Learning Compound Decision Functions for Sequential Data in Dialog with Experts

Wojciech Jaworski

Faculty of Mathematics, Computer Science and Mechanics
Warsaw University, Banacha 2, 02-07 Warsaw, Poland
wjaworski@mimuw.edu.pl

Abstract. In this paper, we investigate the problem of learning the decision functions for sequential data describing complex objects that are composed of subobjects. The decision function maps sequence of attribute values into a relational structure, representing properties of the object described by the sequence. This relational structure is constructed in a way that allows us to answer questions from a given language. The decision function is constructed by means of rule system. The rules are learned incrementally in a dialog with an expert. We also present an algorithm that implements the rule system and we apply it to real life problems.

Keywords: Rough sets, Sequential pattern recognition.

Introduction

There are two types of sequential data:

- Data describing changes of objects in some evolving process.
- Data describing structural objects.

The first approach is suitable for physical phenomena, e.g., modeled by differential equations. In this case, surroundings of each point in the sequence is a state of object. The second approach is natural while analysing the data generated as the result of purposeful actions composing objects from simpler objects. Such a property have for example textual data, voice, recorded parameters of car on road.

In this paper, we investigate the problem of the decision function learning for data that belongs to the second mentioned type. The considered decision function maps any sequence of attribute values into a relational structure, representing properties of object described by the sequence. This relational structure is constructed in a way that allows us to answer questions from a given language. The decision function is constructed by means of a rule system. In a consequence, the decision function is compound (consists of vast number of rules), and has compound domain. So it belongs to a huge hypothesis space and according to statistical learning theory [11] cannot be learned only from data.

The function is learned in a dialog with an expert. The expert provides us the domain knowledge: He explains his decisions in natural language. We approximate, in a sense, his language by the rule system. The rule system must be compatible with the expert's way of thinking. Then the rules acquired from the expert are tested on data. Finally, the expert is inquired about the cases than does not match to the rules or are classified improperly. In this way, we extract successive fragments of decision functions which converge to the ideal description of the problem.

Due to the sequential character of data the rule system differs from the one used with data represented by tables. Each object is described by a sequence of attribute values. Complex objects can be decomposed into simple ones, which correspond to the split of the attribute value sequence into smaller parts. However, the distance from the beginning of sequence does not distinguish objects. Decision rules are not applied to the attributes according to their absolute position in sequence. Only the relative position of attributes is important for the rule to recognize the pattern.

Rules are used to recognize objects. Successful rule application means that object described by its construction belongs to the upper approximation of the problem. Information about this object is added to the data as a new attribute. The relational structure that describes the object is assigned to the attribute. The other rules may use such an attribute to recognize more complex objects.

The problem discussed in the paper is relates to the objectives of the Information Extraction (IE) [2,6,10].

IE is a subdiscipline of the Natural Language Processing. Its task is to find information, in text written in a natural language, needed to fill a table with description of some event. The attributes of the event are defined a priori.

The main differences follow the characteristics of the data. IE bases on the fact that in modern languages words are marked out with spaces and meaning of ambiguous parts of document can be determined using heuristic methods [1].

Our aim is to process sequential data in general. This implies that we can't take advantage of any a priori defined partition of the data sequence. We concentrate on solving the problem of ambiguity without the necessity of choosing one out of the contradicting interpretations.

We discover the relational structures during the process of learning instead of using a prori defined table.

We adopted the idea of syntax and semantic parsers well known in computer science [3,4,5,8]. Yet we propose our own approach to representation of rules and parser (rule-applying algorithm).

In Section 1, we formally define objects and attributes. In Section 2, we describe rules system properties. In Section 3 and 4, we define rules. In Section 5, we discuss the representation of data. In Section 6, we propose efficient algorithm for applying rules. In Section 7, we discuss the problem of learning the decision rules. In Section 8, we present applications.

1 Objects, Attributes, Meanings and Relational Structures

We are given a set of *objects* $\mathcal{U}$, a set of *attributes* $\mathcal{A}$, a set of *meanings* $\mathcal{M}$ and a set of *relational structures* $\mathcal{E}$.

white figures: ○ △ □ ◿
black figures: ● ▲ ■ ◢
shapes with undefined colours: ● ▲ ■ ◢
colours: ◔ ◕
letters (signs): a,. . . ,z,*space*

Fig. 1. Example of the set of objects

The signature of every structure in $\mathcal{E}$ is identical to the signature of the language of questions, which is a set of sentences in some logic. Having a structure from $\mathcal{E}$ we can deduce answers to the questions expressed in the logic.

The example of the set of objects is presented on Fig. 1 and the example of corresponding relational structures is on Fig. 1.

constants: a,. . . , z, space, white, black, circle, triangle, square
unary relations: Shape, Colour
binary relations: Figure

Fig. 2. Signature for the set of relational structures

Each attribute $a \in \mathcal{A}$ is a function $a : \mathcal{U} \to V$, where V is the set of attribute values. Each meaning $e \in \mathcal{M}$ is a function $e : \mathcal{U} \to \mathcal{E}$. Only information defined by attributes from $\mathcal{A}$ is available about objects from $\mathcal{U}$. For each attribute a the function $h : \mathcal{A} \to \mathcal{M}$ returns $h(a)$ — the meaning of a.

We are given an infinite sequence of attributes $\{a_i\}_{i=-\infty}^{\infty}$, where $a_i : \mathcal{U} \to (V \cup \{\#\})$, and a finite set X of attribute value sequences for some elements of $\mathcal{U}$. Each value sequence is finite, i.e., for each $u \in \mathcal{U}$ there exists $n \in \mathbb{N}$ such, that for $0 < i \leq n$ we have $a_i(u) \neq \#$ and for $i > n$ $a_i(u) = \#$ and $a_i(u) = \#$, when $i \leq 0$. In other words, each sequence from X has a finite interval of values from V and every attribute beyond that interval is equal to $\#$.

$$V = \{\texttt{a}, \ldots, \texttt{z}, \texttt{space}, \texttt{colour}, \texttt{shape}, \texttt{figure}\}$$

Fig. 3. Set of attribute values

Objects in $\mathcal{U}$ have a hierarchical structure: $u \in \mathcal{U}$ may be composed of some $u_1, u_2, \ldots, u_k \in \mathcal{U}$. In terms of attributes' sequence it means that the whole sequence describes an object, every $a_i(u)$ describes an object, and every subsequence $\{a_i(u)\}_{i=k}^{n}$ may describe an object.

u_1 in Fig. 1 is composed of two smaller objects

- object ◕ described by sequence $\{a_1(u_1), \ldots, a_5(u_1)\}$ with associated meaning: Colour(black)
- object ● described by sequence $\{a_7(u_1), \ldots, a_{13}(u_1)\}$ with associated meaning: Shape(circle)

objects'	attributes													
ids	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}
u_1	w	h	i	t	e		c	i	r	c	l	e	#	#
u_2	c	i	r	c	l	e	#	#	#	#	#	#	#	#
u_3	b	l	a	c	k		p	o	l	y	g	o	n	#
u_4	w	h	i	t	e		t	r	i	a	n	g	l	e

Fig. 4. Object descriptions in terms of attribute value sequences

We denote the object described by a single a_i as an *atomic object* and we assume that the meaning of atomic objects is a simple function of value of their attributes.

In our example meaning for a_i is equal to its value:

$$h(a_i)(u) = a_i(u).$$

We say that objects u_1, u_2 represented by sequences $\{a_{n+i}(u_1)\}_{i=0}^{k}$ and $\{a_{m+i}(u_2)\}_{i=0}^{k}$ are *indiscernible* if and only if $a_{n+i}(u_1) = a_{m+i}(u_2)$ for each i such that $0 \leq i \leq k$.

- $\{a_1(u_1), \ldots, a_{12}(u_1)\}$ defines ○ in an exact way
- $\{a_1(u_2), \ldots, a_6(u_2)\}$ is incomplete, and thus does not distinguish ○ and ●
- $\{a_1(u_3), \ldots, a_{13}(u_3)\}$ does not discern ●, ▲, ■ and ◢ because it is too general
- On the other hand our set of relational structures does not provide distinction between △ and ⊿, so $\{a_1(u_4), \ldots, a_{14}(u_4)\}$ is as exact as possible definition of △.

Information about objects from $\mathcal{U}$ is finite but it is not bounded. The number of attributes does not provide itself any useful information. Only the order of attributes is important.

2 Rule System

Our task is to discover the structure of objects from X having their description provided in terms of attributes.

We assume that elements of U are constructed out of some smaller objects. And these objects are represented by subsequences of elements of X. We recognize them by means of rules. The rule application can be interpreted as a process of object recognition. The rule recognizes the sequence of attribute values describing object and returns its meaning.

The rules can be divided into two parts: syntactical and semantic. Each syntactical rule recognizes sequence of attribute values and returns a value of new attribute constructed out from them. Any semantic rule operates on meanings of recognized sequence of attribute values and generates meaning for the newly constructed attribute.

We may derive the following rules for the example presented in previous section:

`colour`	$::=$	`w h i t e`	$\_ \rightarrow \{\text{Colour(white)}\}$
`colour`	$::=$	`b l a c k`	$\_ \rightarrow \{\text{Colour(black)}\}$
`shape`	$::=$	`c i r c l e`	$\_ \rightarrow \{\text{Shape(circle)}\}$
`shape`	$::=$	`t r i a n g l e`	$\_ \rightarrow \{\text{Shape(triangle)}\}$
`shape`	$::=$	`p o l y g o n`	$\_ \rightarrow \{\text{Shape(circle)}, \text{Shape(triangle)}, \text{Shape(square)}\}$
`figure`	$::=$	`colour space shape`	$(C, \_, S) \rightarrow \bigcup_{\text{Colour}(c) \in C} \bigcup_{\text{Shape}(s) \in S} \{\text{Figure}(c, s)\}$
`figure`	$::=$	`shape`	$S \rightarrow \bigcup_{\text{Shape}(s) \in S} \{\text{Figure(black}, s), \text{Figure(white}, s)\}$

If we apply the rules to u_4 we obtain the following attributes:

$a_{1,5}(u_4) =$ `colour`	$h(a_{1,5})(u_4) = \{\text{Colour(white)}\}$
$a_{7,14}(u_4) =$ `shape`	$h(a_{7,14})(u_4) = \{\text{Shape(triangle)}\}$
$a_{1,14}(u_4) =$ `figure`	$h(a_{1,14})(u_4) = \{\text{Figure(white,triangle)}\}$
$a_{7,14}(u_4) =$ `figure`	$h(a_{7,14})(u_4) = \{\text{Figure(white,triangle)}, \text{Figure(black,triangle)}\}.$

So after the rule application we obtained meaning for the entire sequence.

We construct the upper approximation of the set of meanings in a sense that for each sequence we find all possible meanings. For indiscernible objects we generate rules that recognize the same sequence and return a different attribute value or meaning.

3 Syntactic Rules

Now, we consider the problem of rule representation. Since attribute value sequence may be arbitrary long, there exists infinite number of possible rules. Only a finite subset of them can be learned. We must have decided what class of languages will be recognized by our rule system. We chose regular languages as a trade of between language strength and implementability. Yet for specific tasks another choice could be more appropriate.

We represent syntactic rules using a modification of context-free grammars by adding some special rule, called, a *term accumulation rule*. Formally, let

$$G = (\Sigma, N, X_I, R, +)$$

be such that

- $\Sigma = \bigcup_{u \in \mathcal{U}} \bigcup_i \{a_i(u)\}$ is finite set of atomic objects' names (terminal symbols),
- $N = V \setminus \Sigma$ is a finite set of non-atomic objects' names (non-terminal symbols),
- $X_I \in N$ is the start-symbol of grammar,
- R is a finite set of production rules. Each production has the form $A \rightarrow \alpha$ or $A \rightarrow \beta+$, where A is a non-terminal and α is a sequence of terminals and non-terminals and $\beta \in \Sigma \cup N$; $A \rightarrow \beta+$ is a shortcut for the set of rules: $A \rightarrow \beta, A \rightarrow \beta\beta, A \rightarrow \beta\beta\beta, \ldots$.

- $\prec$ is a binary relation of $\Sigma \cup N$ such that $A \prec B$ if and only if there is a rule $A \to \alpha$ in R such that B belongs to α or there is a rule $A \to B+$,
- $\prec$ is an irreflexive and transitive partial order.

Proposition 1. *A language L can be recognized by a grammar of the defined above type if and only if L is a regular language.*

The purpose of syntactic rules is to parse any sequence from U into X_I.

4 Semantic Rules

In order to obtain the object meaning instead of recognizing its presence we add semantic interpretations to symbols and rules. Let $\mathcal{E}$ be set of relational structures (see Section 1 for a definition and Section 8 for an example). For terminal symbols we define

$$[\![\cdot]\!]_\Sigma : \Sigma \to \mathcal{E}.$$

For each $A \to \alpha_1 \ldots \alpha_n$ rule we define

$$f_{A \to \alpha_1 \ldots \alpha_n} : \mathcal{E}^n \to \mathcal{E}$$

For each $A \to \beta+$ rule we define

$$f_{A \to \beta+} : \mathcal{E}^+ \to \mathcal{E}$$

These semantic functions operates on the relational structures. They compose greater structures out of smaller ones.

Now, we define semantics interpretation of symbols: For each $\sigma \in \Sigma$ let

$$[\![\sigma]\!] = [\![\sigma]\!]_\Sigma.$$

For each $A \in N$ if A was derived using $A \to \alpha_1 \ldots \alpha_n$ rule let

$$[\![A]\!] = f_{A \to \alpha_1 \ldots \alpha_n}([\![\alpha_1]\!], \ldots, [\![\alpha_n]\!]),$$

and if A was derived using $A \to \beta+$ rule as $\beta_1 \ldots \beta_n$ sequence let

$$[\![A]\!] = f_{A \to \beta+}([\![\beta_1]\!], \ldots, [\![\beta_n]\!])$$

Let u be an object, and a an attribute, then

$$[\![a(u)]\!] = h(a)(u)$$

Note that we may add many different semantic actions to each syntactic rule. Obtaining rules that are grammatically identical but differ on semantic level.

5 Data Sequence Representation

Our goal is to find all possible semantic interpretations (the upper approximation) for a given rule set and an attribute value sequence. We represent objects recognized in data sequence as directed acyclic graph whose edges are labelled by attribute values.

Having given attribute value sequence $\{\sigma_i\}_1^n, \sigma_i \in \Sigma$ we create graph with vertexes $V = \{v_0, \ldots, v_n\}$ and set of edges $E = \{v_0 \xrightarrow{\sigma_1} v_1, \ldots, v_{n-1} \xrightarrow{\sigma_n} v_n\}$

Applying the rule $A \rightarrow \alpha_1, \ldots, \alpha_k$ consists in finding all paths

$$v_{a_0} \xrightarrow{\alpha_1} v_{a_1} \xrightarrow{\alpha_2} v_{a_2} \ldots v_{a_{k-1}} \xrightarrow{\alpha_k} v_{a_k}$$

and adding for each of them the edge

$$v_{a_0} \xrightarrow{A} v_{a_k}$$

to the graph. Formally, applying the $A \rightarrow \beta+$ consisting in finding all paths

$$v_{a_0} \xrightarrow{\beta} v_{a_1} \xrightarrow{\beta} v_{a_2} \ldots v_{a_{k-1}} \xrightarrow{\beta} v_{a_k}$$

and adding for each of them the edge

$$v_{a_0} \xrightarrow{A} v_{a_k}$$

to the graph.

6 Rule-Applying Algorithm

We divide set of symbols into layers: Let $N_0 = \Sigma$ and let

$$N_{n+1} = \{A : \exists_{A \rightarrow \alpha_1 \ldots \alpha_k} \forall_i (\alpha_i \in N_n) \vee \exists_{A \rightarrow \beta+} (\beta \in N_n)\}$$

Now we divide the rule set R into layers. Let $R_{-1} = \emptyset$ and

$$R_n = (\{A \rightarrow \alpha_1 \ldots \alpha_k : \forall_i \alpha_i \in N_n\} \cup \{A \rightarrow \beta+ : \beta \in N_n\}) \setminus R_{n-1}.$$

We begin with graph (V, E_0), where $E_0 = E$. We obtain graph (V, E_{n+1}) by applying to (V, E_n) rules from R_n.

In order to do it efficiently we create prefix tree out of every layer: For each rule $A \rightarrow \alpha_1 \ldots \alpha_k$ in R_n we create path from the root labelled by symbols α_1 till α_k and we label the leaf tree node by A. For each node we merge path that have identical labels. Using this data structure we can apply all $A \rightarrow \alpha$ rules in layer in $\mathcal{O}(|E_n| l \log |\Sigma \cup E| + |E_n||R_n^+|)$ time, where

$$l = \max_{R_n} \{k : A \rightarrow \alpha_1 \ldots \alpha_k \in R_n\}$$

Since l, $\log |\Sigma \cup E|$ and number of layers is relatively small $|E_n|$ is crucial for parser performance. The problem is that E_n contains information that we want to obtain as a result of parsing process. If text have exponential number of interpretations $|E_n|$ will increase exponentially. To handle this problem we must reduce the number of interpretations either by throwing away part of them or merging them.

7 Learning

Experiments indicated that the rules can be divided into two kinds:

The rules that describe the structure of sequences. They have complex semantics and, therefore, they must be designed manually in dialog with experts. Fortunately, in typical task there is only a small number of such rules.

The rules that contain the "vocabulary" of the problem. There may be lots of them, but they split into a few groups and all the rules in each group produce the same semantic function. These rules may be learned automatically.

The learning process may be performed analogically as in case of learning decision tables: We create the training sample and using it we generate a classifier. As a classification algorithm we use one of well known classifiers adjusted for sequential data. For example, if we wish to use k-Nearest Neighbour algorithm we introduce a similarity measure on attributes values and use the edit distance to determine similarity of a pair of attribute value sequences.

The other possibility is specific for the sequential data. We learn the context in which the sample is likely to appear instead of learning the sample itself.

One can also combine the above mentioned methods.

In the example presented in the following section we use a semiautomatic way of learning, which takes the advantage of the fact that samples appear in a small number of contexts. We derive rules automatically from the attribute value sequences that are within the context which we manually indicated.

8 Application Example: Semantic Analyser for Sumerian Ur III Economic Text Corpus

The Ur III economic text corpus consists of circa 43500 documents. They describe the process of redistribution of goods in Ancient Sumer during the Ur III period (2100BC-2000BC). The Ur III economic texts were the subject of sumerological research for many years. The tablets were transliterated by many researchers, who didn't have the strict conventions for describing the tablet content. As a result the corpus does not have a uniform format, the description of text arrangement on tablets is not standardized and mixed with Sumerian text. The other problem is that the Sumerian texts itself are often ambiguous on both syntactic and semantic level.

The signature for structures included in $\mathcal{E}$ is composed of following elements:

- constants such as numbers, names, commodities and dates
- unary relations: **Number**, **Name**, **Day**, **Month**, **Year**, **Date**, **Quantity**, **Commodity Supplier**, **Receiver**
- 5-tuple relation **Transaction**.

We arranged a rule set into the corpus. Most of the rules (about 3500) recognize Sumerian personal names, names of gods and cities, etc. These rules were generated in semiautomatic way described in Sect. 7. The remaining rules (about 200) describe the structure of the language.

We described contents of the whole tablet as a set of transactions:

`&P123831 = OIP 121, 101`	
`tablet`	
`obverse`	
`1. 1(disz) sila4 ur-mes ensi2`	*1 lamb Urmes governor*
`2. 1(disz)# sila4 da-da dumu lugal`	*1 lamb Dada son of king*
`3. 1(disz)# sila4 id-da-a`	*1 lamb Idda*
`reverse`	
`1. u4 2(u) 3(asz@t)-kam`	*Day 23*
`$ 1 line blank`	
`3. mu-DU`	*delivery*
`4. ab-ba-sa6-ga i3-dab5`	*Abbasaga received*
`5. iti sze-KIN-ku5`	*month sze-kin-ku5*
`6. mu en {d}inanna ba-hun`	*Year when high priest of goddess Innana was elevated to office*
`left`	
`1. 3(disz)`	*3*

Fig. 5. Example of transliterated cuneiform tablet

Date	**Quantity**	**Commodity**	**Supplier**	**Receiver**
23-11-AS05	1	sila4	ur-mes ensi2	ab-ba-sa6-ga
23-11-AS05	1	sila4	da-da dumu lugal	ab-ba-sa6-ga
23-11-AS05	1	sila4	id-da-a	ab-ba-sa6-ga

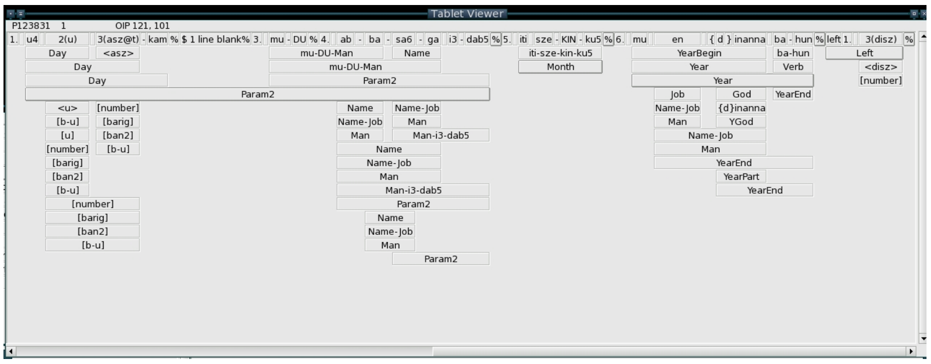

Fig. 6. Rule applying algorithm at work

The system allowed us to transform automatically the data (in this case Sumerian economic document) from the sequential form into a table representing a relational structure. We defined relational structure that represents information contained in texts in a way convenient for further analysis.

9 Conclusions

We investigated the problem of learning the decision functions for sequential data. We proposed a rule system that allows us to map data into a compound decision value.

We introduced a heuristic distinction between rules that may and may not be learned automatically. We outlined the general ideas for the classification algorithms construction.

The question about kinds of rules that may be learned from data and details of construction various classification algorithms require further studies.

In our future work, we plan to define properties of query languages relevant for certain applications and extend the rule system for numerical data. We also plan to combine the process of constructing the relational structure with deductive reasoning, by creating an implicative interpretation for object recognition rules.

We would like also extend the process of learning to the higher level concepts (e.g. soft concepts) that cannot be expressed in an exact way by means of our relational structures.

Acknowledgement. The research has been supported by the grant 3 T11C 002 26 from Ministry of Scientific Research and Information Technology of the Republic of Poland.

References

1. E. Brill "A Simple Rule-Based Part-of-Speech Tagger". *In Proceedings of the Third Conference on Applied Computational Linguistics (ACL)*, Trento Italy.
2. J. Cowie, W. Lehnert. "Information extraction". *Special NLP Issue of the Communications of the ACM*, 1996.
3. Daniel Jurafsky and James H. Martin "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition", Prentice-Hall, 2000
4. L. Kartunen, J.P. Chanod, G. Grefenstette and A. Schiller "Regular expressions for language engineering", *Natural Language Engineering*, 2:2:305-328,1996
5. D. E. Knuth, *Semantics of context-free languages.* Mathematical Systems Theory, 2(2):127–145, 1968.
6. Ludovic Lebart, Andre Salem, and Lisette Berry, "Exploring Textual Data", Kluwer Academic Publishers, 1997. Review available in Computational Linguistics, 25(1), 1999.
7. Marie-Francine Moens, "Information Extraction: Algorithms and Prospects in a Retrieval Context" *The Information Retrieval Series , Vol. 21*, Springer 2006,
8. G. van Noord , D. Gerdemann "An Extendible Regular Expression Compiter for Finite-State Approaches in Natural Language Processing", *Workshop on Implementing Automata*, Postnam, Germany, 1997
9. S. K. Pal, L. Polkowski, A. Skowron (Eds.), *Rough-Neural Computing: Techniques for Computing with Words*, Cognitive Technologies. Springer-Verlag, 2004.
10. Ed.: Maria Teresa Pazienza. "Information Extraction in the Web Era. Natural Language Communication for Knowledge Acquisition and Intelligent Information Agents". Berlin 2003 Springer-Verlag 8 s. XI, 162. Lecture Notes in Artificial Intelligence, 2700. subser. of Lecture Notes in Computer Science, 2700 ISBN 3540405798
11. V. N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.