{P} \in \mathcal{P}(\mathcal{X}, \mathbb{T}_d)} D_{KL}(\mathbb{P}^0 \| \mathbb{P}).$$

Then $\mathbb{P}^{(\sigma)0}$ is the reverse I-projection of $\mathbb{P}^0$ and the corresponding structure $\sigma^{(0)} = (i_r^0, j_r^0)_{j=2}^d$ is the Chow-Liu dependence structure. If $\mathbb{P} \in \mathcal{P}(\mathcal{X}, \mathbb{T}_d)$, then $\mathbb{P}^{(\sigma)0} = \mathbb{P}^0$.

Let $\mathcal{T}_d^0$ denote the tree structure corresponding to $\sigma^{(0)}$, then

$$\mathbb{W}(\mathcal{T}_d^0) = \sum_{r=2}^{d} I^0(i_r^0, j_r^0)$$

where $I^0(i_r^0, j_r^0)$ are the mutual informations computed with $\mathbb{P}^{0\downarrow(i_r^0, j_r^0)}$, which is the tree of maximal weight.

Let

$$\widehat{\mathbb{P}}_{ML;n}^{(\sigma_{ML})} = \text{argmin}_{\mathbb{P} \in \mathcal{P}(\mathcal{X}, \mathbb{T}_d)} D_{KL}(\mathbb{P}_n \| \mathbb{P})$$

where $\mathbb{P}_n$ denotes the empirical distribution and σ_{ML} denotes the maximum likelihood Chow-Liu dependence structure. Let $\mathbb{W}(\mathcal{T}; n)$ denote the weight of tree $\mathcal{T}$ based on probability distribution $\mathbb{P}_n$ and, in particular, let $\mathbb{W}(\mathcal{T}_d^{(ML)}(n); n)$ denote the Chow Liu dependence tree weight based on σ_{ML} and $\widehat{\mathbb{P}}_n$. Then the following result holds:

Theorem 15.16.

$$\mathbb{W}(\mathcal{T}_d^{(ML)}(n); n) \overset{n \to +\infty}{\longrightarrow} \mathbb{W}(\mathcal{T}_d^0) \qquad \mathbb{P}^0 - a.s.$$

Proof This is a consequence of the strong law of large numbers; firstly, since $\mathcal{X}$ is finite, the strong law of large numbers gives that

$$\max_{x\epsilon\mathcal{X}} |\mathbb{P}_n(x) - \mathbb{P}^0(x)| \overset{n\to+\infty}{\longrightarrow} 0 \qquad \mathbb{P}^0 - a.s.$$

from which a.s. convergence of all empirical marginal distributions follows and in particular

$$\widehat{I}_n(i,j) \overset{n\to+\infty}{\longrightarrow} I^0(i,j)$$

for all pairs (i,j). It follows that for each tree $\mathcal{T}_d$,

$$\mathbb{W}(\mathcal{T}_d;n) \overset{n\to+\infty}{\longrightarrow} \mathbb{W}(\mathcal{T}_d) \qquad \mathbb{P}^0 - a.s.$$

and hence, since $\mathbb{T}_d$ is a finite set,

$$\max_{\mathcal{T}_d\epsilon\mathbb{T}_d} |\mathbb{W}(\mathcal{T}_d;n) - \mathbb{W}(\mathcal{T}_d)| \overset{n\to+\infty}{\longrightarrow} 0.$$

Note that, by construction, $\mathbb{W}(\mathcal{T}_d;n) \leq \mathbb{W}(\mathcal{T}_d^{(ML)}(n);n)$ for all $\mathcal{T}_d$ and each n. Now let

$$\mathbb{T}_d^0 = \{\mathcal{T}_d \in \mathbb{T}_d | \mathbb{W}(\mathcal{T}_d) = \mathbb{W}(\mathcal{T}_d^0)\}.$$

Since $\mathbb{T}_d$ is finite, there is a positive constant δ such that

$$\delta = \min_{\mathcal{T}_d\epsilon\mathbb{T}_d\setminus\mathbb{T}_d^0} |\mathbb{W}(\mathcal{T}_d^0) - \mathbb{W}(\mathcal{T}_d)| > 0.$$

Choose n large enough such that $\mathbb{P}^0$ a.s.

$$\max_{\mathcal{T}_d\epsilon\mathbb{T}_d} |\mathbb{W}(\mathcal{T}_d;n) - \mathbb{W}(\mathcal{T}_d)| \leq \frac{\delta}{2}.$$

There is an n_δ such that this holds for all $n \geq n_\delta$ and such that there is a tree in $\mathbb{T}_d^0$, say $\mathcal{T}_d^0$ such that $\mathbb{W}(\mathcal{T}_d^{(ML)}(n)) = \mathbb{W}(\mathcal{T}_d^0)$ and such that

$$|\mathbb{W}(\mathcal{T}_d^0;n) - \mathbb{W}(\mathcal{T}_d^0)| \leq \frac{\delta}{2}.$$

In other words, for any $\epsilon > 0$ with $\frac{\delta}{2} > \epsilon$, it holds that for $n > n_\epsilon$,

$$|\mathbb{W}(\mathcal{T}_d^{(ML)}(n);n) - \mathbb{W}(\mathcal{T}_d^0)| < \epsilon \qquad \mathbb{P}^0 - a.s.$$

□

This result does not assert convergence of the sequence of trees $\mathcal{T}_d^{(ML)}(n)$ unless the set $\mathbb{T}_d^0$ contains exactly one element.

15.6 Classification

Many of the techniques of supervised learning, or classification, involve a Bayes rule and an approximate distribution. Variables are of two types, *symptom* variables $\underline{X}_O$ (O for observable) and *class* variables, or *diagnosis* variables, $\underline{X}_C$. A prior distribution $\mathbb{P}_C$ is placed over the class variables, evidence is obtained in the form of an instantiation $\underline{x}_O$ of $\underline{X}_O$ of the symptom variables and the posterior distribution over the class variables obtained using Bayes rule;

$$\mathbb{P}_{C|O} = \frac{\mathbb{P}_C \mathbb{P}_{O|C}}{\mathbb{P}_O} \propto \mathbb{P}_C \mathbb{P}_{O|C}.$$

In *supervised* classification, the probabilities $\mathbb{P}_{O|C}$ are *learned*, by observing the instantiations $\underline{x}_O$ in training examples where $\underline{x}_C$ is given. When classifying (where the class $\underline{x}_C$ is unknown, the class that maximises $\mathbb{P}_C \mathbb{P}_{O|C}$ is chosen, for a given set of symptoms $\underline{x}_O$.

Often in classification, the distribution $\mathbb{P}_{O|C}$ has too many states and instead a set of lower dimensional marginals is considered:

$$\mathcal{P}(\underline{x}_C) = \{\mathbb{P}_{A_j|B_j,C}(.|.,\underline{x}_C) \qquad j = 1,\ldots,s\}$$

where for each $\underline{x}_C$, $S_C := (A_j, B_j)_{j=1}^s$ is a dependence structure. The dependence structures may depend on $\underline{x}_C$. The class variable $\underline{x}_C$ is then chosen to maximise

$$\mathbb{P}_C \mathbb{P}_{A_1|C} \prod_{j=1}^{s} \mathbb{P}_{A_j|B_j,C}.$$

The Naïve Classifier The *naïve classifier* considers $X_1,\ldots,X_d$ to be independent conditioned on C, so that the approximation $\mathbb{Q}$ to the probability distribution $\mathbb{P}$, given by

$$\mathbb{Q}_{\underline{X}|C} = \prod_{j=1}^{d} \mathbb{P}_{X_j|C}$$

is used. The aim is then, for an observation $\underline{x}$, to find the value c that maximises $\mathbb{P}_C \mathbb{Q}_{\underline{X}|C}(\underline{x}|.)$.

Classification comes in two stages; firstly, constructing the classifier. For constructing the classifier, a large number of observations of $\underline{X}$ are made, assumed independent, for each value of C, where the value of C is known. From this, $\mathbb{P}_{X_j|C}$ is estimated by $\hat{\mathbb{P}}_{X_j|C}(x|c) = \frac{n(x,c)}{n(c)}$ where $n(x,c)$ is the number of observations with $(X_j, C) = (x, c)$ in the sample.

If a prior distribution $\mathbb{P}_C$ has been placed over the class variable C, the score function is then

$$\mathbb{P}_C(c) \prod_{j=1}^{d} \hat{\mathbb{P}}_{X_j|C}(x_j|c)$$

and an observation $\underline{x}$ is assigned to the class c that maximises this function. If there is no prior, then the likelihood function $\prod_{j=1}^{d} \hat{\mathbb{P}}_{X_j|C}(x_j|c)$ is used.

Example 15.17.

In the article [28] by Chow and Liu, the example of character recognition is discussed. A person writes a number, $0, 1, 2, 3, 4, 5, 6, 7, 8$ or 9 in a rectangular space and the machine has to recognise which of the ten characters has been written. The rectangle is split into a 12×8 grid and each of the 96 spaces is coded as a 1 or a 0 depending on which character has been written. In the example, 7000 numerals were used as training examples to construct the classifier, which was then applied to 12000 examples, with a success rate of 91%.

Chow Liu Tree Suppose that $V = \{X_1, \ldots, X_d, C\}$, where $\underline{X} = (X_1, \ldots, X_d)$ is a random vector to be observed and C is a class variable. With classification, an observation $\underline{x}$ is assigned to the category c that maximises $p_C(.)p_{\underline{X}|C}(\underline{X}|.)$

The Chow - Liu tree presents an improvement over the naïve classifier. For each category $c \in \mathcal{C}$, the best fitting Chow Liu tree is estimated from the training variables;

$$\mathbb{Q}_{\underline{X}|C} = \prod_{j=1}^{d} \hat{\mathbb{P}}_{X_j|X_{\pi_c(j)},C}$$

and then the observation $\underline{x}$ is assigned to the category c that maximises the score function $S_{C,\underline{X}} = \mathbb{P}_C \mathbb{Q}_{\underline{X}|C}(\underline{x}|.)$ if there is a prior $\mathbb{P}_C$ over the categories, or the score function $S_{C,\underline{X}} = \mathbb{Q}_{\underline{X}|C}(\underline{x}|.)$ if the initial assessment is that all categories are equally likely.

The article [28] which introduced the Chow - Liu tree considers the problem of machine recognition of handwritten numerals, $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$. There are $c = 10$ pattern classes. Let a_i denote the numeral i. There is a prior distribution $\underline{p} = (p_0, p_1, \ldots, p_9)$ over the numerals. The number is written on a 12×8 rectangle and 96 binary measurements are used to represent the numeral; 1 if the cell contains writing and 0 otherwise. In the example given in [28], 19000 numerals produced by 4 inventory clerks were scanned. 7000 of these were employed as training examples, to find the best fitting trees and estimate the probabilities $p_0, \ldots, p_9$. The optimal trees for each of the 10 numerals were obtained. For the remaining numerals, the observation $\underline{x} = (x_1, \ldots, x_{96})$ was considered. By Bayes rule,

$$\mathbb{P}_{C|\underline{X}}(a_k|\underline{x}) = \frac{p_{\underline{X}|C}(\underline{x}|a_k)\mathbb{P}_C(a_k)}{\mathbb{P}_{\underline{X}}(\underline{x})} = \frac{\mathbb{P}_{\underline{X}|C}(\underline{x}|a_k)\mathbb{P}_C(a_k)}{\mathbb{P}_{\underline{X}}(\underline{x})}.$$

The quantity $\mathbb{P}_{\underline{X}|C}(\underline{x}|a_k)$ was estimated by $\mathbb{Q}_{\underline{X}|C}(\underline{x}|a_k)$ and the following classification rule was used: the numeral was declared to be of class a_k if $\mathbb{P}_C(a_k)\mathbb{Q}_{\underline{X}|C}(\underline{x}|a_k) \geq \mathbb{P}_C(a_i)\mathbb{Q}_{\underline{X}|C}(\underline{x}|a_i)$ for all $i \neq k$. Using the trees, the error rate was reduced from 0.09 to 0.04 compared with the model produced by assuming independence between the contents of the 96 cells.

Chapter 16

Constraint-Based Structure Learning Algorithms

16.1 Structure Learning

Let $X = (X_1, \ldots, X_d)$ be a random (row) vector and $\mathbf{X}$ an $n \times d$ random matrix, where each row is an i.i.d. copy of X. Let $\mathbf{x}$ be an $n \times d$ data matrix, where the modelling assumption is that $\mathbf{x}$ is an instantiation of $\mathbf{X}$.

The object of *structure learning* is to learn the DAG of a Bayesian Network for X from $\mathbf{x}$.

Structure learning algorithms fall broadly into two categories; *search and score*, and *constraint based*. Many of the learning algorithms available are *hybrid* algorithms, which involve both constraint-based and search-and-score principles.

For search and score algorithms, a score function is used to score each network, giving a high score if it returns a high value for the *score function*. This is usually based on the likelihood function (often the Cooper-Herskovitz likelihood, Equation (12.15)), which is combined with a penalisation term for models that have a large number of parameters. Since the search space tends to be large, these algorithms tend to be computationally expensive. They are considered in later chapters.

The other category of structure learning algorithm is *constraint based*. They tend to make the rather bold assumption that there exists a faithful DAG for the underlying probability distribution and work according to the principle of Theorem 2.3. That is, starting with a complete graph, they remove an edge $\alpha \sim \beta$ from the skeleton whenever a conditional independence statement $X_\alpha \perp X_\beta | X_S$ is established for some subset $S \subseteq V \backslash \{\alpha, \beta\}$. Here, using ϕ to denote the empty set, $X_\alpha \perp X_\beta | X_\phi$ means: $X_\alpha \perp X_\beta$. A resulting vee- structure $\alpha - \gamma - \beta$ (where $\alpha \not\sim \beta$) is declared an *immorality* if $\gamma \notin S$; it is declared *not* to be an immorality if $X_\gamma \in S$. The edges of the skeleton corresponding to immoralities are directed accordingly, the other compelled edges are directed and the algorithm returns an essential graph.

16.2 Testing for Conditional Independence

16.2.1 Gaussian variables

For multivariate Gaussian, a test of $X \perp Y|S$ may be carried out using *partial correlation*.

Definition 16.1 (Partial Correlation). *The* partial correlation $\rho_{X,Y|S}$ *between* X *and* Y *given* S *is defined as*

$$\rho(X - \Sigma_{XS}\Sigma_{SS}^{-1}S, Y - \Sigma_{YS}\Sigma_{SS}^{-1}S).$$

where Σ_{SS} is the covariance matrix of S, Σ_{XS} is the covariance between X and S, Σ_{YS} is the covariance between Y and S. The partial correlation may be viewed in the following way: regress X on S and regress Y on S; the partial correlation is the correlation between the residuals from these two regressions.

For multivariate Gaussian, $X \perp Y|S$ if and only if $\rho_{X,Y|S} = 0$. To test this, first regress X against S and store the residuals R_1, and regress Y against S and store the residuals R_2. The estimated partial correlation is the sample correlation between R_1 and R_2. Fisher's z-transform of the partial correlation is defined as:

$$z(\widehat{\rho}_{X,Y|S}) = \frac{1}{2}\log\left(\frac{1+\widehat{\rho}_{X,Y|S}}{1-\widehat{\rho}_{X,Y|S}}\right).$$

Consider the null hypothesis $H_0 : \rho_{X,Y|S} = 0$ versus the alternative $H_1 : \rho_{X,Y|S} \neq 0$ (two sided test). The null hypothesis is rejected at significance level α if and only if

$$\sqrt{n - |S| - 3}|z(\widehat{\rho}_{X,Y|S})| \geq z_{\alpha/2}$$

where z_α is the value such that $\mathbb{P}(Z \geq z_\alpha) = \alpha$ for $Z \sim N(0,1)$. The distribution of the sample partial correlation was described by Fisher (1924) [42].

Under the assumption that the variables are multivariate Gaussian, the statement $X \perp Y|S$ (where X and Y are variables and S is a vector) may be tested by considering $\widehat{\Sigma}^{-1}$, where Σ is the covariance matrix of (X, Y, S) and $\widehat{\Sigma}^{-1}$ is either the inverse, or a generalised inverse, of Σ. If $X \perp Y|S$ then $(\Sigma^{-1})_{XY} = 0$.

16.2.2 Discrete Variables

For discrete variables, testing for conditional independence is carried out, quite simply, using the usual χ^2 test. To test whether or not $X \perp Y|S$, let $n(x, y, s)$ denote the number of times $(X, Y, S) = (x, y, s)$ appears in the data, $n(x, s)$, $n(y, s)$, $n(s)$ the number of instances of $(X, S) = (x, s)$, $(Y, S) = (y, s)$, $S = s$ respectively. The G^2 statistic, which is standard, is defined as

$$G^2(X, Y, S) = 2\sum_{x,y,s} n(x, y, s)\log\frac{n(x, y, s)n(s)}{n(x, s)n(y, s)}. \tag{16.1}$$

Asymptotically, this is distributed as a χ^2 distribution on $(j_x - 1)(j_y - 1)j_s$ degrees of freedom, where j_x, j_y and j_s are the number of values that X, Y and S respectively can take.

16.2.3 Hypothesis Testing and Statistical Theory

There are two basic difficulties with the method of declaring $X \perp Y|S$ when the null hypothesis is not rejected at a significance level α. The first is that while the *nominal* significance is α, there are rather many tests carried out. All that can be said about the true significance level is that it is less than $N\alpha$, where N is the total number of tests carried out. Nevertheless, for a single hypothesis test, a result 'reject H_0' at significance level α is a good indicator that the data suggests that the alternative hypothesis H_1 is true; if H_0 is rejected, then the dependence represented by H_1 is clearly and distinctly present in the data matrix $\mathbf{x}$, even if it is not necessarily present in the probability distribution of the random vector X.

There is, however, a much more serious problem. In statistical theory, the conclusion reached when a test fails to reject the null hypothesis is, simply, 'there is insufficient evidence to reject the null hypothesis'. The 'court of law' metaphor is appropriate here; a 'not guilty' verdict may simply mean that the evidence is insufficient to establish guilt beyond all reasonable doubt. It does not establish that the defendant did not commit the crime. There are two possible reasons for a failure to reject a null hypothesis: either the null hypothesis happens to be true, or else the null hypothesis is false, but the test is not sufficiently powerful to detect this. The constraint based algorithms discussed all *accept* an independence statement $X \perp Y|S$ if the result of the test is 'do not reject independence'. The problem is that these independence statements are added to the list of constraints, and the output network satisfies the D-separation statements, even if they contradict some of the 'reject conditional independence' statements that have been obtained by *rejecting* a null hypothesis of conditional independence.

Several approaches have been suggested to try and limit acceptance of conditional independence that is incompatible with independence statements rejected. A. Fast in [41] suggests using the power of the test, but points out the computational difficulties with this. He does not, though, address the problem that if $X \not\perp Y|S$, then the corresponding D-connection statement should be in the network. Blomberg and Margaritis in [9] formalise the identification of all inconsistencies that stem from standard probability theory and provide respective algorithms.

All the constraint based algorithms discussed need to be modified to ensure that the conditional dependence statements obtained by *rejecting* conditional independence statements correspond to D-connection statements in the resulting DAG.

Some of the most prominent constraint-based algorithms are now described.

16.3 The K2 Structural Learning Algorithm

Let $\underline{X} = (X_1, \ldots, X_d)$ and $V = \{X_1, \ldots, X_d, C\}$, where C is a class variable, $(X_1, \ldots, X_d)$ are variables to be observed, from which the class should be inferred.

The K2 structure learning algorithm, introduced by Cooper and Herskovitz, is an algorithm to locate associations between the variables $(X_1, \ldots, X_d)$. Since the number of entries required to define the conditional probability functions increases exponentially with the number of parents, the algorithm limits the number of parents a node can take. An upper limit of four parents is a value widely used.

The algorithm *assumes* that an order has been established for the d nodes $X_1, \ldots, X_d$ so that, for each i, the parent nodes Pa_i for variable X_i are established among the nodes $X_1, \ldots, X_{i-1}$. For $j = 1, \ldots, i-1$, the empirical Kullback - Leibler divergence between the two empirical probability distributions of $(X_1, \ldots, X_i)$, one determined by the graphs with and the other determined by the graph without the directed edge (i, j), is measured and the edge is retained if a) the divergence is sufficiently large and b) node i does not already have 4 parents. That is, if Pa_i is the current parent set of X_i and X_j is under consideration, the quantity in question is

$$Q(i,j) = \sum \widehat{\mathbb{P}}_{X_i,\mathrm{Pa}_i,X_j} \log \frac{\widehat{\mathbb{P}}_{X_i,\mathrm{Pa}_i,X_j}\widehat{\mathbb{P}}_{\mathrm{Pa}_i}}{\widehat{\mathbb{P}}_{X_i,\mathrm{Pa}_i}\widehat{\mathbb{P}}_{\mathrm{Pa}_i,X_j}} = \sum \widehat{\mathbb{P}}_{X_i,\mathrm{Pa}_i,X_j} \log \frac{\widehat{\mathbb{P}}_{X_i|\mathrm{Pa}_i,X_j}}{\widehat{\mathbb{P}}_{X_i|\mathrm{Pa}_j}}.$$

Under the null hypothesis, that $X_i \perp X_j | \mathrm{Pa}_i$, $2nQ(i,j) \sim \chi^2_{n(\mathrm{Pa}_i)(n(X_i)-1)(n(X_j)-1)}$, where $n(X_i)$ denotes the number of elements in the state space of X_i; similarly for X_j and Pa_i.

The resulting algorithm is a *greedy algorithm*, with all the advantages and disadvantages that this implies.

When the K2 algorithm is used, the learnt structure depends entirely on the order chosen for the variables generated before the learning process starts. It is therefore usual to repeat the algorithm with several different randomly chosen orders (say 1000) and choose the best; the one with the lowest Kullback Leibler divergence between the fitted distribution and the empirical distribution.

Example 16.2 (Robotics)**.**

This example is taken from the paper [79]. It shows an application to Bayesian network learning techniques for task execution in mobile robots. The task here is for the robot to locate an open door and travel through it.

The robot emits sonar pulses and is equipped with eight detectors, which detect the echoes. From this information, it has to decide where the door is located.

An action has to be taken: step to left, right, or straight ahead. This is the class variable and the class has to be determined by the signals received by the eight detectors. Since the signals are not independent of each other (the echoes may be created by the same object), the model is improved by incorporating a dependence structure.

In this experiment, the problem is to learn the *structure* of the Bayesian network and to estimate the probability potentials from the training data base.

The K2 algorithm is employed to establish a suitable structure. For the robot learning example, the maximum number is set to four. The size of the probability potentials cannot be too large, since the robot is expected to find the door and travel through it in real time.

The intensity of an echo may be modelled as a continuous random variable, but the variables are *discretised* for computational convenience. In general, it is not convenient to use a variable with more than 20 different values.

In the Bayesian Robotics experiment, the experiments were repeated 1000 times and nets with optimal values selected.

The resulting network for the eight variables is shown in Figure 16.1.

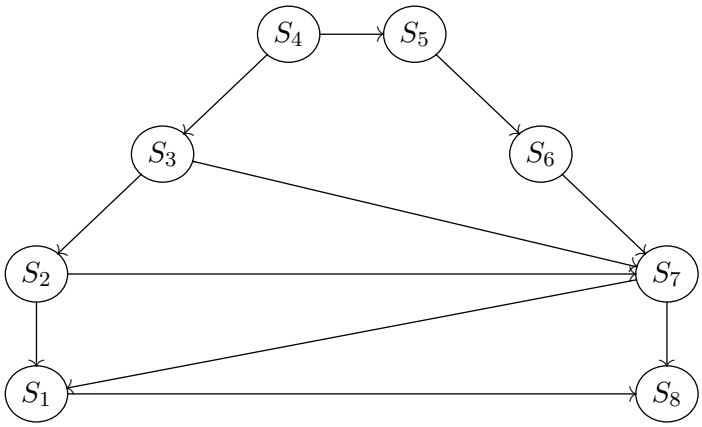

Figure 16.1: Network produced by the K2 algorithm. Here the nodes S_j represent the signals received by the sensors. The variable C, not shown, which is a parent to all the variables shown, denotes the class variable, the action to be performed.

In addition to the 8 variables shown in the network, there is also a *class* variable C, the direction to be taken, which is a *parent* of all the nodes in $X = (S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8)$. The network is estimated using a uniform prior distribution over C, which is an ancestor variable for each random ordering chosen for the nodes in X; the action performed is the action that maximises $\widehat{p}_{X,C}$ where $\widehat{p}$ is the estimate of the distribution from the training examples, factorised according to the DAG in Figure 16.1. □

16.4 Three phase dependency analysis

The *three phase dependency analysis* algorithm (denoted TPDA) was introduced by Cheng, Greiner, Kelly, Bell and Liu (2002) [23], who write, 'this TPDA algorithm is correct (i.e., will produce the perfect model of the distribution) given a sufficient quantity of training data whenever the underlying model is monotone DAG faithful.' The algorithm requires the faithfulness assumption to hold and relies on Theorem 2.3. The TPDA algorithm works in three phases; *draughting*, *thickening* and *thinning*, outlined in Algorithm 4, which gives the main steps of the algorithm. A precise description of the algorithm and proof that it returns a faithful DAG when it exists, is straightforward to establish and is found in [23].

Strictly speaking, the TPDA algorithm is a hybrid algorithm, since the first stage (draughting) is the Chow-Liu tree, which is a search and score procedure.

16.5 Fast Adjacency Search (FAS) algorithm

The FAS algorithm is perhaps the simplest constraint-based algorithm. Firstly, an *input order* of the variables, $(X_1, \ldots, X_d)$ and then Algorithm 5 is applied. The algorithm works on the principle

Algorithm 4 The Three Phase Dependency Analysis Algorithm

Stage 1: Draughting Locate the Chow - Liu tree

This stage is simply Kruskal's algorithm

Stage 2: Thickening Add edges

for $i = 1, \ldots, d-1$, $j = i+1, \ldots, d$ **do**

Let $C_{i,j}$ be the set of neighbours of X_i or X_j on a path between X_i and X_j

if $X_i \not\perp X_j | C$ for any subset $C \subseteq C_{ij}$ **then**

add an edge $X_i \sim X_j$

else

do not add an edge $X_i \sim X_j$

and let $S_{i,j}$ denote the set such that $X_i \perp X_j | S_{ij}$

end if

end for

Stage 3: Thinning Removing unnecessary edges

for $i = 1, \ldots, d-1$, $j = i+1, \ldots, d$ **do**

Let C_{ij} denote common neighbours of X_i and X_j

if $X_i \sim X_j$ and there is a set $C \subseteq C_{ij}$ such that $X_i \perp X_j | C$ **then**

remove the edge between X_i and X_j.

end if

end for

Stage 4: Directing edges For each vee structure $X_i \sim X_k \sim X_j$, (X_i, X_k, X_j) is an immorality if $X_k \notin S_{ij}$, otherwise it is not. Once the immoralities have been added, the additional compelled edges are obtained using Meek's rules.

that there exists a faithful graphical representation for the probability distribution. First, a complete (undirected) graph is created. Then for $n = 0, 1, 2, \ldots$ an edge $\langle \alpha, \beta \rangle$ is removed if and only if there is a set $S_{\alpha,\beta}$ of size n such that $X_\alpha \perp X_\beta | X_{S_{\alpha,\beta}}$. This is the approach to finding the skeleton. A vee structure (α, γ, β) is declared to be an immorality if and only if $\gamma \notin S_{\alpha,\beta}$ (known as the minimal sepset). The remaining compelled edges are added using Meek's rules to obtain the essential graph. These are edges $\alpha \sim \beta$ that appear in structures given in Figure 2.10 Definition 2.16 Page 42 are directed as in the Figure 2.10.

16.6 PC and MMPC Algorithms

The PC algorithm was introduced by Spirtes, Glymour and Scheines [127] (1993) and was modified to produce the MMPC algorithm in [137] (2006). It is algorithm for locating the skeleton of a faithful DAG (should such a DAG exist) and hence to construct the essential graph. It works in three stages. Firstly, a forward stage starts with an empty graph, and adds in all possible edges. There are possibly too many edges after this stage. Secondly, a backward stage removes some of the edges. The resulting graph, after the second stage, will contain no false negatives, but may still contain some false positives. A third stage is implemented to remove the false positives. The algorithm runs as follows:

The algorithm starts with an input order for the variables $(X_1, \ldots, X_d)$. Stage 1 of the PC algorithm is given in Algorithm 6.

The MMPC differs from the PC in one aspect: there is a gentle change whereby at each stage the *best* variable is added into the parent set. Stage 1 of the MMPC algorithm is given in Algorithm 7.

After the first stage of the PC / MMPC algorithm, the candidate parent/children sets may contain too many variables. The next stage prunes them. This is Algorithm 8.

After Stages 1 and 2 of the PC / MMPC algorithm, there may still be false positives. Suppose a probability distribution may be represented by the DAG in Figure 16.2. Working from T, the node C may enter the output, and remain in the output.

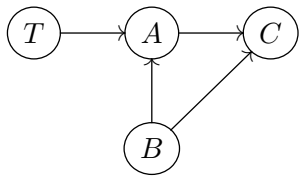

Figure 16.2: MMPC: A False Positive from Algorithm 8

This is because C is dependent on T, conditioned on all subsets of T's parents and children; namely, ϕ (the empty set) and $\{A\}$. Note that the *collider* connection TAB, is opened when A is instantiated so that, when A is instantiated and B is uninstantiated, T is d-connected with C. For ϕ (the empty set), TAC is a chain connection, where A is uninstantiated, so that T is D-connected to C.

Algorithm 5 The FAS Algorithm

Start with Full Undirected Graph
Stage 0
for $\alpha = 1, \ldots, d-1$, $\beta = i+1, \ldots, d$ **do**
  if $X_\alpha \perp X_\beta$ **then**
    remove edge $\langle \alpha, \beta \rangle$
    set $S_{\alpha,\beta} = \phi$
  end if
end for
for $k \geq 1$ **do**
  Stage k
  for $\alpha = 1, \ldots, d-1$, $\beta = i+1, \ldots, d$ **do**
    Work through sets $S \subset V \backslash \{\alpha, \beta\}$
    of size k from lowest to highest sum of indices
    if current graph contains edge $\langle \alpha, \beta \rangle$ **then**
      Test $X_\alpha \perp X_\beta | X_S$. If true, then remove edge $\langle \alpha, \beta \rangle$.
      set $S_{\alpha,\beta} = S$ and move to next (α, β) pair.
    end if
  end for
end for
Termination: Terminate when either all nodes have less than k neighbours, or else a pre-specified terminal value is reached.
for $\alpha = 1, \ldots, d-1, \beta = \alpha+1, \ldots, d, \gamma = 1, \ldots, d; \gamma \neq \alpha, \beta$ **do**
  if $\alpha - \gamma - \beta$ is a vee-structure **then**
    if $\gamma \notin S_{\alpha,\beta}$ **then**
      $\alpha - \gamma - \beta$ is an immorality
    else
      $\alpha - \gamma - \beta$ is not an immorality
    end if
  end if
end for
Now direct the additional compelled edges are obtained using Meek's rules.

Algorithm 6 The PC Algorithm: Stage 1

for For $i = 1, \ldots, d$ **do**
  initialise $\mathcal{Z}_0^{(i)} = \phi$, the empty set.
  these will become the parent/children sets of the variables
end for
for $i = 1, \ldots d$ **do**
  for $j = 1, \ldots, d$, $j \neq i$ **do**
    check whether $X_i \perp X_j | \mathcal{Z}_j^{(i)}$. If it is not, let $\mathcal{Z}_{j+1}^{(i)} = \mathcal{Z}_j^{(i)} \cup \{X_j\}$. If the independence statement holds, then set $\mathcal{Z}_{j+1}^{(i)} = \mathcal{Z}_j^{(i)}$ and set $S_{X_i X_j} = \mathcal{Z}_j^{(i)}$. $S_{X_i X_j}$ is known as the *sepset*, or separating set; a set that satisfies $X_i \perp X_j | S_{ij}$.
  end for
end for
Set $\mathcal{Z}^{(i)} = \mathcal{Z}_d^{(i)}$.

Algorithm 7 The MMPC Algorithm: Stage 1

for For $i = 1, \ldots, d$ **do**
  initialise $\mathcal{Z}_0^{(i)} = \phi$, the empty set.
  these will become the parent/children sets of the variables
end for
for $i = 1, \ldots, d$ **do**
  for $k = 1, \ldots, d$, $k \neq i$ **do**
    Let $j_k^* = \text{argmax}_{j \notin \{j_1^*, \ldots, j_{k-1}^*\}} G(X_i, X_j, \mathcal{Z}_{k-1}^{(i)})$.
    Check whether $X_i \perp X_{j^*} | \mathcal{Z}_{k-1}^{(i)}$. If it is not, let $\mathcal{Z}_k^{(i)} = \mathcal{Z}_{k-1}^{(i)} \cup \{X_{j^*}\}$. If the independence statement holds, then set $\mathcal{Z}_k^{(i)} = \mathcal{Z}_{k-1}^{(i)}$ and set $S_{X_i X_{j^*}} = \mathcal{Z}_{k-1}^{(i)}$. $S_{X_i X_{j^*}}$ is known as the *sepset*, or separating set; a set that satisfies $X_i \perp X_{j^*} | S_{ij^*}$.
  end for
end for
if $i = d$ **then**
  $\mathcal{Z}_d^{(i)} = \mathcal{Z}_{d-1}^{(i)}$
end if
Set $\mathcal{Z}^{(i)} = \mathcal{Z}_d^{(i)}$.

Algorithm 8 The PC / MMPC Algorithm: Stage 2

Suppose that $\mathcal{Z}^{(i)}$ contains k variables.
Label them $Y_1, \ldots, Y_k$. Let $\mathcal{Z}_k^{(i)} = \mathcal{Z}^{(i)}$.
for $j = 0, \ldots, k-1$ **do**
  Check whether there exists a set $S \subseteq \mathcal{Z}_{k-j} \backslash \{Y_{k-j}\}$ such that $X_i \perp Y_{k-j} | S$
  if there is **then**
    set $\mathcal{Z}_{k-j-1}^{(i)} = \mathcal{Z}_{k-j}^{(i)} \backslash \{Y_{k-i}\}$, and set $S_{X_i Y_{k-i}}$
  else
    Set $\mathcal{Z}_{k-j-1}^{(i)} = \mathcal{Z}_{k-j}^{(i)}$.
  end if
end for
Let $\mathcal{Z}^{(i)} = \mathcal{Z}_1^{(i)}$
This set contains all the variables which have an edge either to or from the variable X_i.

T and C are D-separated if and only if A and B are simultaneously instantiated; that is, $T \perp C | \{A, B\}$. But if B is independent from T given the empty set, so it will be removed from $\mathcal{Z}$. Therefore, the link TC will not be removed.

This is corrected by considering the parent / child sets of the other variables. When working from C, both A and B will be in the parent / child set, and $T \perp C | \{A, B\}$. The third stage of the algorithm (Algorithm 9) removes these false positives.

Algorithm 9 The PC / MMPC Algorithm: Stage 3

Let $(\mathcal{Z}^{(i)})_{i=1}^d$ denote the parent / child sets for all the variables arrived at after Algorithm 8.
Let $X_{\sigma(1)}, \ldots, X_{\sigma(k)}$ denote the set of variables in $\mathcal{Z}^{(i)}$, the parent child set for X_i arrived at after Algorithm 8
Set $\mathcal{Y}_0^{(i)} = \mathcal{Z}^{(i)}$.
for $j = 1, \ldots, k$ **do**
  set

$$\mathcal{Y}_j^{(i)} = \begin{cases} \mathcal{Y}_{j-1}^{(i)} \backslash \{X_{\sigma(j)}\} & X_i \notin \mathcal{Z}^{(\sigma(j))} \\ \mathcal{Y}_{j-1}^{(i)} & X_i \in \mathcal{Z}^{(\sigma(j))}. \end{cases}$$

end for
for $i = 1, \ldots, d$ **do**
  Set $\mathcal{Z}^{(i)} = \mathcal{Y}_k^{(i)}$
end for
This returns the complete parent / child set for X_i.
The sepsets for the variables removed in the third stage have already been established.

Establishing the Essential Graph Having recorded the sepsets, sets such that $X \perp Y | S_{XY}$, it is now straightforward to construct the essential graph. For each vee structure (X, Z, Y) (that is a structure such that $\{X, Y\} \subset \mathcal{Z}^{(Z)}$, but $X \notin \mathcal{Z}^{(Y)}$), check whether or not $Z \in S_{XY}$. If $Z \in S_{XY}$, then (X, Z, Y) is not an immorality; the edges $X - Z - Y$ remain undirected at this stage. If $Z \notin S_{XY}$, then (X, Z, Y) is an immorality.

Finally, add in the additional compelled edges using Meek's rules; edges $\alpha \sim \beta$ that appear in structures given in Figure 2.10 Definition 2.16 Page 42 are directed as in the Figure 2.10.

16.7 Recursive Autonomy Identification

The *Recursive Autonomy Identification* algorithm is from Yehezkel and Lerner [150] (2009). Like the FAS algorithm, it tries to keep the size of the sepsets as small as possible. The general idea is similar to the FAS algorithm, but it tries to locate, and use, more of the chain graph structure of the essential graph at each state. At stage $n+1$, instead of simply checking all possible subsets of size $n+1$ to determine whether or not there is a set S such that $X \perp Y | S$, only those components of the current chain graph after stage n that can have influence are considered. This reduces the number of tests that have to be carried out at stage $n+1$.

The algorithm assumes that there is a faithful graph and aims to locate its essential graph. When testing for independence, it checks all relevant tests of $X \perp Y | S$ for $X, Y \in V$ and $S \subset V$ for $|S| = n$ (the subset S has n variables) before making tests of $X \perp Y | S$ where $|S| = n+1$, since tests are less reliable when the conditioning sets are larger.

The first step of the algorithm is as follows.

- Starting with a variable set V, the initial graph is the complete graph, with undirected edges between each pair of variables $\{X, Y\}$.
- For each pair $\{X, Y\} \subset V$, it is checked whether or not $X \perp Y$ and if this holds, the edge $X - Y$ is removed. Record $S_{X,Y} = \phi$, the empty set ($S_{X,Y}$ is the separator).
- For each vee structure $X - Z - Y$ where there is no edge $X - Y$, the triple (X, Z, Y) is an immorality $X \to Z \leftarrow Y$.
- The remaining compelled edges are added.

For each pair $\{X, Y\}$ that do not have an edge between them, the set $S_{X,Y}$ used to determine the edge removal using $X \perp Y | S_{X,Y}$, is recorded.

After this initialisation (stage 0), the algorithm proceeds recursively. At stage n+1, do the following.

- Start with the *skeleton* from stage n. For each vee-structure $\alpha - \gamma - \beta$, the vee-structure is an immorality if $\gamma \notin S_{\alpha,\beta}$ and it is not an immorality if $\gamma \in S_{\alpha,\beta}$. Add in the remaining compelled edges. The resulting graph is the Stage n essential graph, which is a chain graph. Locate the chain components.

- Starting with a chain component that has no descendants and proceeding backwards, consider in turn each chain component $\mathcal{G}_C$ and the subgraph $\mathcal{G}_D$ formed by taking the chain component $\mathcal{G}_C = (C, U_C)$ together with the chain components that have parent variables of $\mathcal{G}_C$ and all all the directed edges connecting these chain components. Let D denote variable set for $\mathcal{G}_D$. For each $Y \in C$ and each neighbour X of Y (consider first the parents in different connected components, and then the undirected neighbours in the component $\mathcal{G}_C$), check whether there is a set $S_{XY} \subset D$ of size n such that $X \perp Y | S$. If there is, then remove the edge between X and Y and record S_{XY}. Remove the chain component $\mathcal{G}_C$ and proceed recursively until the whole graph has been considered.

This is repeated until the size of the largest neighbour set in the undirected graph is equal to n, then the algorithm terminates. Undirect all the edges, find the immoralities and add in the remaining compelled edges. The output is the resulting essential graph.

Note In [150], the algorithm presented is slightly different; once an edge is directed, it is not subsequently undirected. It is difficult to see the theoretical justification for this; the modification presented here ensures that, when there is a faithful graph and assuming a perfect oracle, the output graph is the essential graph.

Example 16.3 (Example for Recursive Autonomy Identification)**.**

Suppose that the DAG in Figure 16.3 is faithful to the distribution $\mathbb{P}_{X_1,X_2,X_3,X_4,X_5,X_6,X_7}$.

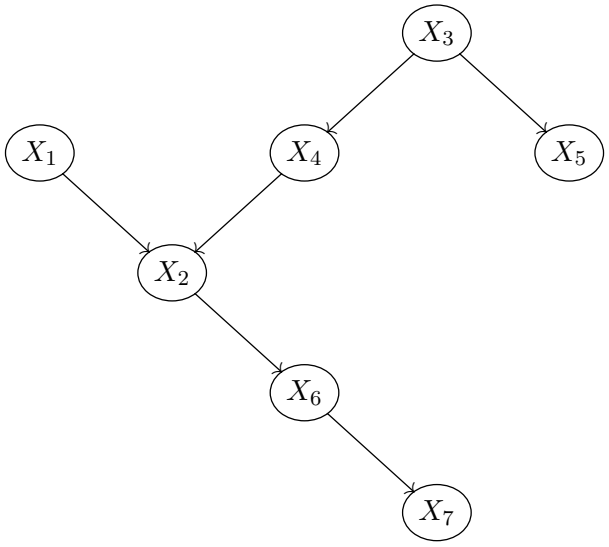

Figure 16.3: Example to illustrate RAI algorithm

Suppose also a 'perfect oracle'; independence tests give the correct results. After the first round, $X_1 \not\perp X_2$, $X_1 \not\perp X_6$, $X_1 \not\perp X_7$, but $X_1 \perp \{X_3, X_4, X_5\}$. $X_2 \not\perp X_j$ for any j, $X_3 \not\perp X_j$ for $j = 4, 5, 6, 7$, $X_4 \not\perp X_j$ for $j = 5, 6, 7$, $X_5 \not\perp X_j$ for $j = 6, 7$ and $X_6 \not\perp X_7$.

After the CI tests with conditioning sets size 0 have been carried out, the immoralities are determined;

$$(X_1, X_2, X_3), (X_1, X_6, X_3), (X_1, X_7, X_3), (X_1, X_2, X_4), (X_1, X_6, X_4)$$

$$(X_1, X_7, X_4), (X_1, X_2, X_5), (X_1, X_6, X_5), (X_1, X_7, X_5).$$

The edges $X_3 - X_4$, $X_3 - X_5$ and $X_4 - X_5$, $X_2 - X_6$, $X_2 - X_7$ $X_6 - X_7$ remain undirected.

Removing the undirected edges, the chain components are $A_1 = \{X_1\}$, $D = \{X_2, X_6, X_7\}$ and $A_2 = \{X_3, X_4, X_5\}$. D stands for *descendant*, A for *ancestor*.

Within D, $X_2 \perp X_7 | X_6$ and this is the only CI statement with a conditioning set size 1. The edge $X_2 - X_7$ is therefore removed and $X_2 - X_6 - X_7$ is not an immorality, since $X_6 \in S_{2,7}$ (the sep set).

Within A_2, $X_4 \perp X_5 | X_3$, hence $X_4 - X_5$ is removed and $X_4 - X_3 - X_5$ is not an immorality since $X_3 \in S_{4,5}$.

Now consider the directed edges from A_1 and A_2 to D. $\{X_3, X_4, X_5\} \perp \{X_6, X_7\} | X_2$, leading to removal of the 6 corresponding directed edges. $\{X_3, X_5\} \perp \{X_2\} | \{X_4\}$. Finally, Meek's rules may be used to direct $X_2 \to X_6$ and $X_6 \to X_7$ giving the essential graph in Figure 16.4.

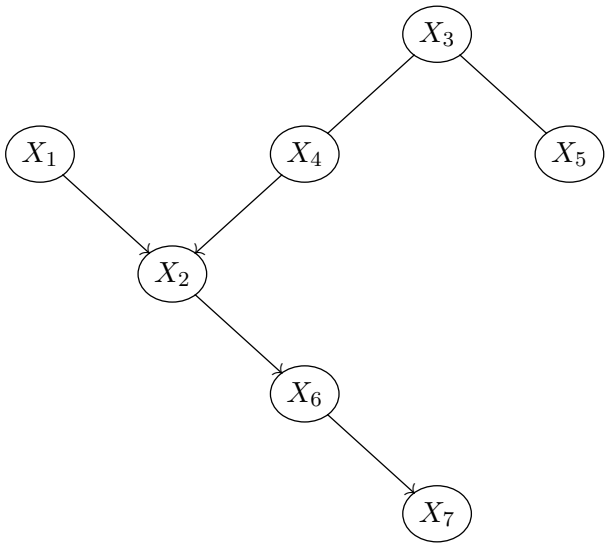

Figure 16.4: Example for RAI algorithm: essential graph

At this point, D is now removed. Since A_1 and A_2 have no ancestors, they are considered separately and the algorithm is finished. If A_1 and A_2 were descendants of other chain components, a chain component with no descendants would be chosen and the algorithm continues until all chain components have been considered.

The algorithm is then repeated with conditioning sets of size 2, and so on, until the termination condition is satisfied.

16.8 Incompatible Immoralities: EDGE-OPT Algorithm

It is possible that in the constraint based structure learning algorithms, immoralities that are incompatible with each other may emerge. This could, for example, be vee structures declared not to be immoralities, which lead to a cycle of length ≥ 4 without a chord, or two immoralities that give oppositie orientations for an edge.

There are two possible reasons for incompatibilities; either the independence tests give inaccurate results, or else there does not exist a faithful DAG.

A. Fast in [41] uses a constraint based method for dealing with this. EDGE-OPT ALGORITHM starts with the edges of the skeleton produced by a constraint based algorithm, either FAS, MMPC or RAI, chooses an orientation of the edges at random to produce a DAG and then considers each vee-structure in turn, deciding locally whether or not it should be a collider.

The list of constraints (that is, for each X, Y with no edge between them, the statement $X \perp Y | S_{XY}$ and $X \sim Y$ if there is no sepset) is established before EDGE-OPT start. At each stage, it chooses a vee-structure, examines the possibilities for orientation of the edges in the vee structure, and chooses the orientation that satisfies the largest number of constraints.

16.9 Hybrid Algorithms

We now consider various *hybrid* algorithms, which start by constraining the space and then using search and score techniques within the constrained space.

16.9.1 The Maximum Minimum Hill Climbing Algorithm

The MMHC algorithm by Tsamardinos, Brown and Aliferis (2006) [137] is a *hybrid* algorithm. Firstly, the set of edges of which the skeleton is a subset, is obtained using the constraint-based Maximum Minimum Parents Children algorithm. The sep sets, though, are not recorded, since they are unnecessary. Having obtained the skeleton, the orientation of the edges is obtained via a search-and-score procedure, known as the MMHC algorithm. It works as follows: let $V = \{X_1, \ldots X_d\}$ and denote the current graph by $\mathcal{G} = (V, D)$.

- Start with the empty graph $\mathcal{G} = (V, D)$ where $D = \phi$.
- At each stage, either *add* a directed edge to D, choosing an edge in $\mathcal{E}$ and directing it; any direction that does not produce a cycle is admissible, or *delete* an edge from $\mathcal{G}$, or *reverse* an edge in $\mathcal{G}$, or leave the graph unaltered. From all the possibilities of 'add an edge', 'delete an edge', 'reverse an edge', 'leave the graph unaltered' choose the one that gives the greatest score; that is, the operation that produces the greatest reduction in the Kullback Leibler divergence between the probability modelled along the graph and the empirical probability.
- Repeat until the score is not changed.

The algorithm may be modified as follows: instead of the best change, make the best change that results on a graph that has not already appeared. When 15 changes occur without an increase in the best score ever encountered during the search, the algorithm terminates. The DAG that produced the best score is then returned. which starts with the constraint based MMPC stage to locate the skeleton and then carries out a search and score based MMHC stage, using the skeleton obtained from MMPC as the candidate edge set. Two other hybrid methods are described below.

16.9.2 L1-Regularisation

One method, introduced by Schmidt, Niculescu-Mizil and Murphy (2007) [123], places constraints on the model and then uses an L^1 score function, described below, as the basis of a search and score within the constrained space.

The method can be employed with Gaussian or binary variables. The binary case is outlined here.

In this algorithm, there is no restriction on the number of parents that a variable may have, but there is a constraint on the way in which the parents influence the variable. The state space of variable X_j is $\{-1, 1\}$ for each j and the conditional probabilities are modelled so that the *logit* function is linear:

$$\ln\left(\frac{p_{X_j|\mathrm{Pa}_j}(1|\underline{\pi}_j)}{1-p_{X_j|\mathrm{Pa}_j}(1|\underline{\pi}_j)}\right) = \left(\theta_{j,0} + \sum_{k=1}^{p_j}\theta_{j,k}\pi_{j,k}\right) \tag{16.2}$$

where $\underline{\pi}_j = (\pi_{j1}, \ldots, \pi_{jp_j})$, the configuration of Pa_j, is a sequence of ± 1 corresponding to the states of the parent variables. The parent variables are only permitted to influence $\ln\frac{p}{1-p}$ linearly; no interactions are permitted. This permits a large number of parents, since the number of parameters is linear, rather than exponential, in the number of parents.

The algorithm works in two stages: like the MMPC, it first produces candidate parent children sets for each variable. Having constrained the search space, it then uses a search and score algorithm to determine the candidate parent / children sets. Having determined the parent / children sets, it runs the hill climbing part of the MMHC algorithm of Tsamardinos, Brown and Aliferis to obtain the structure, keeping the conditional probabilities of the form in Equation (16.2). For a vector $\underline{x} = (x_1, \ldots, x_d)$ of 1's and -1's, let Pa_j denote all the variables without j. That is, all variables permitted as possible parents for j at this stage. Let $\underline{\tilde{x}}^{(j)}$ denote the vector $\underline{x}$ without x_j. Let

$$LL(j, \underline{\theta}_j, \underline{x}) = \log p_{X_j|\mathrm{Pa}_j}(x_j|\underline{\tilde{x}}^{(j)})$$

denote the log likelihood function and, for $\mathbf{x}$ the data matrix with rows $\underline{x}_{(1)}, \ldots, \underline{x}_{(n)}$, let

$$LL(j, \underline{\theta}_j, \mathbf{x}) = \sum_{k=1}^{n} LL(j, \underline{\theta}_j, \underline{x}_{(k)}).$$

The parameters $\underline{\theta}_j$ are chosen to maximise the $L1$ regularisation score function,

$$L_1R(\underline{\theta}_j, \mathbf{x}) = LL(j, \underline{\theta}_j, \mathbf{x}) - \lambda\|\underline{\theta}_j\|_1,$$

where $\|\underline{\theta}_j\|_1 = \sum_{k=1}^{d-1} |\theta_{jk}|$ and λ is chosen appropriately. The sum is over the parameters corresponding to dependence on parent variables; the parameter $\theta_{j,0}$ is not included. The article [123] has some discussion about the appropriate choice of λ.

The L^1 regularisation, if λ is appropriately chosen, has the effect of choosing vectors $\underline{\theta}_j$ with a substantial number of zero components. Because of this property, it tends to favours a lower number of parameters in the model. For this reason, L^1 regularisation is a technique that is developing increasing importance.

16.9.3 Gibbs sampling

A related approach to the problem of structure learning is found in Bulashevska and Eils (2005) [11]. The structure learning algorithm is intended for analysis of gene expression data, to locate gene regulatory interactions. As with Schmidt, Niculescu-Mizil and Murphy (2007) [123], the parents influence the offspring independently of each other and the algorithm forms 'noisy OR' and 'noisy AND' gates. The parent sets are chosen using Gibbs sampling. The generic techniques of Gibbs sampling are found in Gamerman and Lopes (2006) [49].

16.10 A Junction Tree Framework for Undirected Graphical Model Selection

Let $X = (X_1, \ldots, X_d)$ be a random vector, with indexing set $V = \{1, \ldots, d\}$. An *undirected graphical model* is simply an undirected graph $\mathcal{G} = (V, U)$ where U is a set of undirected edges, such that $\mathcal{G}$ is the independence graph of X.

The edge set U contains an edge $\langle \alpha, \beta \rangle$ if and only if $X_\alpha \not\perp X_\beta | X_{-(\alpha,\beta)}$ (i.e. X_α not independent of X_β conditioned on all the other components of the random vector X).

Learning the independence graph may be problematic when d is large, since the conditional independence tests available have lower power when the number of states of the conditioning set is large.

The process may be facilitated if additional a-priori information is available, of the form: $U \subseteq \widetilde{U}$, where $\widetilde{U}$ is an undirected edge set with node set V. Let $\mathcal{H} = (V, \widetilde{U})$.

If $\mathcal{H}$ is triangulated, a junction tree may be constructed from the cliques. It is clear that $\langle \alpha, \beta \rangle \notin U$ if none of the cliques contain both α and β, since this implies that $\langle \alpha, \beta \rangle \notin \widetilde{U}$.

If there is a clique C such that $\alpha, \beta \in C$, then it is not necessary to consider $X_{-(\alpha,\beta)}$; let $\mathcal{C}$ denote the clique-set, and let C denote a generic element of $\mathcal{C}$. Let $C_\alpha = \{C \in \mathcal{C} : \alpha \in C\}$ and let $W_{\alpha,\beta} = (\cup_{C \in C_\alpha} C) \cup \left(\cup_{C \in C_\beta} C\right)$. In other words, $W_{\alpha,\beta}$ is the collection of nodes contained in cliques which contain either α or β (or both). Then, by obvious properties of the independence graph, $\langle \alpha, \beta \rangle \in U$ if and only if

$$X_\alpha \not\perp X_\beta | X_{W_{\alpha,\beta} \setminus \{\alpha,\beta\}}.$$

In other words, only those cliques containing either α or β (or both) need to be considered, leading to a reduction in the size of the conditioning sets and hence to more accurate conditional independence tests.

There may be some additional gain, in terms of reducing the size of the conditioning sets, if the junction tree can be successively updated, by removing from $\widetilde{U}$ edges that have been considered, for which it has been established that they are not in U.

Vats and Nowak [139](2014) provide a framework for this, by considering the so-called *region graph*. A *region graph* is simply a directed acyclic graph, where a node the region graph (which we call a region-node) is a subset of V, the node set. The region graph of interest is constructed as follows: the first generation of regions, $\mathcal{R}^1$ is the collection $\mathcal{C}$ of cliques of a junction tree. These are the *ancestor* nodes of the region graph. Generation $\mathcal{R}^{i+1}$ is the set of all pairwise intersections of sets in $\mathcal{R}^i$ with cardinality greater than or equal to 2, for $i = 1, \ldots, L-1$, where L is the maximum value of i for which $\mathcal{R}^i$ constructed in this way is non-empty.

The edge set of a region graph contains an edge $R \to S$ if and only if $R \in \mathcal{R}^i$ and $S \in \mathcal{R}^{i+1}$ for some $i \in \{1, \ldots, L-1\}$ and there is a set $T \in \mathcal{R}^i$ such that $R \cap T = S$.

Vats and Nowak propose an algorithm for locating the independence graph $\mathcal{G} = (V, U)$, given a decomposable graph $\mathcal{H} = (V, \widetilde{U})$ where $U \subseteq \widetilde{U}$. The algorithm is given as Algorithm 10; some further notation is needed before introducing it.

For a region R of a region graph, let

$$\overline{R} = \cup_{S \in \{\mathrm{an}(R), R\}} S \tag{16.3}$$

In other words, $\overline{R}$ is the union of region R and all its ancestors. In terms of the junction tree for $\mathcal{H}$, this is the union of all cliques which have R as a subset.

For a node set S, let $K(S)$ denote the complete undirected graph with node set S. For a set of nodes R, let denote the edge set W restricted to R and let

$$W'_R = W_R \backslash \{\cup_{S \in \mathrm{ch}(R)} K(S)\}. \tag{16.4}$$

Algorithm 10 returns the independence graph $\mathcal{G} = (V, U)$.

Correctness of Algorithm 10 It remains to show that, assuming a perfect oracle, Algorithm 10 returns the independence graph $\mathcal{G} = (V, U)$. Firstly, it is clear that the algorithm thus constructed considers all the edges of $\widetilde{U}$. Secondly, it is straightforward (and left as an exercise) to show that, if the distribution factorises along the junction tree, then $X_\alpha \not\perp X_\beta | X_{\overline{R} \backslash \{\alpha, \beta\}} \Leftrightarrow X_\alpha \not\perp X_\beta | X_{V \backslash \{\alpha, \beta\}}$. From this it follows that, assuming a perfect oracle, the algorithm returns the independence graph.

Example 16.4 (Region Graph)**.**

Suppose the independence graph $\mathcal{G} = (V, U)$ is given on the left of Figure 16.5 and it is known that $U \subseteq \widetilde{U}$, where the graph $\mathcal{H} = (V, \widetilde{U})$ on the right. Here $V = \{1, 2, 3, 4, 5, 6, 7\}$.
The algorithm proceeds as follows:

Algorithm 10 Finding Independence Graph Given Decomposable Graph as Wrapper

Input: A graph $\mathcal{H} = (V, \widetilde{U})$ such that $U \subseteq \widetilde{U}$

Output: The independence graph $\mathcal{G} = (V, U)$

Step 1: Initialise: $\widehat{U}$ as $\widehat{U} = \phi$ (empty set) and find the region graph of $\mathcal{H}$.

Step 2: Suppose the regions are $\mathcal{R}^1, \ldots, \mathcal{R}^L$
and j is the smallest value such that there exists a region $R \in \mathcal{R}^j$ such that $\widetilde{U}'_R \neq \phi$
where $\widetilde{U}'_R$ is defined by (16.4)

for each $R \in \mathcal{R}^j$ **do**

Compute $\overline{R}$ defined by (16.3) and $\widetilde{U}'_R$ defined by (16.4). For each edge $\langle \alpha, \beta \rangle \in \widetilde{U}'_R$, remove the edge from $\widetilde{U}$. Add the edge $\langle \alpha, \beta \rangle$ to $\widehat{U}$ if and only if $X_\alpha \not\perp X_\beta | X_{\overline{R} \setminus \{\alpha, \beta\}}$.

end for

(Note: at this stage, $\widetilde{U} \cup \widehat{U} \supseteq U$).

Step 3: Compute a new junction tree and region graph using an efficient triangulation of edge set $\widetilde{U} \cup \widehat{U}$

Step 4: If $\widetilde{U} = \phi$ then terminate, otherwise go to **Step 2**.

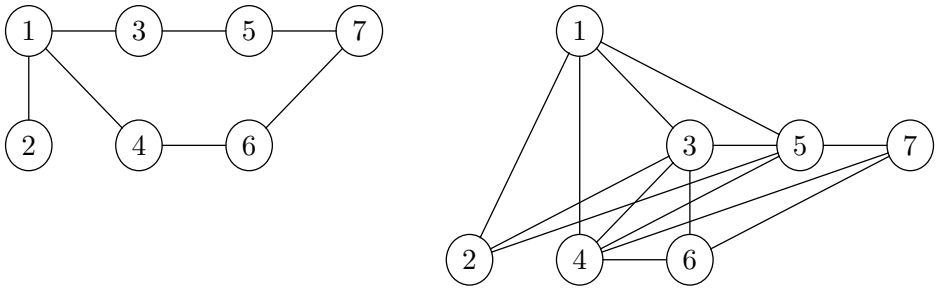

Figure 16.5: Graph $\mathcal{G} = (V, U)$ and $\mathcal{H} = (V, \widetilde{U})$; $\widetilde{U} \supseteq U$

- Clique $1, 2, 3, 5$ has child $1, 3, 5$. Remove edges from the child gives edges $\langle 1, 2 \rangle$, $\langle 2, 3 \rangle$, $\langle 2, 5 \rangle$ from $\widetilde{U}$ to be estimated. Therefore, look at $\mathcal{R}^1$ (the cliques of the junction tree with the complete graphs of the separators removed.

  Clique $1, 2, 3, 5$ edges $\langle 1, 2 \rangle$, $\langle 2, 3 \rangle$, $\langle 2, 5 \rangle$ considered; $\langle 2, 3 \rangle$ and $\langle 2, 5 \rangle$ removed. Edge $\langle 1, 2 \rangle$ added to $\widehat{U}$

  Clique $1, 3, 4, 5$ Children are $1, 3, 5$ and $3, 4, 5$. Therefore, only edge $\langle 1, 4 \rangle$ is considered. This is retained. It is therefore removed from $\widetilde{U}$ and added to $\widehat{U}$.

  Clique $3, 4, 5, 6$ Children are $3, 4, 5$ and $4, 5, 6$. Only edge $\langle 3, 6 \rangle$ is considered. It is removed from $\widetilde{U}$.

  Clique $4, 5, 6, 7$ Child is $4, 5, 6$. Edges considered are: $\langle 4, 7 \rangle$, $\langle 5, 7 \rangle$ and $\langle 6, 7 \rangle$. They are removed from $\widetilde{U}$. Edges $\langle 5, 7 \rangle$ and $\langle 6, 7 \rangle$ are added to $\widehat{U}$.

- At this stage, a new junction tree may be computed, using the edges from $\widetilde{U} \cup \widehat{U}$. This may be

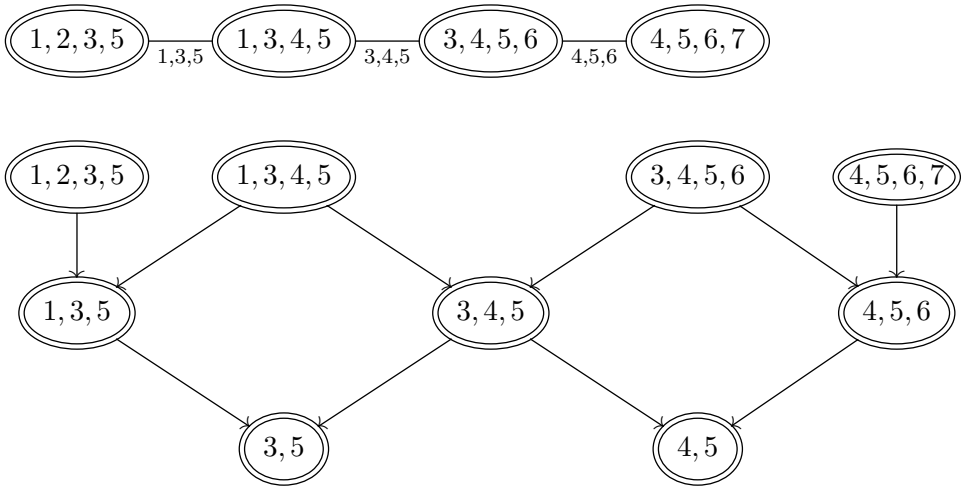

Figure 16.6: Junction Tree (above) and Region Graph (below) for Figure 16.5

more efficient. Alternatively, we may continue with the same junction tree.

After deleting these edges from $\widetilde{U}$, generation $\mathcal{R}^2$ is the first generation that satisfies the property.
Region $1,3,5$: This has one child, which is $3,5$. The edges under consideration are therefore $\langle 1,3\rangle$ and $\langle 1,5\rangle$. These have not been considered before. They are removed from $\widetilde{U}$ and edge $\langle 1,3\rangle$ is added to $\widehat{U}$.
Region $3,4,5$: This has two children, $3,5$ and $4,5$. Only one edge is considered; $\langle 3,4\rangle$. This is removed from $\widetilde{U}$. It is not present in U and therefore (assuming a perfect oracle) is not added to $\widehat{U}$.
Region $4,5,6$: This has one child, $4,5$. The edges under consideration are therefore $\langle 4,6\rangle$ and $\langle 5,6\rangle$. They are removed from $\widetilde{U}$. The edge $\langle 4,6\rangle$ is added to $\widehat{U}$.

- At this stage, a new junction tree may be computed. If we proceed with the current junction tree, we look at $\mathcal{R}^3$. This contains regions $3,5$ and $4,5$. Each region consists of two nodes; for each region, there is one edge under consideration. The edge $\langle 3,5\rangle$ is removed from $\widetilde{U}$ and added to $\widehat{U}$; similarly with $\langle 4,5\rangle$.

At this stage, $\widetilde{U} = \phi$ and $\widehat{U} = U$. □

16.11 The Xie-Geng Algorithm for Learning a DAG

The algorithm of Xie-Geng [148] (2008) provides a framework for learning a DAG which is essentially different from FAS, PC/MMPC, RAI. This algorithm again assumes that there exists a faithful DAG and, at the final stage of edge removal, removes edges according to the principle of Theorem 2.3, although this last stage may be omitted if one does not have a priori information that there exists a faithful DAG. The algorithm starts by finding the *independence graph* (Definition 5.6), which is useful

by Theorem 5.7. Recall the definition of *weak decomposition* (Definition 7.17). The independence graph is subsequently decomposed, the sep-sets recorded at each stage. With the reconstruction, an edge $\langle \alpha, \beta \rangle$ appears in the final graph if and only if it appears in all parts of the decomposition that contain both nodes α and β; otherwise a suitable immorality is added, dictated by the sep-sets in the usual manner. The compelled edges are then added and the essential graph is returned.

The algorithm assumes that independence statements $X_\alpha \perp X_\beta | X_{-\{\alpha,\beta\}}$ can be verified. This may be a weak point for large numbers of random variables, since the power of conditional independence tests decays proportionally to the number of variables in the conditioning set. The algorithm may be combined with the algorithm of Section 16.10 from Vats and Nowak [139] to reduce the size of the conditioning sets if there is additional a-priori information that the independence graph is contained within a decomposable graph $\mathcal{H} = (V, \widetilde{U})$.

Theorem 2.3 is essential for proving that the algorithm returns a faithful DAG when it exists. From Definition 5.6 (the definition of the independence graph) together with Theorem 5.7, it follows that graphical separation statements in the independence graph are equivalent to the corresponding conditional independence statements for the probability distribution. By Theorem 5.5 together with the definition of the independence graph (Definition 5.6), the independence graph is equivalent to the moral graph of a faithful DAG, when a faithful DAG exists.

The following two theorems are used crucially in proving that the graph returned by the algorithm is the skeleton of a DAG along which the distribution may be factorised.

Theorem 16.5. *Let $\mathcal{G} = (V, D)$ be a DAG. Suppose that $A \perp\!\!\!\perp B \| _{\mathcal{G}} S$ for three subsets $A, B, S \subset V$. Let $\alpha \in A$ and $\beta \in A \cup S$. Then $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} R$ for some $R \subset A \cup B \cup S$ if and only if $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} R'$ for a subset $R' \subset A \cup S$.*

Theorem 16.6. *Let $\mathcal{G} = (V, D)$ be a DAG and suppose that $A, B, S \subset V$ such that $A \perp\!\!\!\perp B \| _{\mathcal{G}} S$. Let $\alpha, \beta \in S$. Then there is a subset $R \subseteq A \cup B \cup S$ such that $\alpha \perp \beta | R$ if and only if either there is a subset $R' \subset A \cup S$ or there is a subset $R' \subset B \cup S$ such that $\alpha \perp\!\!\!\perp \beta \| R'$.*

These statements appear, at face value, precisely what one would expect. Their proofs, though, are somewhat involved and non-trivial. The Xie-Geng algorithm is based on these statements, which enable the edge set for the whole graph to be concluded from examining subsets of the variables. The proofs of these theorems are given later, after the description of the algorithm.

16.11.1 Description of the Xie-Geng Algorithm

The algorithm proceeds as follows:

- An undirected graph is constructed. This is the graph $\mathcal{G} = (V, U)$ where $\langle \alpha, \beta \rangle \in U$ if and only if $X_\alpha \not\perp X_\beta | X_{-(\alpha,\beta)}$. This is the independence graph (Definition 5.6). It is therefore equivalent to the moral graph of a faithful DAG if a faithful DAG exists (Exercise 6 page 352).
- A weak decomposition (A, B, S) (Definition 7.17) of the moral graph is found, if such a decomposition exists.

- For each $\alpha \in A \backslash S$ and $\beta \in B \backslash S$, set $S_{\alpha,\beta} = S$, the separator of α, β.
- Construct $\mathcal{G}^i_{A \cup S}$ and $\mathcal{G}^i_{B \cup S}$, where for each $\gamma, \delta \in A \cup S$, $\langle \gamma, \delta \rangle \in U^i_{A \cup S}$ if and only if $X_\gamma \not\perp X_\delta | X_{(A \cup S) \backslash \{\gamma, \delta\})}$, similarly for $\mathcal{G}^i_{B \cup S}$. These are the *independence graphs* for $A \cup S$ and $B \cup S$ respectively.
- Find weak decompositions of $\mathcal{G}^i_{A \cup S}$ and $\mathcal{G}^i_{B \cup S}$, the independence graphs associated with these weak decompositions and continue recursively until it is not possible to decompose any of these pieces further. At each stage, if $W \subset V$ is decomposed into A', B', S', set $S_{\alpha,\beta} = S'$ for each $\alpha \in A', \beta \in B'$.

Before the assembly stage, the following additional stage is carried out on the cliques which are obtained from the recursive decomposition:

- For each clique A in the decomposition and each pair $\{\alpha, \beta\} \subseteq A$, check whether there is a subset $S \subset A \backslash \{\alpha, \beta\}$ such that $X_\alpha \perp X_\beta | X_S$. If there is, then remove the edge $\langle \alpha, \beta \rangle$ and let $S_{\alpha,\beta} = S$, the sep-set of $\{\alpha, \beta\}$.

From these pieces, the DAG is constructed as follows:

- Two sub-skeletons $L_{A \cup S} = (A \cup S, U_{A \cup S})$ and $L_{B \cup S} = (B \cup S, U_{B \cup S})$ are combined to form

$$L_{A \cup B \cup S} = (A \cup B \cup S, U_{A \cup B \cup S})$$

  where

$$U_{A \cup B \cup S} = U_{A \cup S} \cup U_{B \cup S} \backslash \{\langle \alpha, \beta \rangle | \alpha, \beta \in S, \quad \langle \alpha, \beta \rangle \notin U_{A \cup S} \cap U_{B \cap S}\}.$$

- This is done recursively until all the pieces have been added.
- For each separator $S_{\alpha,\beta}$, orient a vee-structure (α, γ, β) as an immorality $\alpha \to \gamma \leftarrow \beta$ if $\gamma \notin S_{\alpha,\beta}$.
- Orient the compelled edges.

Establishing Correctness If there is a faithful DAG for the distribution and a perfect oracle, then the algorithm returns the essential graph of the faithful DAG. This is established as follows:

Suppose that $\mathcal{G}_{A \cup C}$ and $\mathcal{G}_{B \cup C}$ are faithful for the distributions over $A \cup C$ and $B \cup C$ respectively and are combined according to the rules given to give $\mathcal{G}_{A \cup B \cup C}$. The results of Theorems 16.5 and 16.6 may be used to establish the D-separation properties:

For any $\alpha \in A$ and $\beta \in B$, $S_{\alpha,\beta} = C$ and therefore there is no edge $\alpha \sim \beta$ in a faithful DAG for the distribution over $A \cup B \cup C$. Following the reconstruction, there is no edge in $\mathcal{G}_{A \cup B \cup C}$.

For $\alpha, \beta \in C$, the reconstruction has an edge $\alpha \sim \beta$ in $\mathcal{G}_{A \cup B \cup C}$ if and only if there are edges in both $\mathcal{G}_{A \cup C}$ and $\mathcal{G}_{B \cup C}$. Theorem 16.6 states that if $\mathcal{G}(A \cup B \cup C)$ is a DAG over the variables $A \cup B \cup C$ and there is a set $R \subseteq A \cup B \cup C$ such that $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}(A \cup B \cup C)} R$ if and only if either there is a set $R' \subset A \cup C$ such that $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}(A \cup C)} R'$ or there is a set $R' \subset B \cup C$ such that $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}(B \cup C)} R'$. Therefore, $\mathcal{G}(A \cup B \cup C)$ is a faithful graph for $p_{A \cup B \cup C}$ then its skeleton contains an edge $\alpha \sim \beta$ between two variables in C if and only if both $\mathcal{G}_{A \cup C}$ and $\mathcal{G}_{B \cup C}$ contain the edge.

Example 16.7 (Example for the Xie-Geng Algorithm)**.**

Suppose that the probability distribution $\mathbb{P}_{A,B,C,D,E,F,G,H}$ and the DAG in Figure 16.7 are faithful.

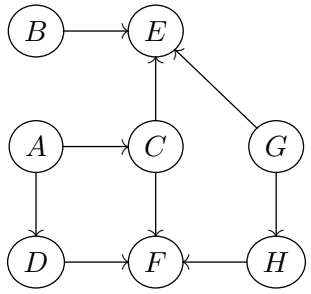

Figure 16.7: A faithful DAG to illustrate the Xie-Geng algorithm

Suppose that we also have a perfect oracle (each conditional independence test gives the correct result). The first step of the algorithm is to construct the independence graph, given in Figure 16.8. This is constructed by starting with the empty graph and adding an undirected edge $\langle \alpha, \beta \rangle$ if and only if $X_\alpha \not\perp X_\beta | X_{-(\alpha,\beta)}$. If the DAG in Figure 16.7 is *faithful*, the independence graph is the moral graph.

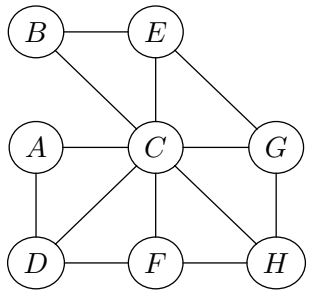

Figure 16.8: The Moral / Independence Graph for the DAG of Figure 16.7

The graph is now decomposed recursively; for example, take $\{C, G\}$, then this decomposes the graph into $\{A, C, D, F\}$ and $\{B, E, C, F, G, H\}$. The independence graphs for these two sets of variables are illustrated in Figure 16.9.

Now consider the piece on the left hand side of Figure 16.9. The set $\{C, D\}$ may be used as the separation set, and the independence graphs of the two pieces $\{A, C, D\}$ and $\{C, D, F\}$ are shown in Figure 16.10.

The edge $C - D$ does not appear in the first graph, since $C \perp D | A$. This is clear from the DAG in Figure 16.7, which is faithful to the distribution. It therefore follows that in the reconstruction stage, the edge $C - D$ will not be present and that $C - F - D$ will be an immorality.

More fully, the decomposition phase can proceed as follows:

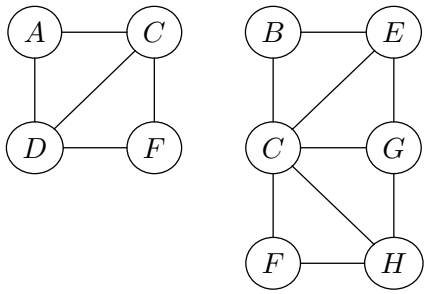

Figure 16.9: First stage of decomposition for the Xie-Geng algorithm

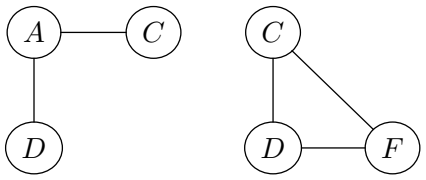

Figure 16.10: Further stage of decomposition for the Xie-Geng algorithm

- $\{A\}$ and $\{B, E, F, G, H\}$ are separated by $\{C, D\}$; $A \perp \{B, E, F, G, H\}|\{C, D\}$. The two pieces are: $\{A, C, D\}$ and $\{B, C, D, E, F, G, H\}$.
- Consider $\{A, C, D\}$. The graph is; $A - D - C$, since for variables $\{A, D, C\}$, $C \perp D|A$.
- This is decomposed further into $\{A, D\}$ and $\{A, C\}$; $C \perp D|A$. This decomposition is complete; the pieces are cliques and cannot be decomposed further.
- Consider $\{B, C, D, E, F, G, H\}$. Then $B \perp \{D, F, H, G\}|\{C, E\}$. The decomposition is into $\{B, C, E\}$ and $\{C, D, E, F, G, H\}$. The piece $\{B, C, E\}$ is a clique, since $B \not\perp C|E$.
- For $\{C, D, E, F, G, H\}$, $E \perp \{D, F, H\}|\{C, G\}$, so it is decomposed into $\{C, D, F, G, H\}$ and $\{C, E, G\}$. $\{C, E, G\}$ is a clique at this stage, since $C \not\perp G|E$.
- For $\{C, D, F, G, H\}$, $C \perp G|\{D, F, H\}$, so the graph of this piece does not contain the edge $C - G$.
- $G \perp \{C, F, D\}|H$, so decompose $\{C, D, F, G, H\}$ into $\{G, H\}$ and $\{C, D, F, H\}$.
- Now consider $\{C, D, F, H\}$ and decompose into $\{C, D, F\}$ and $\{H, D, F\}$; $C \perp H|\{D, F\}$. Since $C \not\perp D|F$, the piece $\{C, D, F\}$ is a clique. Since $D \not\perp H|F$, this is also a clique.

Now the cliques are considered and edges removed according to the principle of Theorem 2.3.

- For $\{C, D, F\}$ the edge $C - D$ is not removed; $C \not\perp D$ and $C \not\perp D|F$.

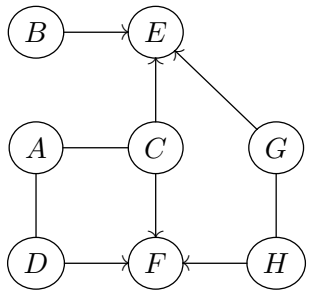

Figure 16.11: Structure learning example: essential graph

- For $\{D, F, H\}$, the edge $D - H$ is removed, with separation set (sep set) ϕ, since $D \perp H$ (with no instantiated notes, there is an open collider in each trail in the original DAG).
- For the final stage of the 'deconstruction' phase, edge $B - C$ is removed because $B \perp C$, with sepset $S_{BC} = \phi$.

Reconstruction For the reconstruction, these are put together, using the rule that $\mathcal{G}_{A \cup B \cup C}$ has an edge between two variables in C if and only if both $\mathcal{G}_{A \cup C}$ and $\mathcal{G}_{B \cup C}$. At this stage, the edge $C - D$ is removed from the final graph, since at the earlier stage $S_{CD} = \{A\}$. Similarly, $C - G$ is removed because $S_{CG} = \{D, F, H\}$. Vee structures (α, γ, β) are immoralities if and only if $\gamma \notin S_{\alpha,\beta}$. This gives the essential graph of Figure 16.11.

16.11.2 Proofs of Theorems 16.5 and 16.6

Finally, in the discussion of the Xie-Geng algorithm, we prove Theorems 16.5 and 16.6, thus establishing the correctness of the Xie-Geng algorithm under the assumptions that there exists a faithful graph and that there is a perfect oracle.

Theorem 16.5 requires some preparatory lemmas:

Lemma 16.8. *Let $\mathcal{G} = (V, D)$ be a directed acyclic graph. Let $\alpha, \beta \in V$. Let $\mathcal{F} = \mathcal{G}^m_{An(\{\alpha,\beta\} \cup S)}$ where $An(W)$ denotes the set W together with all nodes that are ancestor nodes in $\mathcal{G}$ for any node in W. First, the subgraph is taken, then it is moralised. Prove that S separates α and β in $\mathcal{F}$ if and only if $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} S$.*

Proof of Lemma 16.8 Assume that there is a path from α to β in $\mathcal{F}$ that has no nodes in S. Then a trail from α to β in $\mathcal{G}$ may be found by taking the directed edge in $\mathcal{G}$ if it corresponds to an edge in $\mathcal{F}$ or two edges to form a collider if there is no corresponding edge in $\mathcal{F}$; the two directed edges corresponding to the immorality that was removed when the graph was moralised.

If the collider node, or any of its descendants is in S, then the node is S-active. Assume that there is one collider γ that is not S-active. Then each parent node (they are both in $\mathcal{F}$) is either an ancestor

of α or an ancestor of β and hence the collider node is either an ancestor of α or an ancestor of β. It follows that there is a directed path from that node to α or β that does not pass through S. Assume that it is α and consider the trail between α and β with the part between α and γ replaced by this directed path from γ to α.

Proceeding inductively, a trail can be constructed such that the only colliders are S-active and there are no other nodes in S on the trail. It follows that $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} S$.

Now assume that all paths from α to β in $\mathcal{F}$ have at least one node in S. Consider any trail in $\mathcal{G}$ between α and β. The skeleton of any trail that has only fork or chain connections is in $\mathcal{F}$ and hence has a node in S. Consider any trail in $\mathcal{G}$ and consider the S-active collider connections. In $\mathcal{H}$, there is an undirected edge $\langle X, Y \rangle$ for any collider connection (X, Z, Y) such that Z is S-active. If the trail has nodes not in $\mathrm{An}(\{\alpha, \beta\} \cup S)$, then it clearly has a collider that is uninstantiated and has no descendants in S. If all the nodes of the trail are in $\mathrm{An}(\{\alpha, \beta\} \cup S)$, then since the undirected path in $\mathcal{H}$ formed by taking the directed edge $\langle X, Y \rangle$ instead of $\langle X, Z \rangle, \langle Z, Y \rangle$ has a node in S, it follows that the original trail has a fork or chain node in S and hence is blocked. The proof of Lemma 16.8 is complete. □

Lemma 16.9. *Let $\mathcal{G} = (V, D)$ and let $S \subset V$. Two nodes $\{\alpha, \beta\}$ are D-separated by S if and only if they are D-separated by $an(\{\alpha, \beta\}) \cap S$, where $an(W) = An(W) \backslash W$.*

Proof of Lemma 16.9 Set $S' = \mathrm{an}(\{\alpha, \beta\}) \cap S$. Since $S \supseteq S'$, it follows that if $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} S'$ then (trivially) there is a subset $R \subset S$ such that $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} R$.

Now suppose that $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} S'$. By Lemma 16.8, there is a path ρ connecting α and β in $\mathcal{G}^m_{\mathrm{An}(\{\alpha,\beta\})}$ that does not contain any vertex of S' and hence that ρ does not contain any vertex in $S \backslash \{\alpha, \beta\}$.

Suppose that α and β are D-separated by $S_0 \subseteq S$. Since $\mathrm{an}(\{\alpha, \beta\}) \cap S_0 \subseteq S'$, it follows that ρ does not contain any vertex in $\mathrm{an}(\{\alpha, \beta\}) \cap S_0$ and hence, by Lemma 16.8, $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} S_0$. It follows that if there is a subset $R \subseteq S$ such that $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} R$, then $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \mathrm{an}(\{\alpha, \beta\}) \cap S$. The proof of Lemma 16.9 is complete. □

Lemma 16.10. *Let $\mathcal{G} = (V, D)$ be a DAG and suppose that ρ is a trail between two non adjacent vertices α and β. If there are any nodes in ρ that are not in $An(\{\alpha, \beta\})$, then the trail ρ is blocked by any subset $S \subseteq an(\{\alpha, \beta\})$*

Proof It is clear that such a trail contains a collider connection, where the collider node is not in $\mathrm{An}(\{\alpha, \beta\})$ and hence the node does is not in $\mathrm{an}(\{\alpha, \beta\})$, nor does it have a descendant in this set. The proof of Lemma 16.10 is complete. □

We are now in a position to prove Theorem 16.5.

Proof of Theorem 16.5 Since $A \cup B \cup S \supseteq A \cup S$, it follows trivially that existence of a suitable subset of $A \cup S$ implies existence of a suitable subset of $A \cup B \cup S$.

To prove that existence of a subset in $A \cup B \cup S$ implies existence of a subset in $A \cup S$, assume that α and δ are two vertices in A and $A \cup S$ respectively, that are D-separated by a subset of $A \cup B \cup S$.

Let

$$S' = (\mathrm{an}(\{\alpha\}) \cup \mathrm{an}(\{\delta\}) \cap (A \cup S).$$

By Lemma 16.9, it is sufficient to show that S' blocks every trail ρ between α and δ. There are two cases:

- ρ not contained completely in $\mathrm{An}(\{\alpha, \delta\})$
- ρ contained completely in $\mathrm{An}(\{\alpha, \delta\})$.

By Lemma 16.10, in the first case, ρ is blocked by S' since $S' \subset \mathrm{an}(\{\alpha\}) \cup \mathrm{an}(\{\delta\})$.

For the second case, $A \perp\!\!\!\perp B \| _{\mathcal{G}} S$ implies that $\{\alpha\} \cup (S' \cap A) \perp \{\beta\} | S$ for each $\beta \in B$ and hence (using Exercise 2 page 22) that $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} (S' \cap A) \cup S$. Since $S' \subseteq A \cup S$, it follows that

$$\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} (S' \cup S).$$

Now suppose there is a trail ρ contained in $\mathrm{An}(\{\alpha, \delta\})$ between α and δ that is not blocked by S'. Let $W = S' \cup S$. Then W blocks ρ. There is therefore at least one node in ρ that is in $W \backslash S'$. Note that $W \subseteq B$. Let $\gamma \in W \backslash S'$ denote the first node on the trail ρ, starting from α, that is in $W \backslash S'$. Let ρ' denote the sub-trail of ρ between α and γ. Since ρ is not blocked by S', neither is ρ'. Since γ is the only node of ρ' that is in B, it follows that if ρ' is S' active, it is also W active and hence $\alpha \not\perp\!\!\!\perp \gamma \| _{\mathcal{G}} (S' \cup S)$, which is a contradiction.

If every sequence satisfies these properties, then clearly it satisfies these properties for every trail and hence, from the definition, $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} S$.

If $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} S$, then consider any such sequence of nodes. Take a subsequence by removing the loops so that any node appears at most once. This is a trail. Since D-separation holds, the trail has the property listed. The property therefore holds for the original sequence.

It is clear that if there is a set $R' \subset A \cup S$ or $R' \subset B \cup S$, then $R = R' \subset A \cup B \cup S$ satisfies the criterion.

Now suppose there is a set $\tilde{R} \subset A \cup B \cup S$ such that $\alpha \perp \beta \| _{\mathcal{G}} \tilde{R}$ and let $\gamma_1, \gamma_2 \in S$ such that $\gamma_1 \perp \gamma_2 \| _{\mathcal{G}} \tilde{R}$. By Lemma 16.9, $\gamma_1 \perp \gamma_2 \| _{\mathcal{G}} R$ where

$$R = (\mathrm{an}(\gamma_1) \cup \mathrm{an}(\gamma_2)) \cap (A \cup B \cup S).$$

Suppose that γ_2 is not an ancestor of γ_1. This can be done without loss of generality, by exchanging the roles of γ_1 and γ_2 if necessary.

Let

$$R_1 = (\mathrm{an}(\gamma_1) \cup \mathrm{an}(\gamma_2)) \cap (A \cup S).$$

$$R_2 = (\mathrm{an}(\gamma_1) \cup \mathrm{an}(\gamma_2)) \cap (B \cup S).$$

To prove that R_1 or R_2 D-separate γ_1 and γ_2, it is sufficient to show that for two trails ρ_1 in $A \cup S$ and ρ_2 in $B \cup S$ either ρ_1 is R_1 active, or ρ_2 is R_2 active, or both.

Consider the two cases separately:

- One of the trails ρ_j is not completely contained in $\mathrm{An}(\{\gamma_1, \gamma_2\})$

- both trails γ_1 and γ_2 are contained in $\text{An}(\{\gamma_1, \gamma_2\})$.

For the first case, since both R_1 and R_2 are subsets of $\text{an}(\gamma_1) \cup \text{an}(\gamma_2)$, it follows from Lemma 16.10 that ρ_j is blocked by both R_1 and R_2.

Now consider the second case. Suppose that ρ_1 is R_1 active and ρ_2 is R_2 active. Both ρ_1 and ρ_2 are blocked by $R = R_1 \cup R_2$. It follows that ρ_1 has a node in $R \backslash R_1$ and ρ_2 has a node in $R \backslash R_2$. Let δ_1 and δ_2 denote the nodes on ρ_1 and ρ_2 respectively that are closest to γ_1. As with the previous exercise, $\gamma_1 \in R \backslash R_1 \subseteq B$ and $\gamma_2 \in R \backslash R_2 \subseteq A$. Let ρ_1' and ρ_2' denote the subtrails of ρ_1 and ρ_2 respectively between $\gamma_1 \leftrightarrow \delta_1$, and $\gamma_1 \leftrightarrow \delta_2$ respectively. Note that ρ_1' is R_1 active, and ρ_2' is R_2 active. Connecting at γ_1 gives a sequence ρ' between δ_1 and δ_2 through γ_1. Note that ρ' may not be a trail, since there may be repeated nodes.

Any node that is not a collider node in ρ_1', since it is in $\text{an}(\gamma_1) \cup \text{an}(\gamma_2)$ and since neither ρ_1 nor ρ_1' are blocked by R_1, is not in $R_1 \cup S$. Similarly S does not contain any collider node on ρ_2'. Therefore, except perhaps for γ_1, ρ' does not have any collider connections where the collider node is in S.

Let ν_1 denote the neighbour of γ_1 on ρ_1'. Since $\nu_1 \in \text{an}(\gamma_1) \cup \text{an}(\gamma_2)$ and it is not γ_2, it is an ancestor of γ_1 or γ_2. If the orientation is $\gamma_1 \to \nu_1$, then γ_2 is an ancestor of γ_1, contradicting the assumption. Therefore the edge is oriented $\nu_1 \to \gamma_1$. Similarly, for ν_2 a neighbour of γ_1 on ρ_2'. It follows that (ν_1, γ_1, ν_2) is a collider on ρ'. Therefore S does not contain any nodes on ρ' that are not collider nodes on the trail.

Consider any collider node c in ρ_j' (that is, the centre of a collider connection in ρ_j'). It is either in R_j or else has a descendant in R_j. Since $c \in \text{an}(\gamma_1) \cup \text{an}(\gamma_2)$, it follows that $\gamma_1 \in S$ or $\gamma_2 \in S$ is a descendant of c. Since $\gamma_1 \in S$, it follows that each collider node in ρ' is either in S or has a descendant in S.

It follows that $\delta_1 \not\perp\!\!\!\perp \delta_2 \| _{\mathcal{G}} S$, contradicting $A \perp\!\!\!\perp B \| _{\mathcal{G}} S$. It follows that either $\gamma_1 \perp\!\!\!\perp \gamma_2 \| _{\mathcal{G}} R_1$ or $\gamma_1 \perp\!\!\!\perp \gamma_2 \| _{\mathcal{G}} R_2$. The proof of Theorem 16.5 is complete. □

Another preparatory lemma is needed, before proving Theorem 16.6.

Lemma 16.11. *Two non adjacent nodes α and β in a directed acyclic graph $\mathcal{G} = (V, D)$ are D-separated by a set $S \subset V$ if and only if for any sequence $\lambda = (\alpha, \lambda_1, \ldots, \lambda_{n-1}, \beta)$ (where the same node can appear more than once) with edges between each consecutive pair*

- *either λ contains a chain or a fork connection such that the chain node or fork node is in S or*
- *λ contains a collider connection such that the collider node is not in S and has no descendant in S.*

A sequence λ with edges between each consecutive pair that satisfies this property is said to be blocked by S.

Proof of Lemma 16.11 The result of Theorem 1.24 page 15, stating that a DAG $\mathcal{G} = (V, D)$ has an edge between α and β in D if and only if $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} S$ for any subset S, is used crucially here, together with the definition of 'faithful', that conditional independence statements and D-separation statements are equivalent.

For $\alpha \in A$ and $\beta \in C$, the graph $\mathcal{G}_{A \cup B \cup C}$ in the reconstruction has an edge $\alpha \sim \beta$ if and only if there is an edge $\alpha \sim \beta$ in the graph $\mathcal{G}_{A \cup C}$. Theorem 16.5 states that if $\mathcal{G}(A \cup B \cup C)$ is a DAG over the variables $A \cup B \cup C$ then there is a set $R \subset A \cup B \cup C$ such that $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}(A \cup B \cup C)} R$ if and only if there is a set $R' \subseteq A \cup C$ such that $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}(A \cup C)} R'$. It follows that if $\mathcal{G}(A \cup B \cup C)$ is faithful for $p_{A \cup B \cup C}$ then its skeleton contains an edge $\alpha \sim \beta$ between two variables $\alpha \in A$ and $\beta \in C$ if and only if $\mathcal{G}(A \cup C)$ contains an edge between α and β. The proof of Lemma 16.11 is complete. □

Theorem 16.6 now follows almost directly:

Proof of Theorem 16.6 If $\alpha, \beta \in V$, then any vee-structure $\alpha - \gamma - \beta$ such that $\gamma \notin S_{\alpha,\beta}$ is an immorality, hence the immoralities are correct. The proof of Theorem 16.6 is complete. □

16.12 The Ma-Xie-Geng Algorithm for Learning Chain Graphs

We now turn attention to learning chain graphs. As with the Xie-Geng algorithm for learning DAGs, the Ma-Xie-Geng algorithm for learning chain graphs starts by learning the independence graph. From the independence graph, a separation tree can be learned (Theorem 5.23) and, from the separation tree, edges may be deleted and edges oriented according to the principles described in 5.2.

The region graph of 16.10 can provide an effective alternative to the separation tree.

16.12.1 Skeleton Recovery with a Separation Tree

The *skeleton* of the chain graph may be recovered from the separation tree with the help of Theorem 5.25. The assumption is that there exists a chain graph which is *faithful* to the probability distribution. The recovery follows Algorithm 11. The algorithm consists of three main parts:

- Local skeletons are recovered for each individual tree-node of the separation tree. By Condition 1 of Theorem 5.25, edges deleted in any *local* skeleton are also absent in the global skeleton. This is the same principle used in the Xie-Geng algorithm for DAGs.

- All the information from local skeletons is combined to give a global undirected graph, which has all the edges of the skeleton, but may contain additional edges.

- Finally, the extra edges are eliminated.

Theorem 16.12. *Suppose there is a chain graph faithful to a probability distribution $\mathbb{P}$. Given a perfect oracle (i.e. each test for conditional independence gives the correct answer, rejecting CI when it is false and not rejecting when the CI statement is true), Algorithm 11 returns the skeleton of a faithful chain graph.*

Algorithm 11 Recovering the Skeleton of a Chain Graph

Input: A separation tree $\mathcal{T}$ of $\mathcal{G}$ and the set of independence statements of $\mathbb{P}$

Output: The skeleton of $\mathcal{G}$ and a set $\mathcal{S}$ of C-separators

Stage 1: recover local skeletons

Set $\mathcal{S} = \phi$

for each tree node C_h **do**

Start from a complete undirected graph $\mathcal{G}_h$ with vertex set C_h

for each pair of nodes $\{\alpha, \beta\} \subset C_h$ **do**

if $\exists S_{\alpha,\beta} \subset C_h$ such that $X_\alpha \perp X_\beta | X_{S_{\alpha,\beta}}$ **then**

Delete the edge $\langle \alpha, \beta \rangle$ in $\mathcal{G}_h$.

Add $S_{\alpha,\beta}$ to $\mathcal{S}$

end if

end for

end for

Stage 2: Combine Local Skeletons

Combine the graphs $\mathcal{G}_h = (C_h, E_h)$ into an undirected graph $\mathcal{G}' = (V, \cup_h E_h)$.

for each pair of nodes $\{\alpha, \beta\}$ contained in more than one tree-node and $\langle \alpha, \beta \rangle \in \mathcal{G}$ **do**

if $\exists C_h$ such that $\{\alpha, \beta\} \subset C_h$ and $\langle \alpha, \beta \rangle \notin E_h$ **then**

Delete the edge $\langle \alpha, \beta \rangle$ from $\mathcal{G}'$

end if

end for

Stage 3: Remove Extra Edges

for each pair of nodes $\{\alpha, \beta\}$ contained in more than one tree-node and $\langle \alpha, \beta \rangle \in \mathcal{G}'$ **do**

if $X_\alpha \perp X_\beta | X_{S_{\alpha,\beta}}$ for some $S_{\alpha,\beta} \subset N_{\mathcal{G}'}(\alpha)$ or $N_{\mathcal{G}'}(\beta)$ which is not a subset of any C_h with $\{\alpha, \beta\} \subset C_h$ **then**

Delete $\langle \alpha, \beta \rangle$ from $\mathcal{G}'$

Add S_{uv} to $\mathcal{S}$

end if

end for

Proof This uses Theorem 5.25. There is an edge between two nodes in a chain graph if and only if $\alpha \not\perp\!\!\!\perp \beta \| _{\mathcal{G}} S$ for any subset $S \subseteq V \backslash \{\alpha, \beta\}$. The three lines which delete edges therefore only delete edges which cannot appear in the skeleton. The output therefore returns a graph which contains all the edges of the skeleton.

At the same time, if $\alpha \not\sim \beta$, then one of the three conditions of Theorem 5.25 holds. For condition 1, there is no edge $\langle \alpha, \beta \rangle$ if α and β do not appear in the same tree node.

If condition 2 holds, then the edge $\langle \alpha, \beta \rangle$ is removed in Stage 1 or Stage 2.

If condition 3 holds, then either $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \mathrm{Pa}(\alpha)$ or $\alpha \perp\!\!\!\perp \beta \| _{\mathcal{G}} \mathrm{Pa}(\beta)$ (or both) where $\mathrm{Pa}(\gamma)$ denotes $\{\delta : (\delta, \gamma) \in D \quad \text{or} \quad \langle \delta, \gamma \rangle \in U\}$. The edge $\langle \alpha, \beta \rangle$ is therefore removed in Stage 3. □

16.12.2 Recovering the Complexes

Algorithm 12 locates and orients the complex arrows of $\mathcal{G}$, after the skeleton $\mathcal{G}'$ has been located.

Algorithm 12 Complex Recovery

Input: The conditional independence statements of $\mathbb{P}$, the skeleton $\mathcal{G}'$ of $\mathcal{G}$ and the set $\mathcal{S}$ of C-separators from Algorithm 11.
Output: The *pattern* $\mathcal{G}^*$ of $\mathcal{G}$
Initialise: $\mathcal{G}^* = \mathcal{G}'$
for each ordered pair (α, β) such that $S_{\alpha,\beta} \in \mathcal{S}$ **do**
  for each $\langle \alpha, \gamma \rangle$ in $\mathcal{G}^*$ **do**
    if $X_\alpha \perp X_\beta | X_{S_{\alpha,\beta} \cup \{\gamma\}}$ **then**
      Orient $\langle \alpha, \gamma \rangle$ as (α, γ) in $\mathcal{G}^*$
    end if
  end for
end for
The resulting graph $\mathcal{G}^*$ is the pattern of $\mathcal{G}$.

Before proving that Algorithm 12 orients the edges correctly, the following preparatory lemma is necessary.

Lemma 16.13. *Any arrow oriented by Algorithm 12 gives the same orientation as the arrow in* $\mathcal{G}$.

Proof The result is trivially clear and requires faithfulness; lack of C-separation implies that the corresponding conditional independence statement does not hold.

If all trails $\alpha \leftrightarrow \beta$ are blocked by $S_{\alpha,\beta}$, but opened by γ, then γ is either a node in the region of a complex on the trail between α and β or a descendant of such a node. Since γ is adjacent to α, it follows that $\mathcal{G}$ contains the arrow (α, β). □

Theorem 16.14. *If* $\mathcal{G}'$ *is the skeleton of a chain graph* $\mathcal{G}$ *which is faithful to the probability distribution* $\mathbb{P}$ *over* X*, then the output* $\mathcal{G}^*$ *of Algorithm 12 is the pattern of* $\mathcal{G}$.

Proof This follows from Theorem 16.12 and Proposition 5.26. Firstly, by Theorem 16.12 provides the correct skeleton. Clearly, if all the C-sep-sets were recorded, Algorithm 12 would consider all ordered pairs of nodes (α, β); for each γ such that $\langle \alpha, \gamma \in \mathcal{G}'$ (the skeleton) and determine whether or not it was a *complex arrow* (Definition **??**). The algorithm would then return the *pattern*; the graph where all the complex arrows are directed and the others are undirected.

The only remaining issue is whether or not the sep-sets provided by Algorithm 11 are sufficient.

By Proposition 5.26, for any complex $(\alpha, \rho_1, \ldots, \rho_n, \beta)$, there is a tree-node C which contains both parents α and β of the complex. Hence the set of C-sep-sets returned by Algorithm 11 is sufficient and hence Algorithm 11 returns the correct pattern. □

Example 16.15.

Suppose the chain graph in Figure 5.7 gives a faithful graphical representation of the conditional independence structure of a probability distribution $\mathbb{P}$. Suppose that we derive the separation tree of Figure 5.8. This separation tree is not optimal, in the sense that the tree-node $FGKH$ could be decomposed further into two tree-nodes FKG and GH separated by G. Given a perfect oracle, Algorithm 11 will return the correct skeleton and the set of C-sep-sets will be sufficient for Algorithm 12 will return the correct pattern.

The C-sep-sets found by Algorithm 1 are:

- **Stage 1** (each tree node): $S_{BC} = \{A\}$, $S_{CD} = \{B\}$ (we cannot separate $B - C$ at this stage, nor $D - E$), $S_{FG} = \{D\}$, $S_{FH} = \{G\}$, $S_{KH} = \{G\}$.
- **Stage 2** this simply looks at the edges removed from each tree-node. An edge between two nodes is present in the skeleton if and only if it is present between the two nodes for every tree-node.

After Stage 2, the only additional edge, still present in the graph which is not present in the skeleton, is $D - E$.

- **Stage 3** The edge $D - E$ is removed with sep set $S_{DE} = \{C, F\}$ using tree-nodes CDE and DEF.

The complex arrows are $D \to F$, $C \to E$, $F \to K$ and $G \to K$. Algorithm 12 detects these because $G \not\perp F | S_{GF} \cup K$, i.e. $G \not\perp F | \{D, K\}$ for the immorality $G \to K \leftarrow F$.

For the other complex, $D \not\perp C | S_{CD} \cup F$ and $D \not\perp C | S_{CD} \cup E$.

The pattern has thus been established. □

16.13 Structure Learning and Faithfulness: an Evaluation

16.13.1 Faithfulness and 'real world' data

The Recursive Autonomy Identification algorithm was analysed by B. Barros (2012) [4], applying it both to data simulated from test networks and to a financial data set. When applied to simulated data, simulated from the ALARM network, the algorithm performed very well; the performance was consistent with the results described by Yehezkel and Lerner [150]. For a data set generated by a

probability distribution for which there exists a faithful DAG, the results verified that the algorithm is efficient and produces a graph that corresponds well to the distribution that generated the data, with low computational overheads. The feature of the algorithm of making all required tests with smaller conditioning sets before moving on to larger increases accuracy over methods that do not do this. The additional use made of the structure, identifying the chain components of the essential graph at each stage, ensures that fewer statistical calls (references to the data set) are required.

Some features were noted in the performance of the algorithm. In earlier stages, some contradictory directions appeared. That is, pairs of immoralities $X \to Y \leftarrow Z$, $Y \to Z \leftarrow W$, in situations where the edge $Y \sim Z$ would be deleted in subsequent rounds of the algorithm following tests with larger conditioning sets. The direction chosen for the edge during that round was dictated by which immorality appeared first. If the test $X \perp Z | S_{X,Z}$, yielding a sep-set $S_{X,Z}$ was carried out first, then the edge would take the direction $Y \leftarrow Z$. After carrying out the CI tests and determining the directions, Meek's orientation rules were applied to determine the structures for the next round of the algorithm.

The algorithm worked very well; with 10000 observations, it produced a graph that had the correct skeleton and only 4 edges with incorrect orientation.

The test of performance of an algorithm is based on the ability of the algorithm to recover a probability distribution used to simulate data. There are several standard networks, including the ALARM network, that are used. Data is simulated from the network and the algorithm applied to the simulated data. Freedman and Humphreys (2000) p 33,34 [43] are somewhat scathing in their assessment of this procedure for verifying the utility of an algorithm, of using simulated data from a distribution known to have good properties. They write,

> The ALARM network is supposed to represent causal relations between variables relevant to hospital emergency rooms, and Spirtes Glymour Scheines (1993) [126] p 11 claim to have discovered almost all the adjacencies and edge directions 'from sample data'. However, these 'sample data' are simulated; the hospitals and patients exist only in the computer program. The assumptions made by SGS (1993) [126] are all satisfied by fiat, having been programmed into the computer: the question of whether they are satisfied in the real world is not addressed. After all, computer programs operate on numbers, not on blood pressures or pulmonary ventilation levels (two of the many evocative labels on nodes in the ALARM network).

Freedman and Humphreys continue by stating,

> These kinds of simulations tell us very little about the extent to which modelling assumptions hold true for substantive applications.

The constraint based algorithms all depend crucially on the modelling assumption that there is a DAG that is faithful to the set of conditional dependence / independence statements that can be established. We have already pinpointed two difficulties that can arise in the 'real world'; interaction effects without main effects and hidden common causes.

16.13.2 Interaction effects without main effects

Example 2.7 gives an example of a situation where these constraint based algorithms will miss key associations between the variables. Any situation where factors taken individually give no information, but where there are two-factor, or higher order factor interaction without main effects, will not be detected. If applied to genetic data, for example, the algorithm will not be able to detect situations where a single gene by itself has no apparent effect, but where the genome pathway may be opened by two genes acting together.

This situation will not lead to internal inconsistencies in the functioning of the algorithms; associations of this type will simply be missed and the output will be a DAG that does not show these associations, but it may not lead to reversed edges (situations where the algorithm has to choose between two contradictory directions for an edge).

16.13.3 Hidden variables

In a 'real world' situation, there may well be hidden variables which are not measured and the experimenter may be unaware of their existence. This can lead to reversed edges, as the following example illustrates. Suppose that X, Y, Z, W are variables that are recorded, while H is a hidden variable, a common cause of X and Y, whose presence is not suspected by the researcher. Suppose that the causal relations between H, X, Y, Z, W are given by Figure 16.12.

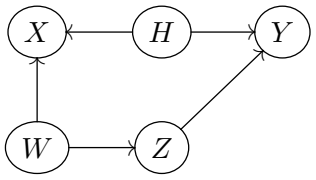

Figure 16.12: H is hidden and does not appear in the data matrix

If the RAI algorithm is applied to the variables X, Y, Z, W, whose associations are described by the d-connection statements of the DAG in Figure 16.12, then $X \perp Z|W$, giving $X \rightarrow Y \leftarrow Z$ and $Y \perp W|Z$, giving the immorality $Y \rightarrow X \leftarrow W$. Even if there is a perfect oracle (sufficient data to give correct results for each CI test so that the results are consistent with the probability distribution over (X, Y, Z, W)), the edge between X and Y is a *reversed edge*, $X \leftrightarrow Y$. This notation means that, from the CI tests, one test gives a direction $X \rightarrow Y$; the other gives a direction $X \leftarrow Y$ and the algorithm will choose the direction depending on the order in which the tests are carried out.

In the RAI algorithm, the direction that an edge takes in the output graph, under such circumstances is determined by the order of the variables; if the test results $X \perp Z|W$ appears first, the output graph will contain $X \rightarrow Y$ and thus the graph will contain the false d-separation statement $W \perp Y|\{X, Z\}$, while if the result $W \perp Y|Z$ appears first, the output graph will contain the edge $Y \rightarrow X$ and the false d-separation statement $X \perp Z|\{W, Y\}$. The two possibilities are given in Figure 16.13.

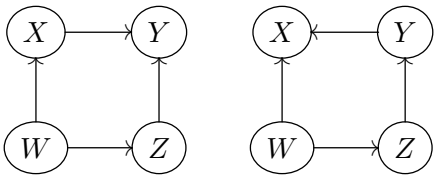

Figure 16.13: Possible outputs applying constraint based algorithm to variables (X, Y, Z, W) from Figure 16.12

16.13.4 The scope of structure learning

Algorithms can detect associations, at the level of 'descriptive statistics', without reference to the process that generates the data and the nature of randomness. At the level of descriptive statistics, the scope of constraint based algorithms is viewed along the following lines: from the $n \times d$ data matrix, an empirical distribution can be established (or, at least, if d is very large, empirical probability distributions of the marginalisation to subsets of the variables can be established). Any test result that produces $X \not\perp Y|S$ corresponds to a d-connection statement that is to be retained in the output graph; any test result where $X \not\perp Y|S$ is not rejected does not have to be retained in the output graph. The output graph attempts to have as few edges as possible, while retaining all the d-connection statements that were established through rejecting independence.

For large numbers of variables, there are clear difficulties that make serious inferential statistics impossible. The assumption is that the $n \times d$ data matrix represents n independent instantiations of a d-random vector X. This assumption, together with an assumption that n is sufficiently large for a central limit theorem effect to hold is required for the test statistics to be approximately χ^2. Even if the nominal significance level α chosen for rejecting a null hypothesis can be considered as a measure of a probability in any serious way the number of tests required is large that the overall significance level could be close to 1. In terms of descriptive statistics, the output graph can be informative, but it is difficult to reach inferential conclusions from the output of these algorithms.

16.13.5 Application of FAS and RAI to financial data

After testing the Fast and RAI algorithms on the training example of the ALARM network, where it performed well, the work of Barros [4] proceeded to run these algorithms on a financial data set, composed of the closing values of 18 stock market indices (Amsterdam stock index, Austrian traded index, Brussels stock index, etc ...) from 1st January 2005 to 1st January 2011, approximately 1000 instantiations of 18 variables.

The aim of the thesis was to detect *changes* in associations between the variables, to learn a structure, detect when the structure was no longer appropriate and update.

In the financial data set, the raw RAI algorithm gave no independence statements after the first round; for each pair of variables (X, Y), the result was 'reject independence'. Therefore, any pair of

variables should be d-connected in the output graph. Yet the output graph, following application of the raw RAI algorithm, gave pairs of d-separated variables, which indicates that conditional independence was falsely accepted due to weak tests.

In order to deal with the situation where 'accept independence' from tests with large conditioning sets contradicted d-connection statements with lower order conditioning sets, Barros adopted a more conservative approach than the argumentation of Bromberg and Margaritis [9] and modified the algorithm so that it did not accept an independence statement that resulted in a d-separation in the output graph contradicting a dependence statement that has already been established. This modification worked well.

The output still gave a large number of 'reversed edges'. While the ALARM network gave one or two, the financial data set gave approximately 28 reversed edges, indicating situations that appeared in the DAG in Figure 16.12, with possible output graphs corresponding to Figure 16.13.

The presence of a *substantial* number of 'common cause' hidden variables would explain this.

This was a randomly chosen 'real world' data set and probably not appropriate for an algorithm based on a 'faithfulness' assumption. The variables here do not satisfy one of the motivating features of the faithfulness assumption, that the variables stand in *causal* relation to each other; their association is more likely to be a result of hidden common causes, such as government policies, or global financial considerations that influence the various stock markets.

The same difficulties seemed to arise in other applications. The RAI algorithm was applied to the genetic data found in Friedman et. al. [46]. Tentative results seem to give substantially different output depending on the input order of the variables, suggesting hidden common causes.

16.13.6 Conclusion

Constraint based algorithms offer a fast approach, which is convenient with data matrices when d, the number of variables, is very large. They can be many times faster than search and score algorithms. Unfortunately, these algorithms tend to assume 'faithfulness' and work on the principle of removing an edge whenever a conditional independence test gives the result 'do not reject $X \perp Y|S$'. This leads to several difficulties. Firstly, since tests with larger conditioning sets are weaker, it can lead to situations where deletion of an edge can contradict earlier d-connection statements. This difficulty is present even if there is a faithful DAG corresponding to the independence structure. Secondly, two-factor, or higher order interactions are not detected if there are no 'main effects'. Thirdly, hidden variables can lead to contradictory edges, resulting in d-separation statements not present in the probability distribution. If there is no faithful DAG that describes the underlying independence structure, this can manifest itself in other ways.

Modifications to remove the first of these difficulties have been considered, for example by Bromberg and Margaritis [9] using argumentation and the more conservative approach of Barros [4] retaining all dependence statements that have been established through rejecting independence.

The second and third of these difficulties have not been fully addressed by constraint based algorithms.

16.13.7 The 'Causal Discovery' Controversy

The discussion about structure learning has described various methods to locate structures that represent the independence relations within a data set. All these methods, search and score, constraint based, hybrid, yield results that fall under the heading of *descriptive statistics*. The search and score methods simply examine some of the available structures and choose the structure with the highest score of those examined. On the 'classical' side, there is no measure of confidence for the structure chosen; on the 'Bayesian' side, even if a prior distribution is placed over the structure space and the posterior used as the basis of a score function, there is no posterior assessment of the probability for the structure to lie in a certain subspace of the set of possible structures; only a small number of structures are visited and the structure chosen is the one visited that gives the largest score. With constraint based methods, even if the hypothesis that the data matrix represents n instantiations of i.i.d. random vectors held, the number of tests is so large that even with a small nominal significance level for each test, the overall significance level approaches 1.

The output structure can give useful information at the level of descriptive statistics, but little or no formal inference can be made. This is generally the case in multivariate statistics, where methods are often more successful as descriptive than inferential tools.

Assume, though, that statistical associations have been established. Substantial parts of the literature suggest claims that a rigorous engine for inferring *causation* from association has been established. For example, Spirtes, Glymour and Scheines (1993) [126] claim to have algorithms for discovering causal relations based only on empirical data. The underlying assumption seems to be that, for a large class of problems, when immoralities are learned from data and Meek's rules then applied, cause to effect can be inferred for the directed edges of the essential graph. Schmidt, Niculesu-Mizil and Murphy (2007) [123] write, explaining why they are constructing techniques to produce *directed* graphs,

> '... undirected models cannot be used to model causality in the sense of Pearl [109], which is useful in many domains such as molecular biology, where interventions can be performed.'

The thrust of the quote is that directed edges whose direction can be interpreted as cause to effect, can be learned from data. But placing a causal interpretation on a directed arrow in a graph that has been learned purely by applying a structure learning algorithm to data can be misleading.

In a situation where interventions can be performed, a *causal* directed graph can be obtained from the undirected graph through further controlled experiments. Consider the situation on three variables (X, Y, Z) where $X \perp Z|Y$, but $X \not\perp Y$, $X \not\perp Z$, $Y \not\perp Z$, $Y \not\perp X|Z$ and $Y \not\perp Z|X$. There are three DAGs along which the distribution $p_{X,Y,Z}$ may be factorised, given in Figure 16.14. Suppose that an intervention may be carried out on the variable Y, forcing its state. This has the effect of removing arrows from parents of Y to Y. If the state $Y \leftarrow y$ is forced, this gives the graphs in Figure 16.15.

If all the states of Y can be explored, in a controlled experiment, by randomly assigning levels of the 'treatment' variable Y, the causal structure can be determined from the Markov structure, but not otherwise.

Markowetz and Spang [91] discuss the application of intervention calculus for perturbation experiments that are inferring gene function and regulatory pathways.

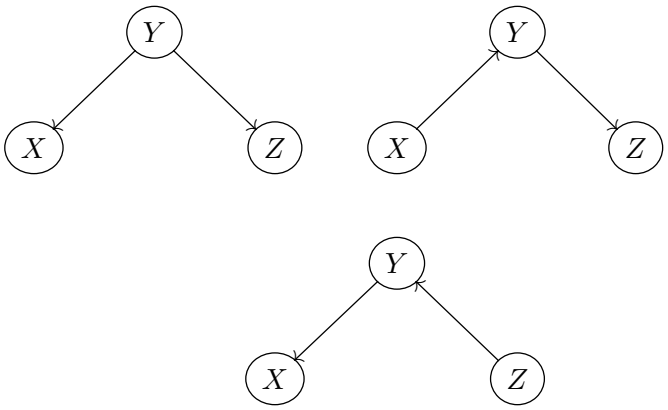

Figure 16.14: Three Markov equivalent DAGs

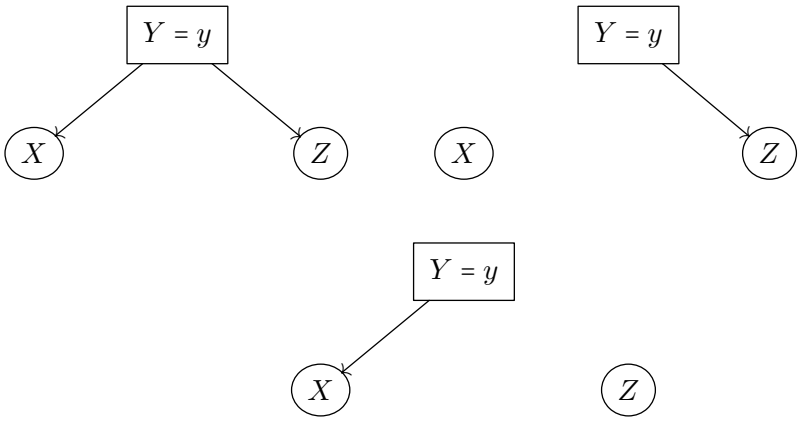

Figure 16.15: Intervention $Y \leftarrow y$ in Figure 16.14

As Freedman and Humphreys point out (1999) [43], commenting on automated causal learning, 'these claims are premature at best and the examples used in [126] to illustrate the algorithms are indicative of failure rather than success.' They point out that 'the gap between association and causation has yet to be bridged.'

16.13.8 Faithfulness and the great leap of faith

One of the leading assumptions behind 'causal discovery' is the assumption that distributions of interest satisfy the faithfulness assumption, that there is a DAG $\mathcal{G}$ with variable set $V = (U, O)$ where U denotes the unobserved variables and O the observed variables and a probability distribution $\mathbb{P}$ over (U, O) such that $\mathbb{P}$ factorises along $\mathcal{G}$ and $\mathcal{G}$ gives a faithful graphical representation of the independence structure.

This is described as follows;

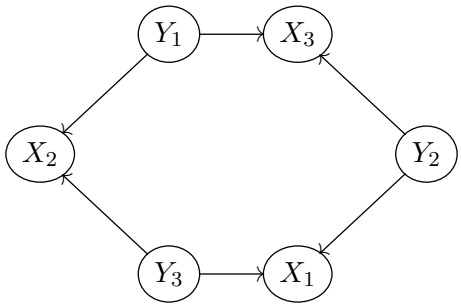

Figure 16.16: DAG for the natural factorisation; it is not faithful

> ' .... the faithfulness condition can be thought of as the assumption that conditional independence relations are due to causal structure rather than to accidents of parameter values.' Spirtes et. al. (2000) [127]

Example 2.7 gives an instance of a situation where the probability distribution does not have faithful graphical representation. For the variables $(Y_1, Y_2, Y_3, X_1, X_2, X_3)$, the DAG that best represents the associations between the variables is given by Figure 16.16. In this graph, $X_1 = 1$ if $Y_2 = Y_3$ and 0 otherwise. $X_1 \perp Y_2$ and $X_1 \perp Y_3$, but $X_1 \not\perp \{Y_2, Y_3\}$. In this situation the influence of Y_2 and Y_3 on X_1 is not seen if the variables are considered separately, but the interaction effect is decisive.

Another statement of the same principle is found in Meek (1995) [93]

> In cases where $\mathcal{P}(\mathcal{G})$ (the set of distributions that factorise along a graph $\mathcal{G}$) can be parametrised by a family of distributions with a parameter of finite dimensions, the set of unfaithful distributions typically has Lebesgue measure zero. (Spirtes et. al. (2000) [127] pp 42 - 2)

This assumption, that the set of observable variables O may be extended to a set $V = (U, O)$ where U represents unobserved common causes, or confounders, and that there will exist a DAG over V that is faithful to the probability distribution over V, is re-stated in Robins, Scheines, Spirtes and Wasserman (2003) [117]. There is strong interest in classes of faithful distributions in the literature; the work of Zhang and Spirtes [151] requires that the class of distributions under consideration satisfy a stronger assumption than faithfulness in order to obtain uniform consistency in causal inference for a certain class of problems; [117] illustrates non-existence of uniform consistency when only faithfulness is assumed, because of the possibility of non-faithful distributions in the closure of the set of distributions under consideration.

Consider again Example 2.7 and suppose that $O = (X_1, X_2, X_3)$, the values for (X_1, X_2, X_3) are observable and $U = (Y_1, Y_2, Y_3)$, the results of (Y_1, Y_2, Y_3) are hidden. Clearly, the set of distributions over 6 binary variables that factorises over the DAG in Figure 16.16 can be described by a finite parameter space; 15 parameters are required to describe the entire set of distributions; the parameter space is $[0, 1]^{15}$. Furthermore, it is clear that the parameters to describe the distribution over $(Y_1, Y_2, Y_3, X_1, X_2, X_3)$ in Example 2.7 correspond to exactly one point in the parameter space, which

has Lebesgue measure zero. Nevertheless, examples where knowledge of two causes is required to explain the effect and where knowledge only of a single cause tells you nothing about an effect arise all the time in practise, in the real world.

Furthermore, the parametrisation of any distribution that has an independence structure has Lebesgue measure zero in the parameter space of all distributions over the variables in question. Meek's argument can equally well be used to argue against searching for any independence structure at all.

Faithfulness appears a convenient hypothesis to produce beautiful mathematics (and the relation between DAGs and probability distributions under this assumption has produced a very elegant and attractive mathematical theory), but it is difficult to see that it necessarily applies to real world situations; the real world does not respect the fact that the set of parameters that describe the situation have Lebesgue measure zero in a mathematical parameter space. Divergence between 'real world' behaviour and the assumption that it should fit into a convenient mathematical framework has been termed 'The Mind Projection Fallacy' by E.T. Jaynes (2003) [70].

16.13.9 Inferring non-causation and causation

Robins, Scheines, Spirtes and Wasserman (2003) [117] describe situations where *non*-causation can be inferred. A situation where such an inference can be made is given by Figure 16.12 representing the *causal* associations between variables, where H is hidden and X, Y, W are observable. In this example, X is not a cause of Y, neither is Y a cause of X. This can be inferred from the CI tests; from the results $X \perp Z|W$ and $Y \perp W|Z$, it is possible to infer that the relation between X and Y is not cause to effect in either direction and that a common cause H would explain the test results.

The discovery of an immorality, though, does not necessarily imply causation. Suppose H_1 and H_2 are hidden and X, Z, Y are observable in Figure 16.17. The distribution over (X, Z, Y) factorises according to Figure 16.18.

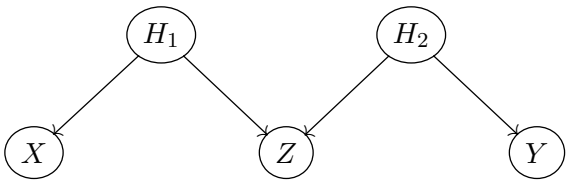

Figure 16.17: H_1 and H_2 hidden

If one were using immoralities as a guide to causation, one would conclude that X and Y were common causes of Z. As Freedman and Humphreys point out in [43], commenting on Spirtes Glymour Scheines (1993) [126] on a DAG produced from a sociological data set,

> The graph says, for instance, that race and religion cause region of residence.

In the context, this is non-sensical and raises a timely note of caution when inferring causality.

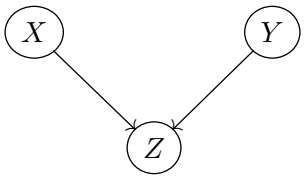

Figure 16.18: DAG for (X, Y, Z) from Figure 16.17

16.13.10 Summarising causal discovery

Freedman and Humphreys go on to summarise the attempts to automate 'causal discovery' with the example of smoking and lung cancer,

> The epidemiologists discovered an important truth - smoking is bad for you. The epidemiologists made this discovery by looking at the data and using their brains, two skills that are not readily automated. .... The examples in SGS (1993) [126] count against the automation principle, not for it.'

The conclusion drawn by the authors of this article is that the output produced by structure learning algorithms provides invaluable information. It can give good information about associations and can certainly point towards the possibility of causal relations, but they do not even begin to automate the process of learning causality; it is still necessary for researchers to use their brains to design experiments, examine the data and use their brains again, taking into account circumstances and contexts additional to the raw data, to reach conclusions. As the example from SGS (1993) [126], extended by Freedman and Humphreys [43] shows, causation cannot be deduced from the presence of an immorality and, indeed, cannot be inferred from the output of structure learning algorithms alone.

Notes The PC algorithm was introduced by Spirtes et. al. [126](1993), while the MMPC was introduced by Tsamardinos et. al. [137](2006). The FAS algorithm is discussed in Fast [41]. Recursive Autonomy Identification is due to Yehezkel and Lerner [150](2009).

16.14 Exercises

1. This problem is motivated by the following consideration: when searching for a graph with a suitable structure to fit a given data set with reasonable accuracy, Markov chain Monte Carlo techniques are often used. These algorithms are computationally more efficient if they change as few edges as possible at each transition, while ensuring that the chain can move through the entire space of graphs. It is also more efficient to search the space of essential graphs, to ensure that the chain does not spend time moving between graphs that are Markov equivalent.

   This exercise shows that even in a simple setting, it is necessary to change at least two edges per move to ensure that the algorithm can move from the current essential graph a different essential graph.

   (a) Consider a collider connection $A \to B \leftarrow C$. Is this an essential graph?

   (b) List all the essential graphs on three variables.

   (c) List all the graphs that may be obtained by altering one edge of the graph $A \to B \leftarrow C$, through either adding or removing a directed edge or an undirected edge, or from directing an undirected edge, or from 'un-directing' a directed edge, or reversing the direction of a directed edge. Which of these graphs are essential graphs?

2. Let Y_1, Y_2, Y_3 be three independent identically distributed variables with probability function $\mathbb{P}(1) = \mathbb{P}(0) = \frac{1}{2}$. Let

$$X_1 = \begin{cases} 1 & Y_2 = Y_3 \\ 0 & \text{otherwise} \end{cases}$$

$$X_2 = \begin{cases} 1 & Y_1 = Y_3 \\ 0 & \text{otherwise} \end{cases}$$

$$X_3 = \begin{cases} 1 & Y_1 = Y_2 \\ 0 & \text{otherwise} \end{cases}$$

   (a) Let $V = \{X_1, X_2, X_3\}$. Construct an undirected graph by adding an edge between two nodes α and β if and only if $\alpha \not\perp \beta | S$ for any subset $S \subseteq V \backslash \{\alpha, \beta\}$.

   (b) Construct the *independence graph*.

   (c) What happens if $V = \{Y_1, Y_2, Y_3, X_1, X_2, X_3\}$?

3. Consider the second structure in Figure 16.19.

   (a) Is it a chain graph?

   (b) Is it an essential graph? If not, why not?

   (c) If it is a chain graph, what are the chain components? Are they triangulated?

   (d) Do there exists any substructures on three variables from the graph on the right of the form of the graph on the left?

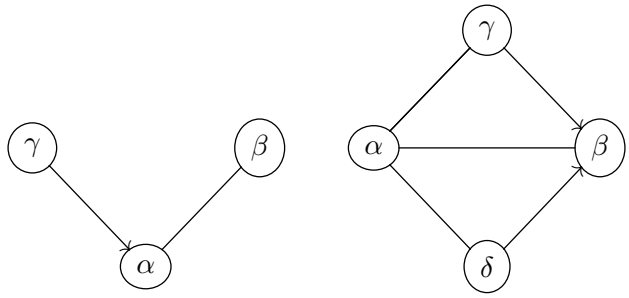

Figure 16.19: Figure for Exercise 3

The following two exercises are taken from Chickering [24].

4. For any DAG $\mathcal{G} = (V, D)$, an edge $(X, Y) \in D$ is said to be *covered* in $\mathcal{G}$ if $\text{Pa}_X = \text{Pa}_Y \backslash \{X\}$. Let $\mathcal{G}_1 = (V, D_1)$ be a DAG and let $\mathcal{G}_2 = (V, D_2)$ be obtained by reversing the edge $(X, Y) \in D_1$. Prove that $\mathcal{G}_2$ is Markov equivalent to $\mathcal{G}_1$ if and only if (X, Y) is covered in $\mathcal{G}_1$.

5. Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two Markov equivalent DAGs and suppose that there are exactly m edges in $\mathcal{G}_1$ with the opposite orientation in $\mathcal{G}_2$. Using Exercise 4, prove that there is a sequence of exactly m distinct edge reversals in $\mathcal{G}_1$ with the following properties:

   - Each edge reversed is covered when it is reversed.
   - After each reversal, the resulting graph $\mathcal{H}$ is Markov equivalent to $\mathcal{G}_2$.
   - After all reversals, $\mathcal{H} = \mathcal{G}_2$.

6. Let $\mathcal{G} = (V, D)$ be a directed acyclic graph. Prove that $\mathcal{G}^m$, the moral graph, contains an undirected edges $\langle X, Y \rangle$ if and only if $X \not\perp\!\!\!\perp Y \| _{\mathcal{G}} V \backslash \{X, Y\}$ (X and Y are not d-separated by $V \backslash \{X, Y\}$).

7. Recall the *Recursive Autonomy Identification* algorithm, Subsection 16.7 page 321.

   (a) In the description of stage 0, where an edge between X and Y is removed if and only if $X \perp Y$, assume that the resulting skeleton is correct. Why is (X, Z, Y) an immorality if there are edges $X - Y$ and $Y - Z$ but no edge $X - Y$?

   (b) Assume that the graph in Figure 16.20 is a faithful graph for $\mathbb{P}_{X_1, X_2, X_3, X_4}$. Assume that the data set is sufficiently large so that each test for independence gives the correct result. Outline how the algorithm proceeds, sketching the graphs returned at each stage of the algorithm, stating the reasons for deleting edges and directing edges.

   (c) Assume that the graph in Figure 16.21 is a faithful graph for $\mathbb{P}_{X_1, X_2, X_3, X_4}$ and that each independence test gives the correct result. Outline how the algorithm proceeds.

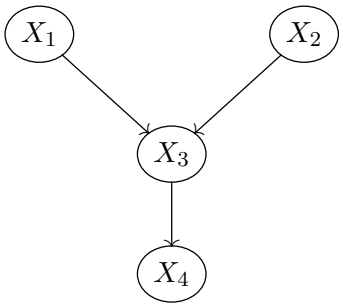

Figure 16.20: Directed acyclic graph for algorithm, example 1

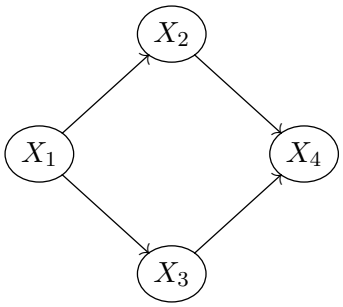

Figure 16.21: Directed acyclic graph for algorithm, example 2

(d) Assume that the graph in Figure 2.3 is faithful to the distribution $\mathbb{P}_{U_1,Z_1,Z_2,Z_3,Z_4}$ and that variable U_1 is hidden. What is the output of the RAI algorithm if the input is (Z_1, Z_2, Z_3, Z_4)? What is the output of the RAI algorithm if the input order is (Z_4, Z_3, Z_2, Z_1)?

16.15 Answers

1. (a) Yes: $A \to B \leftarrow C$ is an essential graph.

   (b) Recall that the essential graph is the graph where directions are retained on and only on those edges that retain the same direction in every graph in the Markov equivalence class. Hence $A - B \leftarrow C$ is not an essential graph since $A \to B \leftarrow C$ and $A \leftarrow B \leftarrow C$ are not Markov equivalent; if $B \leftarrow C$ is present and (A, B, C) is not an immorality, this forces $A \leftarrow B$.

   With this in mind, the essential graphs are:

   The three graphs $A \to B \leftarrow C$, $B \to A \leftarrow C$, $A \to C \leftarrow B$, the three graphs with one (undirected) edge between two of the nodes and the third node unconnected, the graph with no edges between any of the nodes, the three graphs with two undirected edges $A - B - C$, $A - C - B$, $C - A - B$. The graph with three undirected edges between A, B and C.

   (c) $A, B \leftarrow C$; $A - B \leftarrow C$; $A \leftarrow B \leftarrow C$; $A \to B, C$; $A \to B - C$; $A \to B \to C$. None of them are essential graphs.

2. (a) The graph contains no edges; since $X_1 \perp X_2$, $X_1 \perp X_3$ and $X_2 \perp X_3$.

   (b) The graph is complete; $X_1 \not\perp X_2 | X_3$, $X_1 \not\perp X_3 | X_2$ and $X_2 \not\perp X_3 | X_1$.

   (c) Again, the graph constructed according to the 'faithfulness' principle (no edge $X \sim Y$ whenever $X \perp Y | S$ for some S) is empty since $Y_1 \perp X_1$, $Y_1 \perp X_2$, $X_1 \perp X_2$. Taking $S = \phi$ (the empty set) in each case, any pair of variables is independent.

3. (a) Yes; it is a chain graph.

   (b) No; the edge $\alpha - \beta$ appears in a *compelled* configuration and should be directed $\alpha \mapsto \beta$.

   (c) There is one chain component, which contains all the nodes $\{\alpha, \beta, \gamma, \delta\}$, with the edges $\gamma \mapsto \beta$ and $\delta \mapsto \beta$ removed. It is a tree and it is clearly triangulated, because it contains no cycles.

   (d) No; $\{\gamma, \alpha, \delta\}$ has two undirected edges, $\{\gamma, \beta, \delta\}$ forms an immorality centred at β, $\{\alpha, \delta, \beta\}$ has three edges and $\{\alpha, \gamma, \delta\}$ has three edges.

   This shows that to show that a chain graph is an essential graph, it is not sufficient simply to show that the chain components are triangulated and that the substructure on the left in Figure 16.19 does not appear.

4. When edge (X, Y) is reversed in a DAG $\mathcal{G}_1$ to form a new graph $\mathcal{G}_2$, the two graphs are Markov equivalent if and only if it has the same skeleton, and the same immoralities and there are no cycles in $\mathcal{G}_2$.

   When (X, Y) is removed and (Y, X) is added, there are no new immoralities if and only if for each $Z \in \mathrm{Pa}(X)$, there is link between Y and Z. The link is (Z, Y), otherwise there is a cycle in $\mathcal{G}_1$. Therefore $\mathrm{Pa}(X) \subseteq \mathrm{Pa}(Y)$ in $\mathcal{G}_1$.

   No immoralities are removed and no cycles are introduced if and only if for any $Z \in \mathrm{Pa}(Y) \backslash \{X\}$, $Z \in \mathrm{Pa}(X)$, so $\mathrm{Pa}(Y) \backslash \{X\} \subseteq \mathrm{Pa}(X)$. It follows that $\mathrm{Pa}(X) = \mathrm{Pa}(Y) \backslash \{X\}$.

5. Assume that none of the m edges are covered. Then using the previous exercise, for each edge (X,Y) to be altered, either there is a node $Z \in \text{Pa}(Y)\backslash\text{Pa}(X)$ or there is a node $Z \in \text{Pa}(X)\backslash\text{Pa}(Y)$. If there is a node $Z \in \text{Pa}(Y)\backslash\text{Pa}(X)$ then (X,Y,Z) is an immorality, so that the direction $X \to Y$ remains the same in any Markov equivalent graph. It follows that for each of the m edges (X,Y), there is a variable $Z \in \text{Pa}(X)\backslash\text{Pa}(Y)$. If the direction of the edge (Z,X) is not also reversed, then (Z,X,Y) is an immorality in the new graph, which is a contradiction. It follows that there is at least one covered edge among the m edges. Change the orientation of this edge. After the change, there is a covered edge among the remaining $m-1$ and by induction the target graph is obtained after m changes.

6. Firstly, note that the moral graph contains an edge $X - Y$ if and only if $Y \in MB(X)$, the Markov blanket of X. $MB(X)$ is the set X, together with $\text{Pa}(X)$ (the parents of X) and $\text{Ch}(X)$ (the children of X) and all parents that share a child with X. That is, X together with all neighbours of X and those variables that are linked to X when the graph is moralised.

   Note that

   $$X \perp V\backslash MB(X) \| _{\mathcal{G}} MB(X)$$

   so that, using the weak union result of Exercise 2 page 22, if $Y \notin MB(X)$,

   $$X \perp Y \| _{\mathcal{G}} V\backslash\{X,Y\}.$$

   If $Y \in MB(X)$, then the moral graph contains an edge $X - Y$ and $X \not\perp Y \| _{\mathcal{G}} V\backslash\{X,Y\}$.

7. (a) If $X - Z - Y$ is a fork or chain connection, $X \not\perp Y$ so that $X - Y$ would not be removed. It follows that if the vee structure $X - Z - Y$ remains in the final graph, it is an immorality.

   (b) Stage 1: add all edges, all are undirected.

   Stage 2: $X_1 \perp X_2$ so remove $X_1 - X_2$. No other edges removed. $X_1 - X_3 - X_2$ is a collider, $X_1 - X_4 - X_2$ is a collider. No additional compelled edges.

   The chain components are: $\{X_1\}$, $\{X_2\}$ and $\{X_3, X_4\}$.

   Stage 3: Consider chain component $\{X_3, X_4\}$ together with chain components containing parents. $X_1 \perp X_4|X_3$, $X_2 \perp X_4|X_3$ so remove $X_1 - X_4$ and $X_2 - X_4$. Now $X_3 \to X_4$ is compelled.

   There are now no parent sets of size greater than 1 hence the algorithm terminates.

   (c) Stage 1: add all edges to make the complete undirected graph.
   Stage 2: test $X \perp Y|\phi$; no edges removed, none of the variables are (pairwise) independent. There is therefore only one chain component after this stage; the complete graph. Stage 3: test $X \perp Y|Z$ for each triple (12 tests). The only independence result is $X_2 \perp X_3|X_1$, therefore edge $\langle X_2, X_3\rangle$ is removed. The triple (X_2, X_4, X_3) is an immorality. The triple (X_2, X_1, X_3) is not an immorality. By Meek's rules, the edge $\langle X_1, X_4\rangle$ is compelled $X_1 \to X_4$.
   Stage 4: the chain components are the subgraph with $\{X_2, X_1, X_4\}$ and the subgraph $\{X_4\}$. Start with $\{X_4\}$, this is connected to $\{X_2, X_1, X_4\}$. At this stage, $X_1 \perp X_4|\{X_2, X_3\}$ so that

the edge $X_1 \rightarrow X_4$ is removed. The algorithm now terminates, returning the essential graph with undirected edges $\langle X_1, X_2 \rangle$ and $\langle X_1, X_3 \rangle$ and directed edges $\langle X_2, X_4 \rangle$ and $\langle X_3, X_4 \rangle$.

Chapter 17

Bayesian Networks in R: Structure and Parameter Learning

17.1 Bayesian Networks with bnlearn

This tutorial is based predominantly on the **bnlearn** package, a package by Marco Scutari. It supports a wide variety of structure learning algorithms. These are found in the documentation. They are:

Constraint Based Algorithms

1. **Grow-Shrink** `gs`: based on the Grow-Shrink Markov Blanket, the first (and simplest) Markov blanket detection algorithm used in a structure learning algorithm.

2. **Incremental Association** `iamb`: based on the Markov blanket detection algorithm of the same name, which is based on a two-phase selection scheme (a forward selection followed by an attempt to remove false positives).

3. **Fast Incremental Association** `fast.iamb`: a variant of IAMB which uses speculative stepwise forward selection to reduce the number of conditional independence tests.

4. **Interleaved Incremental Association** `inter.iamb`: another variant of IAMB which uses forward stepwise selection to avoid false positives in the Markov blanket detection phase.

Search and Score Learning Algorithms

1. **Hill-Climbing** `hc`: a hill climbing greedy search on the space of the directed graphs. The optimised implementation uses score caching, score decomposability and score equivalence to reduce the number of duplicated tests.

2. **Tabu Search** `tabu`: a modified hill climbing able to escape local optima by selecting a network that minimally decreases the score function.

Hybrid Learning Algorithms

1. **Max-Min Hill-Climbing** `mmhc`: a hybrid algorithm which combines the Max-Min Parents and Children algorithm (to restrict the search space) and the Hill-Climbing algorithm (to find the optimal network structure in the restricted space).

2. **Restricted Maximization** `rsmax2`: a more general implementation of the Max-Min Hill-Climbing, which can use any combination of constraint-based and score-based algorithms.

Other (Constraint-Based) Learning Algorithms

These algorithms learn the structure of the undirected graph underlying the Bayesian network, which is known as the skeleton of the network or the (partial) correlation graph. Therefore all the arcs are undirected, and no attempt is made to detect their orientation. They are often used in hybrid learning algorithms.

1. **Max-Min Parents and Children** `mmpc`: a forward selection technique for neighbourhood detection based on the maximization of the minimum association measure observed with any subset of the nodes selected in the previous iterations.

2. **Hiton Parents and Children** `si.hiton.pc`: a fast forward selection technique for neighbourhood detection designed to exclude nodes early based on the marginal association. The implementation follows the Semi-Interleaved variant of the algorithm.

3. **Chow-Liu** `chow.liu`: an application of the minimum-weight spanning tree and the information inequality. It learn the tree structure closest to the true one in the probability space.

4. **ARACNE** `aracne`: an improved version of the Chow-Liu algorithm that is able to learn polytrees.

17.1.1 Creating and Manipulating Network Structures

The following illustrates how to create objects of class `bn`. We consider the `marks` data set, which gives the exam scores of 88 students across five different topics: mechanics, vectors, algebra, analysis and statistics. The original data set was investigated by Mardia et. al. (1979) [90] and subsequently became a bench mark for structure learning (e.g. Whittaker (1990) [144]. It is a data set within the **bnlearn** package under the name `marks`.

```
> library(bnlearn)
> data(marks)
> str(marks)
'data.frame': 88 obs. of  5 variables:
 $ MECH: num  77 63 75 55 63 53 51 59 62 64 ...
 $ VECT: num  82 78 73 72 63 61 67 70 60 72 ...
```

```
 $ ALG : num  67 80 71 63 65 72 65 68 58 60 ...
 $ ANL : num  67 70 66 70 70 64 65 62 62 62 ...
 $ STAT: num  81 81 81 68 63 73 68 56 70 45 ...
```

First create an empty network with the nodes corresponding to the variables using the `empty.graph` function:

```
> ug<-empty.graph(names(marks))
```

The arcs presented in Whittaker (1990) from Figure 17.1 may be added as follows:

```
> arcs(ug,ignore.cycles=TRUE)=matrix(
+ c("MECH","VECT","MECH","ALG","VECT","MECH",
+ "VECT","ALG","ALG","MECH","ALG","VECT",
+ "ALG","ANL","ALG","STAT","ANL","ALG",
+ "ANL","STAT","STAT","ALG","STAT","ANL"),
+ ncol=2, byrow = TRUE,
+ dimnames=list(c(),c("from","to")))
> plot(ug)
```

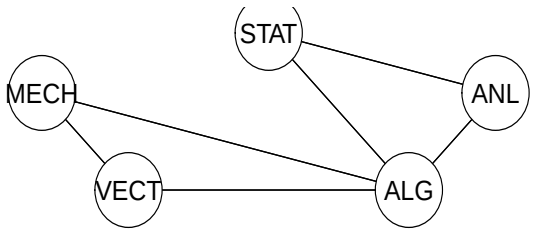

Figure 17.1: Marks network: undirected graph

The resuting `ug` object belongs to graph `bn`. There are several arguments: `ug$learning`, `ug$nodes`, `ug$arcs`.

`learning` is not useful in this example, since this argument gives information about the results of the structure learning algorithm used to generate the network and its tuning parameters (which were not used here).

`$nodes` gives information about the Markov blanket of each node, while `$arcs` gives the arcs presented in the network.

```
> ug

  Random/Generated Bayesian network

  model:
    [undirected graph]
  nodes:                                5
  arcs:                                 6
    undirected arcs:                    6
    directed arcs:                      0
  average markov blanket size:          2.40
  average neighbourhood size:           2.40
  average branching factor:             0.00

  generation algorithm:                 Empty

> dag = empty.graph(names(marks))
> arcs(dag)=matrix(c("VECT","MECH","ALG","MECH","ALG","VECT",
+ "ANL","ALG","STAT","ALG","STAT","ANL"),
+ ncol=2,byrow=TRUE,
+ dimnames=list(c(),c("from","to")))
> dag

  Random/Generated Bayesian network

  model:
   [STAT][ANL|STAT][ALG|ANL:STAT][VECT|ALG][MECH|VECT:ALG]
  nodes:                                5
  arcs:                                 6
    undirected arcs:                    0
    directed arcs:                      6
  average markov blanket size:          2.40
  average neighbourhood size:           2.40
  average branching factor:             1.20

  generation algorithm:                 Empty
```

A dag can be specified by its adjacency matrix. The function `all.equal()` indicates whether two graphs are equal.

```
> mat=matrix(c(0,1,1,0,0,0,0,1,0,0,0,0,
```

```
+ 0,1,1,0,0,0,0,1,0,0,0,0,0),
+ nrow=5,
+ dimnames=list(nodes(dag),nodes(dag)))
> mat
     MECH VECT ALG ANL STAT
MECH    0    0   0   0    0
VECT    1    0   0   0    0
ALG     1    1   0   0    0
ANL     0    0   1   0    0
STAT    0    0   1   1    0
> dag2=empty.graph(nodes(dag))
> amat(dag2)=mat
> all.equal(dag,dag2)
[1] TRUE
```

A new `bn` object may be created by adding (`set.arc`), dropping (`drop.arc`) or reversing `rev.arc`) arcs from the original. For example:

```
> dag3 = empty.graph(nodes(dag))
> dag3 = set.arc(dag3,"VECT","MECH")
> dag3 = set.arc(dag3,"ALG","MECH")
```

A topological ordering of the nodes (from ancestors to descendants) may be obtained by the function `node.ordering()`. The neighbours and Markov blanket may be found using `nbr()` and `mb()` respectively. The `%in%` command may be used to establish membership.

```
> node.ordering(dag)
[1] "STAT" "ANL"  "ALG"  "VECT" "MECH"
> nbr(dag,"ANL")
[1] "ALG"  "STAT"
> mb(dag,"ANL")
[1] "ALG"  "STAT"
> "ANL" %in% mb(dag,"ALG")
[1] TRUE
```

We can check that the Markov blanket of a variable consists of parents, children and children of parents:

```
> chld=children(dag,"VECT")
> par=parents(dag,"VECT")
> o.par=sapply(chld,parents,x=dag)
> unique(c(chld,par,o.par[o.par != "VECT"]))
[1] "MECH" "ALG"
> mb(dag,"VECT")
[1] "MECH" "ALG"
```

17.1.2 Visualising Graphical Models

The structures from **bnlearn** may be plotted using functions provided by **graph** and **Rgraphviz** packages (Gentry et. al. [52](2012)). The `graphviz.plot` function takes a `bn` object and returns the corresponding `graph` object.

In **bnlearn**, vee-structure refers to a collider connection.

```
> library(Rgraphviz)
Loading required package: grid
> h = list(arcs=vstructs(dag2,arcs=TRUE),lwd=4,col="black")
> graphviz.plot(dag2,highlight=h,layout="fdp",main="dag2")
```

The output is shown in Figure 17.2.

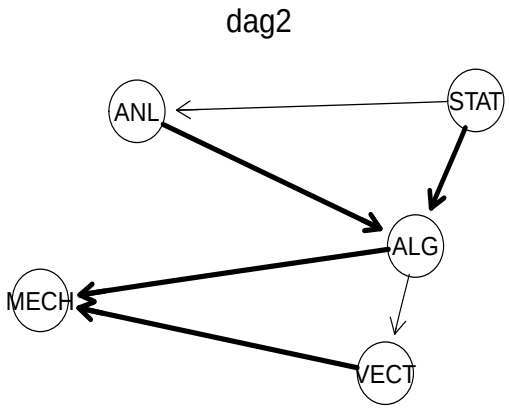

Figure 17.2: Plot obtained using `graphviz.plot`

The *essential graph*, showing the Markov equivalence class is returned by `cpdag`.The function `moral` returns the moral graph.

```
> plot(cpdag(dag2))
```

for example gives a plot of the essential graph corresponding to `dag2`.

17.1.3 Structure Learning

In **bnlearn**, the Maximum Minimum Parents Children (MMPC) algorithm is referred to as the *grow-shrink* algorithm. The name is natural following the procedure; first the maximum parents / children set for each node is established and then unnecessary nodes are removed. This algorithm is implemented simply by the function `gs`.

```
> bn.gs <- gs(marks)
> bn.gs
```

```
  Bayesian network learned via Constraint-based methods

  model:
    [undirected graph]
  nodes:                                5
  arcs:                                 6
    undirected arcs:                    6
    directed arcs:                      0
  average markov blanket size:          2.40
  average neighbourhood size:           2.40
  average branching factor:             0.00

  learning algorithm:                   Grow-Shrink
  conditional independence test:        Pearson's Correlation
  alpha threshold:                      0.05
  tests used in the learning procedure: 44
  optimized:                            TRUE
```

The parameter value $\alpha = 0.05$ is the nominal significance level for each χ^2 test for independence. The `mmhc` algorithm learns a different network, but it is Markov equivalent to the network learned by the `gs` algorithm and has the same BIC score.

These structure learning algorithms often only direct an edge when a particular direction gives a better fit, leaving other edges undirected. The function `cextend()` gets one graph out of the Markov equivalence class, which may be used for scoring purposes. The BIC score for the learned graph may be obtained as follows. The documentation lists other scoring criteria that are available (such as AIC).

```
> bn.gsdirect <- cextend(bn.gs)
> bn.gsdirect

  Bayesian network learned via Constraint-based methods

  model:
   [STAT][ANL|STAT][ALG|ANL:STAT][VECT|ALG][MECH|VECT:ALG]
  nodes:                                5
  arcs:                                 6
    undirected arcs:                    0
    directed arcs:                      6
  average markov blanket size:          2.40
  average neighbourhood size:           2.40
```

```
  average branching factor:              1.20

  learning algorithm:                    Grow-Shrink
  conditional independence test:         Pearson's Correlation
  alpha threshold:                       0.05
  tests used in the learning procedure:  44
  optimized:                             TRUE

> score(bn.gsdirect, data=marks, type="bic-g")
[1] -1720.15
```

17.1.4 Parameter Learning

Having established the network, the next task is to learn the parameters. With **bnlearn**, this is performed by the `bn.fit` function.

```
> fitted = bn.fit(bn.gsdirect, data=marks)
> fitted

  Bayesian network parameters

  Parameters of node MECH (Gaussian distribution)

Conditional density: MECH | VECT + ALG
Coefficients:
(Intercept)         VECT          ALG
-12.3647583    0.4658693    0.5484053
Standard deviation of the residuals: 13.97432

  Parameters of node VECT (Gaussian distribution)

Conditional density: VECT | ALG
Coefficients:
(Intercept)          ALG
 12.4183094    0.7543653
Standard deviation of the residuals: 10.48167

  Parameters of node ALG (Gaussian distribution)

Conditional density: ALG | ANL + STAT
Coefficients:
```

```
(Intercept)          ANL         STAT
 24.7254768    0.3482454    0.2273881
Standard deviation of the residuals: 6.871428

  Parameters of node ANL (Gaussian distribution)

Conditional density: ANL | STAT
Coefficients:
(Intercept)         STAT
 24.5824229    0.5223601
Standard deviation of the residuals: 11.86392

  Parameters of node STAT (Gaussian distribution)

Conditional density: STAT
Coefficients:
(Intercept)
   42.30682
Standard deviation of the residuals: 17.25559
```

The type of estimator (maximum likelihood or Bayes) can be specified by either `mle` (maximum likelihood estimates) or `Bayes` the posterior Bayesian estimate arising from a flat, non-informative prior. Only `mle` is available with continuous (Gaussian) data; the `Bayes` considers Dirichlet densities over the parameter space.

The parameters of a fitted network can easily be replaced. For example, `ALG` has two parents, `ANL` and `STAT`. For the Gaussian network, the restriction is that the standard deviation for the residuals at each node is the same. We consider

$$\texttt{ALG} = \beta_0 + \texttt{ANL}\beta_1 + \texttt{STAT}\beta_2 + \epsilon$$

where $\epsilon \sim N(0, \sigma^2)$, independent identically distributed. This is carried out by:

```
> fitted$ALG = list(coef=c("(Intercept)"=25, "ANL"=0.5, "STAT"=0.25),sd=6.5)
> fitted$ALG

  Parameters of node ALG (Gaussian distribution)

Conditional density: ALG | ANL + STAT
Coefficients:
(Intercept)          ANL         STAT
      25.00         0.50         0.25
Standard deviation of the residuals: 6.5
```

A `bn.fit` object can be created from scratch using the `custom.fit` function. For example:

```
> MECH.par = list(coef=c("(Intercept)"=-10, "VECT"=0.5, "ALG"=0.6),sd = 13)
> VECT.par = list(coef=c("(Intercept)"=10, "ALG"=1),sd=10)
> ALG.par=list(coef=c("(Intercept)"=25,"ANL"=0.5,"STAT"=0.25),sd=6.5)
> ANL.par=list(coef=c("(Intercept)"=25,"STAT"=0.5),sd=12)
> STAT.par=list(coef=c("(Intercept)"=43),sd=17)
> dist=list(MECH=MECH.par,VECT=VECT.par,ALG=ALG.par,ANL=ANL.par,STAT=STAT.par)
> fitted2 = custom.fit(bn.gsdirect,dist=dist)
```

17.1.5 Discretisation

The only continuous models that can be accommodated are Gaussian. When the data is manifestly not Gaussian, it is better to discretise it and to construct a Bayesian network over multinomial variables. There are several methods of discretisation available; look up the documentation for `discretize`. For example:

```
> ?discretize
> dmarks = discretize(marks, breaks=2, method="quantile")
> bn.dgs=gs(dmarks)
> plot(bn.dgs)
> all.equal(cpdag(bn.dgs),cpdag(bn.gsdirect))
[1] "Different number of directed/undirected arcs"
```

The network learned from the discretised data is different; `MECH` is independent of the other variables.

The parameters may be fitted to the structure using the discretised data:

```
> fitted3=bn.fit(cextend(bn.dgs),data=dmarks)
> fitted3$ALG

  Parameters of node ALG (multinomial distribution)

Conditional probability table:

         ANL
ALG          [9,49]   (49,70]
  [15,50] 0.7777778 0.2558140
  (50,80] 0.2222222 0.7441860
```

17.1.6 Latent Variables

Probability distributions often fail to have a faithful graphical representation because there are latent (or hidden) variables missing from the model.

For the `marks` data, Edwards (2000) [40] assumed that the students fell into two distinct groups (which we call A and B). He then used a classification technique involving the EM algorithm to assign the students to two differenc classes. The results were as follows: group A contained students 1-44 and 46-52 while group B contained students 45 and 53 - 88. We add in this latent variable and we construct a network for group A and another network for group B. We then discretize the variables and learn the network when the latent variable is included. The results are:

```
> latent=factor(c(rep("A",44),"B",rep("A",7),rep("B",36)))
> bn.A = hc(marks[latent=="A",])
> bn.B = hc(marks[latent=="B",])
> modelstring(bn.A)
[1] "[MECH][ALG|MECH][VECT|ALG][ANL|ALG][STAT|ALG:ANL]"
> modelstring(bn.B)
[1] "[MECH][ALG][ANL][STAT][VECT|MECH]"
> dmarks=discretize(marks,breaks=2,method="interval")
> dmarks2=cbind(dmarks,LAT=latent)
> bn.LAT=hc(dmarks2)
> bn.LAT

  Bayesian network learned via Score-based methods

  model:
   [MECH][ANL][LAT|MECH:ANL][VECT|LAT][ALG|LAT][STAT|LAT]
  nodes:                                 6
  arcs:                                  5
    undirected arcs:                     0
    directed arcs:                       5
  average markov blanket size:           2.00
  average neighbourhood size:            1.67
  average branching factor:              0.83

  learning algorithm:                    Hill-Climbing
  score:                                 BIC (disc.)
  penalization coefficient:              2.238668
  tests used in the learning procedure:  40
  optimized:                             TRUE
```

Note that for the learned network, variable `LAT` has two parents; `MECH` and `ANL`. If `MECH`, `VECT`, `ALG`, `ANL`, `STAT` were continuous, this distribution would therefore not fall into the CG framework.

17.1.7 Application to Gene Expression Data

The analysis for large arrays of gene expression data is dealt with in the following steps:

1. Outliers are removed. This is because, for continuous data, Bayesian Networks only supports multivariate Gaussian distributions; outliers make the Gaussian modelling assumptions less likely to hold.

2. Structure learning is repeated several times, so that there is more chance of finding a global maximiser for the score function.

3. The networks discovered in the previous step are *averaged*. This is a technique from Claskens and Hjort (2008) [29]. The averaged network uses arcs present in (say) 85% of the networks.

We try this on the `sachs.data.txt` data set, found in the data directory of the course home page:

```
> library(bnlearn)
> sachs.data <- read.delim("~/data/sachs.data.txt")
> sachs<-sachs.data
> dsachs=discretize(sachs,method="hartemink",breaks=3,ibreaks=60,idisc="quantile")
```

Each variable in the `dsachs` data frame is a factor with three levels, corresponding approximately to low, normal and high expression. Now apply bootstrap resampling to learn a set of 500 networks to be used for model averaging:

```
> boot=boot.strength(data=dsachs,R=500,algorithm="hc",algorithm.args=list(score="bde",iss=10))
> boot[(boot$strength>0.85)&(boot$direction>=0.5),]
        from       to strength direction
1       praf     pmek    1.000 0.5180000
23      plcg     PIP2    1.000 0.5100000
24      plcg     PIP3    1.000 0.5220000
34      PIP2     PIP3    1.000 0.5120000
56    p44.42 pakts473    1.000 0.5620000
57    p44.42      PKA    0.992 0.5665323
67  pakts473      PKA    1.000 0.5690000
89       PKC      P38    1.000 0.5100000
90       PKC     pjnk    1.000 0.5100000
100      P38     pjnk    0.954 0.5062893
```

The virtual sample size is 10, which is very low. Arcs are significant if they appear in at least 85% of the networks and in the direction that appears most frequently. The averaged network is formed quite simply using the `averaged.network` function:

```
> avg.boot = averaged.network(boot,threshold=0.85)
```

An alternative approach is to average the results of several hill climbing searches, each starting from a different network. The initial condition can be generated using a distribution over the space of connected graphs. An algorithm to do this was proposed by Ide and Cozman [69](2002). It is implemented by the function `random.graph()`. It is carried out as follows:

```
> library("bnlearn", lib.loc="~/R/x86_64-redhat-linux-gnu-library/3.1")
> nodes=names(dsachs)
> start=random.graph(nodes=nodes,method="ic-dag",num=500)
> netlist=lapply(start,function(net){
+ hc(dsachs,score="bde",iss=10,start=net)})
> rnd=custom.strength(netlist,nodes=nodes)
> rnd[(rnd$strength>0.85)&(rnd$direction>=0.5),]
        from       to strength direction
1       praf     pmek        1     0.500
11      pmek     praf        1     0.500
23      plcg     PIP2        1     0.500
24      plcg     PIP3        1     0.620
33      PIP2     plcg        1     0.500
34      PIP2     PIP3        1     0.620
56    p44.42 pakts473        1     0.500
57    p44.42      PKA        1     0.507
66  pakts473   p44.42        1     0.500
67  pakts473      PKA        1     0.507
89       PKC      P38        1     0.500
90       PKC     pjnk        1     0.500
99       P38      PKC        1     0.500
100      P38     pjnk        1     0.500
109     pjnk      PKC        1     0.500
110     pjnk      P38        1     0.500
> avg.start=averaged.network(rnd,threshold=0.85)
Warning messages:
1: In averaged.network.backend(strength = strength, nodes = nodes,  :
  arc pjnk -> PKC would introduce cycles in the graph, ignoring.
2: In averaged.network.backend(strength = strength, nodes = nodes,  :
  arc pjnk -> P38 would introduce cycles in the graph, ignoring.
> all.equal(cpdag(avg.boot),cpdag(avg.start))
[1] TRUE
```

The networks have the same skeleton, although some of the directions are different.

The score is computed first by taking `cpdag` to get an essential graph and then by taking `cextend` to form a dag.

```
> score(cextend(cpdag(avg.start)),dsachs,type="bde",iss=10)
[1] -8498.877
```

The **bnlearn** package contains a default level for the threshold, which is found in `averaged.network`

```
> averaged.network(boot)

  Random/Generated Bayesian network

  model:
   [praf][plcg][p44.42][PKC][pmek|praf][PIP2|plcg][pakts473|p44.42][P38|PKC]
   [pjnk|PKC][PIP3|plcg:PIP2][PKA|p44.42:pakts473]
  nodes:                                 11
  arcs:                                  9
    undirected arcs:                     0
    directed arcs:                       9
  average markov blanket size:           1.64
  average neighbourhood size:            1.64
  average branching factor:              0.82

  generation algorithm:                  Model Averaging
  significance threshold:                0.954
```

The default threshold is computed as follows: Let

$$\widehat{\mathbf{p}}_{(.)} = \{0 \leq \widehat{p}_{(1)} \leq \ldots \leq \widehat{p}_{(k)} \leq 1\}$$

denote the order statistics for the arc strengths stored in `boot`. Now, let $\widehat{t}$ denote a threshold and set

$$\widetilde{p}_{(k)}(t) = \begin{cases} 1 & \widehat{p}_{(k)} \geq t \\ 0 & \widehat{p}_{(k)} < t. \end{cases}.$$

This denotes the 'empirical' probability function for arc strengths for the graph where arcs are present if and only if $\widehat{p}_{(k)} \geq t$. Let $\widetilde{\mathbf{p}}_{(.)}$ denote the resulting vector.

Now choose $\widehat{t}$ to minimise

$$L_1(t, \widehat{\mathbf{p}}_{(.)}) := \int |F_{\widehat{\mathbf{p}}_{(.)}} - F_{\widetilde{\mathbf{p}}_{(.)}}| dx$$

where $F_{\widehat{\mathbf{p}}_{(.)}}$ and $F_{\widetilde{\mathbf{p}}_{(.)}}$ are the empirical distribution functions of $\widehat{\mathbf{p}}_{(.)}$ and $\widetilde{\mathbf{p}}_{(.)}$ respectively. Then $\widetilde{t}$, the threshold is chosen to minimise this.

17.1.8 Interventional Data

The data set in `sachs.interventional.txt` gives data from different experiments, where the interventions to force the levels of certain variables, differ from experiment to experiment.

```
> isachs <- read.table("~/data/sachs.interventional.txt",header=TRUE,colClasses="factor")
```

It is important that `colClasses = "factor"`.

One (less useful) way of dealing with the situation is to include the intervention `INT` in the network and make all the variables depend on it. This is done using the `whitelist` command, which contains all possible arcs from `INT` to the other nodes. These arcs are then forced to be present in the learned network structure.

```
> wh = matrix(c(rep("INT",11),names(isachs)[1:11]),ncol=2)
> bn.wh = tabu(isachs,whitelist=wh,score="bde",iss=10,tabu=50)
```

The `tabu` learning algorithm gives more stable results here.

Not all the arcs in `wh` are necessary. The `tiers2blacklist` function may be used to blacklist all arcs going towards `INT`, thus ensuring that only outgoing arcs are present.

```
> tiers=list("INT",names(isachs)[1:11])
> bl = tiers2blacklist(nodes=tiers)
> bn.tiers=tabu(isachs,blacklist=bl,score="bde",iss=10,tabu=50)
```

While the two methods given above, producing `bn.wh` and `bn.tiers` show how to force certain arrows into a network, they do not involve the structure of the intervention.

The way to model an intervention is described as follows: the value of `INT` identifies which node is subject to an intervention. Therefore, we start by constructing a named list of which observations are manipulated for each node.

```
> INT2=sapply(1:11,function(x){which(isachs$INT==x)})
> nodes=names(isachs)[1:11]
> names(INT2)=nodes
```

Now pass the list to `tabu` as an additional argument for `mbde` (the *modified* BDe score function).

```
> start=random.graph(nodes=nodes,method="melancon",num=500,burn.in=10^5,every=100)
> netlist=lapply(start,function(net){
+ tabu(isachs[,1:11],score="mbde",exp=INT2,iss=10,start=net,tabu=50)})
> bn.mbde=averaged.network(arcs,threshold=0.85)
Warning messages:
1: In averaged.network.backend(strength = strength, nodes = nodes,  :
  arc pjnk -> PKA would introduce cycles in the graph, ignoring.
```

```
2: In averaged.network.backend(strength = strength, nodes = nodes,  :
  arc PKC -> PKA would introduce cycles in the graph, ignoring.
3: In averaged.network.backend(strength = strength, nodes = nodes,  :
  arc PKC -> P38 would introduce cycles in the graph, ignoring.
4: In averaged.network.backend(strength = strength, nodes = nodes,  :
  arc pjnk -> P38 would introduce cycles in the graph, ignoring.
> bn.mbde2 <- cextend(cpdag(bn.mbde))
> graphviz.plot(bn.mbde2)
```

17.2 Exercises

1. This exercise uses the `asia` data set found in the **bnlearn** package.

   (a) Create a `bn` object with the network structure shown in Figure 17.3.

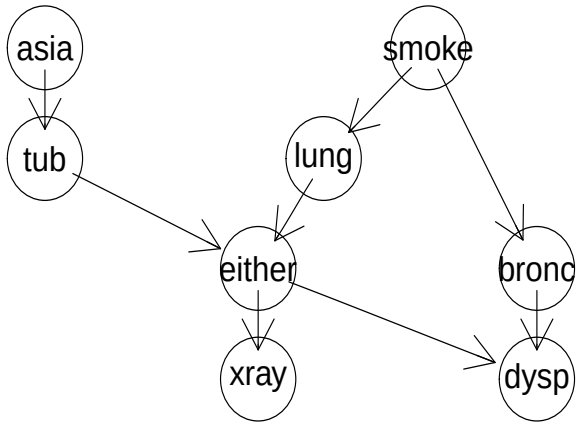

Figure 17.3: Asia Network

   (b) Derive the skeleton, the moral graph, and the essential graph representing the Markov equivalence class. Plot them using `graphviz.plot`.

   (c) Identify the parents, the children, the neighbours and the Markov blanket of each node.

   (d) For the network in Figure 17.3, estimate the CPPs.

   (e) Using the data `asia`, use the MMPC algorithm (called 'grow-shrink' in **bnlearn**) to learn the skeleton followed by hill climbing to learn the direction of the arrows. Is the output DAG Markov equivalent to the graph in Figure 17.3?

2. The `marks` data set is found in the **bnlearn** package.

   (a) Discretise the data using a quantile transform and different numbers of intervals (say 2 to 5). Learn the network structure. How does the structure change with the discretisation?

   (b) Repeat the discretisation using `interval` discretisation, using up to five intervals. Compare the resulting networks with those obtained previously using `quantile` discretisation.

   (c) Does Hartemink's discretisation algorithm perform better than either quantile or interval discretisation? How does its behaviour depend on the number of initial breaks?

3. The ALARM network is a standard network used to test new algorithms. A synthetic data set `alarm` is found in the **bnlearn** package. Type:

```
> library(bnlearn)
> ?alarm
```

On the bottom right quadrant of Rstudio, click on `ALARM Monitoring System (synthetic) data set`. This gives a description. Go to the bottom under **Examples**. You will find the structure of the 'true' network.

(a) Create a `bn` object for the true network using the model string provided in the documentation.

(b) Compare the networks learned from the data using different constraint based algorithms with the true network, both in terms of structural differences and also using either BIC or BDe.

(c) How are these constraint based strategies affected by different choices of α (the nominal significance level of each test)?

(d) Now learn the structure with hill-climbing and tabu search, using the posterior density BDe as a score function. How does the network change with the hyper parameters `iss` (imaginary sample size)?

(e) Does the length of the `tabu` list have a significant impact on the network structures learned using `tabu`?

(f) Does the learned network depend on whether BDe or BIC is being used as a score criterion?

4. Now consider the data from Sachs et. al., found in `sachs.data.txt` on the course home page. Use the original data set; not the discretised data set.

(a) Evaluate the networks leanred by hill-climbing with BIC and BGe, using cross-validation and the log-likelihood loss function.

(b) Use bootstrap resampling to evaluate the distribution of the number of arcs present in each of the networks learned. Do they differ significantly?

(c) Compute the averaged network structure for `sachs` using hill-climbing with BGe and different hyperparameters (imaginary sample sizes). How does the value of the significance threshold change as `iss` increases?

Chapter 18

Monte Carlo Algorithms for Graph Search

There are various Monte Carlo approaches to locating a structure. These involve running a stochastic process through the space of possible structures and using this either to build up a posterior distribution over the space of structures (Markov Chain Monte Carlo) or else designing a process with sufficient mobility, that is attracted to highly scoring structures and scoring each structure visited. The output from a stochastic optimisation algorithm is simply the structure visited with the highest score.

As usual, $\underline{X} = (X_1, \ldots, X_d)$ denotes the random vector of variables, $\mathbf{X} = \begin{pmatrix} \underline{X}_{(1)} \\ \vdots \\ \underline{X}_{(n)} \end{pmatrix}$ denotes an $n \times d$ random matrix of n independent copies of $\underline{X}$, $\mathbf{x}$ denotes the data matrix, an instantiation of $\mathbf{X}$.

18.1 A Stochastic Optimisation Algorithm for Essential Graphs

The following discussion is loosely based on the Markov chain Monte Carlo model composition algorithm, known as MC^3, and the augmented Markov chain Monte Carlo model composition (AMC^3) algorithm from Madigan, Andersson, Perlman and Volinsky [88] (1997). This algorithm provides a stochastic process which works through the space of essential graphs. It is not intended to provide a process that gives the correct stationary distribution; the aim is simply to find a process which is sufficiently mobile, where the direction is biased towards highly scoring structures and where the stochastic component will help the process to escape from local maxima.

Let $\mathcal{E}$ denote the space of edge sets for essential graphs. The aim is to construct a Markov chain $\{E(t), j = 1, 2, \ldots\}$ with state space $\mathcal{E}$.

Firstly, assume that $\mathbb{P}_{\mathcal{D}}(D)$ is equal for each $D \in \text{equiv}(E)$, where E denotes the edge set of an essential graph and equiv(E) denotes the space of DAGs which have E as their essential graph. The prior over DAGs, then

$$\mathbb{P}_{\mathcal{E}}(E) = n(E)\mathbb{P}_{\mathcal{D}}(D)$$

where $n(E)$ is the number of DAGs within the equivalence class and $D \in \text{equiv}(E)$. The posterior is then given by:

$$\mathbb{P}_{\mathcal{E}|\mathbf{X}}(E|\mathbf{x}) \propto \mathbb{P}_{\mathcal{E}}(E)L(D,\mathbf{x}) \qquad D \in \text{equiv}(E).$$

A penalisation may be useful; let $\kappa \in (0,1)$ and let $|E|$ denote the number of edges in the graph. If sparser graphs are desirable, then it may be useful to consider a score function

$$\mathbb{S}_{\mathcal{E}|\mathbf{X}}(E|\mathbf{x}) = \kappa^{|E|}\mathbb{P}_{\mathcal{E}|\mathbf{X}}(E|\mathbf{x}) \qquad \kappa \in (0,1). \tag{18.1}$$

The difficulty with constructing Markov chains over the set of essential graphs is that if only a single edge is modified at a time, the chain may not move. This is seen rather simply with the immorality $A \to B \leftarrow C$. This is an essential graph on three variables. Any alteration of a single edge (either by adding in one of (A,C), (C,A) or $\langle A,C\rangle$, or un-directing one of the directed edges or changing the direction of an edge) gives a graph that is not an essential graph. It is therefore not possible to move in a single step from the immorality (A,B,C) (where B is the collider node) to a different essential graph on the variables (A,B,C). Filling in the details is left as an exercise (Example 1, Page 351).

The $(MC)^3$ algorithm therefore considers *triples* of nodes and works as follows. Let E_0 be an edge set of an arbitrarily chosen essential graph. To move from E_j to E_{j+1}, do the following:

- Choose three nodes (X_i, X_j, X_k) at random, where $X_i \neq X_j$, $X_j \neq X_k$, $X_i \neq X_k$, taking any possible triple of nodes each with equal probability.

- Let E denote the current edge set. As usual, $E = D \cup U$ where D denotes the directed edges and U denotes the undirected edges. $\langle\alpha,\beta\rangle \in U$ denotes an undirected edge; $(\alpha,\beta) \in D$ denotes a directed edge $\alpha \mapsto \beta$. For F_{ij} and F_{jk} where F_{pq} is defined below, consider the 16 possible graphs generated by keeping all other edges the same and modifying any edges between the two pairs $[X_i, X_j]$ and $[X_j, X_k]$ (where $[\alpha,\beta]$ simply denotes the ordered pair of vertices) according to the four possibilities for each pair:

$$F_{pq} = \begin{cases} 1 & (X_p,X_q) \notin D,\ (X_q,X_p) \notin D,\ \langle X_p,X_q\rangle \notin U \\ 2 & (X_q,X_p) \in D \\ 3 & (X_p,X_q) \in D \\ 4 & \langle X_p,X_q\rangle \in U. \end{cases} \tag{18.2}$$

- Suppose the current state is $E^{(0)}$ and label the 16 possible graphs $E^{(0)}, E^{(1)}, \ldots, E^{(15)}$ generated by all the possibilities of F_{ij} and F_{jk}. For each graph, check whether it is an essential graph, using the criteria of Theorem 5.3.

  That is, it has to be a *chain graph* (for $\alpha \in V_i$ and $\beta \in V_j$ where V_i and V_j are two separate chain components) there is no cycle containing both α and β (that is, a sequence $\rho_0, \ldots, \rho_m, \rho_{m+1} = \rho_0$ with either $(\rho_i, \rho_{i+1}) \in D$ or $\langle\rho_i, \rho_{i+1}\rangle \in U$ for each $i = 0, \ldots, m$). The chain components have to be triangulated and the graph must not contain forbidden substructures (those in Figure 5.1).

- For each possible graph $E^{(l)} : l = 0, \ldots 15$, set

$$y_l = \begin{cases} 0 & E^{(l)} & \text{not essential} \\ \mathbb{S}_{\mathcal{E}|\mathbf{X}}(E^{(l)}|\mathbf{x}) & E^{(l)} & \text{essential} \end{cases}$$

- where $\mathbb{S}_{\mathcal{E}|\mathbf{X}}$ is defined by (18.1), or indeed any other reasonable score function. Select $E(t+1) = E^{(l)}$ with probability y_l, $l = 0, 1, \ldots, 15$.

This gives a process which works through the space of essential graphs, guiding the process (at least locally) to highly scoring structures, while the stochastic element ensures that the process can escape from a local maximum with positive probability.
Since the aim is to examine each graph $E(0), \ldots, E(N)$ visited, together with all those that were checked as candidates when the transition probabilites were computed and then choose the one that maximises $S(E) : E \in \{\text{graphs evaluated}\}$, the following variation may be more efficient.

- Start with an empty graph. Let $E(0)$ denote the empty graph and let $E(t)$ denote the graph selected at step t.

- For each cycle of $\frac{1}{2}d(d-1)(d-2)$ steps, randomly select σ, an ordering of $\{1, \ldots, d\}$, each with probability $\frac{1}{d!}$ and, for $j = 1, \ldots, d$, $i = 1, \ldots, d-1$, $i \neq j$, $k = i+1, \ldots, d$, $k \neq j$, do the following:

1. For the triple of nodes $\{X_{\sigma(i)}, X_{\sigma(j)}, X_{\sigma(k)}\}$, consider all 16 possibilities of $(F_{\sigma(i),\sigma(j)}, F_{\sigma(j),\sigma(k)})$ (defined in Equation (18.2)) when applied to the current essential graph and record those for which the new graph is an essential graph.

2. Let $E^{(0)} = E(t)$ and let $E^{(1)}, \ldots, E^{(15)}$ denote the other 15 possibilities. For $E^{(0)}, \ldots, E^{(15)}$, set $y_k = 0$ if $E^{(k)}$ is not an essential graph, otherwise, set $y_k = R(E^{(k)})$, where R is a suitable score function.

3. Let $E(t+1) = E^{(k)}$ where $y_k = \max_j \{y_j | E^{(j)} \notin \{E(0), \ldots, E(t)\}\}$.

- After the algorithm has run for the required length of time (several cycles of length $\frac{1}{2}d(d-1)(d-2)$), the graph E that gives $\max_{t \in \{0,\ldots,N\}} E(t)$ is selected.

Difficulties with Metropolis Hastings This algorithm has computational advantages. Only three nodes at a time are considered, with the possibility of at most 15 different essential graphs. It provides a *stochastic search* algorithm, where the aim is to find a highly scoring structure. But it seems very difficult to modify it to produce a Metropolis Hastings scheme with a 'theoretically' correct stationary distribution. If $N(E)$ denotes the space of all essential graphs that can be obtained by such a procedure, then the $Q(E, E')$, the probability of proposing E' given a current state E does not have a convenient expression and neither does the acceptance probability $\alpha_{E,E'} = \min\left(1, \frac{\mathbb{S}(E')Q(E',E)}{\mathbb{S}(E)Q(E,E')}\right)$.

18.2 Structure MCMC

The classical MCMC method for learning the underlying structure of a Bayesian network dates back to Madigan and York [89](1995). A prior $\mathbb{P}_{\mathcal{D}}$ is required over the space of directed edge sets $\mathcal{D}$. The score function used is the Cooper-Herskovitz likelihood, $L(D, \mathbf{x})$, given by Equation (12.15). The aim is to construct a Markov chain with posterior distribution

$$\mathbb{P}_{\mathcal{D}|\mathbf{X}}(D|\mathbf{x}) \propto \mathbb{P}_{\mathcal{D}}(D) \times L(D, \mathbf{x}).$$

The Markov chain is generated by the operations of *addition* and *deletion* of single edges. Given directed edge set $D(t)$ at iteration t, let $N(D(t))$ denote all directed edge sets which may be derived from $D(t)$ by one edge added or deleted, together with $D(t)$ itself. A new edge set D' is sampled from the set $N(D(t))$ with proposal probability

$$Q(D(t), D') = \begin{cases} \frac{1}{|N(D(t))|} & D' \in N(D(t)) \\ 0 & \text{otherwise.} \end{cases}$$

The *acceptance* probability is:

$$\alpha_{D(t),D'} = \min\left\{1, \frac{Q(D', D(t))\mathbb{P}(D'|\mathbf{x})}{Q(D(t), D')\mathbb{P}(D(t)|\mathbf{x})}\right\} = \min\left\{1, \frac{|N(D(t))|\mathbb{P}(D'|\mathbf{x})}{|N(D')|\mathbb{P}(D(t)|\mathbf{x})}\right\}.$$

The stationary distribution of this chain is $\mathbb{P}_{\mathcal{D}|\mathbf{X}}(.|\mathbf{x})$.

There are modifications of the basic algorithm: let $N(D(t))$ denote the space of all DAGs obtained by *addition*, *deletion*, or *reversal* of a single edge from the current DAG. This is a straightforward modification; the work of Giudici and Castelo [53](2003) shows that it leads to substantial gains in efficiency.

The samples are generated from randomly chosen starting points and the sequence of DAGs recorded after some suitable *burn-in* period. This should give enough information to decouple the chains from their starting points.

18.3 Edge Reversal Moves

The main problem with structure McMC is slow convergence. The following edge reversal move was introduced by Grzegorczyk and Husmeier [58](2008).

If an edge $X_i \mapsto X_j$ is to be reversed, the two nodes X_i and X_j are first *orphaned*; that is, links from Pa_i to X_i are removed and links from Pa_j to X_j are removed. This involves removing $X_i \mapsto X_j$.

Next, the node $X_j \mapsto X_i$ is inserted. Then the remainder of the new parent set of X_i is established according to a suitable score function and finally a new parent set for X_j is established.

This move is clearly reversible under mild conditions on the way that the new parent sets are established; consider the new graph. Suppose that $X_j \mapsto X_i$ is to be reversed. Firstly, all the edges that have just been added, establishing the new parent sets are removed. Then $X_i \mapsto X_j$ is inserted, then additional parents of X_j and finally parents of X_i are established.

Notation Let D denote a directed edge set. For a node X_i, let $D^{(X_i)\leftarrow\pi}$ denote the graph D where the edges $\text{Pa}_i \mapsto X_i$ are removed and a new parent set π is imposed on X_i. For a graph D, let $\mathbf{1}(D)$ denote the indicator function, returning value 1 if D is a DAG and 0 otherwise. Let

$$Z(X_i|D) = \sum_{\pi:\mathbf{1}(D^{X_i\leftarrow\pi})=1} L_i(\pi|D)$$

where, for a given ordering of the nodes, $L_i(\pi|\mathbf{x})$ denotes a score function for node i having parent set π. Let

$$Z^*(X_i|D, X_j) = \sum_{\pi:\mathbf{1}(D^{X_i\leftarrow\pi})=1, X_j\in\pi} L_i(\pi|D).$$

Choice of New Parent Sets Let D_0 denote the graph D *after* links $\mathrm{Pa}_i \mapsto X_i$ and $\mathrm{Pa}_j \mapsto X_j$ have been removed. The new parent set for X_i, $\widetilde{\pi}_i$, is sampled from the distribution:

$$Q(\widetilde{\pi}_i|D_0, X_j) = \frac{L_i(\widetilde{\pi}_i|\mathbf{x})\mathbf{1}(D_0^{X_i\leftarrow\widetilde{\pi}_i})\mathbf{1}(X_j \in \widetilde{\pi}_i)}{Z^*(X_i|D_0, X_j)}.$$

Having sampled $\widetilde{\pi}_i$, $\widetilde{\pi}_j$ is now sampled from the distribution:

$$Q(\widetilde{\pi}_j|D_0^{X_i\leftarrow\widetilde{\pi}_i}) := \frac{L_j(\widetilde{\pi}_j|\mathbf{x})\mathbf{1}(((D_0)^{X_i\leftarrow\widetilde{\pi}_i})^{X_j\leftarrow\widetilde{\pi}_j})}{Z(X_j|D_0^{X_i\leftarrow\widetilde{\pi}_i})}$$

Conditioned on choosing REV (deciding to make a move of reverse-edge type), the proposal probability for the move $D \mapsto D'$, where D' is obtained by exchanging the parent sets (π_i, π_j) of nodes (X_i, X_j) by $\widetilde{\pi}_i, \widetilde{\pi}_j)$ is:

$$Q(D, D') = \frac{1}{N(D)} Q(\widetilde{\pi}_i|D_0, X_j) Q(\widetilde{\pi}_j|D_0^{X_i\leftarrow\widetilde{\pi}_i})$$

where $N(D)$ is the number of edges in D. The acceptance is:

$$\alpha(D, D') = \min\left(1, \frac{N(D)}{N(D')} \frac{Z^*(X_i|D_0, X_j)}{Z^*(X_j|D_0', X_i)} \frac{Z(X_j|D_0^{X_i\leftarrow\widetilde{\pi}_i})}{Z(X_i|D_0'^{X_j\leftarrow\pi_j})}\right).$$

Adding Reverse Move to the Sampler A value $p_R \in (0, 1)$ is chosen. If the current graph is not empty, then with probability p_R, it is decided is to make a *reverse* move and with probability $p_S = 1 - p_R$ it is decided to make a *standard* move (addition or deletion). Since the standard moves comprise an ergodic Markov chain (albeit not with the desired level of mobility), the mixture is also ergodic.

18.4 Order MCMC

The *order MCMC* algorithm was introduced by Friedman and Koller [47](2003), to establish the ordering of the nodes. The nodes $1, \ldots, d$ according to a given permutation σ. The DAGs that correspond to a given order are simply those where each node may only have parents of a lower order. Once the posterior distribution over orders has been established, the DAG can be constructed relatively easily using other methods (for example, the K2 algorithm).

Recall that the Cooper-Herskovits Likelihood (12.15) has product form, which may be written as:

$$L(D|\mathbf{x}) = \prod_{j=1}^{d} \tilde{L}(j, \pi_j|\mathbf{x})$$

where j denotes node j in D and π_j denotes its parent set. Assume that the prior $\mathbb{P}_{\mathcal{D}}(D)$ also has form:

$$\mathbb{P}_{\mathcal{D}}(D) = \prod_{j=1}^{d} Q(j, \mathrm{Pa}_j)$$

and set

$$S(j, \pi_j|\mathbf{x}) = Q(j, \pi_j)\tilde{L}(j, \mathrm{Pa}_j|\mathbf{x}). \tag{18.3}$$

The *score* $R(\sigma|\mathbf{x})$ for a given ordering σ, given the data $\mathbf{x}$, is given by:

$$R(\sigma|\mathbf{x}) = \sum_{D \epsilon \sigma} \mathbb{P}(D|\mathbf{x}) \propto \prod_{j=1}^{d} \sum_{\mathrm{Pa}_{\sigma(j)} \epsilon \sigma} S(\sigma(j), \mathrm{Pa}_{\sigma(j)}|\mathbf{x}) \tag{18.4}$$

where S is a score function, $D \in \sigma$ denotes a DAG compatible with node ordering σ and $\mathrm{Pa}_{\sigma(j)} \in \sigma$ denotes that the parent set of $\sigma(j)$ is compatible with node ordering σ.

A hard limit K is placed on the size of each parent set. This reduces the complexity of scoring each node to order n^K.

It is much easier to consider moves between node orders. There are a variety of proposals for moves from σ to σ'; for example, choose two at random and flip them. The move $\sigma \mapsto \sigma'$ is *proposed* with probability $Q(\sigma, \sigma')$ the proposal is *accepted* with probability

$$\alpha_{\sigma,\sigma'} = \min\left(1, \frac{Q(\sigma', \sigma)R(\sigma'|\mathbf{x})}{Q(\sigma, \sigma')R(\sigma|\mathbf{x})}\right).$$

Sampling the DAG Having converged to the stationary distribution over orders σ, orderings σ^* are then sampled proportionally to $R(\sigma|\mathbf{x})$. A DAG is sampled for a *fixed* order, in the following way: the parent sets are sampled independently for each variable X_i; for X_i, the scre function . This makes the problem much easier; the parent sets for each variable X_i are sampled independently, according to the score function (18.3).

The Problem with Bias The posterior distribution over orderings is;

$$\mathbb{P}(\sigma|\mathbf{x}) = \sum_{D} \mathbb{P}(\sigma, D|\mathbf{x}) = \sum_{D \epsilon \sigma} \mathbb{P}(\sigma|D)\mathbb{P}(D|\mathbf{x}).$$

Here $\mathbb{P}(D|\mathbf{x})$ is simply the Cooper-Herskovitz likelihood. This differs from the score function (18.4) through the term $\mathbb{P}(\sigma|D)$, which is simply the inverse of the number of orders that the DAG belongs to. On average, the number of orders that each DAG belongs to is exponentially large. (It can range from 1 to $d!$). Neglecting this term in the order MCMC algorithm then wieghts DAGs by the number of orders they belong to.

18.5 Partition MCMC for Directed Acyclic Graphs

Partition MCMC was introduced recently by Kuipers and Moffa [76](2015). With *partition* MCMC, the moves are *not* between DAGs and the aim of the algorithm is not to end up with a distribution over DAGs; rather, it is to end up with a distribution over *layerings* of DAGs (defined below).

Layering of a DAG The nodes of a DAG may be *layered*. A *layering* is a partition satisfying the condition that no node in the same layer is either an ancestor or descendant of any other node in layer k. The layers are indexed by $\mathbb{N} = \{1, 2, 3, \ldots\}$. It is known as a *minimal* layering if each node has the minimal index value such that the partition is a layering.

The minimal layering clearly satisfies (for example) that all ancestor nodes are in layer 1. Furthermore, all nodes in layer k have at least one parent in layer $k-1$.

Consider a minimal layering with m levels and let $(k_1, \ldots, k_m)$ denote the number of nodes in each layer. The number of DAGs belonging to such a partition is given by:

$$a_{k_1,\ldots,k_m} = \frac{d!}{k_1! \ldots k_m!} \prod_{j=2}^{m} \left(2^{k_{j-1}} - 1\right)^{k_j} \prod_{j=3}^{m} 2^{k_j S_{j-2}}.$$

where $S_j = \sum_{i=1}^{j} k_i$. The first term is simply the number of ways of distributing d nodes in m partition elements of size $k_1, \ldots, k_m$ respectively. The second is the number of ways that nodes in each partition can have parents in the previous partition. Subtracting 1 excludes the case where nodes receive no edges. The third term is the number of ways that nodes can have parents from partitions other than the one directly below.

18.5.1 Scoring Partitions

A score $S(P)$ is assigned to each partition P. This is done as follows: let $\lambda = (k_1, \ldots, k_m)$ denote a partition. This gives the shape of a layering; $\lambda = (k_1, \ldots, k_m)$ where $k_1 + \ldots + k_m = d$ specifies that there are k_1 nodes in the first layer, k_2 in the second, k_j in the jth for $j = 1, \ldots, m$ and there are m layers. Furthermore, $k_j \geq 1$ for each $j \in \{1, \ldots, m\}$.

Let σ denote a *permuation* of the nodes. This specifies which nodes are in which layer. If $\lambda = (k_1, \ldots, k_m)$, then $X_{\sigma(1)}, \ldots, X_{\sigma(k_1)}$ belong to layer 1; nodes $X_{\sigma(k_1+\ldots+k_j+1)}, \ldots, X_{\sigma(k_1+\ldots+k_j+k_{j+1})}$ are in layer $j+1$ for $j = 1, \ldots, m-1$.

Permuting nodes within a layer does not change anything. Let $\pi_{\lambda,\sigma}$ denote a representative permutation; that is, λ together with permutation $\pi_{\lambda,\sigma}$ gives the same layering as λ together with σ. Let $\Lambda = (\lambda, \pi_{\lambda,\sigma})$. The score for Λ is:

$$S(\Lambda|\mathbf{x}) = \sum_{D} \mathbb{P}(\Lambda|D, \mathbf{x}) \mathbb{P}(D|\mathbf{x}) = \sum_{D \in \Lambda} \mathbb{P}(D|\mathbf{x}) \propto \prod_{j=1}^{d} \sum_{\mathrm{Pa}_j \in \Lambda} S(X_j, \mathrm{Pa}_j|\mathbf{x}).$$

where $D \in \Lambda$ denotes that the DAG D is compatible with the layering specified by Λ and $\mathrm{Pa}_j \in \Lambda$ denotes that the parent set of variable j is compatible with Λ.

The MCMC will *propose* a move $\Lambda \mapsto \Lambda'$, by defining a set $N(\Lambda)$ of neighbours and choosing each with equal probability. The *acceptance* is:

$$\alpha_{\Lambda,\Lambda'} = \min\left(1, \frac{|N(\Lambda)|}{|N(\Lambda')|}\frac{S(\Lambda'|\mathbf{x})}{S(\Lambda|\mathbf{x})}\right) \tag{18.5}$$

18.5.2 Partition Moves

The basic partition move involves

- Splitting a layer in two;
- Merging two adjacent layers.

When splitting a layer of size k into two parts, one of size c and one of size $k-c$, there are $\binom{k}{c}$ ways to do it. There are $m-1$ ways to merge two partitions. The size of the neighbourhood is therefore:

$$(m-1) + \sum_{i=1}^{m}\sum_{c=1}^{k_i-1}\binom{k_i}{c} = m-1+\sum_{i=1}^{m}\left(2^{k_i}-2\right) = \left(\sum_{i=1}^{m}2^{k_i}\right) - m - 1.$$

When merging layer i with layer $i+1$, the score changes simply with the indicator function of whether the parent sets are legal under the new layering. The alterations are only in those in the layer labelled $i+2$ before the merge; the number of possible parent sets has increased - and (of course) those in layer $i+1$ before the merge. Instead of being forced to have at least one parent from layer i before the merge, links with these variables are excluded; now the variables from $i+1$ (before merge) are forced to have a parent in layer $i-1$.

Splitting and merging thus defined give reversible moves, so that the acceptance defined by (18.5) is positive.

It is straightforward to see that the chain is irreducible; from one partition any other partition can be reached in a finite number of moves which have positive probability. If necessary, the chain can stay still with positive probability to ensure aperodicity.

18.5.3 Permutation Moves

The *permutation moves* are simpler. Two strategies can be adopted

- Choose two nodes at random, with the constraint that they are in different layers, and swap them. There are
$$\sum_{i=1}^{m}\frac{k_i(n-k_i)}{2}$$
possibilities, each chosen with equal probability. The move is accepted with probability $\frac{M(\Lambda)S(\Lambda'|\mathbf{x})}{M(\Lambda')S(\Lambda|\mathbf{x})}$.
- Choose two nodes at random, with the constraint that they are in adjacent layers. There are
$$M(\Lambda) = \sum_{i=1}^{m-1} k_i k_{i+1}$$
possible choices of pairs. They are chosen each with equal probability and the move is accepted with probability $\frac{M(\Lambda)S(\Lambda'|\mathbf{x})}{M(\Lambda')S(\Lambda|\mathbf{x})}$.

18.5.4 Combination with Edge Reversal

The Edge Reversal Move discussed in Section 18.3 may be combined with these. In this context, firstly a DAG is chosen compatible with Λ, with probability proportional to its score. Then the reverse move is proposed and accepted with the probabilities given in Section 18.3. Then the corresponding partition / permutation Λ' is computed. The Edge-Reversal move is not ergodic, but if probabilities $p_R > 0$, $p_\lambda > 0$, $p_\sigma > 0$ for the probabilities of taking an Edge Reversal, Partition and Permutation move respectively are specified in advance, where $p_R + p_\lambda + p_\sigma = 1$, the process is ergodic, with the correct stationary distribution.

Explicitly, let $\mathbb{P}_{D'|D}(\Lambda'|\Lambda)$ denote the probability of a transition to Λ' through an edge reversal move D to D'. Let $\mathbb{Q}(D'|D)$ denote the transition probability of a move from D to D' given that the move is edge reversal. Then

$$\mathbb{P}_{D'|D}(\Lambda'|\Lambda) = \frac{\mathbb{P}(D|\mathbf{x})}{\mathbb{P}(\Lambda|\mathbf{x})}\mathbb{Q}(D'|D).$$

This move satisfies the detailed balance equation;

$$\frac{\mathbb{P}(D'|\mathbf{x})}{\mathbb{P}(D|\mathbf{x})} = \frac{\mathbb{Q}(D'|D)}{\mathbb{Q}(D|D')}$$

from which

$$\frac{\mathbb{P}_{D'|D}(\Lambda'|\Lambda)}{\mathbb{P}_{D|D'}(\Lambda|\Lambda')} = \frac{\mathbb{P}(\Lambda'|\mathbf{x})}{\mathbb{P}(\Lambda|\mathbf{x})}. \tag{18.6}$$

Finally, there may be more than one path between layerings;

$$\mathbb{P}(\Lambda'|\Lambda) = \sum_{D,D'} \mathbb{P}_{D'|D}(\Lambda'|\Lambda)$$

is the total transition. Now, from Equation (18.6), it follows that:

$$\mathbb{P}(\Lambda|\mathbf{x})\mathbb{P}(\Lambda'|\Lambda) = \mathbb{P}(\Lambda'|\mathbf{x})\mathbb{P}(\Lambda|\Lambda').$$

Chapter 19

Dynamic Bayesian Networks

19.1 Introduction

Dynamic Bayesian networks (DBNs) are an important tool that have proved useful for a large class of problems. The thesis of Kevin Murphy (2002) [97] provides a comprehensive introduction to the topic.

The first mention of dynamic Bayesian networks seems to be by Dean and Kanazawa (1989) [34]. The DBN framework provides a way to extend Bayesian network machinery to model probability distributions over collections of random variables $(\underline{Z}_t)_{t\geq 0}$. The parameter $t \in \{0, 1, 2, \ldots\}$ represents time. Typically, the variables at a time slice t are partitioned into $\underline{Z}_t = (\underline{U}_t, \underline{X}_t, \underline{Y}_t)$ representing the input, hidden and output variables of the model. The term 'dynamic' refers to the fact that the system is dynamic; the basic structure remains the same over time.

Definition 19.1. *A* k - slice Dynamic Bayesian network *is a DAG corresponding to a factorisation of the probability distribution over the variables* $\{\underline{Z}_0, \underline{Z}_1, \ldots\}$ *such that for* $t \geq k$,

$$\mathbb{P}_{Z_0,\ldots,Z_t} = \mathbb{P}_{Z_0} \prod_{s=1}^{k-1} \mathbb{P}_{Z_s|Z_0,\ldots,Z_{s-1}} \prod_{s=k}^{t} \mathbb{P}_{Z_s|Z_{s-k},\ldots,Z_{s-1}}$$

where, for $t \geq k$,

$$\mathbb{P}_{Z_t|Z_{t-k-1},\ldots,Z_{t-1}} = \prod_j \mathbb{P}_{Z_t^j|Pa(Z_t^j)},$$

Z_t^j *is the* j*th node at time* t*, which could be a component of either* X_t*,* Y_t *or* U_t *and the set* $Pa(Z_t^j)$ *of parents of* Z_t^j *belongs to the collection*

$$\underline{Z}_{t-k}, \ldots, \underline{Z}_{t-1}, \{Z_t^1, \ldots, Z_t^{j-1}\}.$$

The arrows within the same time slice do not represent causality.

The requirement is that the subgraph restricted to $\{\underline{Z}_t, \ldots, \underline{Z}_{t+k-1}\}$ *is the same for each* $t \geq 0$ *and the conditional probabilities* $\mathbb{P}_{Z_t^j|Pa(Z_t^j)}$ *are the same for each* $t \geq k$*. Furthermore, for* $1 \leq i \leq j \leq k$*, and each* $s \geq j$*, the subgraph restricted to* $\{\underline{Z}_{s+i}, \ldots, \underline{Z}_{s+j}\}$ *is a subgraph of the subgraph restricted to* $\{\underline{Z}_{s+i-1}, \ldots, \underline{Z}_{s+j}\}$.

The arcs between slices are from left to right and reflecting the causal flow of time. If there is an arc from Z_{t-1}^j to Z_t^j, the node Z^j is said to be *persistent*. The arcs *within* a slice may have arbitrary direction, so long as the overall DBN is a DAG. The arcs within a time slice may be undirected, since they model correlation or constraints rather than causation. The resulting model is then a (dynamic) chain graph.

The parameters of the conditional probabilities $\mathbb{P}_{Z_t^j|\mathrm{Pa}(Z_t^j)}$ are time-invariant for $t \geq k$, i.e., the model is time-homogeneous. If parameters can change, they may be added to the state-space and treated as random variables or alternatively a hidden variable may be added that selects which set of parameters to use.

Within the engineering community, DBNs have become a popular tool, because they can express a large number of models and are often computationally tractable.

DBNs have been successfully applied to in the reconstruction of genetic networks, where genes do not remain static, but rather their expression levels fluctuate constantly. Increased expression level of a gene will result in increased levels of mRNA from that gene which will in turn influence the expression levels of other genes. DBNs have proved to be a successful way of analysing genetic expression data.

With a *Dynamic Bayesian Network*, the $n \times d$ data matrix no longer represents n *independent* instantiations of a random d-vector. Rather, the rows represent time slices of a *process* $\{\underline{X}(t) : t \in \mathbb{N}\}$.

Some assumptions (for example time homogeneity) have to be made in order to learn structure and parameters.

If the number of instantiations n available is large in comparison to d, then standard multivariate time series techniques may be used effectively. If n is small compared with d, other techniques (such as LASSO L^1 regularisation) should be used.

19.2 Multivariate Time Series

A VARMA(p,q) model (vector auto regressive moving average, lags p and q for the auto-regressive and moving average parts respectively) is a model:

$$\underline{X}(t) = \underline{\mu}_0 + t\underline{\mu}_1 + \sum_{j=1}^{p} A_j \underline{X}(t-j) + \sum_{k=1}^{q} B_k \underline{\epsilon}_{t+1-q}$$

where $\underline{\epsilon}_t \sim N(\underline{0}, \Sigma)$ are i.i.d. (the distribution is not necessarily normal, but the normality assumption, if true, leads to sharper estimation).

The MA part often leads to instability for estimation; we therefore only consider VAR(p) processes;

$$\underline{X}(t) = \underline{\mu}_0 + t\underline{\mu}_1 + \sum_{j=1}^{p} A_j \underline{X}(t-j) + \underline{\epsilon}_t$$

The package **vars** fits a vector auto regressive model:

```
> install.packages("vars")
> library(vars)
```

Within **vars**, there is a test data-set `Canada`, which contains 4 macroeconomic indicators; `prod` (labour productivity), `e` (employment), `U` (unemployment rate) and `rw` (real wages). A VAR(2) model is fitted quite simply with the command:

```
> data(Canada)
> can = VAR(Canada,p=2)
> summary(can)

VAR Estimation Results:
=========================
Endogenous variables: e, prod, rw, U
Deterministic variables: const
Sample size: 82
Log Likelihood: -175.819
Roots of the characteristic polynomial:
0.995 0.9081 0.9081 0.7381 0.7381 0.1856 0.1429 0.1429
Call:
VAR(y = Canada, p = 2)


Estimation results for equation e:
==================================
e = e.l1 + prod.l1 + rw.l1 + U.l1 + e.l2 + prod.l2 + rw.l2 + U.l2 + const

          Estimate Std. Error t value Pr(>|t|)
e.l1     1.638e+00  1.500e-01  10.918  < 2e-16 ***
prod.l1  1.673e-01  6.114e-02   2.736  0.00780 **
rw.l1   -6.312e-02  5.524e-02  -1.143  0.25692
U.l1     2.656e-01  2.028e-01   1.310  0.19444
e.l2    -4.971e-01  1.595e-01  -3.116  0.00262 **
prod.l2 -1.017e-01  6.607e-02  -1.539  0.12824
rw.l2    3.844e-03  5.552e-02   0.069  0.94499
U.l2     1.327e-01  2.073e-01   0.640  0.52418
const   -1.370e+02  5.585e+01  -2.453  0.01655 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 0.3628 on 73 degrees of freedom
Multiple R-Squared: 0.9985,Adjusted R-squared: 0.9984
F-statistic:  6189 on 8 and 73 DF,  p-value: < 2.2e-16
```

```
Estimation results for equation prod:
=====================================
prod = e.l1 + prod.l1 + rw.l1 + U.l1 + e.l2 + prod.l2 + rw.l2 + U.l2 + const

          Estimate Std. Error t value Pr(>|t|)
e.l1      -0.17277    0.26977  -0.640  0.52390
prod.l1    1.15043    0.10995  10.464 3.57e-16 ***
rw.l1      0.05130    0.09934   0.516  0.60710
U.l1      -0.47850    0.36470  -1.312  0.19362
e.l2       0.38526    0.28688   1.343  0.18346
prod.l2   -0.17241    0.11881  -1.451  0.15104
rw.l2     -0.11885    0.09985  -1.190  0.23778
U.l2       1.01592    0.37285   2.725  0.00805 **
const   -166.77552  100.43388  -1.661  0.10109
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 0.6525 on 73 degrees of freedom
Multiple R-Squared: 0.9787,Adjusted R-squared: 0.9764
F-statistic: 419.3 on 8 and 73 DF,  p-value: < 2.2e-16


Estimation results for equation rw:
===================================
rw = e.l1 + prod.l1 + rw.l1 + U.l1 + e.l2 + prod.l2 + rw.l2 + U.l2 + const

          Estimate Std. Error t value Pr(>|t|)
e.l1     -0.268833   0.322619  -0.833    0.407
prod.l1  -0.081065   0.131487  -0.617    0.539
rw.l1     0.895478   0.118800   7.538 1.04e-10 ***
U.l1      0.012130   0.436149   0.028    0.978
e.l2      0.367849   0.343087   1.072    0.287
prod.l2  -0.005181   0.142093  -0.036    0.971
rw.l2     0.052677   0.119410   0.441    0.660
U.l2     -0.127708   0.445892  -0.286    0.775
const   -33.188339 120.110525  -0.276    0.783
---
```

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 0.7803 on 73 degrees of freedom
Multiple R-Squared: 0.9989,Adjusted R-squared: 0.9987
F-statistic:  8009 on 8 and 73 DF,  p-value: < 2.2e-16


Estimation results for equation U:
==================================
U = e.l1 + prod.l1 + rw.l1 + U.l1 + e.l2 + prod.l2 + rw.l2 + U.l2 + const

         Estimate Std. Error t value Pr(>|t|)
e.l1     -0.58076    0.11563  -5.023 3.49e-06 ***
prod.l1  -0.07812    0.04713  -1.658 0.101682
rw.l1     0.01866    0.04258   0.438 0.662463
U.l1      0.61893    0.15632   3.959 0.000173 ***
e.l2      0.40982    0.12296   3.333 0.001352 **
prod.l2   0.05212    0.05093   1.023 0.309513
rw.l2     0.04180    0.04280   0.977 0.331928
U.l2     -0.07117    0.15981  -0.445 0.657395
const   149.78056   43.04810   3.479 0.000851 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 0.2797 on 73 degrees of freedom
Multiple R-Squared: 0.9726,Adjusted R-squared: 0.9696
F-statistic:   324 on 8 and 73 DF,  p-value: < 2.2e-16



Covariance matrix of residuals:
              e      prod       rw        U
e     0.131635 -0.007469 -0.04210 -0.06909
prod -0.007469  0.425711  0.06461  0.01392
rw   -0.042099  0.064613  0.60886  0.03422
U    -0.069087  0.013923  0.03422  0.07821

Correlation matrix of residuals:
```

```
             e     prod      rw       U
e     1.00000 -0.03155 -0.1487 -0.6809
prod -0.03155  1.00000  0.1269  0.0763
rw   -0.14870  0.12691  1.0000  0.1568
U    -0.68090  0.07630  0.1568  1.0000
```

The default value, which estimates $\underline{\mu}_0$ and sets $\underline{\mu}_1 = 0$ is `const`. To set $\underline{\mu}_0 = 0$ and $\underline{\mu}_1 = 0$, type:

```
> VAR(Canada,p=2,type="none")
```

To set $\underline{\mu}_0 = 0$ while estimating an unknown trend $\underline{\mu}_1$, type:

```
> VAR(Canada,p=2,type="trend")
```

To estimate both an intercept $\underline{\mu}_0$ and a trend $\underline{\mu}_1$, type:

```
> VAR(Canada,p=2,type="both")
```

The `stability` function verifies the covariance stationarity of a VAR process, using cumulative sums of residuals. This may be carried out by:

```
> var.2c=VAR(Canada,p=2,type="const")
> stab=stability(var.2c,type="OLS-CUSUM")
> plot(stab)
```

There are several tests for normality which come under `normality.test`.

```
> normality.test(var.2c)
$JB

JB-Test (multivariate)

data:  Residuals of VAR object var.2c
Chi-squared = 5.094, df = 8, p-value = 0.7475


$Skewness

Skewness only (multivariate)

data:  Residuals of VAR object var.2c
Chi-squared = 1.7761, df = 4, p-value = 0.7769
```

```
$Kurtosis

Kurtosis only (multivariate)

data:  Residuals of VAR object var.2c
Chi-squared = 3.3179, df = 4, p-value = 0.5061
```

The function `serial.test` carries out the Portmanteau (i.e. Ljung-Box) test

```
> serial.test(var.2c,lags.pt=16,type="PT.adjusted")

Portmanteau Test (adjusted)

data:  Residuals of VAR object var.2c
Chi-squared = 231.5907, df = 224, p-value = 0.3497
```

The VARMA model is standard and is treated in any reasonable text on Time Series, for example [**?**].

19.3 Lasso Learning

One of the most prominent applications of DBNs is to gene expression data and locating regulatory pathways. The main difficulty is that n (the number of instantiations) tends to be small compared with d (the number of genes under investigation). On the other hand, gene expression networks tend to be sparse.

One technique that has developed and is quite effective in such situations is L^1 regularisation, or LASSO learning.

LASSO and Least Angle Regression Given a set of input measurements $(x_{j,1}, \ldots, x_{j,d})$ for $j = 1, \ldots, n$ and outcome measurement $y_j : j = 1, \ldots, n$, taken as observations on independent variables, the lasso fits a linear model

$$\widehat{y_j} = \widehat{\beta}_0 + \sum_{j=1}^{d} x_j \widehat{\beta}_j.$$

The criterion it uses is:

Minimise $\sum_{j=1}^{n} (y_j \widehat{y_j})^2$ subject to $\sum_{j=0}^{d} |\beta_j| \leq s$ for a constraint value s.

The bound s is a tuning parameter. When s is sufficiently large, the constraint has no effect and the solution is simply the usual multiple linear least squares regression of y on $x_1, \ldots, x_d$.

For smaller values of s ($s \geq 0$), the solutions are *shrunken* versions of the least squares estimates. The L^1 penalisation often forces some of the coefficient estimates $\widehat{\beta}_j$ to be zero.

The choice of s therefore plays a similar role to choosing the number of predictors in a regression model.

Cross-validation is the standard tool for estimating the best value for s.

Forward stepwise regression achieves the same objective as regularisation by adding in explanatory variables one at a time:

- Start with all coefficients β_j equal to zero.
- Find the predictor x_j which is most correlated to y and add it into the model. Take residuals $r = y - \widehat{y}$.
- Continue, at each stage adding to the model the predictor most correlated with r.
- Until: all predictors are in the model

The *Least Angle Regression* procedure follows the same general scheme, but does not add a predictor *fully* into the model. The coefficient of that predictor is increased only until that predictor is no longer the one most correlated with the residual r. Then some other competing predictor is included.

Least Angle Regression algorithm The algorithm proceeds as follows:

- Start with all coefficients β_j equal to zero.
- Find the predictor x_j most correlated with y.
- Increase the coefficient β_j in the direction of the sign of its correlation with y. Take residuals $r = y - \widehat{y}$. Stop when some other predictor x_k has as much correlation with r as x_j has.
- Increase (β_j, β_k) in their joint least squares direction, until some other predictor x_m has as much correlation with the residual r.
- Continue until: all predictors are in the model

It can be shown that, with one modification, this procedure gives the entire path of lasso solutions, as s is varied from 0 to infinity. The modification needed is: if a non-zero coefficient hits zero, remove it from the active set of predictors and recompute the joint direction.

Cross-Validation Cross validation is a model evaluation method where some of the data is removed before training begins. Then when training is done, the data that was removed can be used to test the performance of the learned model on "new" data. This is the basic idea for the class of model evaluation methods called cross validation.

- **Holdout** The holdout method is the simplest kind of cross validation. The data set is separated into two sets; the *training* set and the *testing* set. The function approximator fits a function using

the training set only. Then the function approximator is asked to predict the output values for the data in the testing set (it has never seen these output values before). The errors it makes are accumulated as before to give the mean absolute test set error, which is used to evaluate the model.

- **K-fold Cross Validation** K-fold cross validation is one way to improve over the holdout method. The data set is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set k-1 times. The variance of the resulting estimate is reduced as k is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation. A variant of this method is to randomly divide the data into a test and training set k different times. The advantage of doing this is that you can independently choose how large each test set is and how many trials you average over.

- **Leave-one-out** Leave-one-out cross validation is K-fold cross validation taken to its logical extreme, with K equal to n, the number of data points in the set. That means that the function approximator is trained on all the data except for one point n separate times and a prediction is made for that point. As before the average error is computed and used to evaluate the model. The evaluation given by leave-one-out cross validation error (LOO-XVE) is good, but at first pass it seems very expensive to compute.

19.3.1 Implementation

There are several packages available in R for DBN learning. One of the most prominent is the **lars** package by Hastie and Efron [60] (2012). Other packages available are: **glmnet** package by Friedman et. al. [44] (2010) and **penalized** by Goeman [54] (2012). For illustration, we use the `arth800MTS` data set from the **GeneNet** package. This describes the expression levels of 800 genes of the Arabidopsis thaliana during the diurnal cycle. We consider a subset `arth12` of 12 of the genes.

```
> library(lars)
> library(GeneNet)
> data(arth800)
> subset=c(60,141,260,333,365,424,441,512,521,578,799)
> arth12=arth800.expr[,subset]
```

Now **lars** is used to estimate a model for a target variable speified by a vector (say `y`) and a set of possible parents specified by a matrix of predictors (say `x`). The `arth800` data set consists of two time series, each of 11 points in length. That is, there are two repeated measurements for each time point. To estimate a VAR(1) process, firstly remove the two repeated measurements for the first time point

of `y` and the two repeated measurements for the last time point of `x`. They cannot be used for LASSO, since $y(t)$ needs $x(t-1)$.

```
> x = arth12[1:(nrow(arth12)-2),]
> y = arth12[-(1:2),"265768_at"]
> lasso.fit = lars(y=y,x=x,type="lasso")
> plot(lasso.fit)
```

The plot is shown in Figure 19.1.

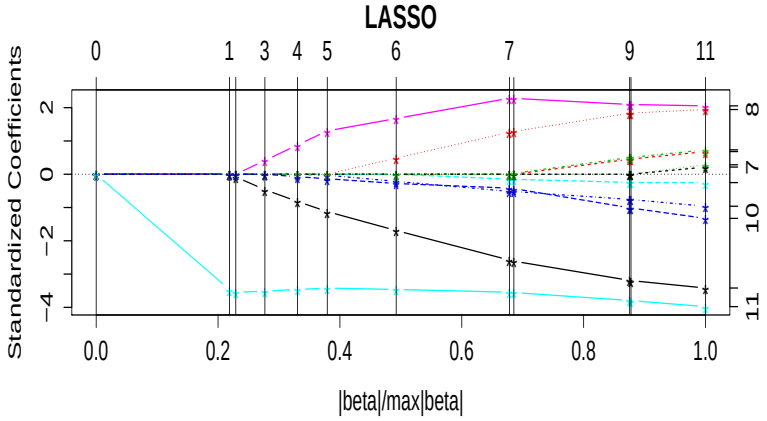

Figure 19.1: Lasso output

The figure is interpreted as follows: the aim is to predict $y(t)$ (the expression levels for gene labelled `265768_at`) by the expression levels one time unit earlier (given at time index $t-2$ because we have double measurements for each time); $x(t-2)$. The regression is carried out by evaluating the coefficients $\underline{\beta}$ which minimise $\sum_{t=3}^{22}(y(t)-\sum_{j=1}^{11} x_j(t-2)\beta_j)^2$, subject to a constraint that $\sum_{j=1}^{11}|\beta_j| \leq t$ for t increasing. For the x-axis, this is presented as $|\beta|/\max|\beta|$, where $|\beta| = \sum_{j=1}^{11}|\beta_j|$ and $\max|\beta|$ is the value of $\sum_{j=1}^{11}|\beta_j|$ for the unconstrained problem.

The values of the coefficients are denoted by different colours and the plot shows how they change as the value of t increases. The vertical lines indicate the points at which new coefficients are introduced. The coefficients may be obtained by

```
> coef(lasso.fit)
```

Structure learning (i.e. deciding which directed edges to include in the network) is carried out via *cross-validation*. The `cv.lars` function does this.

```
> lasso.cv=cv.lars(y=y,x=x,mode="fraction")
```

The *output* gives the MSE (mean squared error) as a function of $|\beta|/\max|\beta|$ (where $|\beta|$ denotes the constraint and $\max|\beta|$ denotes the value of $\sum_{j=1}^{11}|\beta_j|$ for the unconstrained problem) and the output is shown in Figure 19.2. The optimal set of arcs is chosen to minimise the mean squared error.

```
> frac=lasso.cv$index[which.min(lasso.cv$cv)]
> predict(lasso.fit,s=frac,type="coef",mode="fraction")
$s
[1] 0.1919192

$fraction
[1] 0.1919192

$mode
[1] "fraction"

$coefficients
 265768_at  263426_at  260676_at  258736_at  257710_at  255764_at
 0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 255070_at  253425_at  253174_at  251324_at  245319_at  245094_at
 0.0000000  0.0000000  0.0000000  0.0000000  0.0000000 -0.6420806
```

The non-zero coefficients indicate the arcs to be included on the gene `265768_at` for the optimal value `s=frac` computed by `cv.lars`.

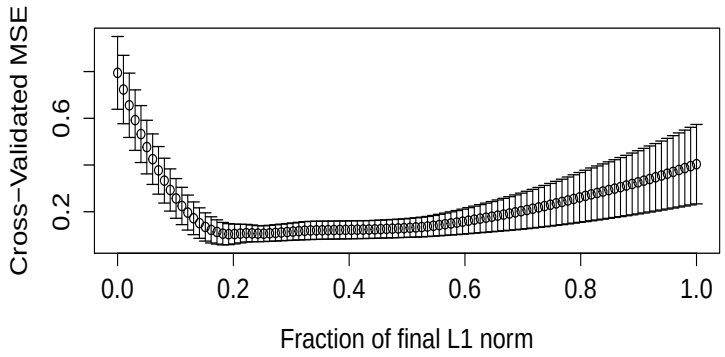

Figure 19.2: Lasso cross validation

The number of steps can be controlled by setting the `mode` argument of `predict` to `step`.

```
> predict(lasso.fit,s=3,type="coef",mode="step")$coefficients
  265768_at   263426_at   260676_at   258736_at   257710_at   255764_at   255070_at   253425_at
-0.02152962  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000  0.00000000
  253174_at   251324_at   245319_at   245094_at
 0.00000000  0.00000000  0.00000000 -0.72966658
```

The L^1 penalty can be specified with `mode = ''lambda''`

```
> predict(lasso.fit,s=0.2,type="coef",mode="lambda")$coefficients
 265768_at  263426_at  260676_at  258736_at  257710_at  255764_at  255070_at  253425_at
 0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
 253174_at  251324_at  245319_at  245094_at
 0.0000000  0.0000000  0.0000000 -0.6961228
```

The **lars** package also fits *least angle regression* and *stepwise regression.*

```
> lar.fit=lars(y=y,x=x,type="lar")
> lar.cv=cv.lars(y=y,x=x,type="lar")
> step.fit=lars(y=y,x=x,type="stepwise")
> step.cv=cv.lars(y=y,x=x,type="stepwise")
```

19.4 simone: Statistical Inference for MOdular NEtworks

The **simone** package by Chiquet et. al. [27](2009) implements LASSO specifically for dynamic Bayesian networks. Install the package, activate it and get information using

```
> install.packages("simone")
> library(simone)
> ?simone
```

It works on the principle that the $n \times d$ data matrix contains n sequential observations of the d variables and it fits a VAR(1) model. The default is `clustering = FALSE`.

```
> result = simone(arth12,type="time-course")
```

The output is the number of edges in the network depending on the penalisation (default: BIC). A sequencing display of the network as the penalty is reduced is obtained by:

```
> plot.simone(result)
```

The analysis can be carried out *with* clustering; edges are penalised if latent clustering is discovered while constructing the network.

```
> resultcluster=simone(arth12,type="time-course",clustering=TRUE,control=ctrl)
```

The sequencing display of the network indicates that clustering has not changed the output much.

19.5 GeneNet, GIDBN

```
> install.packages("G1DBN")
> library(G1DBN)
> data(arth800line)
> data(arth800line)
> subset=c(60,141,260,333,365,424,441,512,521,578,789,799)
> arth12=as.matrix(arth800line[,subset])
```

Learning is carried out in two stages: firstly, learning the graph encoding the first order partial dependencies with `DBNScoreStep1`.

```
> step1=DBNScoreStep1(arth12,method="ls")
> edgesG1=BuildEdges(score=step1$S1ls,threshold=0.50,prec=6)
> nrow(edgesG1)
[1] 27
```

The help commands describe the second step.

```
> step2=DBNScoreStep2(step1$S1ls,data=arth12,method="ls",alpha1=0.50)
> edgesG=BuildEdges(score=step2,threshold=0.05,prec=6)
```

19.6 Inference for Dynamic Bayesian Networks

For a given DBN (where the network structure and the conditional probability potentials have been specified), the queries of interest are usually those of computing the marginal distribution of $X_i(t)$ conditioned on all nodes other than $X_i(t)$ at times $1, \ldots, T$. In line with standard time series problems, these problems fall into three categories:

- If $T = t$, the query is called *filtering*.
- If $T > t$ (node $X_i(t)$ is omitted), the query is called *smoothing*. It returns a smoothed value of $\widehat{X}_i(t)$; the aim of the query is noise reduction.
- If $T < t$, the query is called *prediction*.

Queries which ask for the Most Probable Explanation can be performed for filtering, smoothing and prediction with the **lars** package.

To see how it works, consider the `arth12` data set:

```
> library(GeneNet)
> data(arth800)
> subset = c(60, 141, 260, 333, 365, 424, 441, 512,
+ 521, 578, 789, 799)
> arth12 = arth800.expr[, subset]
> library(lars)
> x = arth12[1:(nrow(arth12) - 2), ]
> y = arth12[-(1:2), "265768_at"]
```

`y` contains the expression levels of gene `265768_at` at all times except for time 0 (recall that there are two measurements at each time). `x` contains the whole data set for all times except for the last one, labelled 24.

```
> lasso.fit = lars(y = y, x = x, type = "lasso")
> lasso.cv = cv.lars(y = y, x = x, mode = "fraction")
> frac = lasso.cv$index[which.min(lasso.cv$cv)]
```

`frac` contains the value of the index that minimises the cross variation. Therefore, this is the value that is used to build the model. Estimation for the expression levels of `265768_at` may be carried out quite simply by:

```
> lasso.est = predict(lasso.fit, type = "fit",
+ newx = x, s = frac,
+ mode = "fraction")$fit
> lasso.est
     0-1      0-2      1-1      1-2      2-1      2-2      4-1
```

```
7.099782 6.894064 7.166249 7.157744 7.592092 7.379432 7.990548
     4-2      8-1      8-2     12-1     12-2     13-1     13-2
8.078921 8.353137 8.333108 8.940241 8.780302 8.816387 8.758480
    14-1     14-2     16-1     16-2     20-1     20-2
8.542374 8.417818 7.446577 7.329513 6.717392 6.747178
```

The estimated expression levels at `20-1` and `20-2` are a result of *filtering*, while the others given here are a result of *smoothing*.

The values of 24-1 and 24-2 can be predicted by:

```
> lasso.pred = predict(lasso.fit, type = "fit",
+ newx = arth12[c("24-1", "24-2"), ],
+ s = frac, mode = "fraction")$fit
> lasso.pred
    24-1     24-2
6.822643 6.882054
```

The **penalized** package fits LASSO models which are compatible with **bnlearn**. Therefore, more complex conditional probability queries can be carried out using `cpquery` and `cpdist` if the model is first learned in this way.

```
> library(penalized)
> lambda = optL1(response = y, penalized = x)$lambda
> lasso.t = penalized(response = y, penalized = x,
+ lambda1 = lambda)
# nonzero coefficients: 2
> coef(lasso.t)
(Intercept)   245094_at
 14.0402894  -0.7059011
```

The only parent of gene `256768_at` is `245094_at`, which seems to act as an inhibitor.

This suggests that a model with this explanatory variable might be useful. Such a DBN can be created in the following way:

```
>dbn1 =
+  model2network("[245094_at][265768_at|245094_at]")
>xp.mean = mean(x[, "245094_at"])
>xp.sd = sd(x[, "245094_at"])
>dbn1.fit =
+  custom.fit(dbn1,
+   dist = list("245094_at" = list(coef = xp.mean,
+             sd = xp.sd), "265768_at" = lasso.t))
```

Since the data is continuous, there are two possibilities: either create a Gaussian network, or discretise the variables. The network `dbn1` is Gaussian. The mean `xp.mean` and *standard deviation* `xp.sd` need to be specified.

The regression analysis suggests that high expression levels of `245094_at` at time $t-1$ lead to low expression levels of `265768_at` at time t. The `cpquery` function can be used:

```
>cpquery(dbn1.fit, event = (‘265768_at‘ > 8),
+                  evidence = (‘245094_at‘ > 8))
[1] 0.2454624
>cpquery(dbn1.fit, event = (‘265768_at‘ > 8),
+                  evidence = (‘245094_at‘ < 8))
[1] 0.9829545
```

Note With this package, it is not permitted to condition on events of measure 0. Therefore, *intervals* must be specified *both* for `event` and `evidence`.

The function `cpdist` may be used to generate random observations. To compare the conditional distributions for both pieces of evidence, use:

```
>dist.low = cpdist(dbn1.fit, node = "265768_at",
+             evidence = (‘245094_at‘ < 8))
>dist.high = cpdist(dbn1.fit, node = "265768_at",
+              evidence = (‘245094_at‘ > 8))
```

These may be plotted and the densities compared.

Now suppose that the variables at time t are not independent of those at $t-2$ given $t-1$. It is then a good idea to construct a DBN which depends on lags 1 and 2. To check whether the introduction of $t-2$ to explain t improves the model:

```
> y = arth12[-(1:2), "245094_at"]
> colnames(x)[12] = "245094_at1"
> lambda = optL1(response = y, penalized = x)$lambda
> lasso.s = penalized(response = y, penalized = x,
+                     lambda1 = lambda)
> coef(lasso.s)
 (Intercept)    258736_at    257710_at    255070_at    245319_at
-2.659077706 -0.009220815  0.273648262 -0.444106451 -0.134050990
  245094_at1
 1.589716443
```

The assumption is that the DBN is time homogeneous. These results suggest a network structure which can be created as follows:

```
> dbn2 = empty.graph(c("265768_at", "245094_at",
+                      "258736_at", "257710_at", "255070_at",
+                      "245319_at", "245094_at1"))
> dbn2 = set.arc(dbn2, "245094_at", "265768_at")
> for (node in names(coef(lasso.s))[-c(1, 6)])
+     dbn2 = set.arc(dbn2, node, "245094_at")
> dbn2 = set.arc(dbn2, "245094_at1", "245094_at")
```

The parameters of `dbn2` may be estimated via maximum likelihood. The parameters of `265769_at` and `245094_at` may then be substituted with those from the LASSO models `lasso.t` and `lasso.s`.

19.7 Exercises

1. Consider the `Canada` data set from the **vars** package. Load the data set, make some exploratory analysis and estimate a VAR(1) process for this data set. Estimate the auto-regressive matrix A and the constant matrix B which define the VAR(1) model.

   Compare the results with the LASSO matrix when the L_1 penalty is estimated by cross-validation.

   What are your conclusions?

2. Consider the `arth800` data set from the **GeneNet** package. Load the data set. The time series expression of the 800 genes is included in a data set called `arth800.expr`. Investigate its properties.

   Compute the variances of each of the 800 variables, plot them in decreasing order and create a data set with those variables whose variance is greater than 2.

   Can you fit a VAR process using the **var** package (unlikely)? Suggest alternative approaches (such as LASSO) and apply them. Estimate a DBN with each approach and compare the DBNs. Plot the DBNs using `plot` from **G1DBN**.

Chapter 20

Factor graphs and the sum product algorithm

This chapter describes the *Sum Product Algorithm*, henceforth abbreviated SPA, which was introduced by Wiberg [145] (1996). It is an algorithm for obtaining the marginals of a factorised function. It has also become known as *Loopy Belief Propagation*. It operates on *factor graphs*. SPA can be considered as the most elementary of a family of related algorithms, consisting of *double-loop algorithms* (see Heskes et. al. [63](2003)), *Generalised Belief Propagation* (see Yedidia et. al. [149] (2005)), *Expectation Propagation* (see [94](2001)), *Expectation Consistent Approximate Inference* (see Opper and Winter [102](2005)), the *Max-Product Algorithm* (see Weiss and Freeman [143](2001)), the *Survey Propagation Algorithm* (see Braunstein, Mézard and Zecchina [7] (2004) and [6](2005)) and *Fractional Belief Propagation* (see Tatikonda [133](2003)) to name but a few variants. SPA and its variants provide a natural method for a wide variety of applications: Wiberg [145] discusses applications to error correcting codes, an application developed by McEliece, MacKay and Cheng [92] (1998). It is used for satisfiability problems in combinatorial optimisation [7] and computer vision (stereo matching: Sun, Zheng and Shum [131](2003) and image restoration Tanaka [132](2002)). More recently, a variant known as 'Stochastic Belief Propagation' algorithm was developed by Noorshams and Wainwright [101] (2013) with applications to image analysis. For that situation, the number of states of each variable is large, so that only a few of the states are randomly selected for update in each cycle of the algorithm.

20.1 Factorisation and Local Functions

As usual, let $\widetilde{V} = \{1, \ldots, d\}$, and for each $j \in \widetilde{V}$ let $\mathcal{X}_j = (x_j^{(1)}, \ldots, x_j^{(k_j)})$ denote the finite state space of variable X_j. Let $\mathcal{X} = \times_{j=1}^{d} \mathcal{X}_j$. The space $\mathcal{X}$ is the *configuration space*. Let ϕ denote a function defined on $\mathcal{X}$. Let $\underline{x} = (x_1, \ldots, x_d) \in \mathcal{X}$ denote a configuration and, for a subset $D \subseteq \{1, \ldots, d\}$, where $D = \{j_1, \ldots, j_m\}$, let $\underline{x}_D = (x_{j_1}, \ldots, x_{j_m})$ and $\mathcal{X}_D = \times_{v \in D} \mathcal{X}_v$.

A domain $\mathcal{X}_D$ for $D \subset \{1, \ldots, d\}$ (where the subset is strict) is called a *local domain*.

Definition 20.1 (Factorisability)**.** *The function ϕ is said to be factorisable if it factors into a product of several* local functions *γ_j each defined on local domains, such that*

$$\phi(\underline{x}) = \prod_{j \in J} \gamma_j(\underline{x}_{D_j}) \tag{20.1}$$

for a collection of local domains $\mathcal{X}_{D_j}, j \in J$ *where* $J = \{1, 2, \ldots, q\}$ *and* $q \leq d$.

For a factorisable function ϕ, consider the problem of computing the marginal

$$\phi_i(x_i) = \sum_{\underline{z} \in \mathcal{X}_{\widetilde{V} \setminus \{i\}}} \prod_{j \in J} \gamma_j(\underline{z}, x_i), \tag{20.2}$$

where the domains of the functions have been extended to $\mathcal{X}$ (Definition 7.2). This is also known as the 'one i (eye) problem'. The aim of this chapter is to describe a procedure for computing the marginalisation, which exploits the way in which the global function is factorised and uses the current values to update the values assigned to each variable. The method involves a *factor graph*, which is an example of a *bipartite graph*.

Definition 20.2 (Bipartite Graph)**.** *A graph* $\mathcal{G}$ *is* bipartite *if its node set can be partitioned into two sets* W *and* U *in such a way that every edge in* $\mathcal{G}$ *has one node in* W *and another in* U.

A *factor graph* is a *bipartite graph* that expresses the structure of the factorisation given by Equation (20.1). The graph has the following properties:

- there is a *variable node* (an element of U) for each variable. A capital letter X will be used to denote the variable node, a small letter the value x in the state space $\mathcal{X}_X$ associated with the variable.
- there is a *function node* (an element of W) for each function γ_j. γ_j will be used to denote both the local function and the node.
- an undirected edge connecting variable node X_i to factor node γ_j if and only if X_i is in the local domain of γ_j.

In other words, a factor graph is a representation of the relation 'is an argument of'.

Example 20.3 (A Bayesian Network as a Factor Graph)**.**

A Bayesian Network has a joint probability distribution that factorises according to a DAG. This joint distribution can be converted into a factor graph. Each function is the local function $\mathbb{P}_{X_i|\Pi_i}$ and edges are drawn from this node to X_i and to its parents Π_i. The DAG corresponding to the factorisation

$$\mathbb{P}_{X_1X_2X_3X_4} = \mathbb{P}_{X_1}\mathbb{P}_{X_2|X_1}\mathbb{P}_{X_3|X_1X_2}\mathbb{P}_{X_4|X_3}$$

of $\mathbb{P}_{X_1,X_2,X_3,X_4}$ is shown in Figure 20.1 and the corresponding factor graph in Figure 20.2. □

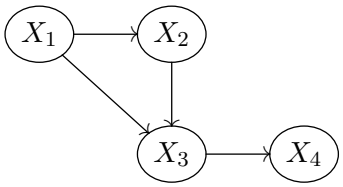

Figure 20.1: A Directed Acyclic Graph

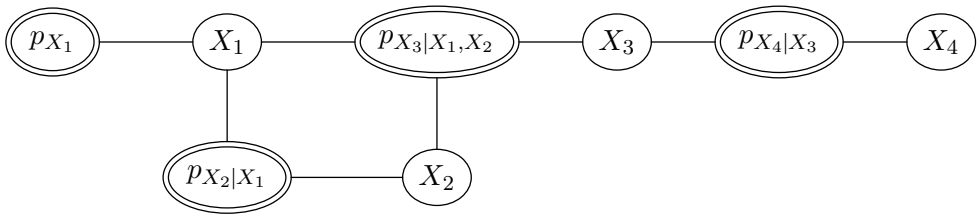

Figure 20.2: The Factor Graph Corresponding to the Directed Acyclic Graph in Figure 20.1

20.2 The Sum Product Algorithm

Figure 20.3 indicates messages to be passed. The following notation is introduced:

$$\mu_{X\to\gamma_j}(x) \qquad x \in \mathcal{X}_X: \qquad \text{Variable to local function}$$

This is the message sent from node X to node γ_j in the sum product algorithm and

$$\mu_{\gamma_j\to X}(x) \qquad x \in \mathcal{X}_X: \qquad \text{Local function to variable.}$$

This is the message sent from the function node γ_j to the variable node X.

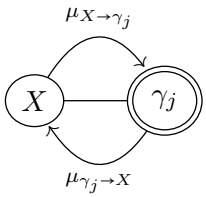

Figure 20.3: Updates in a Factor Graph

Recall the definition of *neighbour* (Definition 1.2). N_v will be used to denote the set of neighbours of a node v. A factor graph is undirected. By the definition of a factor graph, all the neighbours of a node will be of the opposite type to the node itself.

The message sent from node v on edge e is the product of the local function at v (or the *unit* function if v is a variable node) with all messages received at v on edges other than e and then marginalised to the variable associated with e. The messages are defined recursively as follows.

Definition 20.4 (Sum Product Update Rule). *For $x \in \mathcal{X}_k$, and for each $X_k \in N_{\gamma_j}$,*

$$\mu_{X_k \to \gamma_j}(x) = \begin{cases} \prod_{h \in N_{X_k} \setminus \{\gamma_j\}} \mu_{h \to X_k}(x) & \forall x \in \mathcal{X}_k \quad N_{X_k} \neq \phi \\ 1 & N_{X_k} = \phi. \end{cases} \tag{20.3}$$

and for each $\gamma_j \in N_{X_k}$,

$$\mu_{\gamma_j \to X_k}(x) = \sum_{\underline{y} \in \mathcal{X}_{\sim \setminus \{k\}}} \gamma_j(\underline{y}, x) \prod_{Y \in N_{\gamma_j} \setminus \{X_k\}} \mu_{Y \to \gamma_j}(y_j) \qquad \forall x \in \mathcal{X}_k \tag{20.4}$$

where ϕ denotes the empty set, and where the domain of γ_j has been extended to $\mathcal{X}$ and variable X_k takes the last position; y_j is the value taken by variable X_j ($j \neq k$).

The flow of computation in a factor graph is illustrated in Figure 20.4.

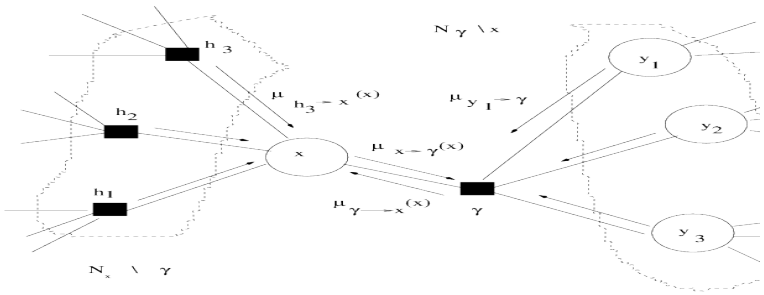

Figure 20.4: Updates in a fragment of a Factor Graph

Definition 20.5 (Initialisation). *The initialisation is*

$$\mu_{X_k \to \gamma_j}(x) = 1 \qquad \forall x \in \mathcal{X}_j$$

for each $X_k \in N_{\gamma_j}$
and

$$\mu_{\gamma_j \to X_k}(x) = 1 \qquad \forall x \in \mathcal{X}_j$$

for each $\gamma_j \in N_{X_k}$. for each variable node X_k and each function node γ_j.

Definition 20.6 (Termination)**.** *The termination at a node is the product of all messages directed towards that node.*

$$\mu_{X_k}(x) = \prod_{\gamma_j \in N_{X_k}} \mu_{\gamma_j \to X_k}(x), \qquad x \in \mathcal{X}_k \tag{20.5}$$

and

$$\mu_{\gamma_j}(\underline{x}_{D_j}) = \prod_{X_k \in N_j} \mu_{X_k \to \gamma_j}(x_k) \qquad \forall \underline{x}_{D_j} \in \mathcal{X}_{D_j}.$$

Note that the function node receives communications from precisely those variables that are in the domain of the function.

After sending sufficiently many messages according to a suitable schedule, the *termination* at the *variable* node yields the *marginalisation*, or a suitable approximation to the marginalisation, over that variable. That is,

$$\mu_{X_i}(x) = \sum_{\underline{y} \in \mathcal{X}_{\tilde{V} \setminus \{i\}}} \phi(\underline{y}, x) \qquad \forall x \in \mathcal{X}_i,$$

where the arguments of ϕ have been rearranged, so that variable X_i appears last.

Note Consider the problem where the potentials initially represent probability distributions over the domains and where hard evidence is inserted rendering the potential over the 'impossible'. For the initialisation, only those states that are possible are included and the initialisation set to 1; the other states are not included (equivalently, the corresponding initialisation is set to zero). The termination at a node then gives the *joint* probability distribution of the variable *and the evidence.* If a conditional probability is required, then the answer has to be normalised.

The Schedule One node is arbitrarily chosen as a root and, for the purposes of constructing a schedule, the edges are *directed* to form a directed acyclic graph, where the root has no parents. If the graph is a tree, then the choice of directed acyclic graph is uniquely defined by the choice of the root node. Computation begins at the leaves of the factor graph.

- Each leaf variable node sends the trivial identity function to its parents.
- Each leaf function node sends a description of γ to its parents.
- Each node waits for the message from all its children before computing the message to be sent to its parents.
- Once the root has received messages from all its children, it sends messages to all its children.
- Each node waits for messages from all its parents before computing the message to be sent to its children.

This is repeated from root to leaves and is iterated a suitable number of times. No iterations are needed if the factor graph is cycle free. This is known as a *generalised forward and backward algorithm.*

The following result was proved by N. Wiberg [145].

Theorem 20.7 (Wiberg)**.** *Let*

$$\phi(\underline{x}) = \prod_j \gamma_j(\underline{x}_{D_j})$$

and let $\mathcal{G}$ be a factor graph with **no cycles***, representing ϕ. Then, for any variable node X_k, the marginal of ϕ at $x \in \mathcal{X}_k$ is*

$$\mu_{X_k}(x) = \sum_{\underline{y} \in \mathcal{X}_{\widetilde{V}\setminus\{k\}}} \phi(\underline{y}, x),$$

where the arguments of ϕ have been rearranged so that the kth variable appears last and $\mu_{X_k}(x)$ is given in Equation (20.5).

Example 20.8.

Before giving a proof of Wiberg's theorem, the following example may be instructive. Consider

$$\phi(x_1, x_2, x_3) = \gamma_1(x_1, x_2)\gamma_2(x_2, x_3).$$

The factor graph is then a tree given in Figure 20.5.

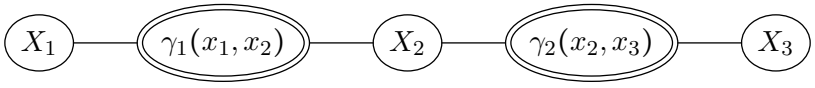

Figure 20.5: An Example on Three Variables and Two Functions

In this case, the messages are:

$$\mu_{X_1\to\gamma_1}(x_1) = \mu_{X_3\to\gamma_2}(x_3) = 1.$$

$$\mu_{\gamma_1\to X_2}(x_2) = \sum_{x_1\in\mathcal{X}_1} \gamma_1(x_1, x_2)\mu_{X_1\to\gamma_1}(x_1) = \sum_{x_1\in\mathcal{X}_1} \gamma_1(x_1, x_2)$$

$$\mu_{\gamma_2\to X_2}(x_2) = \sum_{x_3\in\mathcal{X}_3} \gamma_2(x_2, x_3)\mu_{X_3\to\gamma_2}(x_3) = \sum_{x_3\in\mathcal{X}_3} \gamma_2(x_2, x_3)$$

$$\mu_{X_2\to\gamma_2}(x_2) = \mu_{\gamma_1\to X_2}(x_2) = \sum_{x_1\in\mathcal{X}_1} \gamma_1(x_1, x_2)$$

$$\mu_{X_2\to\gamma_1}(x_2) = \mu_{\gamma_2\to X_2}(x_2) = \sum_{x_3\in\mathcal{X}_3} \gamma_2(x_2, x_3)$$

$$\mu_{\gamma_1\to X_1}(x_1) = \sum_{x_2\in\mathcal{X}_2} \gamma_1(x_1, x_2)\mu_{X_2\to\gamma_1}(x_2) = \sum_{(x_2,x_3)\in\mathcal{X}_2\times\mathcal{X}_3} \gamma_1(x_1, x_2)\gamma_2(x_2, x_3)$$

$$\mu_{\gamma_2 \to X_3}(x_3) = \sum_{(x_1,x_2)\in\mathcal{X}_1\times\mathcal{X}_2} \gamma_1(x_1,x_2)\gamma_2(x_2,x_3).$$

Note that the variable terminations are

$$\mu_{X_1}(x_1) = \mu_{\gamma_1\to X_1}(x_1) = \sum_{(x_2,x_3)\in\mathcal{X}_2\times\mathcal{X}_3} \gamma_1(x_1,x_2)\gamma_2(x_2,x_3)$$

$$\mu_{X_2}(x_2) = \mu_{\gamma_1\to X_2}(x_2)\mu_{\gamma_2\to X_2}(x_2) = \sum_{(x_1,x_3)\in\mathcal{X}_1\times\mathcal{X}_3} \gamma_1(x_1,x_2)\gamma_2(x_2,x_3)$$

$$\mu_{X_3}(x_3) = \mu_{\gamma_2\to X_3}(x_3) = \sum_{(x_1,x_2)\in\mathcal{X}_1\times\mathcal{X}_2} \gamma_1(x_1,x_2)\gamma_2(x_2,x_3),$$

which are the required marginalisation. The theorem of N. Wiberg states that if the factor graph is a tree, then after a full schedule, the terminations give the required marginalisation.

Proof of Theorem 20.7 Consider Figure 20.6. Suppose that a full schedule has been performed on a tree. The proof proceeds in three steps.

Step 1: Decompose the factor graph into n components, $R_1,\ldots,R_n$ Choose a variable X_i and suppose that n edges enter the variable node X_i. Since there are no cycles, the margin $\sum_{\underline{y}\in\mathcal{X}_{\widetilde{V}\setminus\{i\}}} \phi(\underline{y},x_i)$ (where the arguments of ϕ have been suitably rearranged) may be written as

$$\begin{aligned}
\sum_{\underline{y}\in\mathcal{X}_{\widetilde{V}\setminus\{i\}}} \phi(\underline{y},x_i) &= \sum_{\underline{y}\in\mathcal{X}|y_i=x_i} \prod_{j\in R_1}\gamma_j(\underline{y}_{D_j}) \prod_{j\in R_2}\gamma_j(\underline{y}_{D_j})\cdots\prod_{j\in R_n}\gamma_j(\underline{y}_{D_n}) \\
&= \prod_{k=1}^{n} \sum_{\underline{y}_{R_k}\in\mathcal{X}_{R_k}|y_i=x_i} \prod_{j\in R_k}\gamma_j(\underline{y}_{D_j}) \\
&= \prod_{k=1}^{n} \nu_{R_k}(x_i),
\end{aligned}$$

where the notation is clear. The last expression has the same form as the *termination formula*. Therefore the assertion is proved if it can be established that

$$\nu_{R_k}(x_i) = \mu_{\gamma_k^0\to X_i}(x_i), \quad k=1,\ldots,n,$$

where $\gamma_1^0,\ldots,\gamma_n^0$ are the n function nodes that are neighbours of X_i. Due to the clear symmetry, it is only necessary to consider one of these.

Step 2 Consider the decomposition of R_1. The case where γ_1^0 has three neighbours is illustrated in Figure 20.7. In the three variable case shown in Figure 20.7, X_1 is the node under consideration and γ_1^0 is outside R_3 and R_4. Suppose the variables neighbouring γ_1^0 are $X_1,Y_1,\ldots,Y_m$ and the regions corresponding to $Y_1,\ldots,Y_m$ are $R_{11},\ldots,R_{1m}$ respectively. Then ν_{R_1} can be decomposed as

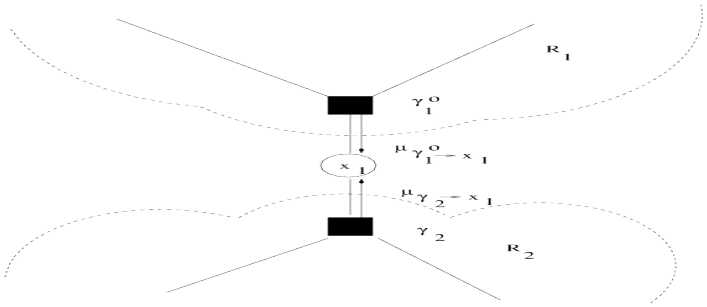

Figure 20.6: Step 1 (chosen variable X_1, which has two neighbours)

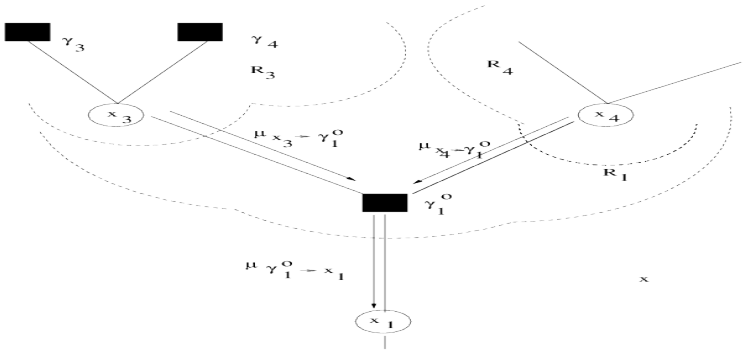

Figure 20.7: Step 2

$$
\begin{aligned}
\nu_{R_1}(x_1) &= \sum_{\underline{y} \in \mathcal{X}_{R_1} | y_1 = x_1} \prod_{j \in R_1} \gamma_j(\underline{y}_{D_j}) \\
&= \sum_{(y_1,\ldots,y_m)} \gamma_1^0(x_1, y_1, \ldots, y_m) \prod_{k=1}^{m} \left(\sum_{\underline{z}_{R_{1k}} \in \mathcal{X}_{R_{1k}} | Y_k = y_k} \prod_{j \in R_3} \gamma_j(\underline{z}_{D_j}) \right) \\
&= \sum_{(y_1,\ldots,y_m)} \gamma_1^0(x_1, y_1, \ldots, y_m) \prod_{k=1}^{m} \widetilde{\nu}_{R_{1k}}(y_k)
\end{aligned}
$$

where the notation $Y_k = y_k$ means that the value of the variable denoted Y_k takes the value y_k in $\underline{z}_{R_{1k}}$. The notation $\mathcal{X}_{R_{1k}}$ denotes all the variable nodes that are neighbours of function nodes in R_{1k}, retaining the same indices as the full set of variables.

Crucially, note that if variable X_j is a leaf node in the graph, then $\widetilde{\nu}_{R_j} \equiv 1$.

The expression for ν_{R_1} has the same form as the *update rule* given for $\mu_{\gamma_j \to X}$ in Equation (20.4). In other words, if $\widetilde{\nu}_{R_j}(y_j) = \mu_{X_j \to \gamma_1^0}(y_j)$ for each j, then the result is proved. The algorithm proceeds to the leaf nodes of the factor graph.

Step 3 There are two cases. If the leaf node is a *function* node (as in step 1, going from a variable to functions), then (clearly from the graph) this is a function (h say) of a single variable (say Y) and (from (20.4)),

$$\nu(y) = h(y) = \mu_{h \to Y}(y).$$

If the leaf node is a *variable* node X (as in step 2, going from functions to variables), then the leaf variable is adjacent to a single function h (or else it is not a leaf), which has neighbours $(Y_1, \ldots, Y_m, X)$, say, then

$$\widetilde{\nu}(x) = 1 = \mu_{X \to h}(x),$$

since if X is a leaf, then h is the only neighbour of X and hence $\mu_{X \to h}(x) \equiv 1$ from (20.3).
By tracing backward from the leaf nodes, it is now clear, by induction, that

$$\sum_{\underline{y} \in \mathcal{X}_{\widetilde{V} \setminus \{i\}}} \phi(\underline{y}, x_i) = \prod_{j=1}^{n} \mu_{\gamma_j^0 \to X_i}(x_i),$$

where $(\gamma_1^0, \ldots, \gamma_n^0)$ are the neighbours of node X_i. □

Termination Consider the termination formula

$$\mu_X(x) = \prod_{\gamma_j \in N_X} \mu_{\gamma_j \to X}(x),$$

together with the formula for the message from a variable node to a function node:

$$\mu_{X \to \gamma_j}(x) = \prod_{h \in N_X \setminus \{\gamma_j\}} \mu_{h \to X}(x).$$

Suppose the factor graph is a tree. Then, since any variable to function message is the product of all but one of the factors in the termination formula, it is clear that $\mu_X(x)$ may be computed as *the product of the two messages that were passed in opposite directions, a) from the variable X to one of the functions and b) from the function to the variable X.*

20.3 The Sum Product Algorithm on General Graphs

The result of Wiberg shows that the sum product algorithm gives the correct answer after a finite schedule when the factor graph is a tree. Unfortunately, even in relatively simple examples (Example 20.3), the factor graph is not a tree. The problem of finding conditions on whether a propagation

scheme converges to the right answer has been considered in [95]. In general, there are two major obstacles.

1. if the sum-product algorithm converges, it is not clear whether the convergence is to the required marginal.

2. the sum-product algorithm does not always converge.

If the factors are all *strictly* positive, a fixed point exists [149]. This does not imply convergence towards the fixed point and there is no guarantee that the fixed point is stable.

Mooij and Kappen [95] give sufficient conditions where the mapping has a fixed point and where there is convergence to the fixed point.

20.4 Stochastic Probability Updates

This section considers the article by Noorshams and Wainwright [101]. Some simplification to the message passing algorithm can be made if it is assumed that the function ϕ over a domain $\mathcal{X} = \times_{j=1}^{d} \mathcal{X}_j$ is of the form:

$$\phi = \prod_{j=1}^{d} \psi_j \prod_{\langle j,k\rangle \in U} \psi_{jk} \tag{20.6}$$

where the domain of ψ_j is $\mathcal{X}_j$ and the domain of ψ_{jk} is $\mathcal{X}_j \times \mathcal{X}_k$; $U \subseteq \{\langle j,k\rangle : 1 \leq j < k \leq d\}$. The *charge* here is:

$$\Phi = \{\psi_j : j \in \widetilde{V}; \psi_{jk} : \langle j,k\rangle \in U\}$$

In this case, the function nodes corresponding to the ψ_js are leaf nodes, while the function nodes ψ_{jk} only receive a message from *one* neighbour before passing a message onto a variable node. Therefore, the message passed on from the function node is identical to the message received by the function node; no multiplication is required.

For functions that factorise according to Equation (20.6), it follows that only variable to variable messages need be considered; messages are propagated along the edges of the undirected graph $\mathcal{G} = (\widetilde{V}, U)$.

Let M_{uv} denote the message transmitted along the edge $\langle u,v\rangle$ in the direction $u \mapsto v$. The message passing algorithm discussed so far, in this setting, may be expressed as:

$$\begin{cases} M_{uv}^0 \equiv 1 \\ M_{uv}^{t+1}(x_v) = \sum_{y \in \mathcal{X}_u} \psi_u(y)\psi_{uv}(y,x_v) \prod_{j \in N(u)\setminus\{v\}} M_{ju}^t \end{cases}$$

where $N(u)$ denotes the neighbours of node u in graph $\mathcal{G}$. If the factor graph is a tree, the messages are sent into a root, then propagated back out to the leaves, resulting in exact marginalisations. If the factor graph contains loops, then a suitable schedule is chosen and the updates are iterated.

Suppose that $M_{uv}^t \overset{t\to+\infty}{\longrightarrow} M_{uv}^*$. The termination is:

$$\mathbb{P}(X_u = x_u^{(k)}) = \psi_u(x_u^{(k)}) \prod_{w \in N(u)} M_{wu}^*(x_u^{(k)}).$$

The Stochastic Probability Updates of Noorshams and Wainwright [101] consider the situation where the state space for each variable $\mathcal{X}_j = (x_j^{(1)}, \ldots, x_j^{(k_j)})$ is large. Therefore, not all elements of the state space are updated at each iteration. The algorithm proceeds as follows:

Off-line Phase For the off-line phase, compute:

$$\widetilde{\Gamma}_{uv}(., x_v^{(j)}) = \frac{\psi_{uv}(., x_v^{(j)})}{\beta_{uv}(x_v^{(j)})} \qquad \beta_{uv}(x_v^{(j)}) = \sum_{i=1}^{k_u} \psi_{uv}(x_u^{(i)}, x_v^{(j)}) \psi_v(x_v^{(j)}).$$

Stochastic Update

1. Initialise message vectors $M_{vu}(x_u^{(k)})^0 \equiv 1$
2. (a) Compute the product of incoming messages

$$\widetilde{M}_{v \setminus u}(x_v^{(j)}) = \prod_{w \in N(v) \setminus \{u\}} M_{wv}^t(x_v^{(j)})$$

   (b) Pick a random index J_{vu}^t according to the probability distribution

$$p_{vu}^t(x_v^{(j)}) \propto \widetilde{M}_{v \setminus u}(x_v^{(j)}) \beta_{vu}(x_v^{(j)}) \qquad j \in \{1, \ldots, k_v\}$$

   (c) Update message vector M_{vu}^{t+1} with step-size $\lambda^t \in (0,1)$ (superscript is an index):

$$M_{vu}^{t+1}(.) = (1 - \lambda^t) M_{vu}^t(.) + \lambda^t \widetilde{\Gamma}_{uv}(., x_v^{(J_{vu}^t)}).$$

Now suppose that $k_j = K$, for some fixed $K \in \mathbb{N}$. The computational complexity of this algorithm is $O(d)$ operations per edge per round.

The number λ^t is chosen as: $\lambda^t = \frac{1}{1+t}$. It has to satisfy:

1. $\lambda^t \to 0$ as $t \to +\infty$,
2. $\sum_{t=1}^{\infty} \lambda^t = +\infty$ to ensure 'infinite travel'.

Application to Image Restoration This algorithm is presented in [101], where results on convergence are established. It is applied to image processing and computer vision; a 200×200 image (40000 pixels), with $K = 256$ grey-scale levels.

The model is the *Potts model*: it is assumed that the state space for each variable is $\mathcal{X}_j = \{1, \ldots, K\}$ and

$$\psi_{uv}(i,j) = \begin{cases} 1 & i = j \\ \gamma & i \neq j \end{cases}$$

For the Potts model,

$$\begin{cases} \beta_{uv}(j) = \psi_u(j)(1 + (K-1)\gamma) \\ \Gamma_{uv}(i,j) = \begin{cases} \frac{1}{1+(K-1)\gamma} & i = j \\ \frac{\gamma}{1+(K-1)\gamma} & i \neq j \end{cases} \end{cases}$$

For the application to image processing, the lattice is used; the edge set is

$$U = \{\langle (x,y),(x+1,y)\rangle : x = 1,\ldots,199, y = 1,\ldots,200; \\ \langle (x,y),(x,y+1)\rangle : x = 1,\ldots,200, y = 1,\ldots,199\}.$$

The parameter in the Potts model is: $\gamma = 0.05$. This is a smoothing parameter. A picture of the moon is taken, which is then contaminated by adding i.i.d. $N(0, 0.1^2)$ variables to each pixel. The algorithm is then run, where evidence is entered on the singleton potentials;

$$\psi_j(x) \leftarrow \begin{cases} 1 & \text{intensity} = x \\ 0 & \text{otherwise} \end{cases}$$

This is slightly different from the earlier discussion of the sum-product algorithm; the single variable potentials ψ_j represent the *raw data*; the edge potentials ψ_{jk} represent smoothing.

The propagation algorithm is applied and the output is the most likely value for each pixel.

The experiments indicate that the Stochastic Probability Update gives good results.

Notes The sum product algorithm is due to N.Wiberg (1996) [145], and was developed further, with applications to Bayesian networks by F.R. Ksischang , B.J. Frey and H-A. Loeliger (2001) [78] and S.M. Aji and R.J McEliece (2000) [1]. The stochastic update algorithm and application to image processing was introduced by N. Noorshams and M.J. Wainwright [101] (2013).

20.5 Exercise

Consider the directed acyclic graph below.

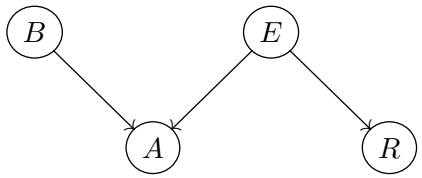

Figure 20.8: Burglary, Earthquake and Radio

The variables are B- Burglary, A - Alarm, E- Earthquake and R - news broadcast.

These are random variables with the states (0 - no (false), 1 - yes(true)). The alarm is reliable for detecting burglary, but also responds to minor earthquakes. Radio broadcasts tell about occurrences of such earthquakes, but are not always correct. The conditional probability distributions for this problem are given below.

$$\mathbb{P}_{R|E} = \begin{array}{c|cc} R\backslash E & 0 & 1 \\ \hline 0 & 0.99 & 0.05 \\ 1 & 0.01 & 0.95 \end{array}$$

$$\mathbb{P}_{A|B,E}(0|.,.) = \begin{array}{c|cc} E\backslash B & 0 & 1 \\ \hline 0 & 0.97 & 0.05 \\ 1 & 0.05 & 0.02 \end{array}$$

$$\mathbb{P}_B(1) = 0.01,\ \mathbb{P}_E(1) = 0.999$$

Assume that the joint distribution $\mathbb{P}_{A,B,E,R}$ factorises recursively according to the Bayesian network shown in the figure. Using the sum - product algorithm, compute

1. the conditional probability $\mathbb{P}_{B|A}(1|1)$
2. the conditional probability $\mathbb{P}_{B|A,R}(1|1,1)$.

20.6 Answer

The computation of $\mathbb{P}_{B|A}(1|1)$ is given. The key point is that when hard evidence $A = 1$ is received, this is accommodated by considering $\mathcal{X}_A = \{1\}$ and only considering $a = 1$. When this is done, the termination at variable B will give the function $\mathbb{P}_{B,A}(.,1)$; this has to be normalised appropriately to give the conditional probability.

The factor graph is given in Figure 20.9

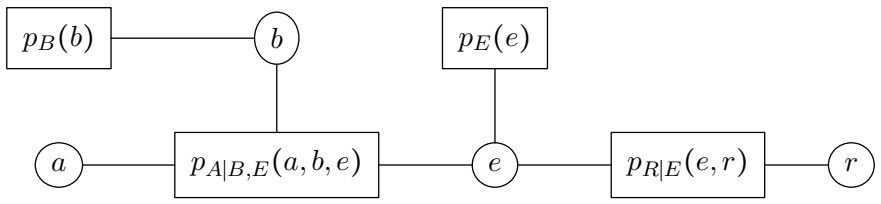

Figure 20.9: Factor Graph

$$\mu_{\mathbb{P}_B \to B} = \mathbb{P}_B = (0.99, 0.01)$$

$$\mu_{B \to \mathbb{P}_{A|B,E}} = \mu_{\mathbb{P}_B \to B} = (0.99, 0.01)$$

A is observed to be 1, so

$$\mu_{A \to \mathbb{P}_{A|B,E}}(1) = 1 \qquad \forall (b,e)$$

Now $\mu_{\mathbb{P}_{A|B,E} \to E}$ needs a marginalisation

$$\begin{aligned}
\mu_{\mathbb{P}_{A|B,E} \to E} &= \sum_b \mathbb{P}_{A|B,E}(1|b,e)\mu_{B \to \mathbb{P}_{A|B,E}}(b)\mu_{A \to \mathbb{P}_{A|B,E}}(1) \\
&= \sum_b \mathbb{P}_{A|B,E}(1|b,e)\mu_{B \to \mathbb{P}_{A|B,E}}(b) \\
&= (0.03 \times 0.99 + 0.95 \times 0.01, 0.95 \times 0.99 + 0.98 \times 0.01) = (0.0392, 0.9503)
\end{aligned}$$

$$\mu_{\mathbb{P}_E \to E} = \mathbb{P}_E$$

$$\mu_{R \to p_{R|E}} = (1,1)$$

$$\mu_{\mathbb{P}_{R|E} \to E} = \sum_r \mathbb{P}_{R|E}(r|.) = (1,1)$$

All messages have been propagated to the root E.
Message $\mu_{E \to \mathbb{P}_E}$ is not involved in the computation of $\mathbb{P}_{B|A}$ so don't compute it.

$$\mu_{E \to \mathbb{P}_{A|B,E}} = \mu_{\mathbb{P}_E \to E} = (0.001, 0.999)$$

Message $\mu_{p_{A|B,E} \to A}$ not needed, so don't compute it. Neither is $\mu_{E \to \mathbb{P}_{R|E}}$ nor $\mu_{\mathbb{P}_{R|E} \to R}$.

$$\begin{aligned}
\mu_{\mathbb{P}_{A|B,E}\to B}(b) &= \sum_e \mathbb{P}_{A|B,E}(1|b,e)\mu_{E\to\mathbb{P}_{A|B,E}}(e)\mu_{A\to\mathbb{P}_{A|B,E}}(1) \\
&= \sum_e \mathbb{P}_{A|B,E}(1|b,e)\mu_{E\to\mathbb{P}_{A|B,E}}(e) \\
&= (0.03\times 0.001 + 0.95\times 0.999, 0.95\times 0.001 + 0.98\times 0.999) = (0.94908, 0.97997)
\end{aligned}$$

Message $\mu_{B\to\mathbb{P}_B}$ not needed, because we are interested in the variable B and we need the product of messages function to the variable B.

Finally,

$$\begin{aligned}
(\mathbb{P}_{B|A}(0|1), \mathbb{P}_{B|A}(1|1)) &= \beta(\mu_{\mathbb{P}_B\to b}(0)\mu_{\mathbb{P}_{A|B,E}\to b}(0), \mu_{\mathbb{P}_B\to b}(1)\mu_{\mathbb{P}_{A|B,E}\to b}(1)) \\
&= \frac{1}{0.968469627}(0.037203936, 0.93165691)
\end{aligned}$$

Chapter 21

Graphical Models and Exponential Families

This chapter deals with multivariate distributions which fall within the framework of *exponential family*. The dependence structure is expressed as a graphical model. For an exponential family of full rank, there is a 1 - 1 mapping between canonical parameters and mean field parameters. We discuss *conjugate duality* and the Fenchel-Legendre transform between the log-partition function $A(\theta) : \theta \in \Theta$ (the canonical parameter space) and $A^*(\mu) : \mu \in \mathcal{M}$ where $\mathcal{M}$ is the mean-value parameter space and μ denotes the mean value vector of the sufficient statistic vector. The Kullback-Leibler divergence has particularly convenient form for exponential families; we discuss the primal, dual and mixed forms in terms of the canonical and mean value parametrisations. We consider *mean field approximations*, to obtain a mean field lower bound for $A(\theta)$.

21.1 Introduction to Exponential Families

The notations are as before. Let $V = \{X_1, \ldots, X_d\}$ denote the random variables. For $j = 1, \ldots, d$, $\mathcal{X}_j$ will denote the state space for variable X_j. If X_j is continuous, then $\mathcal{X}_j \subseteq \mathbb{R}$ (the real numbers). If X_j is discrete, then $\mathcal{X}_j = \{x_j^{(1)}, \ldots, x_j^{(k_j)}\}$, where k_j is possibly $+\infty$. As usual, the notation $\underline{X} = (X_1, \ldots, X_d)$ denotes the row vector of variates. An instantiation of $\underline{X}$ will be denoted $\underline{x} \in \mathcal{X}_1 \times \ldots \mathcal{X}_d \equiv \mathcal{X}$ (when no subscript is employed, $\mathcal{X}$ denotes the product space, which is the state space of the row vector $\underline{X}$).

An *exponential family* is a family of probability distributions satisfying certain properties, listed in Definition 21.1 below. For the purposes of Bayesian Networks, the emphasis is on discrete variables and Gaussian variables.

Definition 21.1 (Exponential Family)**.** *An* exponential family *is a family of probability distributions* $\{\mathbb{P}_\theta : \theta \in \Theta\}$*, where* Θ *is a parameter space. These are defined by a probability mass function* $\mathbb{P}_{\underline{X}}(.|\underline{\theta})$ *if* $\underline{X}$ *are discrete variables, or a probability density function* $\pi_{\underline{X}}(.|\underline{\theta})$ *for continuous variables, indexed by a parameter set* $\Theta \subseteq \mathbb{R}^p$ *(where* p *is possibly infinite), where there is a function* $\Phi : \mathcal{X} \to \mathbb{R}^p$*, a function* $A : \Theta \to \mathbb{R}$ *and a function* $h : \mathcal{X} \to \mathbb{R}$ *such that*

$$\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta}) = \exp\{\langle\underline{\theta}, \Phi(\underline{x})\rangle - A(\underline{\theta})\}h(\underline{x})$$

if $\underline{X}$ *is a discrete random vector and*

$$\pi_{\underline{X}}(\underline{x}|\underline{\theta}) = \exp\{\langle\underline{\theta}, \Phi(\underline{x})\rangle - A(\underline{\theta})\}h(\underline{x})$$

if $\underline{X}$ *is a continuous random vector.*

It is convenient to use the notation $\mathcal{I}$ *to denote the indexing set for the parameters;* $\underline{\theta} = (\theta_\alpha)_{\alpha\in\mathcal{I}}$. *Then* Φ *denotes a collection of functions* $\Phi = (\phi_\alpha)_{\alpha\in\mathcal{I}}$, *where* $\phi_\alpha : \mathcal{X} \to \mathbb{R}$. *The inner product notation is defined as*

$$\langle\underline{\theta}, \Phi(\underline{x})\rangle = \sum_{\alpha\in\mathcal{I}} \theta_\alpha\phi_\alpha(\underline{x}).$$

The parameters in the vector $\underline{\theta}$ *are known as the* canonical parameters *or* exponential parameters.

Attention will be restricted to distributions where $|\mathcal{I}| = p < +\infty$; namely, $\mathcal{I}$ has a finite number, p, of elements.

Since $\sum_{\mathcal{X}} \mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta}) = 1$ for discrete variables and $\int_{\mathcal{X}} \pi_{\underline{X}}(\underline{x}|\underline{\theta})d\underline{x} = 1$ for continuous variables, it follows that the quantity A, known as the *log partition function*, is given by the expression

$$A(\underline{\theta}) = \log \int_{\mathcal{X}} \exp\{\langle\underline{\theta}, \Phi(\underline{x})\rangle\}h(\underline{x})d\underline{x}$$

for continuous variables and

$$A(\underline{\theta}) = \log \sum_{\mathcal{X}} \exp\left\{\langle\underline{\theta}, \Phi(\underline{x})\rangle\right\} h(\underline{x})$$

for discrete variables. It is assumed that h, $\underline{\theta}$ and Φ satisfy appropriate conditions so that A is finite.

Set

$$P(\underline{x}; \underline{\theta}) = \frac{\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta})}{h(\underline{x})}. \tag{21.1}$$

With the set of functions Φ fixed, each parameter vector $\underline{\theta}$ indexes a particular probability function $\mathbb{P}_{\underline{X}}(.|\underline{\theta})$ belonging to the family. The exponential parameters of interest belong to the parameter space, which is the set

$$\Theta = \{\underline{\theta} \in \mathbb{R}^p | A(\underline{\theta}) < +\infty\}. \tag{21.2}$$

It will be seen shortly that A is a convex function of $\underline{\theta}$.

Definition 21.2 (Regular Families)**.** *An exponential family for which the domain* Θ *of Equation* (21.2) *is an open set is known as a* regular *family.*

Attention will be restricted to regular families.

Definition 21.3 (Minimal Representation)**.** *An exponential family, defined using a collection of functions Φ for which there is no linear combination $\langle \underline{a}, \Phi(\underline{x}) \rangle = \sum_{\alpha \in \mathcal{I}} a_\alpha \phi_\alpha(\underline{x})$ equal to a constant is known as a* minimal representation.

For a minimal representation, there is a unique parameter vector $\underline{\theta}$ associated with each distribution.

Definition 21.4 (Over-complete)**.** *An* over-complete representation *is a representation that is not minimal; there is a linear combination of the elements of Φ which yields a constant.*

When the representation is over-complete, there exists an affine subset of parameter vectors $\underline{\theta}$, each associated with the same distribution.

Recall the definition of sufficiency, given in Definition 12.12. The following lemma is crucial. Its proof is left as an exercise

Lemma 21.5. *Let $\underline{X} = (X_1, \ldots, X_d)$ be a random vector with joint probability function*

$$\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta}) = \exp\{\langle \underline{\theta}, \Phi(\underline{x}) \rangle - A(\underline{\theta})\} h(\underline{x}), \qquad \underline{x} \in \mathcal{X}$$

then $\Phi(\underline{X})$, which will be denoted Φ, is a sufficient statistic for $\underline{\theta}$. If the representation is minimal, then $\Phi(\underline{X})$ is a minimal *sufficient statistic for $\underline{\theta}$.*

Proof Exercise 1 page 436. □

21.2 Standard Examples of Exponential Families

The purpose of this section is to take some basic distributions, which are well known, and illustrate that they satisfy the definition of an exponential family.

Bernoulli Consider the random variable X, taking values 0 or 1, with probability function $\mathbb{P}_X(1) = p$, $\mathbb{P}_X(0) = 1 - p$. This may be written as

$$\mathbb{P}_X(x) = \begin{cases} p^x (1-p)^{1-x} & x \in \{0, 1\} \\ 0 & \text{other } x. \end{cases}$$

Then

$$\begin{aligned} p_X(x) &= \exp\left\{ x \log\left(\frac{p}{1-p} \right) + \log(1-p) \right\} \\ &= \exp\left\{ x\theta + \log(1-p) \right\} \\ &= \exp\left\{ x\theta - \log(1+e^\theta) \right\}, \end{aligned}$$

where $\theta = \log\left(\frac{p}{1-p}\right)$.

Notation Here, the quantity θ denotes the *canonical parameter*.

In the language of exponential families, $\mathcal{X} = \{0, 1\}$, $\Phi = \{\phi\}$ where $\phi(x) = x$, $h(0) = h(1) = 1$,

$$\mathbb{P}_X(0|\theta) = e^{-A(\theta)}, \qquad \mathbb{P}_X(1|\theta) = e^{\theta - A(\theta)}$$

In other words

$$\log \mathbb{P}_X(x|\theta) = \theta x - A(\theta),$$

which gives

$$1 = \mathbb{P}_X(0|\theta) + \mathbb{P}_X(1|\theta) = e^{-A(\theta)}(1 + e^{\theta})$$

so that

$$A(\theta) = \log(1 + \exp\{\theta\}).$$

Gaussian Recall that the one dimensional Gaussian density is of the form

$$\pi(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}.$$

This may be expressed in terms of an exponential family as follows: $\mathcal{X} = \mathbb{R}$, $h(x) = 1$, $\Phi = \{\phi_1, \phi_2\}$ where $\phi_1(x) = x$ and $\phi_2(x) = -x^2$.

$$\log \pi(x|\underline{\theta}) = \theta_1 x - \theta_2 x^2 - A(\underline{\theta})$$

where

$$1 = e^{-A(\underline{\theta})} \int_{-\infty}^{\infty} e^{\theta_1 x - \theta_2 x^2} dx.$$

The partition function is therefore

$$A(\underline{\theta}) = \frac{1}{2}\log \pi - \frac{1}{2}\log \theta_2 + \frac{\theta_1^2}{4\theta_2^2}$$

and the parameter space is

$$\Theta = \{(\theta_1, \theta_2) \in \mathbb{R}^2 | \theta_2 > 0\}.$$

Note that in the 'usual' notation

$$\theta_1 = \frac{\mu}{\sigma^2}, \qquad \theta_2 = \frac{1}{\sigma^2}.$$

Exponential Recall that an Exponential density is of the form

$$\pi(x|\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0. \end{cases}$$

This is an exponential family, taking $\mathcal{X} = (0, +\infty)$, $h(x) = dx$, $\Phi = \phi$, where $\phi(x) = -x$, $\theta = \lambda$, so that $e^{-A(\theta)} = \theta$, yielding $A(\theta) = -\log\theta$, $\Theta = (0, +\infty)$.

Poisson Recall that the probability function p for a Poisson distribution with parameter μ is given by

$$\mathbb{P}(x|\mu) = \frac{\mu^x}{x!}e^{-\mu}, \qquad x = 0, 1, 2, \ldots$$

This is an exponential family with $h(x) = \frac{1}{x!}$, $\theta = \log\mu$ so that $\mathbb{P}(x|\mu) = P(x;\theta)h(x)$, where

$$P(x;\theta) = e^{x\theta - e^\theta}.$$

This gives $A(\theta) = \exp\{\theta\}$. Since $\mu \geq 0$ and $\theta = \log\mu$, it follows that $\Theta = \mathbb{R}$.

Beta Recall that the probability density function for a Beta distribution is given by

$$\pi(x|\alpha, \beta) = \begin{cases} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} & x \in [0, 1] \\ 0 & \text{other } x. \end{cases}$$

This is an exponential family, with $\mathcal{X} = (0, 1)$, $h \equiv 1$, $\alpha - 1 = \theta_1$, $\beta - 1 = \theta_2$, $\Phi = \{\phi_1, \phi_2\}$ where $\phi_1(x) = \log x$, $\phi_2(x) = \log(1-x)$. Then

$$\log\pi(x|\underline{\theta}) = \theta_1 \log x + \theta_2 \log(1-x) - A(\underline{\theta}),$$

where the partition function A is given by

$$A(\underline{\theta}) = \log\Gamma(\theta_1 + 1) + \log\Gamma(\theta_2 + 1) - \log\Gamma(\theta_1 + \theta_2 + 2)$$

and the parameter space is $\Theta = (-1, \infty)^2$.

21.3 Graphical Models and Exponential Families

The scalar examples described in section 21.2 serve as building blocks for the construction of exponential families, which have an underlying graphical structure.

Example 21.6 (Sigmoid Belief Network Model)**.**

The *sigmoid belief network model*, described below, was introduced by R. Neal (1992) [98]. It is an exponential family, with an underlying graphical structure.

Consider a directed acyclic graph $\mathcal{G} = (V, D)$, where $V = \{X_1, \ldots, X_d\}$ is the set of variables, along which the probability distribution of $\underline{X} = (X_1, \ldots, X_d)$ may be factorised. Suppose that for each

$X_j \in V$, $j = 1, \ldots, d$, the random variable X_j takes values 0 or 1, each with probability 1/2. For any two components X_s and X_t of the random vector $\underline{X}$, component X_s has a direct causal effect on X_t only if $(X_s, X_t) \in D$.

The notation will be simplified in the following way: V and D will be used to denote the sets of nodes (variables) and directed edges respectively; the same notation will also be used to denote the *indexing* sets of nodes and directed edges. In other words the notations

$$V = \{1, \ldots, d\} \qquad \text{and} \qquad D = \{(s,t) | (X_s, X_t) \in D\}$$

will also be used. The meaning will be clear from the context. The probability distribution over the possible configurations is modelled by an exponential family with probability function $\mathbb{P}_{\underline{X}}(.|\underline{\theta})$ of the form

$$\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta}) = \exp\left\{ \sum_{s=1}^{d} \theta_s x_s + \sum_{(s,t)\in D} \theta_{(s,t)} x_s x_t - A(\underline{\theta}) \right\}.$$

The notation Pa_i denotes the parent set of node X_i and $\pi_i(\underline{x})$ denotes the instantiation of Pa_i corresponding to the instantiation $\{\underline{X} = \mathbf{x}\}$, this may be rewritten as

$$\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta}) = \prod_{i=1}^{d} \mathbb{P}_{X_i|\mathrm{Pa}_i}(x_i|\pi_i(\underline{x}), \underline{\theta}),$$

where (clearly)

$$\mathbb{P}_{X_i|\mathrm{Pa}_i}(x_i|\pi_i(\underline{x}), \underline{\theta}) = \frac{\exp\left\{x_i\left(\theta_i + \sum_{x_j\in\pi_i(\underline{x})} \theta_{(ij)} x_j\right)\right\}}{1 + \exp\left\{\theta_i + \sum_{x_j\in\pi_i(\underline{x})} \theta_{(ij)} x_j\right\}},$$

where the notation $x_j \in \pi_i(\underline{x})$ is clear. The index set is $\mathcal{I} = V \cup D$. The domain $\Theta = \mathbb{R}^n$, where $n = |\mathcal{I}|$. Since the sum that defines $A(\underline{\theta})$ is finite for all $\underline{\theta} \in \mathbb{R}^n$, it follows that the family is regular. It is *minimal*, since there is no linear combination of the functions equal to a constant.

This model may be generalised. For example, one may consider higher order interactions. To include coupling of triples (X_s, X_t, X_u), one would add a monomial $x_s x_t x_u$ with corresponding exponential parameter $\theta_{(s,t,u)}$. More generally, the set $\mathcal{C}$ of indices of interacting variables may be considered, giving

$$\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta}) = \exp\left\{ \sum_{C\in\mathcal{C}} \theta_{(C)} \prod_{s\in C} X_s - A(\underline{\theta}) \right\}.$$

□

Example 21.7 (Noisy 'or' as an Exponential Family)**.**

The QMR - DT (Quick Medical Reference - Decision Theoretic) database is a large scale probabilistic data base that is intended to be used as a diagnostic aid in the domain of internal medicine. It is a

bipartite graphical model; that is, a graphical model where the nodes may be of one of two types. The upper layer of nodes (the parents) represent diseases and the lower layer of nodes represent symptoms. There are approximately 600 disease nodes and 4000 symptom nodes in the database.

An *evidence*, or *finding* will be a set of observed symptoms, denoted by a vector of length 4000, each entry being a 1 or 0 depending upon whether or not the symptom is present or absent. This will be denoted $\underline{f}$, which is an instantiation of the random vector $\underline{F}$. The vector $\underline{d}$ will be used to represents the diseases; this is considered as an instantiation of the random vector $\underline{D}$. Let d_j denote component j of vector $\underline{d}$ and let f_j denote component j of vector $\underline{f}$. Then, if the occurrence of various diseases are taken to be independent of each other, the following factorisation holds:

$$\mathbb{P}_{\underline{F},\underline{D}}(\underline{f},\underline{d}) = \mathbb{P}_{\underline{F}|\underline{D}}(\underline{f}|\underline{d})\mathbb{P}_{\underline{D}}(\underline{d}) = \prod_i \mathbb{P}_{F_i|\underline{D}}(f_i|\underline{d}) \prod_j \mathbb{P}_{D_j}(d_j).$$

This may be represented by *noisy 'or'* model. Let q_{i0} denote the probability that symptom i is present in the absence of any disease and q_{ij} the probability that disease j induces symptom i, then the probability that symptom i is absent, given a vector of diseases $\underline{d}$ is

$$\mathbb{P}_{F_i|\underline{D}}(0|\underline{d}) = (1-q_{i0}) \prod_j (1-q_{ij})^{d_j}.$$

The *noisy or* may then be rewritten in an exponential form:

$$\mathbb{P}_{F_i|\underline{D}}(0|\underline{d}) = \exp\left\{ -\sum_j \theta_{ij} d_j - \theta_{i0} \right\},$$

where $\theta_{ij} \equiv \log(1-q_{ij})$ are the transformed parameters.

21.4 Properties of the log Partition Function

Firstly, some basic properties of the log partition function $A(\underline{\theta})$ are discussed, which are then developed using convex analysis, discussed in [3]. Let $\mathbb{E}_{\underline{\theta}}[.]$ denote expectation with respect to $p(.|\underline{\theta})$ for discrete variables, or $\pi(.|\underline{\theta})$ for continuous variables. Of particular importance is the idea that the vector $\underline{\mu}$, where $\mu_i := \mathbb{E}_{\underline{\theta}}[\phi_i(\underline{X})]$ provides an alternative parametrisation of the exponential family. Here expectation is defined as

$$\mathbb{E}_{\underline{\theta}}[f(\underline{X})] = \int_{\mathcal{X}} \pi_{\underline{X}}(\underline{x}|\underline{\theta}) f(\underline{x}) d\underline{x}$$

if $\underline{X}$ is a continuous random vector and

$$\mathbb{E}_{\underline{\theta}}[f(\underline{X})] = \sum_{\underline{x} \in \mathcal{X}} \mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta}) f(\underline{x})$$

if $\underline{X}$ is a discrete random vector. Recall that, for discrete variables,

$$A(\underline{\theta}) = \log \sum_{\underline{x} \in \mathcal{X}} e^{\langle \underline{\theta}, \Phi(\underline{x}) \rangle} h(\underline{x}). \tag{21.3}$$

Provided expectations and variances exist, it follows that

$$\frac{\partial}{\partial\theta_\alpha}A(\underline{\theta}) = \sum_{\underline{x}\in\mathcal{X}} e^{\langle\underline{\theta},\Phi(\underline{x})\rangle - A(\underline{\theta})}\phi_\alpha(\underline{x})h(\underline{x}) = E_{\underline{\theta}}[\phi_\alpha(\underline{X})]. \tag{21.4}$$

Taking second derivatives yields

$$\frac{\partial}{\partial\theta_\alpha\partial\theta_\beta}A(\underline{\theta}) = \mathbb{E}_{\underline{\theta}}[\phi_\alpha(\underline{X})\phi_\beta(\underline{X})] - \mathbb{E}_{\underline{\theta}}[\phi_\alpha(\underline{X})]E_{\underline{\theta}}[\phi_\alpha(\underline{X})] = \mathrm{Cov}_{\underline{\theta}}(\phi_\alpha(\underline{X}),\phi_\beta(\underline{X})).$$

It is and easy to show, and a standard fact, that any covariance matrix is non negative definite. It now follows that, on Θ, A is a convex function.

Mapping to Mean Parameters Given a vector of functions Φ, set $F(\underline{\theta}) = \mathbb{E}_{\underline{\theta}}[\Phi(\underline{X})]$ and let $\mathcal{M} = F(\Theta)$. For an arbitrary exponential family defined by

$$\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta}) = \exp\{\langle\underline{\theta},\Phi(\underline{x})\rangle - A(\underline{\theta})\}\,h(\underline{x}),$$

a mapping $\Lambda : \Theta \to \mathcal{M}$ may be defined as follows:

$$\Lambda(\underline{\theta}) := E_{\underline{\theta}}[\Phi(\underline{X})].$$

To each $\underline{\theta} \in \Theta$, the mapping Λ associates a vector of *mean parameters* $\underline{\mu} = \Lambda(\underline{\theta})$ belonging to the set $\mathcal{M}$. Note that, by Equation (21.4),

$$\Lambda(\underline{\theta}) = \nabla A(\underline{\theta}).$$

The mapping Λ is one to one, and hence invertible on its image, when the representation is minimal. The image of Θ is the interior of $\mathcal{M}$.

Example 21.8 (Bernoulli Trial)**.**

Consider a Bernoulli random variable X with state space $\{0,1\}$. That is, $p_X(0) = 1-p$ and $p_X(1) = p$. Now consider an Overcomplete exponential representation

$$\mathbb{P}_X(x|\underline{\theta}) = \exp\{\theta_0(1-x) + \theta_1 x - A(\theta_0,\theta_1)\}$$

so that

$$A(\theta_0,\theta_1) = \log\left(e^{\theta_0} + e^{\theta_1}\right).$$

Here $\Theta = \mathbb{R}^2$. $\phi_0(x) = 1-x$ and $\phi_1(x) = x$.

$$\frac{\partial}{\partial\theta_0}A(\underline{\theta}) = e^{\theta_0 - A(\theta_0,\theta_1)} = 1-p = \mu_0$$

$$\frac{\partial}{\partial\theta_1}A(\underline{\theta}) = e^{\theta_1 - A(\theta_0,\theta_1)} = p = \mu_1.$$

The set $\mathcal{M}$ of *mean parameters* is the simplex $\{(\mu_0, \mu_1) \in \mathbb{R}_+ \times \mathbb{R}_+ | \mu_0 + \mu_1 = 1\}$. For any fixed $\underline{\mu} = (\mu_0, \mu_1)$ where $\mu_0 \geq 0, \mu_1 \geq 0, \mu_0 + \mu_1 = 1$, the inverse image is,

$$\Lambda^{-1}(\underline{\mu}) = \left\{ (\theta_0, \theta_1) \in \mathbb{R}^2 \middle| \frac{e^{\theta_0}}{e^{\theta_0} + e^{\theta_1}} = \mu_0 \right\}$$

which may be rewritten as

$$\Lambda^{-1}(\underline{\mu}) = \left\{ (\theta_0, \theta_1) \in \mathbb{R}^2 \middle| \theta_1 - \theta_0 = \log \frac{\mu_1}{\mu_0} \right\}.$$

In an over-parametrised, or over-complete representation, there is no longer a bijection between Θ and $\Lambda(\Theta)$. Instead, there is a bijection between elements of $\Lambda(\Theta)$ and a affine subsets of Θ. A pair $(\underline{\theta}, \underline{\mu})$ is said to be *dually coupled* if $\underline{\mu} = \Lambda(\underline{\theta})$, and hence $\underline{\theta} \in \Lambda^{-1}(\underline{\mu})$.

21.5 Fenchel Legendre Conjugate

The *Fenchel Legendre conjugate* of the log partition function A is defined as follows:

$$A^*(\underline{\mu}) := \sup_{\underline{\theta} \in \Theta} \left\{ \langle \underline{\mu}, \underline{\theta} \rangle - A(\underline{\theta}) \right\}. \tag{21.5}$$

The choice of notation is deliberately suggestive; the variables in the Fenchel Legendre dual turn out to have interpretation as the mean parameters. Recall the definition of P given by Equation (21.1); namely, if $\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta})$ is the probability function (or density function), then

$$P(\underline{x}; \theta) = \frac{\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta})}{h(\underline{x})}.$$

Definition 21.9 (Boltzmann - Shannon Entropy)**.** *The Boltzmann - Shannon entropy of* $\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta})$ *with respect to* h *is defined as*

$$H(\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta})) = -\mathbb{E}_{\underline{\theta}}[\log P(\underline{x}; \underline{\theta})].$$

The following is the main result of the chapter.

Theorem 21.10. *For any* $\underline{\mu} \in \mathcal{M}$*, let* $\underline{\theta}(\underline{\mu}) \in \Lambda^{-1}(\underline{\mu})$*. Then*

$$A^*(\underline{\mu}) = -H(\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta}(\underline{\mu}))).$$

In terms of this dual, for $\underline{\theta} \in \Theta$*, the log partition satisfies be expressed:*

$$A(\underline{\theta}) = \sup_{\underline{\mu} \in \mathcal{M}} \{ \langle \underline{\theta}, \underline{\mu} \rangle - A^*(\underline{\mu}) \}. \tag{21.6}$$

Proof of Theorem 21.10 From the definition $\underline{\mu} = \mathbb{E}_{\underline{\theta}}[\Phi(\underline{X})]$, it follows that

$$-H(\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta})) = \mathbb{E}_{\underline{\theta}}[\log P(\underline{X};\underline{\theta})] = \mathbb{E}_{\underline{\theta}}[\langle \underline{\theta}, \Phi(\underline{X})\rangle] - A(\underline{\theta}) = \langle \underline{\theta}, \underline{\mu}\rangle - A(\underline{\theta}). \tag{21.7}$$

Consider the function

$$F(\underline{\mu}, \underline{\theta}) = \langle \underline{\mu}, \underline{\theta}\rangle - A(\underline{\theta}).$$

Let $\underline{\theta}(\underline{\mu})$ denote a value of $\underline{\theta}$ that maximises $F(\underline{\mu}, \underline{\theta})$ if such a value exists in Θ. The result follows directly by using the definition given by Equation (21.5) together with Equation (21.7). Otherwise, let $\underline{\theta}^{(n)}(\underline{\mu})$ denote a sequence such that $\lim_{n\to+\infty} F(\underline{\mu}, \underline{\theta}^{(n)}(\underline{\mu})) = A^*(\underline{\mu})$. The first statement of the theorem follows directly.

For the second part, choose $\underline{\theta} \in \Theta$ and choose $\underline{\mu}(\underline{\theta}) = \nabla_{\underline{\theta}} A(\underline{\theta})$. By definition of $\mathcal{M}$, note that $\mu(\theta) \in \mathcal{M}$. Since A is convex, it follows that $\underline{\mu}(\underline{\theta})$ maximises $\langle \underline{\theta}, \underline{\mu}\rangle - A(\underline{\theta})$, so that

$$A(\underline{\theta}) = \langle \underline{\mu}(\underline{\theta}), \underline{\theta}\rangle - A^*(\underline{\mu}(\underline{\theta})).$$

But, from the definition of $A^*(\underline{\mu})$, it follows that for all $\underline{\mu} \in \mathcal{M}$,

$$A(\underline{\theta}) \geq \langle \underline{\mu}, \underline{\theta}\rangle - A^*(\underline{\mu}).$$

From this,

$$A(\underline{\theta}) = \sup_{\underline{\mu}\in\mathcal{M}} \{\langle \underline{\mu}, \underline{\theta}\rangle - A^*(\underline{\mu})\}$$

and Theorem 21.10 is established. □

Examples The conjugate dual pair (A, A^*) is now computed for several examples of exponential families.

Bernoulli Recall that $A(\theta) = \log(1 + \exp\{\theta\})$ for $\theta \in \mathbb{R}$. It follows that

$$A^*(\mu) = \sup_{\theta\in\mathbb{R}} \{\theta\mu - \log(1 + e^{\theta})\}$$

The supremum is attained for $\theta(\mu)$ satisfying

$$\mu = \frac{e^{\theta(\mu)}}{1 + e^{\theta(\mu)}}.$$

It follows that

$$e^{\theta(\mu)} = \frac{\mu}{1 - \mu}$$

and

$$\theta(\mu) = \log \mu - \log(1-\mu)$$

so that

$$A^*(\mu) = \mu \log \mu - \mu \log(1-\mu) - \log(1 + \frac{\mu}{1-\mu}),$$

which gives

$$A^*(\mu) = \mu \log \mu + (1-\mu)\log(1-\mu).$$

Gaussian Recall that $\Theta = \{(\theta_1, \theta_2) | \theta_2 > 0\}$ and

$$A(\underline{\theta}) = \frac{1}{2}\log \pi - \frac{1}{2}\log \theta_2 + \frac{\theta_1^2}{4\theta_2}.$$

$$A^*(\underline{\mu}) = \sup_{\underline{\theta} \in \Theta} \{\theta_1\mu_1 + \theta_2\mu_2 - \frac{1}{2}\log \pi + \frac{1}{2}\ln \theta_2 - \frac{\theta_1^2}{4\theta_2}\}.$$

This is maximised when

$$\begin{cases} \mu_1 - \frac{\theta_1(\mu)}{2\theta_2(\mu)} = 0 \\ \mu_2 + \frac{1}{2\theta_2(\mu)} + \frac{\theta_1^2(\mu)}{4\theta_2^2(\mu)} = 0, \end{cases}$$

which gives

$$\begin{cases} \theta_2(\mu_1, \mu_2) = -\frac{1}{2(\mu_1^2 + \mu_2)} \\ \theta_1(\mu) = -\frac{\mu_1}{\mu_1^2 + \mu_2} \end{cases}$$

and

$$A^*(\mu_1, \mu_2) = -\frac{1}{2} - \frac{1}{2}\log \pi - \frac{1}{2}\log(-2(\mu_1^2 + \mu_2)).$$

Note that

$$\mathcal{M} = \{(\mu_1, \mu_2) | \mu_1^2 + \mu_2 < 0\}.$$

Exponential Distribution Recall that $\Theta = (0, +\infty)$ and that $A(\theta) = -\log(\theta)$. By a straightforward computation,

$$A^*(\mu) = -1 - \log(-\mu)$$

and

$$\mathcal{M} = (-\infty, 0).$$

Poisson Distribution Recall that $\Theta = \mathbb{R}$ and that $A(\theta) = \exp\{\theta\}$. It is a straightforward computation to see that

$$A^*(\mu) = \mu \log \mu - \mu$$

and that

$$\mathcal{M} = (0, +\infty).$$

21.6 Kullback Leibler Divergence

Recall Definition 6.9, the Kullback Leibler distance between two probability distributions $\underline{p} \in [0,1]^M$ and $\underline{q} =\in [0,1]^M$

$$D_{KL}(\underline{q}|\underline{p}) = \sum_{j=1}^{M} q_j \ln \frac{q_j}{p_j}.$$

This may be written as

$$D_{KL}(q|p) = \mathbb{E}_q \left[\log \frac{q(X)}{p(X)} \right], \tag{21.8}$$

where X is a random vector with state space $\mathcal{X} = (x_1, \ldots, x_M)$ and $\mathbb{E}_{\underline{q}}$ is expectation with respect to the measure such that $q_j = \mathbb{P}(X = x_j)$. The definition of Kullback Leibler may be extended to continuous distributions using Equation (21.8), where q and p denote the respective density functions. In this case, Equation (21.8) is taken as

$$D_{KL}(q|p) = \int_{\mathbb{R}^d} q(x) \log \frac{q(x)}{p(x)} dx.$$

When q and p are members of the same exponential family, the Kullback Leibler distance may be computed in terms of the parameters. The key result, for expressing the distance in terms of the partition function, is the *Fenchel's inequality* given in Equation (21.9), which can be seen directly from the definition of $A^*(\underline{\mu})$.

$$A(\underline{\theta}) + A^*(\underline{\mu}) \geq \langle \underline{\mu}, \underline{\theta} \rangle, \tag{21.9}$$

with equality if and only if $\underline{\mu} = \Lambda(\underline{\theta})$ and $\underline{\theta} \in \Lambda^{-1}(\underline{\mu})$. That is, for $\underline{\mu} = \Lambda(\underline{\theta})$ and $\underline{\theta} \in \Lambda^{-1}(\underline{\mu})$,

$$A(\underline{\theta}) + A^*(\underline{\mu}) = \langle \underline{\mu}, \underline{\theta} \rangle. \tag{21.10}$$

Consider an exponential family of distributions, and consider two exponential parameter vectors, $\underline{\theta}_1 \in \Theta$ and $\underline{\theta}_2 \in \Theta$. When distributions are from the same exponential family, the notation $D(\underline{\theta}_1|\underline{\theta}_2)$ is used

to denote $D_{KL}(p(.|\theta_1)|p(.|\theta_2))$. Set $\underline{\mu}_i = \Lambda(\underline{\theta}_i)$. Using the parameter to denote the distribution with respect to which the expectation is taken, note that

$$D(\underline{\theta}_1|\underline{\theta}_2) = \mathbb{E}_{\underline{\theta}_1}\left[\log\frac{\mathbb{P}(\underline{X}|\theta_1)}{\mathbb{P}(\underline{X}|\underline{\theta}_2)}\right] = A(\underline{\theta}_2) - A(\underline{\theta}_1) - \langle\underline{\mu}_1, \underline{\theta}_2 - \underline{\theta}_1\rangle. \tag{21.11}$$

The representation of the Kullback Leibler divergence given in Equation (21.11) is known as the *primal form* of the KL divergence.

Taking $\underline{\mu}_1 = \Lambda(\underline{\theta}_1)$ and applying Equation (21.10), the Kullback Leibler distance may also be written

$$D(\underline{\theta}_1|\theta_2) \equiv \widetilde{D}(\underline{\mu}_1|\underline{\theta}_2) = A(\underline{\theta}_2) + A^*(\underline{\mu}_1) - \langle\underline{\mu}_1, \underline{\theta}_2\rangle. \tag{21.12}$$

The representation given in Equation (21.12) is known as the *mixed form* of the KL divergence. Recall the definition of A^* given by

$$A^*(\underline{\mu}) := \sup_{\underline{\theta}\epsilon\Theta}\{\langle\underline{\mu}, \underline{\theta}\rangle - A(\underline{\theta})\}$$

and recall Equation (21.6) from Theorem 21.10,

$$A(\underline{\theta}) = \sup_{\underline{\mu}\epsilon\mathcal{M}}\{\langle\underline{\theta}, \underline{\mu}\rangle - A^*(\underline{\mu})\}.$$

Equation (21.6) may be rewritten as

$$\inf_{\underline{\mu}\epsilon\mathcal{M}}\{A(\underline{\theta}) + A^*(\underline{\mu}) - \langle\underline{\theta}, \underline{\mu}\rangle\} = 0.$$

It follows that $\inf_{\underline{\mu}\epsilon\mathcal{M}} \widetilde{D}(\underline{\mu}|\underline{\theta}) = 0$.

Finally, taking $\underline{\mu}_2 = \Lambda(\underline{\theta}_2)$ and applying Equation (21.10) once again to Equation (21.12) yields the so-called *dual form* of the KL divergence;

$$\widetilde{\widetilde{D}}(\underline{\mu}_1|\underline{\mu}_2) \equiv D(\underline{\theta}_1|\underline{\theta}_2) = A^*(\underline{\mu}_1) - A^*(\underline{\mu}_2) - \langle\underline{\theta}_2, \underline{\mu}_1 - \underline{\mu}_2\rangle. \tag{21.13}$$

21.7 Mean Field Theory

In this section, probability distributions of the form

$$\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta}) = \exp\left\{\sum_\alpha \theta_\alpha\phi_\alpha(\underline{x}) - A(\underline{\theta})\right\}h(\underline{x})$$

are considered. Mean field theory techniques are discussed and it is shown how they may be used to obtain estimates of the log partition function $A(\underline{\theta})$. This is equivalent to the problem of finding an

appropriate normalising constant to make a function into a probability density, a problem that often arises when updating using Bayes rule.

Mean Field Theory is based on the variational principle of Equation (21.6). The two fundamental difficulties associated with the variational problem are the nature of the constraint set $\mathcal{M}$ and the lack of an explicit form for the dual function A^*. Mean field theory entails limiting the optimization to a subset of distributions for which A^* is relatively easy to characterise.

More specifically, the discussion of this chapter is restricted to the case where the functions ϕ_α are either linear or quadratic. The problem therefore reduces to considering a graph $\mathcal{G} = (V, U)$, where the node set V denotes the variables and the edge set U denotes a direct association between the variables. For this discussion, the edges in U are assumed to be *un*directed. As usual, V and U denote the node (variable) and undirected edge sets; the same notation is used for the *indexing* sets. That is, with minor abuse of notation (clear from the context), V and U are also used to mean: $V = \{1, \ldots, d\}$ and $U = \{\langle s, t\rangle | \langle X_s, X_t\rangle \in E\}$. Specifically, the probability distributions under consideration are of the form

$$\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta}) = \exp\left\{\sum_{s\in\widetilde{V}} \theta_s x_s + \sum_{(s,t)\in\widetilde{E}} \theta_{(s,t)} x_s x_t - A(\underline{\theta})\right\}.$$

Let H denote a sub-graph of $\mathcal{G}$ over which it is feasible to perform exact calculations. In an exponential formulation, the set of all distributions that respect the structure of H can be represented by a linear subspace of the exponential parameters. Let $\mathcal{I}(H)$ denote the subset of indices associated with cliques in H. Then the set of exponential parameters corresponding to distributions structured according to H is given by

$$\mathcal{E}(H) := \{\underline{\theta} \in \Theta \mid \theta_\alpha = 0,\ \alpha \in \mathcal{I}\backslash\mathcal{I}(H)\}.$$

The simplest example is to consider the completely disconnected graph $H = (V, \phi)$. Then

$$\mathcal{E}(H) = \{\underline{\theta} \in \Theta \mid \theta(s,t) = 0,\ (s,t) \in E\}.$$

The associated distributions are of the product form

$$\mathbb{P}_{\underline{X}}(\underline{X}|\underline{\theta}) = \prod_{s\in\widetilde{V}} \mathbb{P}_{X_s}(x_s|\theta_s).$$

Optimisation and Lower Bounds Let $\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta})$ denote the *target distribution* that is to be approximated. The basis of mean field approximation is the following: any valid mean parameter specifies a lower bound on the log partition function, established using Jensen's inequality.

Proposition 21.11 (Mean Field Lower Bound)**.**

$$A(\underline{\theta}) \geq \sup_{\underline{\mu}\in\mathcal{M}} \left\{\langle\underline{\theta}, \underline{\mu}\rangle - A^*(\underline{\mu})\right\}$$

Proof The proof is given for discrete variables; the proof for continuous variables is exactly the same, replacing the sum with an integral.

$$\begin{aligned} A(\underline{\theta}) &= \log \sum_{\underline{x} \in \mathcal{X}} \exp\{\langle \underline{\theta}, \Phi(\underline{x}) \rangle\} \\ &= \log \sum_{\underline{x} \in \mathcal{X}} \mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta}) \exp\{\langle \underline{\theta}, \Phi(\underline{X}) \rangle - \log \mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta})\} \\ &= \log \mathbb{E}_{\underline{\theta}}[\exp\{\langle \underline{\theta}, \Phi(\underline{X}) \rangle - \log \mathbb{P}_{\underline{X}}(\underline{X}|\underline{\theta})\}] \\ &\overset{(a)}{\geq} \langle \underline{\theta}, \mathbb{E}_{\underline{\theta}}[\Phi(\underline{X})] \rangle - \mathbb{E}_{\underline{\theta}}[\log \mathbb{P}_{\underline{X}}(\underline{X}|\underline{\theta})\}] \\ &= \langle \underline{\theta}, \underline{\mu} \rangle - A^*(\underline{\mu}). \end{aligned}$$

The inequality (a) follows from Jensen's inequality; the last line follows from Theorem 21.10. □

There are difficulties in computing the lower bound in cases where there is not an explicit form for $A^*(\underline{\mu})$. The mean field approach circumvents this difficulty by restricting to

$$\mathcal{M}(G;H) := \left\{ \underline{\mu} \in \mathbb{R}^d \mid \underline{\mu} = \mathbb{E}_{\underline{\theta}}[\Phi(\underline{X})], \underline{\theta} \in \mathcal{E}(H) \right\}.$$

Note that $\mathcal{M}(G;H) \subset \mathcal{M}$, hence

$$A(\underline{\theta}) \geq \sup_{\underline{\mu} \in \mathcal{M}} \left\{ \langle \underline{\theta}, \underline{\mu} \rangle - A^*(\underline{\mu}) \right\} \geq \sup_{\underline{\mu} \in \mathcal{M}(G;H)} \left\{ \langle \underline{\theta}, \underline{\mu} \rangle - A^*(\underline{\mu}) \right\}.$$

This lower bound is the best that can be obtained by restricting to H.

Let $\underline{\mu}^{(n)}$ denote a sequence such that for each n, $\underline{\mu}^{(n)} \in \mathcal{M}(G,H)$, such that $\underline{\mu}^{(n)} \overset{n \to +\infty}{\longrightarrow} \underline{\mu}$ and such that

$$\langle \underline{\theta}, \underline{\mu}^{(n)} \rangle - A^*(\underline{\mu}^{(n)}) \overset{n \to +\infty}{\longrightarrow} \sup_{\underline{\mu} \in \mathcal{M}(G;H)} \left\{ \langle \underline{\theta}, \underline{\mu} \rangle - A^*(\underline{\mu}) \right\}.$$

Note that $\underline{\mu} \in \overline{\mathcal{M}(G;H)}$. Since $\underline{\theta} \in \Theta$, it follows that $\underline{\mu} \in \mathcal{M}$. The distribution associated with $\underline{\mu}$ minimises the Kullback Leibler divergence between the approximating distribution and the target distribution, subject to the constraint that $\underline{\mu} \in \overline{\mathcal{M}(G;H)}$. Recall the mixed form of the Kullback Leibler divergence; namely, Equation (21.12).

$$\widetilde{D}(\underline{\mu}|\underline{\theta}) = A(\underline{\theta}) - A^*(\underline{\mu}) - \langle \underline{\mu}, \underline{\theta} \rangle.$$

Naive Mean Field Updates In the *naive mean field* approach, a fully factorised distribution is chosen. This is equivalent to the approximation obtained by taking an empty edge set to approximate the original distribution. The naive mean field updates are a set of recursions for finding a stationary point of the resulting optimisation problem.

Example 21.12 (Sigmoid Network Model)**.**

Let $\underline{X} = (X_1, \ldots, X_d)$ be a random vector with state space $\mathcal{X} = \{0,1\}^d$ (d binary variables). Suppose that the distribution may be factorised along an undirected graph $\mathcal{G} = (V, U)$. The probability function is given by

$$\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta}) = \exp\left\{\sum_{j=1}^{n} \theta_j x_j + \sum_{\langle i,j\rangle \in U} \theta_{\langle i,j\rangle} x_i x_j - A(\underline{\theta})\right\}.$$

The *naive mean field* approach involves considering the graph with no edges. In this restricted class,

$$\mathbb{P}_{\underline{X}}(\underline{x}|\underline{\theta}) = \exp\left\{\sum_{j=1}^{n} \theta_j x_j - A(\underline{\theta}^{(H)})\right\},$$

where $\underline{\theta}^{(H)}$ is the collection of parameters $\theta_s^{(H)} = \theta_s$, $s = 1, \ldots, d$ and $\theta^{(H)}(s,t) \equiv 0$. Note that

$$\mu_s = \mathbb{E}_{\underline{\theta}}[\phi_s(\underline{X})] = \mathbb{E}_{\underline{\theta}}[X_s]$$

and

$$\mu_{(s,t)} = \mathbb{E}_{\underline{\theta}}[\phi_{s,t}(\underline{X})] = \mathbb{E}_{\underline{\theta}}[X_s X_t].$$

When $\underline{\theta} \in H$, it follows that $(X_s)_{s=1}^d$ are independent, so that

$$\mu_{(s,t)} = \mathbb{E}_{\underline{\theta}}[X_s X_t] = \mu_s \mu_t.$$

The optimisation is therefore restricted to the set of parameters

$$\mathcal{M}(G;H) = \{(\mu_s)_{s=1}^d, (\mu_{\langle s,t\rangle})_{\langle s,t\rangle \in \{1,\ldots,d\}^2} | 0 \le \mu_s \le 1, \mu_{\langle s,t\rangle} = \mu_s \mu_t.\}$$

With the restriction to product form distributions, $(X_s)_{s=1}^d$ are independent Bernoulli variables and hence

$$A_H^*(\underline{\mu}) = \sum_{s=1}^{d} \{\mu_s \log \mu_s + (1-\mu_s)\log(1-\mu_s)\}.$$

Set

$$F(\underline{\mu};\underline{\theta}) = \sum_{s=1}^{d} \theta_s \mu_s + \sum_{\langle s,t\rangle \in U} \theta_{(s,t)} \mu_s \mu_t - \sum_{s=1}^{d} (\mu_s \log \mu_s + (1-\mu_s)\log(1-\mu_s)),$$

then the lower bound is given by

$$A(\underline{\theta}) \ge \sup_{(\mu_s)_{s=1}^d \in [0,1]^d} F(\underline{\mu};\underline{\theta}).$$

Note that, for each μ_s, the function F is strictly convex. It is easy to see that the maximum is attained when, for all $1 \le s \le t$, $(\mu_t)_{t=1}^d$ satisfies

$$\theta_s + \sum_{t:\langle s,t\rangle \in U} \theta_{\langle s,t\rangle} \mu_t - \log \frac{\mu_s}{1-\mu_s} = 0,$$

or

$$\log \frac{\mu_s}{1-\mu_s} = \theta_s + \sum_{t \in \mathcal{N}(s)} \theta_{\langle s,t \rangle} \mu_t.$$

Note that if

$$\log \frac{y}{1-y} = x,$$

then

$$y = \sigma(x),$$

where

$$\sigma(x) = \frac{1}{1+e^{-x}}.$$

The algorithm then proceeds by setting

$$\mu_s^{(j+1)} = \sigma\left(\theta_s + \sum_{t \in \mathcal{N}(s)} \theta_{\langle s,t \rangle} \mu_t^{(j)}\right).$$

As discussed in [72] (page 222), the lower bound thus computed seems to provide a good approximation to the true value.

Notes

The material for Chapter 21 is taken mostly from Wainright and Jordan [142]. It is developed further in [72]. Possible improvements to the lower bound are proposed by Humphreys and Titterington in [66]. The book by Barndorff - Nielsen [3] is the standard treatise of exponential families and the required convex analysis.

21.8 Exercises: Graphical Models and Exponential Families

1. Prove lemma 21.5.

2. Let (X_1, X_2, X_3) be random variables, with joint probability function

$$p(x_1, x_2, x_3|\eta) = \frac{n!}{x_1!x_2!x_3!} \prod_{j=1}^{3} p_i^{x_i}, \quad x_1 + x_2 + x_3 = n,$$

where $p_1 = \eta^2$, $p_2 = 2\eta(1-\eta)$ and $p_3 = (1-\eta)^2$ and $0 \leq \eta \leq 1$.

   (a) Is this an exponential family?

   (b) Obtain the minimal sufficient statistic for θ.

   (c) Compute the mean parameter in terms of η.

   (d) Compute the Fenchel Legendre Conjugate of the log partition function.

   (e) Prove that the Kullback Leibler Divergence is given by

$$D(\theta_1|\theta_2) = A(\theta_2) - A(\theta_1) - \langle \mu_1, \theta_2 - \theta_1 \rangle.$$

$$\tilde{D}(\mu_1|\theta_2) = A(\theta_2) + A^*(\mu_1) - \langle \mu_1, \theta_2 \rangle$$

$$\tilde{\tilde{D}}(\mu_1|\mu_2) = A^*(\mu_1) - A^*(\mu_2) - \langle \theta_2, \mu_1 - \mu_2 \rangle.$$

   State the definitions of the terms used in this equation.

   (f) Compute the primal form of the Kullback Leibler divergence $D(\theta_1|\theta_2)$, where θ_1 and θ_2 are the canonical parameters, for this example. Compute the dual form, expressed in terms of the mean parameters.

3. (Mean Field Update) Consider a probability function, given by

$$p_{\underline{X}}(\underline{x}|\underline{\theta}) = \exp\left\{ \sum_{j=1}^{n} \theta(j)x(j) + \sum_{(i,j)\in E} \theta(i,j)x(j) - A(\underline{\theta}) \right\},$$

where $\underline{\theta} = \{(\theta(j))_{j=1}^n, (\theta(j,k)), (j,k) \in E\}$, E denotes the edge set and $\mathbf{x} \in \{0,1\}^n$. Let q denote the probability function

$$q_{\underline{X}}(\underline{x}|\underline{\theta}) = \exp\left\{ \sum_{j=1}^{n} \theta(j)x(j) - A_H(\underline{\theta}) \right\}.$$

Let

$$A_H^*(\underline{\mu}) = \sup_{\underline{\theta}} \{ \langle \underline{\mu}, \underline{\theta} \rangle - A_H(\underline{\theta}).$$

   (a) Prove that

$$A_H^*(\underline{\mu}) = \sum_{j=1}^{n} \{ \mu(j) \log \mu(j) + (1 - \mu(j)) \log \mu(j) \}.$$

(b) Prove that

$$A(\underline{\theta}) \geq \sup_{\underline{\mu}} \left\{ \sum_{j=1}^{n} \theta(j)\mu(j) + \sum_{(j,k)\in E} \theta(j,k)\mu(j)\mu(k) - A_H^*(\underline{\mu}) \right\}.$$

(c) Consider the probability distribution

$$p(x_1, x_2, x_3; \theta) = \exp\left\{ \sum_{j=1}^{3} \theta(j)x_j + \theta(1,2)x_1x_2 + \theta(1,3)x_1x_3 - A(\theta) \right\}.$$

Show that the expression in the previous part is maximised for $(\mu(1), \mu(2), \mu(3))$ that satisfy

$$\log \frac{\mu(1)}{1-\mu(1)} = \theta(1) + \theta(1,2)\mu(2) + \theta(1,3)\mu(3)$$

$$\log \frac{\mu(2)}{1-\mu(2)} = \theta(2) + \theta(1,2)\mu(1)$$

$$\log \frac{\mu(3)}{1-\mu(3)} = \theta(3) + \theta(1,3)\mu(1).$$

(d) Write a Matlab code to compute numerical approximations to the values $(\mu(1), \mu(2), \mu(3))$ that give the naive mean field approximation to the log partition function $A(\theta)$.

Chapter 22

Variational Methods for Parameter Estimation

22.1 Complete Instantiations

Let $\mathbf{x}$ be an $n \times d$ data matrix with n i.i.d. instantiations of $\underline{X} = (X_1, \ldots, X_d)$, which has distribution $\mathbb{P}_\theta$. Here θ is the parameter vector; $\theta \in \Theta \subset \mathbb{R}^p$. If $\{\mathbb{P}_\theta : \theta \in \Theta\}$ is an exponential family, there are several useful techniques that may be used for parameter estimation.

The distributions encountered in Bayesian Networks and, more generally graphical models, are usually multinomial, multivariate Guassian, or Conditional Gaussian. All these are exponential families and lend themselves to the techniques discussed.

22.1.1 Triangulated Graphs

Probability distributions that factorise over a triangulated graph present the most straightforward situation for parameter estimation. Such a probability distribution $\mathbb{P}$ may be written in the form:

$$\mathbb{P} = \frac{\prod_{C \epsilon \mathcal{C}} \mathbb{P}_{C,}}{\prod_{S \epsilon \mathcal{S}} \mathbb{P}_S}$$

where $\mathcal{C}$ and $\mathcal{S}$ denote the collections of cliques and separators.

Consider the multivariate setting. Let $x = (x_1, \ldots, x_d)$ denote an instantiation of the random vector $X = (X_1, \ldots, X_d)$. For each $s \in \{1, \ldots, d\}$, $x_s \in \mathcal{X}_s = (1, \ldots, k_s)$, $x \in \mathcal{X} = \times_{s=1}^d \mathcal{X}_s$, $x_C = \{x_s : s \in C\}$ and $x_S = \{x_s : s \in S\}$. For simplicity, the values taken by the variables are noted by their indices.

This is a multinomial distribution written as an exponential family in an over-complete representation. The mean field parameters are simply:

$$p(C, x_C) := \mathbb{P}_C(x_C) : C \in \mathcal{C}, \quad x_C \in \mathcal{X}_C.$$

$$p(S, x_S) = \mathbb{P}_S(x_S) : S \in \mathcal{S}, \quad x_S \in \mathcal{X}_S.$$

The maximum likelihood estimators are:

$$\widehat{p}_C(x_C) = \frac{1}{n}\sum_{j=1}^{n} \mathbf{1}_{x_C}(x_{j,C}) \qquad \widehat{p}_S(x_S) = \frac{1}{n}\sum_{j=1}^{n} \mathbf{1}_{x_S}(x_{j,S}).$$

where $x_{j,C}$ denotes the value for clique C of instantiation $j : j = 1, \ldots, n$ and similarly for $x_{j,S}$. By construction, these are clearly consistent; if $S \subset C$ then $\widehat{p}(S, x_S) = \sum_{x_C \setminus x_S} \widehat{p}(C, x_C)$.

This may be written as an exponential family with over-complete canonical representation:

$$\mathbb{P}(x) = \exp\left\{\sum_{C\in\mathcal{C}} \psi_C(x_C) - \sum_{S\in\mathcal{S}} \psi_S(x_S)\right\} \tag{22.1}$$

where $\psi_C = \log \mathbb{P}_C$ and $\psi_S = \log \mathbb{P}_S$.

Factorisation along a Chow-Liu Tree Now suppose that the distribution factorises along a Chow-Liu tree. Equation 22.1 may now be written:

$$\mathbb{P}(x) = \exp\left\{\sum_{s=1}^{d} \theta(s; x_s) + \sum_{(s,t)\in E} \theta(s, t; x_s, x_t)\right\}$$

where

$$\theta(s; x_s) = \log \mathbb{P}_{X_s}(x_s), \qquad \theta(s, t; x_s, x_t) = \log \frac{\mathbb{P}_{X_s,X_t}(x_s, x_t)}{\mathbb{P}_{X_s}(x_s)\mathbb{P}_{X_t}(x_t)}$$

and E denotes the edge set of the graph. The maximum likelihood estimates of the parameters are given by:

$$\widehat{\theta}(s; x_s) = \log \widehat{p}_s(x_s) \qquad \widehat{\theta}(s, t; x_s, x_t) = \log \frac{\widehat{p}_{s,t}(x_s, x_t)}{\widehat{p}_s(x_s)\widehat{p}_t(x_t)}.$$

22.1.2 Non-Triangulated Graphs

For non-triangulated graphs, there is no closed form expression for the maximum likelihood estimates. Recall that a probability distribution factorises along an undirected graph if $\mathbb{P}$ may be written as:

$$\mathbb{P}(x) = \prod_{C\in\mathcal{C}} \phi_C(x_C)$$

where $\mathcal{C}$ denotes the collection of cliques of the undirected graph. The probability distribution may be written as:

$$\mathbb{P}(x) = \exp\left\{\sum_{C\in\mathcal{C}} \theta_C(x_C) - A(\theta)\right\}$$

where $A(\theta)$ is the log partition function and $\theta = \{\theta_C(x_C) : C \in \mathcal{C}, x_C \in \mathcal{X}_C\}$.

An *iterative proportion fitting* (IPF) method may be used, since the log partition function is convex. Let

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{j=1}^{n}\left(\sum_{C\in\mathcal{C}} \theta_C(x_{j,C}) - A(\theta)\right) = \sum_{C\in\mathcal{C}}\sum_{x_C\in\mathcal{X}_C} \theta_C(x_C)\widehat{p}_C(x_C) - A(\theta) \tag{22.2}$$

then

$$\frac{\partial}{\partial\theta_C(x_C)}\mathcal{L}(\theta) = \widehat{p}_C(x_C) - \frac{\partial}{\partial\theta_C(x_C)}A(\theta) = \widehat{p}_C(x_C) - p_C(x_C). \tag{22.3}$$

Here we've used the fact that we have an exponential family in its canonical form so that $\frac{\partial}{\partial\theta_C(x_C)}A(\theta) = \mathbb{E}_\theta\left[\widehat{p}_C(x_C)\right] = p_C(x_C)$ (from (22.2), $\widehat{p}_C(x_C)$ is the sufficient statistic).

The aim is to find the MLE (where $\frac{\partial}{\partial\theta_C(x_C)}\mathcal{L}(\theta) = 0$). The iterative proportional fitting scheme proceeds as follows:

At iterations $t = 0, 1, 2, \ldots$ let $\theta^{(t)}$denote the current vector of parameter estimates.

- Choose a clique $C = C(t)$ and compute the local marginal distribution

$$p_C^{(t)}(x_C) := \mathbb{P}_{\theta^{(t)}}(X_C = x_C) \qquad \forall x_C \in \mathcal{X}_C.$$

- Update the canonical parameter vector:

$$\theta_C^{(t+1)}(x_C) = \begin{cases} \theta_C^{(t)}(x_C) + \log\frac{\widehat{p}_C(x_C)}{p_C^{(t)}(x_C)} & C = C(t), \quad x_C \in \mathcal{X}_C \\ \theta_C^{(t)}(x_C) & \text{otherwise} \end{cases}$$

The sequence satisfies two important properties, which are stated as a proposition.

Proposition 22.1. *1.*

$$A(\theta^{(t+1)}) = A(\theta^{(t)}).$$

2. For each t, Equation (22.3) *holds.*

Proof Suppose $C(t) = C'$. Defining $\theta^{(t+1)}$ in this way gives:

$$\begin{aligned} A(\theta^{(t+1)}) &= \log\sum_x \exp\left\{\sum_{C\in\mathcal{C}}\theta_C^{(t+1)}(x_C)\right\} \\ &= \log\sum_x \exp\left\{\sum_{C\in\mathcal{C}}\theta_C^{(t)}(x_C) + \log\frac{\widehat{p}_{C'}(x_{C'})}{p_{C'}^{(t)}(x_{C'})}\right\} \\ &= \log\sum_{x'_C}\frac{\widehat{p}_{C'}(x_{C'})e^{\theta_{C'}^{(t)}(x_{C'})}}{p_{C'}^{(t)}(x_{C'})}\sum_{x\setminus x_C}\exp\left\{\sum_{C\in\mathcal{C}\setminus C'}\theta_C^{(t)}(x_C)\right\} \end{aligned}$$

Now use:

$$p_{C',\theta}(x_{C'}) = e^{-A(\theta)+\theta_{C'}(x_{C'})}\sum_{x\setminus x_C} e^{\sum_{C\neq C'}\theta_C(x_C)}$$

from which

$$A(\theta^{(t+1)}) = A(\theta^{(t)}) + \log \sum_{x_{C'}} \widehat{p}_{C'}(x_{C'}) = A(\theta^{(t)}).$$

This follows because $\sum_{x_{C'}} \widehat{p}_{C'}(x_{C'}) = 1$.

For the second part, the parameter update gives:

$$p_{C,\theta^{(t+1)}}(x_C) = \frac{\widehat{p}_C(x_C)}{p_{C,\theta^{(t)}}(x_C)} p_{C,\theta^{(t)}}(x_C) = \widehat{p}_C(x_C).$$

□

It therefore follows that the IPF algorithm corresponds to a co-ordinate ascent method for maximising the objective (22.2).

The Schedule Convexity of the log-partition function gives, by standard results, that the IPF algorithm converges. The main issue is efficiency. One way is to

1. Triangulate the graph and construct a junction tree. Fix a schedule for the junction tree.
2. For each node of the junction tree, consider the true model (the sub-graph of cliques and separators of the true model) and use the IPF scheme to update each clique of the triangulated graph according to the schedule.

22.2 Partially Observed Models and Expectation-Maximisation

Now suppose that the random vector X is not observed directly, but rather a 'noisy' version Y is observed. The expectation-maximisation (EM) algorithm of Dempster et. al. [36] may be used.

22.2.1 Exact EM Algorithm for Exponential Families

Suppose we have a random vector (X, Y) where X are unobserved and Y are observable. Suppose the probability model is:

$$p_\theta(x, y) = \exp\{\langle \theta, \phi(x, y)\rangle - A(\theta)\} h(x)$$

The conditional distribution of X given Y is:

$$p_\theta(x|y) = \frac{\exp\{\langle \theta, \phi(x, y)\rangle\}}{\int_{\mathcal{X}} \exp\{\langle \theta, \phi(x, y)\rangle\} h(x) dx} =: \exp\{\langle \theta, \phi(x, y)\rangle - A_y(\theta)\}.$$

(this is the definition of A_y). For each fixed y, the conditional distribution of X is therefore an exponential family with log partition function A_y given by:

$$A_y(\theta) = \log \int_{\mathcal{X}} \exp\{\langle \theta, \phi(x, y)\rangle\} h(x) dx.$$

The maximum likelihood estimate $\widehat{\theta}$ is obtained by maximising the log probability of the observed data y. This is referred to as the *incomplete log likelihood* in the EM setting. The incomplete log likelihood is given by the integral:

$$\mathcal{L}(\theta; y) = \log \int_{\mathcal{X}} \exp\left\{\langle \theta, \phi(x, y)\rangle - A(\theta)\right\} h(x) dx = A_y(\theta) - A(\theta). \tag{22.4}$$

For each fixed y, the set $\mathcal{M}_y$ of valid mean parameters is defined as:

$$\mathcal{M}_y = \left\{\nu \in \mathbb{R}^p : \mu = \mathbb{E}_\theta[\phi(X, y)] \quad \theta \in \Theta\right\}.$$

The Fenchel-Legendre conjugate may be used to obtain:

$$A_y(\theta) = \sup_{\mu \in \mathcal{M}_y} \left\{\langle \theta, \mu\rangle - A_y^*(\mu)\right\} \tag{22.5}$$

where the conjugate dual is defined variationally as:

$$A_y^*(\mu) := \sup_{\theta \in \text{dom}(A_y)} \left\{\langle \mu, \theta\rangle - A_y(\theta)\right\}. \tag{22.6}$$

From (22.5), it follows that $A_y(\theta) \geq \langle \mu, \theta\rangle - A_y^*(\mu)$ for *any* μ. A lower bound for the incomplete log likelihood is therefore:

$$\mathcal{L}(\theta, y) = A_y(\theta) - A(\theta) \geq \langle \mu, \theta\rangle - A_y^*(\mu) - A(\theta) := \widetilde{\mathcal{L}}(\mu, \theta).$$

With this set up, the EM algorithm is the coordinate ascent function on this function $\widetilde{L}$ which gives a lower bound. The steps of the EM algorithm are:

$$\begin{cases} \mu_y^{(t+1)} = \arg\max_{\mu \in \mathcal{M}_y} \widetilde{\mathcal{L}}(\mu, \theta^{(t)}) & \textbf{E step} \\ \theta^{(t+1)} = \arg\max_{\theta \in \Theta} \widetilde{\mathcal{L}}(\mu_y^{(t+1)}, \theta) & \textbf{M step} \end{cases} \tag{22.7}$$

Note that if $\widetilde{\mathcal{L}}$ were equal to the log likelihood $\mathcal{L}$, then the **E step** would be equivalent to finding the expectation μ for parameter vector $\theta^{(t)}$, while the **M step** would be precisely the problem of finding the maximum likelihood estimator based on expected sufficient statistics $\mu_y^{(t+1)}$.

The maximisation of the **M step** gives $\mathcal{L}(\theta^{(t+1)}, y) = \widetilde{\mathcal{L}}(\mu_y^{(t+1)}, \theta^{(t+1)})$, while the maximisation of the **E step** gives $\mathcal{L}(\theta^{(t)}, y)$.

Example 22.2 (EM for Conditional Gaussian)**.**

The EM algorithm described can be used to estimate the parameters for a Conditional Gaussian model. For example, consider the straightforward setting where $Y = (Y_1, \ldots, Y_r)$ are Gaussian variables, $Y|\{X = j\} = Y_j$ for $j = 1, \ldots, r$. Suppose that X, the index of the components, is unobserved. The state space for X is $\mathcal{X} = \{1, \ldots, r\}$ and X has a multinomial distribution.

The complete likelihood may be written as:

$$L_\theta(x,y) = \exp\left\{\sum_{j=1}^{r} \mathbf{1}_j(x)\left\{\alpha_j + \gamma_j y + \widetilde{\gamma}_j y^2 - A_j(\gamma_j,\widetilde{\gamma}_j)\right\} - A(\alpha)\right\}$$

where $\theta = (\alpha, \gamma, \widetilde{\gamma})$, the parameter $\alpha \in \mathbb{R}^r$ parametrises the the multinomial distribution over the hidden vector X and the pair $(\gamma_j, \widetilde{\gamma}_j)$ parametrises the Gaussian distribution of the jth mixture component. The log-partition function $A(\gamma_j, \widetilde{\gamma}_j)$ is for the conditionally Gaussian distribution of Y given $X = j$, while $A(\alpha) = \log \sum_{j=1}^{r} \exp\{\alpha_j\}$ normalises the multinomial distribution.

When the complete likelihood is viewed as an exponential family, the sufficient statistics are the collection of triples

$$\Psi_j(x,y) := \{\mathbf{1}_j(x), \mathbf{1}_j(x)y, \mathbf{1}_j(x)y^2\} \qquad j = 1,\ldots,r.$$

Consider a collection of i.i.d. observations $(y_1,\ldots,y_n)$. To each observation there is associated a triplet $(\mu_i, \eta_i, \widetilde{\eta}_i) \in \mathbb{R}^r \times \mathbb{R}^r \times \mathbb{R}^r$ corresponding to expectations of the triplet of sufficient statistics $\Psi_j(X, y_i) : j = 1,\ldots,r$. The conditional distribution has form:

$$p(x|y,\theta) \propto \exp\left\{\sum_{j=1}^{r} \mathbf{1}_{\{j\}}(x)\left(\alpha_j + \gamma_j y + \widetilde{\gamma}_j y^2 - A_j(\gamma_j,\widetilde{\gamma}_j)\right)\right\}.$$

It follows that the mean parameter $p_{j|y} = \mathbb{P}(X = j|Y = y)$ is:

$$p_{j|y} = \frac{\exp\left\{\alpha_j + \gamma_j y + \widetilde{\gamma}_j y^2 - A_j(\gamma_j,\widetilde{\gamma}_j)\right\}}{\sum_{k=1}^{r} \exp\left\{\alpha_k + \gamma_k y + \widetilde{\gamma}_k y^2 - A_j(\gamma_k,\widetilde{\gamma}_k)\right\}}$$

Similarly, the remaining mean parameters are:

$$\eta_{j|y} = p_{j|y} y, \qquad \widetilde{\eta}_{j|y} = p_{j|y} y^2.$$

The computations of the mean parameter $\mu_y = (p_{j|y}, p_{j|y}y, p_{j|y}y^2)$ correspond to the **E step**.

The **M step** requires finding $\theta = (\alpha, \gamma, \widetilde{\gamma})$ to maximise

$$\langle \mu_y^{(t+1)}, \theta\rangle - A(\theta).$$

Some computation shows that this problem takes the form of finding $(\alpha, \gamma, \widetilde{\gamma}) \in \Theta$ which maximises:

$$\sum_{j=1}^{r}\sum_{i=1}^{n}\left(\alpha_j p_{j|y_i} + \gamma_j p_{j|y_i} y_i + \widetilde{\gamma}_j p_{j|y_i} y_i^2 - p_{j|y_i} A_j(\gamma_j,\widetilde{\gamma}_j)\right) - nA(\alpha).$$

The optimisation therefore decouples into separate maximisation problems: one for the α vector parametrising the mixtures and one for each of the $(\gamma_j, \widetilde{\gamma}_j)$ pairs specifying the Gaussian mixtures.

The optimum solution is therefore the value α such that

$$\begin{cases} p_{j|\alpha} = \frac{1}{n}\sum_{i=1}^n p_{j|y_i} \\ \mathbb{E}_{\gamma_j,\widetilde{\gamma}_j}[Y|X=j] = \frac{\sum_{i=1}^n p_{j|y_i} y_i}{\sum_{i=1}^n p_{j|y_i}} \\ \mathbb{E}_{\gamma_j,\widetilde{\gamma}_j}[Y^2|X=j] = \frac{\sum_{i=1}^n p_{j|y_i} y_i^2}{\sum_{i=1}^n p_{j|y_i}}. \end{cases}$$

22.2.2 Mean Field Approximate EM

Suppose that it is not feasible to compute the sufficient statistics. Then the **E step** can be replaced by a **Mean Field E step** where the maximum is taken over a reduced space of models:

$$\mu_y^{(t+1)} = \max_{\mu \in \mathcal{M}_{\text{red}}} \left\{ \langle \mu, \theta^{(t)} \rangle - A_y^*(\mu) \right\}.$$

The **E step** no longer closes the gap between the incomplete log-likelihood $\mathcal{L}$ and the auxiliary function $\widetilde{\mathcal{L}}$ and there are no longer guarantees that the algorithm goes uphill.

22.3 Variational Bayes

Assume that the complete distribution lies in an exponential family

$$p(x, y|\theta) = \exp\left\{ \langle \eta(\theta), \phi(x, y) \rangle - A(\eta(\theta)) \right\}$$

where the function $\eta : \mathbb{R}^p \to \mathbb{R}^p$ gives some additional flexibility. Assume, furthermore, that the prior distribution over Θ also lies in an exponential family and is of *conjugate prior form*:

$$p_{\xi,\lambda}(\theta) = \exp\left\{ \langle \xi, \eta(\theta) \rangle - \lambda A(\eta(\theta)) - B(\xi, \lambda) \right\}. \tag{22.8}$$

This exponential family is specified by the sufficient statistics: $\{\eta(\theta), -A(\eta(\theta))\} \in \mathbb{R}^d \times \mathbb{R}$. The log partition function $B(\xi, \lambda)$ is defined in the usual way:

$$B(\xi, \lambda) := \log \int_\Theta \exp\left\{ \langle \xi, \eta(\theta) \rangle - \lambda A(\eta(\theta)) \right\} d\theta.$$

Now consider the problem of computing the *marginal likelihood* $p_{\xi^*,\lambda^*}(y)$ where y is an observed datum and (ξ^*, λ^*) are fixed values of the hyperparameters. This requires averaging over both x (the unobserved variables) and the parameter space Θ.

$$\log p_{\xi^*,\lambda^*}(y) = \log \int \left(\int p(x, y|\theta) dx \right) p_{\xi^*,\lambda^*}(\theta) d\theta = \log \int p_{\xi^*,\lambda^*}(\theta) p(y|\theta) d\theta.$$

A simple application of Jensen's inequality gives:

$$\log p_{\xi^*,\lambda^*}(y) = \log \mathbb{E}_{\xi,\lambda}\left[\frac{p_{\xi^*,\lambda^*}(\Theta)}{p_{\xi,\lambda}(\Theta)} p(y|\Theta) \right] \geq \mathbb{E}_{\xi,\lambda}\left[\log p(y, \Theta) \right] + \mathbb{E}_{\xi,\lambda}\left[\log \frac{p_{\xi^*,\lambda^*}(\Theta)}{p_{\xi,\lambda}(\Theta)} \right]$$

with equality for $(\xi, \lambda) = (\xi^*, \lambda^*)$. From Equation (22.4),

$$\log p(y|\Theta) = A_y(\eta(\Theta)) - A(\eta(\Theta))$$

so that

$$p_{\xi^*,\lambda^*}(y) \geq \mathbb{E}_{\xi,\lambda}\left[A_y(\eta(\Theta)) - A(\eta(\Theta))\right] + \mathbb{E}_{\xi,\lambda}\left[\log \frac{p_{\xi^*,\lambda^*}(\Theta)}{p_{\xi,\lambda}(\Theta)}\right] \tag{22.9}$$

where A_y is the log partition function of the conditional density $p(x|y,\theta)$.

For each fixed y, the set $\mathcal{M}_y$ is the set of mean parameters of the form $\mu = \mathbb{E}[\phi(X,y)]$.

The *variational Bayes* algorithm is based on optimising this lower bound using only distributions of product form over $(\Theta, \mathcal{X})$. Such an optimisation is referred to as 'free form'. Using (22.6),

$$A_y(\eta) \geq \langle \mu, \eta \rangle - A_y^*(\mu)$$

for any μ and hence the right hand side of Equation (22.9) has lower bound:

$$\mathbb{E}_{\xi,\lambda}\left[\langle \mu(\Theta), \eta(\Theta)\rangle - A_y^*(\mu(\Theta)) - A(\eta(\Theta))\right] + \mathbb{E}_{\xi,\lambda}\left[\log \frac{p_{\xi^*,\lambda^*}(\Theta)}{p_{\xi,\lambda}(\Theta)}\right]. \tag{22.10}$$

for any function $\mu(\theta)$. The expression in (22.10), restricting to μ constant, is:

$$\left(\langle \mu, \overline{\eta}\rangle - A_y^*(\mu) - \overline{A}\right) + \mathbb{E}_{\xi,\lambda}\left[\log \frac{p_{\xi^*,\lambda^*}(\Theta)}{p_{\xi,\lambda}(\Theta)}\right], \tag{22.11}$$

where $\overline{\eta} = \mathbb{E}_{\xi,\lambda}\left[\eta(\Theta)\right]$ and $\overline{A} = \mathbb{E}_{\xi,\lambda}\left[A(\Theta)\right]$. Using (22.8),

$$\log \frac{p_{\xi^*,\lambda^*}(\theta)}{p_{\xi,\lambda}(\theta)} = \langle \xi^* - \xi, \eta(\theta)\rangle - (\lambda^* - \lambda)A(\eta(\theta)) - (B(\xi^*,\lambda^*) - B(\xi,\lambda))$$

so that:

$$\mathbb{E}_{\xi,\lambda}\left[\log \frac{p_{\xi^*,\lambda^*}(\Theta)}{p_{\xi,\lambda}(\Theta)}\right] = \langle \overline{\eta}, \xi^* - \xi\rangle + \langle -\overline{A}, \lambda^* - \lambda\rangle - B(\xi^*,\lambda^*) + B(\xi,\lambda).$$

Now recall the definition of B^* (Fenchel Legendre conjugate of B):

$$B^*(\mu_1,\mu_2) = \sup_{\xi,\lambda}\left\{\mu_1\xi + \mu_2\lambda - B(\xi,\lambda)\right\}.$$

Then, since $\{\eta(\theta), -A(\eta(\theta))$ are the sufficient statistics, therefore $\frac{\partial B}{\partial \xi} = \overline{\eta}$ and $\frac{\partial B}{\partial \lambda} = -\overline{A}$, so that:

$$B^*(\overline{\eta},\overline{A}) = \langle \overline{\eta}, \xi\rangle + \langle -\overline{A}, \lambda\rangle - B(\xi,\lambda).$$

Hence the decoupled optimisation problem is equivalent to maximising:

$$\langle \mu + \xi^*, \overline{\eta}\rangle - A_y^*(\mu) + \langle \lambda^* + 1, -\overline{A}\rangle - B^*(\overline{\eta},\overline{A})$$

over $\mu \in \mathcal{M}_y$ and $(\overline{\eta},\overline{A}) \in \mathrm{dom}(B)$.

A coordinate ascent amounts to first maximising over μ and then maximising over the mean parameters $(\overline{\eta},\overline{A})$. This generates a sequence of iterates $(\mu^{(t)}, \overline{\eta}^{(t)}, \overline{A}^{(t)})$. The updates are:

$$\begin{cases} \mu^{(t+1)} = \arg\max_{\mu \in \mathcal{M}_y} \left\{ \langle \mu, \overline{\eta}^{(t)} \rangle - A_y^*(\mu) \right\} & \textbf{VB-E Step} \\ (\overline{\eta}^{(t+1)}, \overline{A}^{(t+1)}) = \arg\max_{(\overline{\eta}, \overline{A})} \left\{ \langle \mu^{(t+1)} + \xi^*, \overline{\eta} \rangle - (1 + \lambda^* \overline{A} - B^*(\overline{\eta}, \overline{A}) \right\} & \textbf{VB-M Step} \end{cases} \tag{22.12}$$

These coordinate-wise optimisations have explicit solutions; the explicit solution of the **VB-E Step** is:

$$\mu^{(t+1)} = \mathbb{E}_{\overline{\eta}^{(t)}} \left[\phi(X, y) \right].$$

Similarly, setting

$$(\xi^{(t+1)}, \lambda^{(t+1)}) = (\xi^* + \mu^{(t+1)}, \lambda^* + 1)$$

then

$$\overline{\eta}^{(t+1)} = \mathbb{E}_{(\xi^{(t+1)}, \lambda^{(t+1)})} \left[\eta(\Theta) \right].$$

Literature Cited

[1] S.M. Aji and R.J McEliece [2000] *The Generalised Distributive Law* IEEE Transactions on Information Theory vol. 46 pp. 325 - 343

[2] S.A. Andersson, D. Madigan, M.D. Perlman and C.M. Triggs [1997] *A graphical characterisation of lattice conditional independence models* Annals of Mathematics and Artificial Intelligence vol. 21 pp. 27 - 50

[3] O. Barndorff - Nielsen [1978]*Information and Exponential Families in Statistical Theory* Wiley

[4] Barros, B. [2012] *Incremental Learning Algorithms for Financial Data Modelling* Master's Thesis, Linköping University, Department of Mathematics LiTH-MAT-INT-A–2012/01–SE

[5] Beeri, C.; Fagin, R.; Maier, D.; Yannakakis, M. [1983] *On the desirability of acyclic database schemes* J. Assoc. Comput. Mach. **30** pp 479 - 513.

[6] [2005] Braunstein, A.; Mézard, M.; Zecchina, R. [2005] *An Algorithm for Satisfiability* Random Structures and Algorithms, vol. 27, no. 2, pp. 201 - 226
`http://dx.doi.org/10.1002/rsa.20057`

[7] Braunstein, A.; Zecchina, R. [2004] *Survey Propagation as Local Equilibrium Equations* Journal of Statistical Mechanics: Theory and Experiment vol. 2004, no. 6 pp. P06007
`https://stacks.iop.org/1742-5468/2004/P06007`

[8] Brockwell, P.J.; Davis, R.A. [1991] *Time Series: Theory and Methods (second edition)* Springer

[9] F. Bromberg, D. Margaritis [2009] *Improving the reliability of causal discovery from small data sets using argumentation* Journal of Machine Learning Research vol. 10 pp. 301 - 340

[10] D.T. Brown [1959] *A Note on Approximations to Discrete Probability Distributions* Information and Control vol. 2 pp. 386 - 392

[11] Bulashevska, S.; Eils, R. [2005] *Inferring genetic regulatory logic from expression data* Bioinformatics vol 21 no 11 pp 2706 - 2713

[12] E. Castillo, J.M. Gutiérrez, A.S. Hadi [1996] *A New Method for Efficient Symbolic Propagation in Discrete Bayesian Networks* Networks vol. 28 no. 1 pp. 31 - 43

[13] E. Castillo, J.M. Gutiérrez, A.S. Hadi [1997] *Sensitivity Analysis in Discrete Bayesian Networks* IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans vol. 27 no. 4

[14] Cayley, A. [1853] *Note on a Question in the Theory of Probabilities* The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science vol. VI. - fourth series July - December, 1853, Taylor and Francis. p. 259

[15] Cayley, A. [1854] *On the theory of groups as depending on the symbolic equation* $\theta^n = 1$ Phil. Mag. vol. 7 no. 4 pp 40 - 47

[16] Cayley, A. [1858] *A Memoir on the Theory of Matrices* Phil. Trans. of the Royal Soc. of London, vol 148 p. 24

[17] Cayley, A. [1869] *A Memoir on Cubic Surfaces* Philosophical Transactions of the Royal Society of London (The Royal Society) vol 159 pp 231–326

[18] Cayley, A. [1878] *Desiderata and suggestions: No. 2. The Theory of groups: graphical representation* Amer. J. Math. vol. 1 no. 2 174–176

[19] Cayley, A. [1889] *A Theorem on Trees* Quarterly Journal of Mathematics vol 23 pp 276-378

[20] H. Chan, A. Darwiche [2005] *A Distance Measure for Bounding Probabilistic Belief Change* International Journal of Approximate Reasoning vol. 38 pp. 149 - 174

[21] H. Chan, A. Darwiche [2002] *When do Numbers Really Matter?* Journal of Artificial Intelligence Research vol. 17 pp. 265 - 287

[22] H. Chan, A. Darwiche [2005] *On the Revision of Probabilistic Beliefs Using Uncertain Evidence* Artificial Intelligence vol. 163 pp. 67-90

[23] Cheng, J.; Greiner, R.; Kelly, J.; Bell, D. A.; Liu, W. [2002] *Learning Bayesian networks from data: An information-theory based approach* Artificial Intelligence vol 137 pp 43 - 90.

[24] D.M. Chickering [1995] *A transformational characterization of Bayesian network structures* In Hanks, S. and Besnard, P., editors, Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, pages 87 - 98 Morgan Kaufmann.

[25] Chickering, D. M. [2002] *Optimal structure identification with greedy search* Journal of Machine Learning Research, 507–554.

[26] D.M. Chickering, D. Heckerman, C. Meek [2004] *Large Sample Learning of Bayesian Networks is NP - Hard* Journal of Machine Learning Research vol. 5 pp. 1287 - 1330

[27] Chiquet, J.; Smith, A.; Grasseau, G.; Matias, C.; Ambroise, C. [2009] *SIMoNe: Statistical Inference for Modular Networks* Bioinformatics 25(3):417–418

[28] C.K. Chow and C.N. Liu [1968] *Approximating Discrete Probability Distributions with Dependence Trees* IEEE Transactions on Information Theory, vol. IT - 14 no. 3

[29] Claeskens G, Hjort NL [2008] *Model selection and model averaging* Cambridge University Press, Cambridge

[30] G.F. Cooper [1990] *The Computational Complexity of Probabilistic Inference using Bayesian Belief Networks* Artificial Intelligence vol. 42 pp. 393 - 405

[31] G.F. Cooper and E. Herskovitz [1992] *A Bayesian Method for the Induction of Probabilistic Networks from Data* Machine Learning vol. 9 pp. 309 - 347

[32] R.G. Cowell, A.P. David, S.L. Lauritzen and D.J. Spiegelhalter [1999] *Probabilistic Networks and Expert Systems* Springer, New York

[33] A.P. Dawid [1992] *Applications of a General Propagation Algorithm for Probabilistic Expert Systems* Statistics and Computing vol. 2 pp. 25 - 36

[34] Dean, T.; Kanazawa, K. [1989] *A Model for Reasoning about Persistence and Causation* Computational Intelligence vol. 5, no. 2, pp.142 - 150.

[35] W.E. Deming and F.F. Stephan [1940] *On a Least Squares Adjustment of a Sampled Frequency Table when the Expected Marginal Totals are Known* Annals of Mathematical Statistics vol. 11 pp. 427 - 444

[36] Dempster, P.; Laird, N.M.; Rubin, D.B. [1977] *Maximum Likelihood from Incomplete Data via the EM Algorithm* Journal of the Royal Statistical Society, Series B, vol. 39, pp. 1 - 38

[37] P. Diaconis and S.L. Zabell [1982] *Updating Subjective Probability* Journal of the American Statistical Association vol. 77 (380) pp. 822 - 830

[38] J.M. Dickey [1983] *Multiple Hypergeometric Functions: Probabilistic Interpretations and Statistical Uses* Journal of the American Statistical Association, 1983, vol. 78 (383) pp. 628 - 637

[39] Drton, M.; Sturmfels, B.; Sullivant, S. [2009] *Lectures on algebraic statistics* Birkhäuser

[40] D. Edwards [2000] *Introduction to Graphical Modelling* chapter 9: Causal Inference. Springer

[41] A. Fast [2010] *Learning the structure of Bayesian networks with constraint satisfaction* Ph.D. thesis, Graduate School of the University of Massachusetts Amherst, Department of Computer Science

[42] Fisher, R.A. [1924] *The Distribution of the Partial Correlation Coefficient* Metron vol. 3 no. 3-4 pp. 329 - 332.

[43] D. Freedman and P. Humphreys [1999]*Are there Algorithms that Discover Causal Structure?* Synthese vol. 121 pp. 29 - 54

[44] Friedman, J.; Hastie, T.; Tibshirani, R. [2010]*Regularisation Paths for Generalised Linear Models via Coordinate Descent* J Stat Softw 33(1):1–22

[45] Friedman, N.; Nachman, I.; Pe'er, D. [1999] *Learning Bayesian network structure from massive datasets: the 'sparse candidate' algorithm* Proc. Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI '99) pp 196 - 205

[46] Friedman, N.; Linial, M.; Nachman, I.; Pe'er, D. [2000] *Using Bayesian Networks to Analyse Expression Data* Journal of Computational Biology **7** no 3/4 pp 601 - 620

[47] Friedman, N.; Koller, D. [2003]*Being Bayesian About Network Structure: A Bayesian Approach to Structure Discovery in Bayesian Networks* Machine Learning, vol. 50 pp. 95 - 125

[48] Friedman, N. [2004] *Inferring Cellular Networks Using Probabilistic Graphical Models* Science Vol 303 no 5659 pp 799-805 DOI: 10.1126/science.1094068

[49] Gamerman, D.; Lopes, H.F. [2006] *Markov chain Monte Carlo: stochastic simulation for Bayesian inference* Chapman and Hall CRC

[50] Garcia, L.D.; Stillman, M.; Sturmfels, B. [2005] *Algebraic geometry of Bayesian networks* Journal of Symbolic Computation **39** pp 331–355

[51] D. Geiger, T. Verma and J. Pearl [1990] *Identifying Independence in Bayesian Networks* Networks vol. 20 pp. 507 - 534.

[52] Gentry J, Long L, Gentleman R, Seth, Hahne F, Sarkar D, Hansen K [2012]*Rgraphviz: provides plotting capabilities for R graph objects.* R package version 1.32.0

[53] Giudici, P.; Castelo, R. [2003]*Improving Markov chain Monte Carlo Model Search for Data Mining* Machine Learning vol. 50 pp. 127 - 158

[54] Goeman, J.J. [2012]*penalized R package* R package version 0.9-41

[55] M.C. Golumbic [2004] *Algorithmic Graph Theory and Perfect Graphs* Elsevier

[56] Greenland, S.; Pearl, J.; Robins, J.M. [1999] *Causal diagrams for epidemiologic research* Epidemiology pp 37 - 48

[57] Greenland, S.; Lash, T. [2008] *Bias Analysis* in: Modern Epidemiology, 3rd ed., Ed. K Rothman, S. Greenland and T. Lash, pp 345 - 380. Philadelphia: Lippincott, Williams and Wilkins.

[58] Grzegorczyk, M.; Husmeier, D. [2008]*Improving the Structure MCMC Sampler for Bayesian Networks by introducing a New Edge Reversal Move* Mach. Learn vol. 71 pp. 265 - 305

[59] Hartmanis, J. [1959] *Application of some Basic Inequalities for Entropy* Information and Control vol. 2 pp 199 - 213

[60] Hastie T.; Efron, B. [2012] *lars: least angle regression, lasso and forward stagewise* R package version 1.1

[61] D. Heckerman [1998] *A Tutorial on Learning with Bayesian Networks* Report # MSR-TR-95-06 Microsoft Research, Redmont, Washington
`http://research.microsoft.com/~ heckerman/`

[62] D. Heckerman, D. Geiger and D.M. Chickering [1995] *Learning Bayesian Networks: The Combination of Knowledge and Statistical Data* Machine Learning vol. 20 pp. 197 - 243

[63] Heskes, T.; Albers, C.; Kappen, H.J. [2003] *Approximate Inference and Constrained Optimisation* in Proc. of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-03) San Fransisco, CA: Morgan Kaufmann Publishers, pp. 313 - 320

[64] Huang, Y.; Valtorta, M. [2006] *Pearl's Calculus of Intervention is Complete* Proceedings of the 22nd Conference on Uncertainty in Artifical Intelligence pp. 217-224 UAI Press

[65] Huang, Y.; Valtorta, M. [2008] *On the Completeness of an Identifiability Algorithm for Semi-Markov Models* Ann Math Artif Intell vol. 54 pp. 363 - 408

[66] K. Humphreys and D.M. Titterington [2000] *Improving the Mean - Field Approximation in Belief Networks using Bahadur's Reparameterisation of the Multivariate Binary Distribution* Neural Processing Letters vol. 12 pp. 183 - 197

[67] Højsgaard, S. [2012] *Graphical Independence Networks with the gRain Package for R* Journal of Statistical Software, vol. 46 no.10 pp. 1-26.
`http://www.jstatsoft.org/v46/i10/`

[68] Højsgaard,S.; Edwards, D.; Lauritzen, S. [2012] *Graphical Models with R* Springer

[69] Ide, J.S.; Cozman, F.G. [2002] *Random generation of Bayesian networks* In: SBIA '02: Proceedings of the 16th Brazilian symposium on artificial intelligence, Springer, pp 366–375

[70] Jaynes, E.T. [2003] *Probability Theory. The Logic of Science* Cambridge University Press

[71] R.C. Jeffrey [1965] *The Logic of Decision* McGraw - Hill, New York (second ed., University of Chicago Press, Chicago, 1983; Paperback correction, 1990)

[72] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, L.K. Saul [1999] *An Introduction to Variational Methods for Graphical Models* Machine Learning vol. 37 pp. 183 - 233

[73] Kellerer, H.G. [1991] *Indecomposable marginal problems* Advances in probability distributions with given marginals: beyond the copulas, Springer Verlag, Berlin, pp 139 - 149

[74] H. Kiiveri, T.P. Speed, J.B. Carlin [1984] *Recursive Causal Models* J. Austral. Math. Soc. (series A) vol. 36 pp. 30 - 52

[75] M. Koivisto and K. Sood [2004] *Exact Bayesian Structure Discovery in Bayesian Networks* Journal of Machine Learning Research vol. 5 pp. 549 - 573

[76] Kuipers, J.; Moffa, G. [2015] *Partition MCMC for Inference on Acyclic Digraphs* preprint: `arxiv:1504.05006v1`

[77] Kuroki, M.; Pearl, J. [2014] *Measurement Bias and Effect Restoration in Causal Inference* Biometrika vol. 101 no. 2 pp. 423 - 437

[78] F.R. Ksischang , B.J. Frey, H-A. Loeliger [2001] *Factor Graphs and the Sum Product Algorithm* IEEE Transactions on Information Theory vol. 47 February, pp. 498 - 519

[79] E. Lazkano, B. Sierra, A. Astigarraga, J.M. Martínez - Otzeta [2007] *On the use of Bayesian Networks to Develop Behaviours for Mobile Robots* Robots and Autonomous Systems vol. 55 pp. 253 - 265

[80] S.L. Lauritzen, D.J. Spiegelhalter [1988]*Local Computations of Probabilities on Graphical Structures and their Applications to Expert Systems* Journal of the Royal Statistical Society B (Methodological) vol. 50 no. 2 pp. 157 - 224

[81] S.L. Lauritzen [1992]*Propagation of Probabilities, Means and Variances in Mixed Graphical Association Models* Journal of the Americal Statistical Association vol. 78 no. 420 pp. 1098 - 1108

[82] S. Lauritzen [2001] *Causal Inference from Graphical Models* in Complex Stochastic Systems pp. 63 - 108, Chapman and Hall

[83] S. Lauritzen and D. Spiegelhalter [1988] *Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems (with discussion)* Journal of the Royal Statistical Society, Series B, vol. 50, pp. 157 - 224

[84] S. Lauritzen [1992] *Propagation of Probabilities, Means and Variances in Mixed Graphical Association Models* Journal of the American Statistical Association vol. 87 no. 420 pp. 1098 - 1108

[85] Lewis II, P.M. [1959]*Approximating Probability Distributions to Reduce Storage Requirements* Information and Control vol. 2 pp 214 - 225

[86] Ma, Z.; X, Xie; Geng, Z. [2008] *Structure Learning of Chain Graphs via Decomposition* J. Mach. Learn Res 9 pp 2847 - 2880

[87] Madgison, J. [1977]*Toward a Causal Model Approach for Adjusting for Pre-Existing Differences in the Non-Equivalent Control Group Situation: A General Alternative to ANCOVA* Eval. Rev. vol. 1 pp. 399 - 420.

[88] D. Madigan, S.A. Andersson, M.D. Perlman, C.T. Volinsky [1996] *Bayesian Model Averaging and Model Selection for Markov Equivalence Classes of Acyclic Digraphs* Communications In Statistics: Theory and Methods vol. 25, no. 11 pp. 2493-2519

[89] D. Madigan and J. York [1995] *Bayesian Graphical Models for Discrete Data* International Statistical Review vol. 63 pp. 215 - 232

[90] Mardia KV, Kent JT, Bibby JM (1979) Multivariate analysis. Academic, London

[91] Markowetz, F.; Spang, R. [2007]*Inferring Cellular Networks - A Review* BMC bioinformatics vol. 8 (Suppl 6) : S5

[92] McEliece, R.J.; MacKay, D.J.C.; Cheng, J.-F. [1998] *Turbo Decoding as an Instance of Pearl's 'Belief Propagation' Algorithm* IEEE J. Select. Areas Commun. vol. 16 pp. 140 - 152

[93] C. Meek [1995] *Causal inference and causal explanation with background knowledge* Proceedings of the Eleventh Conference on Uncertainty in Artificial Intellegence pp 403 - 410

[94] Minka, T. [2001] *Expectation Propagation for Approximate Bayesian Inference* in Proc. of the 17th Annual Conf. on Uncertainty in Artificial Intelligence (UAI-01) San Fransisco, CA: Morgan Kaufmann Publishers, pp. 362 - 369

[95] Mooij, J.M.; Kappen, H.J. [2007] *Sufficient Conditions for Convergence of the Sum-Product Algorithm* IEEE Transactions on Information Theory, vol. 53 no. 12, pp 4422 - 4437

[96] Moore, A.; Wong, W-K. [2003] *Optimal Reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning* Proceedings of the Twentieth International Conference on Machine Learning (ICML - 2003), Washington DC

[97] Murphy, K.P. [2002] *Dynamic Bayesian Networks: Representation, Inference and Learning* University of California, Berkeley, Ph.D. thesis (Computer Science)

[98] R. Neal [1992] *Correctionist Learning of Belief Networks* Artificial Intelligence vol. 56 pp. 71 - 113

[99] R.E. Neapolitan [2004] *Learning Bayesian Networks* Pearson Prentice Hall, Upper Saddle River, New Jersey.

[100] Nelson, E. [1987] *Radically Elementary Probability Theory* Princeton University Press

[101] Noorshams, N; Wainwright, M.J. [2013] *Stochastic Belief Propagation: A Low-Complexity Alternative to the Sum-Product Algorithm* IEEE Transactions on Information Theory vol. 59 no. 4 pp. 1981 - 2000

[102] Opper, M.; Winder, O. [2005] *Expectation Consistent Approximate Inference* Journal of Machine Learning Research vol. 6 pp. 2177 - 2004

[103] J. Pearl [1982] *Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach* AAAI - 82 Proceedings pp. 133 - 136

[104] J. Pearl [1987] *Evidential Reasoning Using Stochastic Simulation of Causal Models* Artifical Intelligence, vol. 32, pp. 245-257.

[105] Pearl, J. [1988] *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* Morgan Kaufmann, San Mateo, CA.

[106] J. Pearl [1990] *Probabilistic Reasoning in Intelligent Systems* 2nd revised printing, Morgan and Kaufman Publishers Inc., San Fransisco

[107] J. Pearl [1995] *Causal Diagrams for Empirical Research* Biometrika vol. 82 pp. 669 - 710

[108] J. Pearl [1995] *Causal Inference from Indirect Experiments* Artificial Intelligence in Medicine vol. 7 pp. 561 - 582

[109] J. Pearl [2000] *Causality* Cambridge University Press

[110] Pearl, J.; Dechter, D. [1996] *Identifying Independencies in Causal Graphs with Feedback* Proceedings of the Twelfth International Conference in Uncertainty in Artificial Intelligence (UAI'96) pp. 420 - 426, Morgan Kaufmann

[111] J. Pearl, D. Geiger and T. Verma [1989] *Conditional Independence and its Representations* Kybernetica vol. 25 no. 2 pp. 33 - 44

[112] J. Pearl and T. Verma [1987] *The Logic of Representing Dependencies by Directed Acyclic Graphs* Proceedings of the AAAI, Seattle, Washington pp. 374 - 379

[113] Pearl, J. [2010] *On Measurement Bias in Causal Inference* In: Proc. 20th Cof. Uncertainty in Artificial Intelligence, pp. 425 - 432, Catalina Island.

[114] J.M. Pẽna [2007] *Approximate Counting of Graphical Models Via MCMC* Proceedings of the 11th Conference in Artificial Intelligence pp. 352- 359

[115] Pistone, G.; Riccomagno, E.; Wynn, H. [2001] *Algebraic Statistics: Computational Commutative Algebra in Statistics* Chapman and Hall, Boca Raton.

[116] M. Ramoni and P. Sebastiani [1997] *Parameter Estimation in Bayesian Networks from Incomplete Databases* Knowledge Media Institute, KMI-TR-57

[117] Robins, J.M.; Scheines, R.; Spirtes, P.; Wasserman, L. [2003] *Uniform consistency in causal inference* Biometrika vol 90 no 3 pp 491- 515

[118] R.W. Robinson [1977] *Counting Unlabelled Acyclic Digraphs* Springer Lecture Notes in Mathematics: Combinatorial Mathematics V, C.H.C. Little (ed.) pp. 28 - 43.

[119] Rosenbaum, P.; Rubin, D. [1983] *The Central Role of Propensity Score in Observational Studies for Causal Effects* Biometrika vol. 70 pp. 41-55

[120] Sadeghi, K.; Lauritzen, S, [2012] *Markov Properties for Mixed Graphs* submitted to Bernoulli, available on arxiv
http://arxiv.org/pdf/1109.5909v2.pdf

[121] J. L. Savage [1966]*Foundations of Statistics* John Wiley and Sons, New York.

[122] R.D. Schachter [1998] *Bayes Ball: The Rational Pass Time for Determining Irrelevance and Requisite Information in Belief Networks and Influence Diagrams* Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (ed G.F. Cooper and S. Moral) pp. 480 - 487, Morgan Kaufmann, San Fransisco, CA.

[123] Schmidt, M.; Niculescu-Mizil, A.; Murphy, K. [2007] *Learning graphical model structure using l1-regularization paths* Proceedings of the National Conference on Artificial Intelligence vol 22 no 2 pp 12- 78

[124] Shpister, I.; Pearl, J. [2006]*Identification of Joint Interventional Distributions in Recursive Semi-Markovian Causal Models* In: Proceedings of the Twenty-First National Conference on Artificial Intelligence. Menlo Park, CA: AAAI Press. pp 1219 - 1226

[125] Shpister, I.; Pearl, J. [2008] *Complete Identification Methods for Causal Hierarchy* Journal of Machine Learning Research vol. 9 pp. 1941 - 1979

[126] Spirtes, P.; Glymour, C.; Scheines, R. [1993] *Causation, Prediction and Search* Lecture Notes in Statistics no. 81 Springer-Verlag New York

[127] P. Spirtes, C. Glymour and R. Scheines [2000] *Causation, Prediction and Search* second edition, The MIT press.

[128] Strotz, R.H.; Wold, H.O.A. [1960] *Recursive versus Nonrecursive Systems: An Attempt at Synthesis* Econometrica vol. 28 pp. 417-427

[129] M. Studený [2005] *Probabilistic Conditional Independence Structures* Springer Verlag.

[130] Sturmfels, B. [2002] *Solving Systems of Polynomial Equations* In: CBMS Lectures Series, American Mathematical Society.

[131] Sun, J.; Zheng, N.-N.; Shum, H.-Y. [2003] *Stereo Matching using Belief Propagation* IEEE Transactions on Pattern Analysis and Machine Intelligence vol. 25 no. 7 pp. 787 - 800

[132] Tanaka, K. [2002] *Statistical-Mechanical Approach to Image Processing* Journal of Physics A: Mathematical and General vol. 35 no. 37 pp. R81 - R150
http://stacks.iop.org/0305-4470/35/R81

[133] Tatikonda, S.C. [2003] *Convergence of the Sum-Product Algorithm* in Proceedings 2003 IEEE Information Theory Workshop

[134] Tian, J.; Pearl, J. [2002] *A General Identification Condition for Causal Effects* Proceedings of the Eighteenth National Conference on Artifical Intelligence, AAAI Press, Menlo Park California pp. 567 - 573.

[135] Tian, J.; Pearl, J. [2002]*On the Testable Implications of Causal Models with Hidden Variables* in Proceedings of UAI-02, pp. 519 - 527

[136] Tian, J.; Shpitser, I. [2010] *On Identifying Causal Effects* In: Dechter, R.; Geffner, H.; Halpern, J. eds., *Heuristics, Probability and Causality: A Tribpute to Judea Pearl* UK: College Publications, pp. 415 - 444.

[137] I. Tsamardinos, L.E. Brown and C.F. Aliferis [2006] *The Max - Min Hill - Climbing Bayesian Network Structure Learning Algorithm* Machine Learning vol. 65 pp. 31 - 78

[138] M. Valtorta, Y.G. Kim, J. Vomlel [2002] *Soft Evidential Update for Probabilistic Multiagent Systems* International Journal of Approximate Reasoning vol. 29 no. 1 pp. 71 - 106

[139] Vats, D.; Nowak, R.D. [2014] *A Junction Tree Framework for Undirected Graphical Model Selection* Journal of Machine Learning Research vol. 15 pp. 147 - 191

[140] P. Verma and J. Pearl [1992] *An Algorithm for Deciding if a Set of Observed Independencies has a Causal Explanation* in Uncertainty in Artificial Intelligence, Proceedings of the Eighth Conference (D. Dubois, M.P. Welman, B. D'Ambrosio and P.Smets, eds.) San Fransisco: Morgan Kaufman pp. 323 - 330

[141] Vorobev, N. N. [1962] *Consistent families of measures and their extensions* Theory of Probability and its Applications vol. 7 pp 147 - 162

[142] M.J. Wainright, M.I. Jordan [2003] *Graphical Models, Exponential Families and Variational Inference* Technical report 649, Department of Statistics, University of California, Berkeley

[143] *On the Optimality of Solutions of the Max-Product Belief-Propagation Algorithm in Arbitrary Graphs* IEEE Transactions on Information Theory vol. 47 no. 2 pp. 736 - 744

[144] Whittaker, J. [1990] *Graphical models in applied multivariate statistics* Wiley

[145] N. Wiberg [1996] *Codes and Decoding on General Graphs* Linköping Studies in Science and Technology. Dissertation 440 Linköpings Universitet, Linköping, 1996

[146] S. Wright [1921] *Correlation and Causation* Journal of Agricultural Research vol. 20 pp. 557 - 585

[147] Wright, S. [1934] *The method of path coefficients* Ann. Math. Statist. vol 5 pp 161 - 215.

[148] X. Xie and Z. Geng [2008] *A recursive method for structural learning of directed acyclic graphs* Journal of machine learning research vol. 9 pp. 459 - 483

[149] Yedidia, J.S.; Freeman, W.T.; Weiss, Y. [2005] *Constructing Free-Energy Approximations and Generalised Belief Propagation Algorithms* IEEE Transactions on Information Theory, vol. 51 no. 7 pp. 2282-2312

[150] R. Yehezkel and B. Lerner [2009] *Bayesian network structure learning by recursive autonomy identification* Journal of Machine Learning Research vol. 10 pp 1527 - 1570

[151] Zhang, J; Spirtes, P. [2002] *Strong faithfulness and uniform consistency in causal inference* Proceedings of the nineteenth conference on uncertainty in artificial intelligence pp 632–639, Morgan Kaufmann Publishers Inc.

Index