Data Mining
Rule-based Classifiers

- What is a classification rule
  - Evaluating a rule
  - Algorithms
    - PRISM
      - Incremental reduced-error pruning
    - RIPPER
  - Handling
    - Missing values
    - Numeric attributes
- Rule-based classifiers versus decision trees → C4.5rules, PART

Classification Rules

- “if...then...” rules
  
  (Blood Type=Warm) ∧ (Lay Eggs=Yes) → Birds
  
  (Taxable_Income<50K) ∧ (Refund=Yes) → Evade=No

- Rule: \((\text{Condition}) \rightarrow y\)
  
  - where
    - \(\text{Condition}\) is a conjunction of attribute tests
      
      \((A_1 = v_1) \text{ and } (A_2 = v_2 \text{ and ... and } (A_n = v_n))\)
    - \(y\) is the class label
  
  - LHS: rule antecedent or condition
  
  - RHS: rule consequent
Rule-based Classifier (Example)

<table>
<thead>
<tr>
<th>Name</th>
<th>Blood Type</th>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>human</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>python</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>reptiles</td>
</tr>
<tr>
<td>salmon</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>fishes</td>
</tr>
<tr>
<td>whale</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>frog</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>amphibians</td>
</tr>
<tr>
<td>komodo</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>reptiles</td>
</tr>
<tr>
<td>bat</td>
<td>warm</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>pigeon</td>
<td>warm</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>birds</td>
</tr>
<tr>
<td>cat</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>leopard shark</td>
<td>cold</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>fishes</td>
</tr>
<tr>
<td>turtle</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>reptiles</td>
</tr>
<tr>
<td>penguin</td>
<td>warm</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>birds</td>
</tr>
<tr>
<td>porcupine</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>eel</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>fishes</td>
</tr>
<tr>
<td>salamander</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>amphibians</td>
</tr>
<tr>
<td>gila monster</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>reptiles</td>
</tr>
<tr>
<td>platypus</td>
<td>warm</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>mammals</td>
</tr>
<tr>
<td>seal</td>
<td>warm</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>birds</td>
</tr>
<tr>
<td>dolphin</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>mammals</td>
</tr>
<tr>
<td>eagle</td>
<td>warm</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>birds</td>
</tr>
</tbody>
</table>

R1: (Give Birth = no) ∧ (Can Fly = yes) → Birds
R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes
R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals
R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles
R5: (Live in Water = sometimes) → Amphibians

Motivation

- Consider the rule set
  - Attributes A, B, C, and D can have values 1, 2, and 3
  
  \[A = 1 \land B = 1 \rightarrow \text{Class} = Y\]
  \[C = 1 \land D = 1 \rightarrow \text{Class} = Y\]
  Otherwise, \text{Class} = N

- How to represent it as a decision tree?
  - The rules need a common attribute
    
    \[A = 1 \land B = 1 \rightarrow \text{Class} = Y\]
    \[A = 1 \land B = 2 \land C = 1 \land D = 1 \rightarrow \text{Class} = Y\]
    \[A = 1 \land B = 3 \land C = 1 \land D = 1 \rightarrow \text{Class} = Y\]
    \[A = 2 \land C = 1 \land D = 1 \rightarrow \text{Class} = Y\]
    \[A = 3 \land C = 1 \land D = 1 \rightarrow \text{Class} = Y\]
    Otherwise, \text{Class} = N
### Motivation

![Diagram showing a decision tree with nodes labeled A, B, C, D, and branches indicating yes (Y) or no (N).]

### Application of Rule-Based Classifier

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

1. $R_1$: $(\text{Give Birth} = \text{no}) \land (\text{Can Fly} = \text{yes}) \rightarrow \text{Birds}$
2. $R_2$: $(\text{Give Birth} = \text{no}) \land (\text{Live in Water} = \text{yes}) \rightarrow \text{Fishes}$
3. $R_3$: $(\text{Give Birth} = \text{yes}) \land (\text{Blood Type} = \text{warm}) \rightarrow \text{Mammals}$
4. $R_4$: $(\text{Give Birth} = \text{no}) \land (\text{Can Fly} = \text{no}) \rightarrow \text{Reptiles}$
5. $R_5$: $(\text{Live in Water} = \text{sometimes}) \rightarrow \text{Amphibians}$

<table>
<thead>
<tr>
<th>Name</th>
<th>Blood Type</th>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>hawk</td>
<td>warm</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>?</td>
</tr>
<tr>
<td>grizzly bear</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>?</td>
</tr>
</tbody>
</table>

The rule $R_1$ covers a hawk $\Rightarrow$ Class = Bird
The rule $R_3$ covers the grizzly bear $\Rightarrow$ Class = Mammal
Rule Coverage and Accuracy

- Quality of a classification rule can be evaluated by
  - **Coverage**: fraction of records that satisfy the antecedent of a rule
    \[
    \text{Coverage}(r) = \frac{|LHS|}{n}
    \]
  - **Accuracy**: fraction of records covered by the rule that belong to the class on the RHS
    \[
    \text{Accuracy}(r) = \frac{|LHS \cap RHS|}{|LHS|}
    \]

(n is the number of records in our sample)

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Marital Status</th>
<th>Taxable Income</th>
<th>Class</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>

(Condition = Single) → No
Coverage = 40%, Accuracy = 50%

How does Rule-based Classifier Work?

- **R1**: (Give Birth = no) ∧ (Can Fly = yes) → Birds
- **R2**: (Give Birth = no) ∧ (Live in Water = yes) → Fishes
- **R3**: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals
- **R4**: (Give Birth = no) ∧ (Can Fly = no) → Reptiles
- **R5**: (Live in Water = sometimes) → Amphibians

<table>
<thead>
<tr>
<th>Name</th>
<th>Blood Type</th>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>lemur</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>?</td>
</tr>
<tr>
<td>turtle</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>sometimes</td>
<td>?</td>
</tr>
<tr>
<td>dogfish shark</td>
<td>cold</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>?</td>
</tr>
</tbody>
</table>

A **lemur** triggers rule **R3** ⇒ class = mammal
A **turtle** triggers both **R4** and **R5** ⇒ voting or ordering rules
A **dogfish shark** triggers none of the rules ⇒ default rule
Building Classification Rules

- **Direct Method**
  - Extract rules directly from data
  - e.g.: **RIPPER, Holte’s 1R (OneR)**

- **Indirect Method**
  - Extract rules from other classification models (e.g. decision trees, etc).
  - e.g: **C4.5rules**

A Direct Method: Sequential Covering

- Let \( E \) be the training set
  - Extract rules one class at a time

For each class \( C \)

1. Initialize set \( S \) with \( E \)
2. While \( S \) contains instances in class \( C \)
   3. Learn one rule \( R \) for class \( C \)
   4. Remove training records covered by the rule \( R \)

**Goal:** to create rules that cover many examples of a class \( C \) and none (or very few) of other classes
A Direct Method: Sequential Covering

- How to learn a rule for a class C?

1. Start from an empty rule \( \{\} \rightarrow \text{class} = C \)
2. Grow a rule by adding a test to LHS \( (a = v) \)
3. Repeat Step (2) until stopping criterion is met

Two issues:
- How to choose the best test? Which attribute to choose?
- When to stop building a rule?

Example: Generating a Rule

- Possible rule set for class “b”:
  - If \( (x \leq 1.2) \) then class = b
  - If \( (x > 1.2) \) and \( (y \leq 2.6) \) then class = b

- Could add more rules, get “perfect” rule set
Simple Covering Algorithm

- **Goal**: Choose a test that improves a quality measure for the rules.
  - E.g. maximize rule’s accuracy
- Similar to situation in decision trees: problem of selecting an attribute to split on
  - Decision tree algorithms maximize overall purity
- Each new test reduces rule’s coverage:

  - $t$ total number of instances covered by rule
  - $p$ positive examples of the class predicted by rule
  - $t - p$ number of errors made by rule
  - Rules accuracy $= \frac{p}{t}$

When to Stop Building a Rule

- When the rule is perfect, i.e. accuracy $= 1$
- When increase in accuracy gets below a given threshold
- When the training set cannot be split any further
**PRISM Algorithm**

For each class C
  Initialize E to the training set
  While E contains instances in class C
    Create a rule R with an empty left-hand side that predicts class C
    Until R is perfect (or there are no more attributes to use) do
      For each attribute A not mentioned in R, and each value v,
        Consider adding the condition A = v to the left-hand side of R
        Select A and v to maximize the accuracy p/t
        (break ties by choosing the condition with the largest p)
        Add A = v to R
    Remove the instances covered by R from E

Available in **WEKA**

---

**Rule Evaluation in PRISM**

\[
\text{Accuracy} = \frac{p}{t}
\]

- Number of instances covered by rule
- Number of instances covered by rule that belong to the positive class

- Produce rules that don’t cover *negative* instances, as quickly as possible
- **Disadvantage:** may produce rules with very small coverage
  - Special cases or noise? (overfitting)
Rule Evaluation

- Other metrics:

\[ \text{FOIL's inf. gain} = p_1 \times \left( \log_2 \frac{p_1}{t} - \log_2 \frac{p_0}{t} \right) \]

\[ \text{Laplace} = \frac{p+1}{t+k} \]

\[ \text{m-estimate} = \frac{p+k}{t+k} \]

- These measures take into account the coverage/support count of the rule

<table>
<thead>
<tr>
<th>Accuracy after a new test is added</th>
<th>Accuracy before a new test is added</th>
</tr>
</thead>
</table>

\( t \): number of instances covered by the rule
\( p \): number of instances covered by the rule that are positive examples
\( k \): number of classes
\( f \): prior probability of the positive class

Missing Values and Numeric attributes

- Common treatment of missing values
  - Algorithm delays decisions until a test involving attributes with no missing values emerges

- Numeric attributes are treated just like they are in decision trees
  1. Sort instances according to attributes value
  2. For each possible threshold, a binary-less/greater than test is considered
     - Choose a test \( A < v \), if it is the one that produces higher accuracy
**Rule Pruning**

- The **PRISM** algorithm tries to get perfect rules, i.e. rules with accuracy = 1 on the training set.
  - These rules can be overspecialized → **overfitting**
  - **Solution**: prune the rules

- Two main strategies:
  - *Incremental pruning*: simplify each rule as soon as it is built
  - *Global pruning*: build full rule set and then prune it

---

**Incremental Pruning: Reduced Error Pruning**

- The data set is split into a training set and a **prune set**

**Reduced Error Pruning**

1. Remove one of the conjuncts in the rule
2. Compare error rate on the prune set before and after pruning
3. If error improves, prune the conjunct

**Sampling with stratification advantageous**

- **Training set** (to learn rules)
  - **Validation set** (to prune rules)
  - **Test set** (to determine model accuracy)
Direct Method: **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

- Available in *Weka*

---

Direct Method: **RIPPER**

- Learn one rule:
  - Start from empty rule
  - Add conjuncts as long as they improve FOIL’s information gain
  - Stop when rule no longer covers negative examples
    - Build rules with accuracy = 1 (if possible)
  - Prune the rule immediately using **reduced error pruning**
  - Measure for pruning: \( W(R) = (p-n)/(p+n) \)
    - \( p \): 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 starts from the last test added to the rule
    - May create rules that cover some negative examples (accuracy < 1)

- A global optimization (pruning) strategy is also applied
Indirect Method: **C4.5rules**

- Extract rules from an unpruned decision tree
- For each rule, \( r: \text{RHS} \rightarrow c \), consider pruning the rule
- Use **class ordering**
  - Each subset is a collection of rules with the same rule consequent (class)
  - Classes described by simpler sets of rules tend to appear first

---

**Example**

<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>no</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>leopardo 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>no</td>
<td>yes</td>
<td>yes</td>
<td>birds</td>
</tr>
</tbody>
</table>
C4.5 versus C4.5rules versus RIPPER

**C4.5rules:**
- (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

**Indirect Method: PART**

- Combines the divide-and-conquer strategy with separate-and-conquer strategy of rule learning
  1. Build a partial decision tree on the current set of instances
  2. Create a rule from the decision tree
     - The leaf with the largest coverage is made into a rule
  3. Discard the decision tree
  4. Remove the instances covered by the rule
  5. Go to step one
- Available in *WEKA*
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
- Can easily handle missing values and numeric attributes

- Available in **WEKA**: *Prism, Ripper, PART, OneR*