

# CSE5334 DATA MINING

Lecture 8: Classification (4)

CSE4392/5334 Data Mining, Fall 2009  
Department of Computer Science and Engineering, University of Texas at Arlington  
Chengkai Li (Slides courtesy of Vipin Kumar)

## Rule-Based Classifier

### Rule-Based Classifier

- Classify records by using a collection of “if...then...” rules
- Rule: (Condition) → y
  - where
    - Condition is a conjunctions of attributes
    - y is the class label
  - LHS: rule antecedent or condition
  - RHS: rule consequent
  - Examples of classification rules:
    - (Blood Type=Warm) ∧ (Lay Eggs=Yes) → Birds
    - (Taxable Income < 50K) ∧ (Refund=Yes) → Evade=No

### 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
seal	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) ∧ (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

### Application of Rule-Based Classifier

- A rule *r* covers an instance *x* if the attributes of the instance satisfy the condition 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 => Bird  
 The rule R3 covers the grizzly bear => Mammal

### Rule Coverage and Accuracy

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

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)  $\wedge$  (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

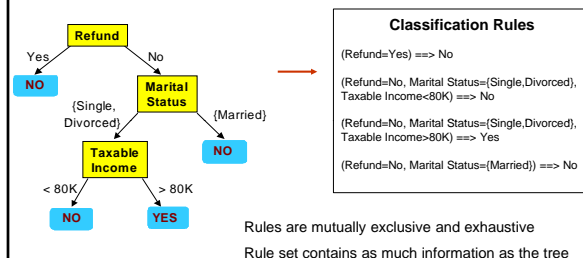
Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

A lemur triggers rule R3, so it is classified as a mammal  
 A turtle triggers both R4 and R5  
 A dogfish shark triggers none of the rules

### Characteristics of Rule-Based Classifier

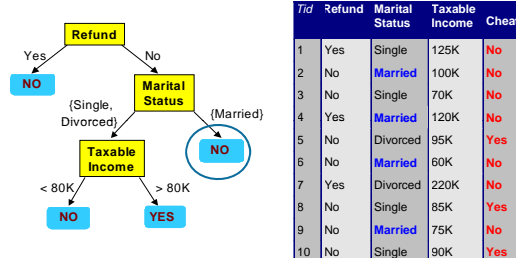
- Mutually exclusive rules
  - Classifier contains mutually exclusive rules if the rules are independent of each other
  - Every record is covered by at most one rule
- Exhaustive rules
  - Classifier has exhaustive coverage if it accounts for every possible combination of attribute values
  - Each record is covered by at least one rule

### From Decision Trees To Rules



Rules are mutually exclusive and exhaustive  
 Rule set contains as much information as the tree

### Rules Can Be Simplified



T/id	Refund	Marital Status	Taxable Income	Cheat
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

Initial Rule: (Refund=No)  $\wedge$  (Status=Married)  $\rightarrow$  No  
 Simplified Rule: (Status=Married)  $\rightarrow$  No

### Effect of Rule Simplification

- Rules are no longer mutually exclusive
  - A record may trigger more than one rule
  - Solution?
    - Ordered rule set
    - Unordered rule set – use voting schemes
- Rules are no longer exhaustive
  - A record may not trigger any rules
  - Solution?
    - Use a default class

### Ordered Rule Set

- Rules are rank ordered according to their priority
  - An ordered rule set is known as a **decision list**
- When a test record is presented to the classifier
  - It is assigned to the class label of the highest ranked rule it has triggered
  - If none of the rules fired, it is assigned to the default class

- 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

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
turtle	cold	no	no	sometimes	?

### Rule Ordering Schemes

- Rule-based ordering
  - Individual rules are ranked based on their quality
- Class-based ordering
  - Rules that belong to the same class appear together

Rule-based Ordering	Class-based Ordering
(Refund=Yes) ==> No	(Refund=Yes) ==> No
(Refund=No, Marital Status=(Single,Divorced), Taxable Income<80K) ==> No	(Refund=No, Marital Status=(Single,Divorced), Taxable Income<80K) ==> No
(Refund=No, Marital Status=(Single,Divorced), Taxable Income>80K) ==> Yes	(Refund=No, Marital Status=(Married)) ==> No
(Refund=No, Marital Status=(Married)) ==> No	(Refund=No, Marital Status=(Single,Divorced), Taxable Income>80K) ==> Yes

### Building Classification Rules

- Direct Method:
  - Extract rules directly from data
  - e.g.: RIPPER, CN2, Holte's 1R
- Indirect Method:
  - Extract rules from other classification models (e.g. decision trees, neural networks, etc).
  - e.g: C4.5rules

### Direct Method: Sequential Covering

Do the following for each class, in certain order of classes:

1. Start from an empty rule
2. Grow a rule using the **Learn-One-Rule** function
3. Remove training records covered by the rule
4. Repeat Step (2) and (3) until stopping criterion is met

### Example of Sequential Covering

(i) Original Data      (ii) Step 1

### Example of Sequential Covering...

(iii) Step 2      (iv) Step 3

### Instance Elimination

- Why do we need to eliminate instances?
  - Otherwise, the next rule is identical to previous rule

Compare rules R2 and R3 in the diagram

- Why do we remove positive instances?
  - Prevent overestimating accuracy of rule
- Why do we remove negative instances?
  - Prevent underestimating accuracy of rule

### Learn-One-Rule

- Objective: extract a rule that covers many positive examples and none (or very few) of the negative examples.
- Optimal rule: computationally expensive
- Greedy approach:
  - ▣ Grow the rule
  - ▣ Stop rule-growing when stop criterion is met
  - ▣ Prune the rule to improve it

### Learn-One-Rule

- Rule Growing
- Rule Evaluation
- Stopping Criterion
- Rule Pruning

### Rule Growing

- Two common strategies

(a) General-to-specific: A tree starting from an empty set {} (Yes: 3, No: 4) and branching into Refund=No, Status=Single, Status=Divorced, Status=Married, and Income > 80K. The Status=Married node is highlighted with a red circle.

(b) Specific-to-general: A tree starting from a specific rule (Refund=No, Status=Single, Income=80K, Class=Yes) and branching into Refund=No, Status=Single, Income=80K, Class=Yes and Refund=No, Status=Single, Income=80K, Class=Yes. The root node is highlighted with a red circle.

### Rule Evaluation

- Which conjunct to add (remove) during rule-growth?
- Metrics:
  - ▣ Accuracy =  $\frac{n_c}{n}$
  - ▣ Laplace =  $\frac{n_c + 1}{n + k}$
  - ▣ M-estimate =  $\frac{n_c + kp}{n + k}$

*n*: Number of instances covered by rule  
*n<sub>c</sub>*: Number of instances covered by rule correctly  
*k*: Number of classes  
*p*: Prior probability

### Rule Evaluation: FOIL's information Gain

- RIPPER Algorithm:
  - ▣ Start from an empty rule: {} => class
  - ▣ Add conjuncts that maximizes FOIL's information gain measure:
    - R0: {} => class (initial rule)
    - R1: {A} => class (rule after adding conjunct)
    - Gain(R0, R1) =  $p1 [ \log (p1/(p1+n1)) - \log (p0/(p0 + n0)) ]$
    - where
      - p0: number of positive instances covered by R0
      - n0: number of negative instances covered by R0
      - p1: number of positive instances covered by R1
      - n1: number of negative instances covered by R1

### Stopping Criterion and Rule Pruning

- Stopping criterion
  - ▣ Compute the gain
  - ▣ If gain is not significant, discard the new rule
- Rule Pruning
  - ▣ Similar to post-pruning of decision trees
  - ▣ Reduced Error Pruning:
    - Remove one of the conjuncts in the rule
    - Compare error rate on validation set before and after pruning
    - If error improves, prune the conjunct

### Summary of Direct Method

- Grow a single rule
- Remove Instances from rule
- Prune the rule (if necessary)
- Add rule to Current Rule Set
- Repeat

### Direct Method: RIPPER

**Class (Rule Set) Ordering**

- 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

### Direct Method: RIPPER

**Rule Growing, Stopping, Pruning:**

- 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

### Direct Method: 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 new description length
  - stop adding new rules when the new description length is d bits longer.

### Direct Method: 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

### Indirect Methods

**Rule Set**

r1: (P=No,Q=No) ==> -  
 r2: (P=No,Q=Yes) ==> +  
 r3: (P=Yes,R=No) ==> +  
 r4: (P=Yes,R=Yes,Q=No) ==> -  
 r5: (P=Yes,R=Yes,Q=Yes) ==> +

### Indirect Method: C4.5rules

- Extract rules from an unpruned decision tree
- For each rule,  $r: A \rightarrow y$ ,
  - ▣ consider an alternative rule  $r': A' \rightarrow y$  where  $A'$  is obtained by removing one of the conjuncts in  $A$
  - ▣ Compare the pessimistic error rate for  $r$  against all  $r'$ 's
  - ▣ Prune if one of the  $r'$ 's has lower pessimistic error rate
  - ▣ Repeat until we can no longer improve generalization error

### Indirect Method: C4.5rules

- Instead of ordering the rules, order subsets of rules (**class ordering**)
  - ▣ Each subset is a collection of rules with the same rule consequent (class)
  - ▣ Compute description length of each subset
    - Description length =  $L(\text{error}) + g L(\text{model})$
    - $g$  is a parameter that takes into account the presence of redundant attributes in a rule set (default value = 0.5)

### 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