From trees to rules

- Simple way: one rule for each leaf
- C4.5rules: greedily prune conditions from each rule if this reduces its estimated error
  - Can produce duplicate rules
  - Check for this at the end
- Then
  - look at each class in turn
  - consider the rules for that class
  - find a “good” subset (guided by MDL)
- Then rank the subsets to avoid conflicts
- Finally, remove rules (greedily) if this decreases error on the training data
C4.5rules: choices and options

- C4.5rules slow for large and noisy datasets
- Commercial version C5.0rules uses a different technique
  - Much faster and a bit more accurate
- C4.5 has two parameters
  - Confidence value (default 25\%): lower values incur heavier pruning
  - Minimum number of instances in the two most popular branches (default 2)
*Classification rules*

- **Common procedure:** *separate-and-conquer*

- **Differences:**
  - Search method (e.g. greedy, beam search, ...)
  - Test selection criteria (e.g. accuracy, ...)
  - Pruning method (e.g. MDL, hold-out set, ...)
  - Stopping criterion (e.g. minimum accuracy)
  - Post-processing step

- **Also:** Decision list
  - vs. one rule set for each class
Test selection criteria

- Basic covering algorithm:
  - keep adding conditions to a rule to improve its accuracy
  - Add the condition that improves accuracy the most
- Measure 1: $p/t$
  - $t$ total instances covered by rule
  - $p$ number of these that are positive
  - Produce rules that don’t cover negative instances, as quickly as possible
  - May produce rules with very small coverage — special cases or noise?
- Measure 2: Information gain $p \left( \log\left(\frac{p}{t}\right) - \log\left(\frac{P}{T}\right) \right)$
  - $P$ and $T$ the positive and total numbers before the new condition was added
  - Information gain emphasizes positive rather than negative instances
- These interact with the pruning mechanism used
*Missing values, numeric attributes*

- Common treatment of missing values: 
  
  _for any test, they fail_
  
  - Algorithm must either
    - use other tests to separate out positive instances
    - leave them uncovered until later in the process

- In some cases it’s better to treat “missing” as a separate value

- Numeric attributes are treated just like they are in decision trees
*Pruning rules*

- Two main strategies:
  - Incremental pruning
  - Global pruning
- Other difference: pruning criterion
  - Error on hold-out set (*reduced-error pruning*)
  - Statistical significance
  - MDL principle
- Also: post-pruning vs. pre-pruning
Rule based classifiers

Each classification rule is of form

\[ r : (\text{Condition}) \rightarrow y \]

- LHS of the rule (Condition), called rule antecedent or precondition, is a conjunction of attribute tests
- RHS, also called the rule consequent, is the class label

Rule set:

\[ R = \{ r_1, r_2, \ldots, r_n \} \]
Classifying Instances with Rules

- A rule $r$ **covers** an instance $x$ if the attributes of the instance satisfy the condition of the rule.

**Example**

- **Rule:**
  
  $r : (\text{Age} < 35) \land (\text{Status} = \text{Married}) \rightarrow \text{Cheat}=\text{No}$

- **Instances:**
  
  $x_1 : (\text{Age}=29, \text{Status}=\text{Married}, \text{Refund}=\text{No})$
  $x_2 : (\text{Age}=28, \text{Status}=\text{Single}, \text{Refund}=\text{Yes})$
  $x_3 : (\text{Age}=38, \text{Status}=\text{Divorced}, \text{Refund}=\text{No})$

- Only $x_1$ is covered by the rule $r$.
Classifying Instances with Rules

- **Rules may not be mutually exclusive**
  - More than one rule may cover the same instance

- **Strategies:**
  - Strict enforcement of mutual exclusiveness
    - Avoid generating rules that have overlapping coverage with previously selected rules
  - Ordered rules
    - Rules are rank ordered according to their priority
  - Voting
    - Allow an instance to trigger multiple rules, and consider the consequent of each triggered rule as a vote for that particular class

- **Rules may not be exhaustive**

- **Strategy:**
  - A *default rule* $r_d : \emptyset \rightarrow y_d$
    - can be added
  - The default rule has an empty antecedent and is applicable when all other rules have failed
  - $y_d$ is known as *default class* and is often assigned to the majority class
Advantages of Rule Based Classifiers

- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees
Definition

- Coverage of a rule:
  - Number (or fraction) of instances that satisfy the antecedent of a rule

- Accuracy of a rule:
  - Fraction of instances that satisfy both the antecedent and consequent of a rule

- Length:
  - Number of descriptors
Example

(Marital Status=Married) → No

- Coverage = 40%,
- Accuracy = 100%
- Length = 1

<table>
<thead>
<tr>
<th>ID</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Construction of a Rule Based Classifier from data

- Generate an initial set of rules
  - Direct Method:
    - Extract rules directly from data
    - Examples: RIPPER, CN2, Holte’s 1R, Boolean reasoning
  - Indirect Method:
    - Extract rules from other classification methods (e.g. decision trees)
    - Example: C4.5 rules

- Rules are pruned and simplified
- Rules can be order to obtain a rule set \( R \)
- Rule set \( R \) can be further optimized
Indirect Method: Conversion from Decision Trees

- Rules are mutually exclusive and exhaustive
- Rule set contains as much information as the tree
- Rules can be simplified

\[(\text{Refund}=\text{No}) \land (\text{Status}=\text{Married}) \rightarrow \text{No} \implies (\text{Status}=\text{Married}) \rightarrow \text{No}\]
Indirect Method: C4.5 rules

Creating an initial set of rules

- Extract rules from an un-pruned decision tree
- For each rule, \( r: A \rightarrow y \)
  - Consider alternative rules \( r': A' \rightarrow y \), where \( A' \) is obtained by removing one of the conjuncts in \( A \)
  - Replace \( r \) by \( r' \) if it has a lower pessimistic error
  - Repeat until we can no longer improve the generalization error

Ordering the rules

- Instead of ordering the rules, order subsets of rules
  - Each subset is a collection of rules with the same consequent (class)
  - The subsets are then ordered in the increasing order of Description length = \( L(\text{exceptions}) + g \cdot L(\text{model}) \)
  - where \( g \) is a parameter that takes in to account the presence of redundant attributes in a rule set. Default value is 0.5
Direct method: Sequential covering

(E: training examples, A: set of attributes)

1. Let \( R = \{ \} \) be the initial rule set
2. While stopping criteria is not met
   1. \( r := \text{Learn-One-Rule} (E, A) \);
   2. Remove instances from \( E \) that are covered by \( r \);
   3. Add \( r \) to rule set: \( R = R + \{r\} \);

- Ex. Stopping criteria = „\( E \) is empty”
(i) Original Data

(ii) Step 1

(iii) Step 2

(iv) Step 3

Association rules 72
Learn one rule (1R)

- The objective of this function is to extract the best rule that covers the current set of training instances
  - What is the strategy used for rule growing
  - What is the evaluation criteria used for rule growing
  - What is the stopping criteria for rule growing
  - What is the pruning criteria for generalizing the rule
Learn One Rule: Rule Growing Strategy

- **General-to-specific approach**
  - It is initially assumed that the best rule is the empty rule, \( r : \{ \} \rightarrow y \), where \( y \) is the majority class of the instances
  - Iteratively add new conjuncts to the LHS of the rule until the stopping criterion is met

- **Specific-to-general approach**
  - A positive instance is chosen as the initial seed for a rule
  - The function keeps refining this rule by generalizing the conjuncts until the stopping criterion is met
Rule Evaluation and Stopping Criteria

- Evaluate rules using rule evaluation metric
  - Accuracy
  - Coverage
  - Entropy
  - Laplace
  - M-estimate

- A typical condition for terminating the rule growing process is to compare the evaluation metric of the previous candidate rule to the newly grown rule
Learn 1R

- Rule Pruning
  - Each extracted rule can be pruned to improve their ability to generalize beyond the training instances
  - Pruning can be done by removing one of the conjuncts of the rule and then testing it against a validation set

- Instance Elimination
  - Instance elimination prevents the same rule from being generated again
  - Positive instances must be removed after each rule is extracted
  - Some rule based classifiers keep negative instances, while some remove them prior to generating next rule
RIPPER

- For 2-class problem, choose one of the classes as positive class, and the other as negative class
  - Learn rules for positive class
  - Negative class will be default class

- For multi-class problem
  - Order the classes according to increasing class prevalence (fraction of instances that belong to a particular class)
  - Learn the rule set for smallest class first, treat the rest as negative class
  - Repeat with next smallest class as positive class
Foil's Information Gain

- Compares the performance of a rule before and after adding a new conjunct.
- Foil's information gain is defined as:
  \[ t \cdot \left[ \log_2 \left( \frac{p_1}{p_1 + n_1} \right) - \log_2 \left( \frac{p_0}{p_0 + n_0} \right) \right] \]
  where \( t \) is the number of positive instances covered by both \( r \) and \( r' \).
Direct Method: RIPPER

- Growing a rule:
  - Start from empty rule
  - Add conjuncts as long as they improve Foil's information gain
  - Stop when rule no longer covers negative examples
  - Prune the rule immediately using incremental reduced error pruning
  - Measure for pruning: \( v = (p - n) / (p + n) \)
    - \( p \): number of positive examples covered by the rule in the validation set
    - \( n \): number of negative examples covered by the rule in the validation set
  - Pruning method: delete any final sequence of conditions that maximizes \( v \)
RIPPER: Building a Rule Set

- Use sequential covering algorithm
  - Finds the best rule that covers the current set of positive examples
  - Eliminate both positive and negative examples covered by the rule

- Each time a rule is added to the rule set, compute the description length
  - Stop adding new rules when the new description length is $d$ bits longer than the smallest description length obtained so far. $d$ is often chosen as 64 bits
RIPPER: Optimize the rule set:

- For each rule \( r \) in the rule set \( R \)
  - Consider 2 alternative rules:
    - Replacement rule (\( r^* \)): grow new rule from scratch
    - Revised rule (\( r' \)): add conjuncts to extend the rule \( r \)
  - Compare the rule set for \( r \) against the rule set for \( r^* \) and \( r' \)
  - Choose rule set that minimizes MDL principle
- Repeat rule generation and rule optimization for the remaining positive examples
<table>
<thead>
<tr>
<th>Name</th>
<th>Give Birth</th>
<th>Lay Eggs</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Have Legs</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>python</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>reptiles</td>
</tr>
<tr>
<td>salmon</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>fishes</td>
</tr>
<tr>
<td>whale</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>frog</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>amphibians</td>
</tr>
<tr>
<td>komodo</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>reptiles</td>
</tr>
<tr>
<td>bat</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>pigeon</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>birds</td>
</tr>
<tr>
<td>cat</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>leopard shark</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>fishes</td>
</tr>
<tr>
<td>turtle</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>reptiles</td>
</tr>
<tr>
<td>penguin</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>birds</td>
</tr>
<tr>
<td>porcupine</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>eel</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>fishes</td>
</tr>
<tr>
<td>salamander</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>sometimes</td>
<td>yes</td>
<td>amphibians</td>
</tr>
<tr>
<td>gila monster</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>reptiles</td>
</tr>
<tr>
<td>platypus</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>owl</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>birds</td>
</tr>
<tr>
<td>dolphin</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>eagle</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>birds</td>
</tr>
</tbody>
</table>
**C 4.5 rules vs. RIPPER**

C 4.5 rules:

- (Give Birth=No, Can Fly=Yes) → Birds
- (Give Birth=No, Live in Water=Yes) → Fishes
- (Give Birth=Yes) → Mammals
- (Give Birth=No, Can Fly=No, Live in Water=No) → Reptiles
- () → Amphibians

RIPPER:

- (Live in Water=Yes) → Fishes
- (Have Legs=No) → Reptiles
- (Give Birth=No, Can Fly=No, Live In Water=No) → Reptiles
- (Can Fly=Yes, Give Birth=No) → Birds
- () → Mammals