Data Mining
Rule-based Classifiers

- **What is a classification rule**

  Section 5.1 of course book.
- **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: *(Condition) → y*
  - where
    - *Condition* is a **conjunction of attribute tests**

      $(A_1 = v_1)$ **and** $(A_2 = v_2)$ **and** … **and** $(A_n = v_n)$
    - *y* is the **class label**
  - *LHS*: rule antecedent or condition
  - *RHS*: rule consequent

Rule-based Classifier (Example)

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles

R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

Motivation

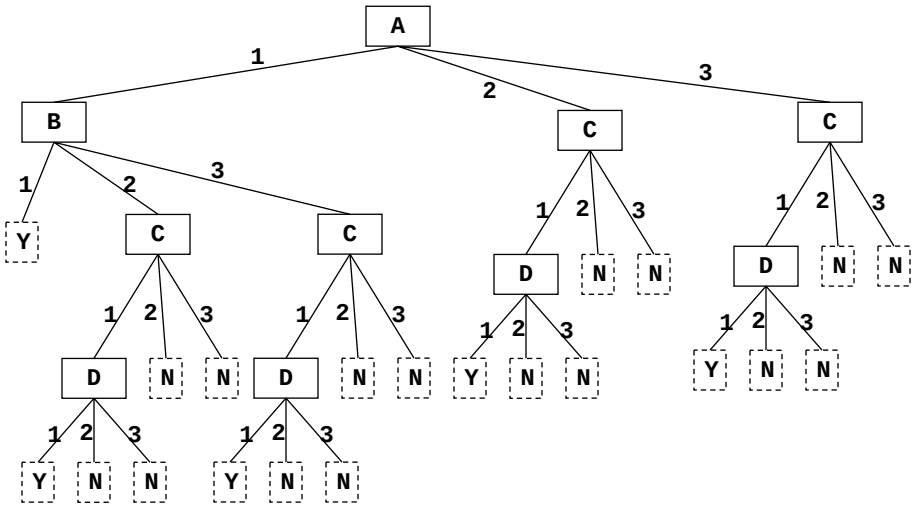
- Consider the rule set
  - Attributes A, B, C, and D can have values 1, 2, and 3

A = 1 $\wedge$ B = 1 $\rightarrow$ Class = Y
C = 1 $\wedge$ D = 1 $\rightarrow$ Class = Y
Otherwise, Class = N

- How to represent it as a decision tree?
  - The rules need a common attribute

A = 1 $\wedge$ B = 1 $\rightarrow$ Class = Y
A = 1 $\wedge$ B = 2 $\wedge$ C = 1 $\wedge$ D = 1 $\rightarrow$ Class = Y
A = 1 $\wedge$ B = 3 $\wedge$ C = 1 $\wedge$ D = 1 $\rightarrow$ Class = Y
A = 2 $\wedge$ C = 1 $\wedge$ D = 1 $\rightarrow$ Class = Y
A = 3 $\wedge$ C = 1 $\wedge$ D = 1 $\rightarrow$ Class = Y
Otherwise, Class = N

Motivation



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

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

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk ⇒ Class = *Bird*
The rule R3 covers the grizzly bear ⇒ 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)

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

(Status = 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

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

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

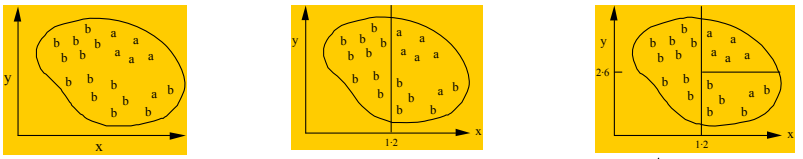
- How to learn a rule for a class C?

1. Start from an empty rule `{} → 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




```
If {}
   then class = a

If (x > 1.2)
   then class = a

If (x > 1.2) and (y > 2.6)
   then class = a
```

- Possible rule set for class "b":

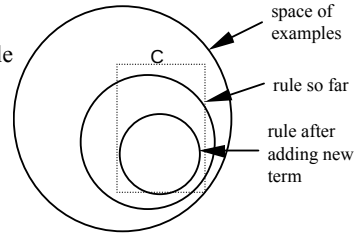
```
If (x ≤ 1.2) then class = b
If (x > 1.2) and (y ≤ 2.6) then class = b
```

- 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** = p/t



When to Stop Building a Rule

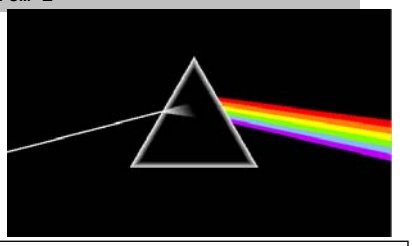
- When the rule is perfect, i.e. accuracy = 1
- When increase in accuracy gets below a given threshold
- When the training set cannot be split any further

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
```

Learn one rule

Available in ***WEKA***



Rule Evaluation in PRISM

$$\text{Accuracy} = \frac{p}{t}$$

t : Number of instances covered by rule

p : 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:

$$\text{FOIL's inf. gain} = p_1 \times (\log_2 \frac{p_1}{t_1} - \log_2 \frac{p_0}{t_0})$$

Accuracy after a new test is added

Accuracy before a new test is added

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

$$\text{m-estimate} = \frac{p+k.f}{t+k}$$

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

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

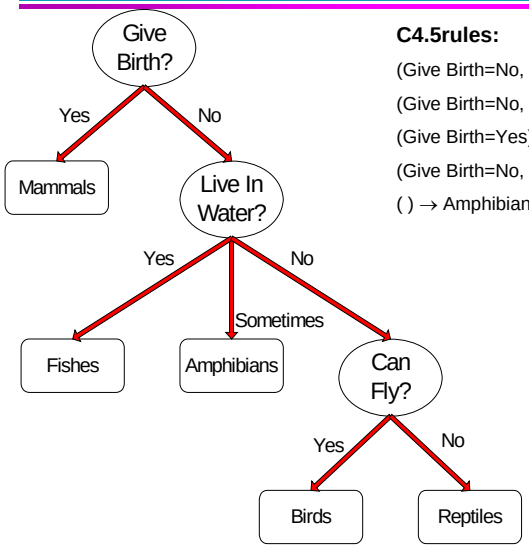
Indirect Method: C4.5rules

- Extract rules from an unpruned decision tree
- For each rule, r: 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

Name	Give Birth	Lay Eggs	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	no	yes	mammals
python	no	yes	no	no	no	reptiles
salmon	no	yes	no	yes	no	fishes
whale	yes	no	no	yes	no	mammals
frog	no	yes	no	sometimes	yes	amphibians
komodo	no	yes	no	no	yes	reptiles
bat	yes	no	yes	no	yes	mammals
pigeon	no	yes	yes	no	yes	birds
cat	yes	no	no	no	yes	mammals
leopard shark	yes	no	no	yes	no	fishes
turtle	no	yes	no	sometimes	yes	reptiles
penguin	no	yes	no	sometimes	yes	birds
porcupine	yes	no	no	no	yes	mammals
eel	no	yes	no	yes	no	fishes
salamander	no	yes	no	sometimes	yes	amphibians
gila monster	no	yes	no	no	yes	reptiles
platypus	no	yes	no	no	yes	mammals
owl	no	yes	yes	no	yes	birds
dolphin	yes	no	no	yes	no	mammals
eagle	no	yes	yes	no	yes	birds

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. Discarded 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***