# 18CSE751 – Introduction to Machine Learning
# Lecture 20,21: RULE EXTRACTION FROM TREES, LEARNING RULES FROM DATA

Dr.Vani Vasudevan

Professor –CSE, NMIT

1

# LECTURE 20

Dr.Vani V

# Rule-Based Classifier

- Classify records by using a collection of "if…then…" rules
- Rule: (*Condition*) $\rightarrow y$
  - *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) $\wedge$ (Lay Eggs=Yes) $\rightarrow$ Birds
    - (Taxable Income < 50K) $\wedge$ (Refund=Yes) $\rightarrow$ 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 |
| 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

# 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

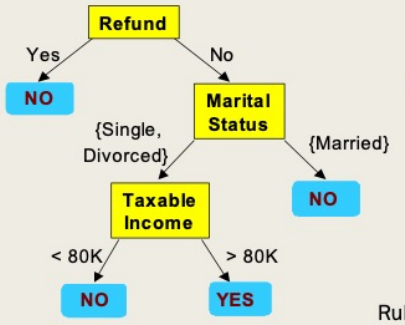
A turtle triggers both R4 and R5

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




**Classification Rules**

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

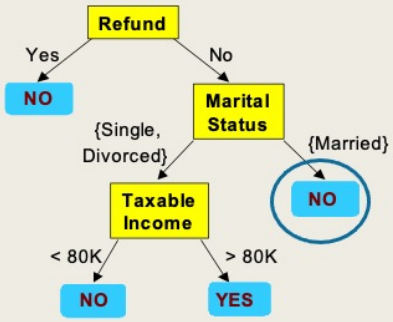
(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

(Refund=No, Marital Status={Married}) ==> No

Rules are mutually exclusive and exhaustive

Rule set contains as much information as the tree

# Rules Can Be Simplified




| Tid | 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: $(\text{Refund}=\text{No}) \wedge (\text{Status}=\text{Married}) \rightarrow \text{No}$

Simplified Rule: $(\text{Status}=\text{Married}) \rightarrow \text{No}$

# 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

# 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**

(Refund=Yes) ==> 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

**Class-based Ordering**

(Refund=Yes) ==> No

(Refund=No, Marital Status={Single,Divorced}, Taxable Income<80K) ==> No

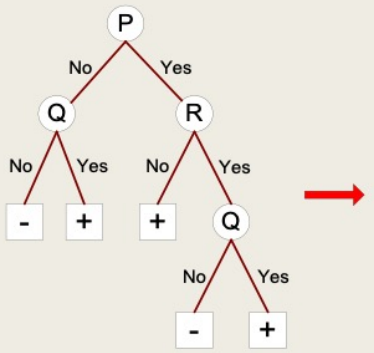
(Refund=No, Marital Status={Married}) ==> No

(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: C4.5 rules

# 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 → y,
  - *consider an alternative rule r': A' → 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(error) + g L(model)
    - g is a parameter that considers the presence of redundant attributes in a rule set (default value = 0.5)

# Sequential Covering Algorithm…

1. Used to extract rules directly from data.
2. Rules are grown in a greedy fashion based on a certain evaluation measure.
3. The algorithm extracts the rules one class at a time for data sets that contain more than two classes.
   - *For the vertebrate classification problem,*
   - *the sequential covering algorithm may generate rules for classifying birds first, followed by rules for classifying mammals, amphibians, reptiles, and finally, fishes*

## Class-Based Ordering

(Skin Cover=feathers, Aerial Creature=yes) ==> Birds

(Body temperature=warm-blooded, Gives Birth=no) ==> Birds

(Body temperature=warm-blooded, Gives Birth=yes) ==> Mammals

(Aquatic Creature=semi)) ==> Amphibians

(Skin Cover=none) ==> Amphibians

(Skin Cover=scales, Aquatic Creature=no) ==> Reptiles

(Skin Cover=scales, Aquatic Creature=yes) ==> Fishes

# Direct Method: Sequential Covering…

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

# Sequential Covering Algorithm…

The criterion for deciding which class should be generated

1. Depends on a number of factors, such as the **class prevalence** (i.e., fraction of training records that belong to a particular class) or
2. The **cost of misclassifying records** from a given class.

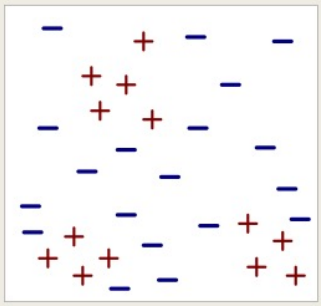
**Algorithm 5.1** Sequential covering algorithm.

1: Let $E$ be the training records and $A$ be the set of attribute-value pairs, $\{(A_j, v_j)\}$.
2: Let $Y_o$ be an ordered set of classes $\{y_1, y_2, \ldots, y_k\}$.
3: Let $R = \{\ \}$ be the initial rule list.
4: **for** each class $y \in Y_o - \{y_k\}$ **do**
5: **while** stopping condition is not met **do**
6: $r \leftarrow$ Learn-One-Rule ($E$, $A$, $y$).
7: Remove training records from $E$ that are covered by $r$.
8: Add $r$ to the bottom of the rule list: $R \longrightarrow R \vee r$.
9: **end while**
10: **end for**
11: Insert the default rule, $\{\} \longrightarrow y_k$, to the bottom of the rule list $R$.

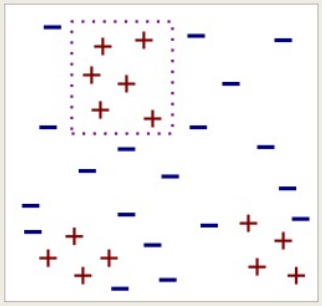
# Sequential Covering Algorithm…

1. The algorithm **starts with an empty decision list, R.**
2. The **Learn-One- Rule function** is then used to **extract the best rule for class y that covers the current set of training records.**
3. During rule extraction, **all training records for class y are considered to be positive examples,** while **those that belong to other classes are considered to be negative examples.**
4. A **rule is desirable if it covers most of the positive examples and none (or very few) of the negative examples.** Once such a rule is found, **the training records covered by the rule are eliminated.**
5. **The new rule is added to the bottom of the decision list R.**
6. This procedure is **repeated until the stopping criterion is met.**
7. The algorithm then proceeds to generate rules for the next class.

# Example of Sequential Covering...

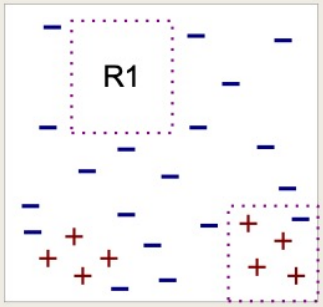


(i) Original Data

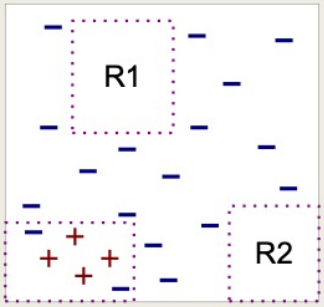


(ii) Step 1

# Example of Sequential Covering




(iii) Step 2



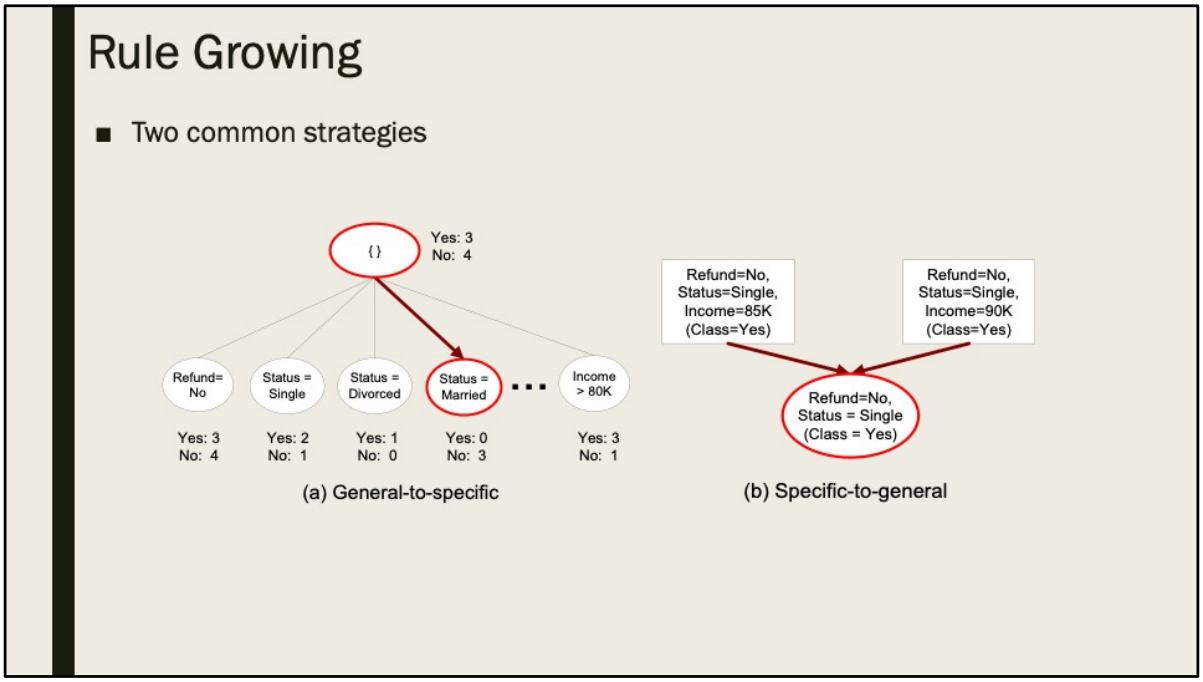

(iv) Step 3

## Learn-One Rule Function

- The **objective of the Learn-One-Rule function is to extract a classification rule that covers many of the positive examples** and none (or very few) of the negative examples in the training set.
- Finding an optimal rule is computationally expensive given the exponential size of the search space.
- The **Learn-One-Rule function addresses the exponential search problem by growing the rules in a greedy fashion.**
- It generates an **initial rule r and keeps refining the rule until a certain stopping criterion is met.**
- The rule is then pruned to improve its generalization error.

# Aspects of Sequential Covering

- Rule Growing
- Instance Elimination
- Rule Evaluation
- Stopping Criterion
- Rule Pruning




General to specific:

r : {} → y  covers all the training samples – poor quality

Conjunct each antecedent

This process continues until the stopping criterion is met (e.g., when the added conjunct does not improve the quality of the rule).

These approaches may produce suboptimal rules because the rules are grown in a greedy fashion.

To avoid this problem**, a beam search may be used**, where k of the best candidate rules are maintained by the algorithm.

Each candidate rule is then grown separately by adding (or removing) a conjunct from its antecedent. The quality of the candidates are evaluated, and the k best candidates are chosen for the next iteration.

# Rule Evaluation…

- 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

$n_c$ : Number of c instances covered by rule

$k$ : Number of classes

$p$ : Prior probability

# Rule Evaluation…

- An evaluation metric is needed to determine which conjunct should be added (or removed) during the rule-growing process.
- Accuracy is an obvious choice because it explicitly measures the fraction of training examples classified correctly by the rule.
- However, **a potential limitation of accuracy is that it does not consider the rule's coverage.**
- For example, consider a training set that contains 60 positive examples and 100 negative examples. Suppose we are given the following two candidate rules:
  - *Rule r1: covers 50 positive examples and 5 negative examples,*
  - *Rule r2: covers 2 positive examples and no negative examples.*
- The accuracies for r1 and r2 are 90.9% and 100%, respectively.
  - *r1 is the better rule despite its lower accuracy. The high accuracy for r2 is potentially spurious because the coverage of the rule is too low.*

# Rule Evaluation…

- The following approaches can be used to handle this problem.

**1. A statistical test can be used to prune rules that have poor coverage.**

- **Likelihood ratio statistic:**

$$R = 2\sum_{i=1}^{k} f_i \log(f_i/e_i),$$

where $k$ is the number of classes,

$f_i$ is the observed frequency of class $i$ examples that are covered by the rule, and

$e_i$ is the expected frequency of a rule that makes random predictions.

# Rule Evaluation…

*Rule r1: covers 50 positive examples and 5 negative examples,*
*Rule r2: covers 2 positive examples and no negative examples.*

- For example, since *r1* covers 55 examples,
  - $e_+ = 55x\ 60/160 = 20.625$,

    $e_- = 55\ x100/160 = 34.375$.
- The likelihood ratio for r1 is $R(r_1) = 2 \times [50 \times \log_2(50/20.625) + 5 \times \log_2(5/34.375)] = 99.9.$
- Similarly, $e_+ = 2x\ 60/160 = 120/160 = ¾ = 0.75$, $e_- = 0$ , the likelihood ratio for r2 is

$$R(r_2) = 2 \times [2 \times \log_2(2/0.75) + 0 \times \log_2(0/1.25)] = 5.66.$$

- This statistic therefore suggests that **r1 is a better rule than r2.**

# Rule Evaluation…

*Rule r1: covers 50 positive examples and 5 negative examples,*

*Rule r2: covers 2 positive examples and no negative examples.*

2. An **evaluation metric that considers the rule coverage can be used.**

$$\text{Laplace} = \frac{f_+ + 1}{n + k},$$

$$\text{m-estimate} = \frac{f_+ + kp_+}{n + k},$$

where , n is the number of examples covered by the rule,

$f_+$ is the number of positive examples covered by the rule,

k is the total number of classes, and

$p_+$ is the prior probability for the positive class.

**Note : m-estimate is equivalent to the Laplace measure by choosing $p_+ = 1/k$.**

The Laplace measure for r1 is 51/57 = 89.47%, r2 is 3/4=75%

# LECTURE 21

Dr.Vani V

# Rule Evaluation…

3. An evaluation metric that considers the **support count** of the rule can be used. One such metric is the **FOIL's information gain.**

- The **support count of a rule corresponds to the number of positive examples covered by the rule.**

- Suppose the rule r : A ⟶ +
  - *covers $p_0$ positive examples and $n_0$ negative examples.*
- After adding a new conjunct B, the extended rule r : A ∧ B ⟶ +
  - *covers p1 positive examples and n1 negative examples.*
- Given this information, the FOIL's information gain of the extended rule is defined as follows:

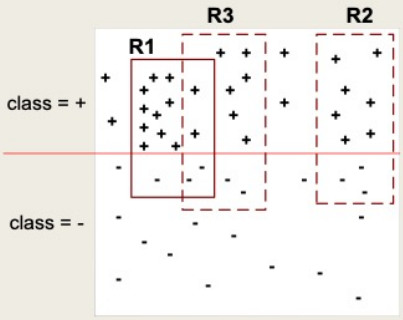
$$\text{FOIL's information gain} = p_1 \times \left( \log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right).$$

# Rule Growing (Examples)

- CN2 Algorithm:
  - *Start from an empty conjunct: {}*
  - *Add conjuncts that minimizes the entropy measure: {A}, {A,B}, …*
  - *Determine the rule consequent by taking majority class of instances covered by the rule*
- 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) = t [ log (p1/(p1+n1)) – log (p0/(p0 + n0)) ]
    - where t: number of positive instances covered by both R0 and R1
      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

# Instance Elimination

- Why do we need to eliminate instances?
  - *Otherwise, the next rule is identical to previous rule*
- Why do we remove positive instances?
  - *Ensure that the next rule is different*
- Why do we remove negative instances?
  - *Prevent underestimating accuracy of rule*
  - *Compare rules R2 and R3 in the diagram*



# Stopping Criterion and Rule Pruning

- Stopping criterion
  - *Compute the gain*
  - *If gain is not significant, discard the new rule*

- Rule Pruning
  - *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

- 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

- Growing a rule:
  - *Start from empty rule*
  - *Add conjuncts if 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 than the smallest description length obtained so far

Description length = L(error) + g L(model)
g is a parameter that considers the presence of redundant attributes in a rule set
(default value = 0.5)

## 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 (Minimum Description Length) principle
  - *Repeat rule generation and rule optimization for the remaining positive examples*

Page 183Compute description length of each subset

Description length = L(error) + g L(model)

g is a parameter that considers the presence of redundant attributes in a rule set

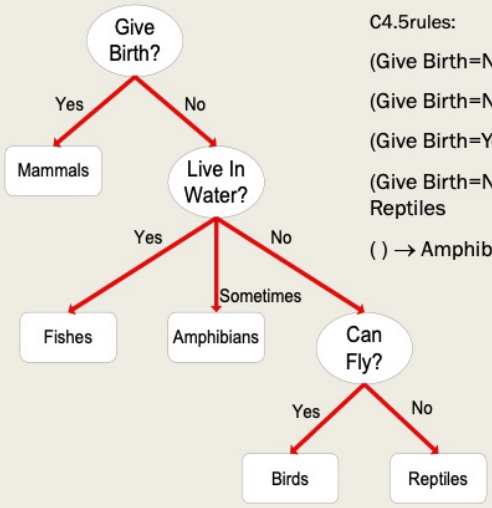
(default value = 0.5)

# 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 |

Total 20 instances
Count of fishes -3
Count of reptiles – 4
Count of birds – 4
Count of amphibians - 2
Count of mammals -7

# 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

## 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

## Characteristics of Rule-Based Classifiers

A rule-based classifier has the following characteristics:

- The expressiveness of a rule set is almost equivalent to that of a decision tree
- **If the rule-based classifier allows multiple rules to be triggered for a given record, then a more complex decision boundary can be constructed.**
- **Rule-based classifiers are generally used to produce descriptive models that are easier to interpret but gives comparable performance to the decision tree classifier.**
- The class-based ordering approach adopted by many rule-based classifiers (such as **RIPPER) is well suited for handling data sets with imbalanced class distributions.**

## Selected Exercises: 1...

- Consider a binary classification problem with the following set of attributes and attribute values:
  - Air Conditioner = {Working, Broken}
  - Engine = {Good, Bad}
  - Mileage = {High, Medium, Low}
  - Rust = {Yes, No}
- Suppose a rule-based classifier produces the following rule set:

  r1: Mileage = High $\longrightarrow$ Value = Low

  r2: Mileage = Low $\longrightarrow$ Value = High

  r3: Air Conditioner = Working, Engine = Good $\longrightarrow$ Value = High

  r4: Air Conditioner = Working, Engine = Bad $\longrightarrow$ Value = Low

  r5: Air Conditioner = Broken $\longrightarrow$ Value = Low

## Selected Exercises:1

(a) Are the rules mutually exclusive?

No

(b) Is the rule set exhaustive?

Yes

(c) Is ordering needed for this set of rules?

Yes, because a test instance may trigger more than one rule.

(d) Do you need a default class for the rule set?

No, because every instance is guaranteed to trigger at least one rule.

# Selected Exercises:2...

Given:

Suppose R1 is covered by 350 positive examples and 150 negative examples, while R2 is covered by 300 positive examples and 50 negative examples. Compute the FOIL's information gain for the rule R2 with respect to R1.

Solution:

$$\text{FOIL's information gain} = p_1 \times \left( \log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right).$$

For this problem, p0 = 350, n0 = 150, p1 = 300, and n1 = 50. Therefore, the FOIL's information gain for R2 with respect to R1 is:

$$Gain = 300\ x\ [log_2 \frac{300}{350} - log_2 \frac{350}{500}] \quad = 87.65$$

2. The RIPPER algorithm (by Cohen [170]) is an extension of an earlier algorithm called IREP (by Fürnkranz and Widmer [184]). Both algorithms apply the **reduced-error pruning** method to determine whether a rule needs to be pruned. The reduced error pruning method uses a validation set to estimate the generalization error of a classifier. Consider the following pair of rules:

$$R_1: \quad A \longrightarrow C$$
$$R_2: \quad A \wedge B \longrightarrow C$$

$R_2$ is obtained by adding a new conjunct, $B$, to the left-hand side of $R_1$. For this question, you will be asked to determine whether $R_2$ is preferred over $R_1$ from the perspectives of rule-growing and rule-pruning. To determine whether a rule should be pruned, IREP computes the following measure:

$$v_{IREP} = \frac{p + (N - n)}{P + N},$$

where $P$ is the total number of positive examples in the validation set, $N$ is the total number of negative examples in the validation set, $p$ is the number of positive examples in the validation set covered by the rule, and $n$ is the number of negative examples in the validation set covered by the rule. $v_{IREP}$ is actually similar to classification accuracy for the validation set. IREP favors rules that have higher values of $v_{IREP}$. On the other hand, RIPPER applies the following measure to determine whether a rule should be pruned:

$$v_{RIPPER} = \frac{p - n}{p + n}.$$

# Selected Exercises:2...

Given : Consider a validation set that contains 500 positive examples and 500 negative examples. For R1, suppose the number of positive examples covered by the rule is 200, and the number of negative examples covered by the rule is 50. For R2, suppose the number of positive examples covered by the rule is 100 and the number of negative examples is 5.

(1) Compute $vIREP$ for both rules. Which rule does $IREP$ prefer?

# Selected Exercises:2...

(1) For the given problem, P = 500, and N = 500.

For rule R1, p = 200 and n = 50. Therefore,

$$V_{IREP}(R1) = \frac{p + (N - n)}{P + N}$$

$$= \frac{200 + (500 - 50)}{1000} = 0.65$$

For rule R2, p = 100 and n = 5.

$$V_{IREP}(R2) = \frac{p + (N - n)}{P + N}$$

$$= \frac{100 + (500 - 5)}{1000} = 0.595$$

Thus, IREP prefers rule R1.

# Selected Exercises:2

(2) Compute $v_{RIPPER}$ for the given problem. Which rule does RIPPER prefer?

For rule R1, p = 200 and n = 50.

$$V_{RIPPER}(R1) = \frac{p-n}{p+n}$$

$$= \frac{200-50}{200+50} = 0.6$$

For rule R2, p = 100 and n = 5.

$$V_{RIPPER}(R2) = \frac{p+n}{p+n}$$

$$= \frac{100-5}{100+5} = 0.9$$

Thus, RIPPER prefers rule R2.

# Selected Exercises:3...

C4.5rules is an implementation of an indirect method for generating rules from a decision tree. RIPPER is an implementation of a direct method for generating rules directly from data.

**(a) Discuss the strengths and weaknesses of both methods.**

The C4.5 rules algorithm generates classification rules from a global perspective. This is because the rules are derived from decision trees, which are induced with the objective of partitioning the feature space into homogeneous regions, without focusing on any classes. In contrast, RIPPER generates rules one-class-at-a-time. Thus, it is more biased towards the classes that are generated first.

# Selected Exercises:3

C4.5 rules is an implementation of an indirect method for generating rules from a decision tree. RIPPER is an implementation of a direct method for generating rules directly from data.

**(b) Consider a data set that has a large difference in the class size (i.e., some classes are much bigger than others). Which method (between C4.5 rules and RIPPER) is better in terms of finding high accuracy rules for the small classes?**

The class-ordering scheme used by C4.5 rules has an easier interpretation than the scheme used by RIPPER.

# Selected Exercises:4...

Consider a training set that contains 100 positive examples and 400 negative examples. For each of the following candidate rules,

R1: A → + (covers 4 positive and 1 negative examples),

R2: B → + (covers 30 positive and 10 negative examples),

R3: C → + (covers 100 positive and 90 negative examples),

Determine which is the best and worst candidate rule according to:

(a) Rule accuracy.

(b) FOIL's information gain.

(c) The likelihood ratio statistic.

(d) The Laplace measure.

(e) The m-estimate measure (with $k = 2$ and $p+ = 0.2$).

# Selected Exercises:4...

(a) Rule accuracy

Solution: R1 : 4/5

R2: 30/40

R3:100/190

The accuracies of the rules are 80% (for R1), 75% (for R2), and 52.6% (for R3), respectively. Therefore, R1 is the best candidate and R3 is the worst candidate according to rule accuracy

# Selected Exercises:4...

(b) Foil's information gain

Solution: Assume the initial rule is $\emptyset \rightarrow +$. This rule covers p0 = 100 positive examples and n0 = 400 negative examples. The rule R1 covers p1 = 4 positive examples and n1 = 1 negative example. Therefore, the FOIL's information gain for this rule is

$$\text{FOIL's information gain} = p_1 \times \left( \log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right).$$

$$Gain(R1) = 4\,x\,[log_2\tfrac{4}{5} - log_2\tfrac{100}{500}] \quad = 8$$

$$Gain(R2) = 57.2$$

$$Gain(R3) = 139.6$$

Therefore, R3 is the best candidate and R1 is the worst candidate according to FOIL's information gain.

4x(-.3219+2.322) = 8.004

# Selected Exercises:4...

(c)The likelihood ratio statistic: $R = 2\sum_{i=1}^{k} f_i \log(f_i/e_i),$

Solution:

The given training set that contains 100 positive examples and 400 negative examples. For each of the following candidate rules,

R1: A ⟶ + (covers 4 positive and 1 negative examples),

For R1, the expected frequency for the positive class is 5 x 100/500 = 1 and the expected frequency for the negative class is 5 x 400/500 = 4. Therefore, the likelihood ratio for R1 is

R(R1) = $2\,x\,[4x\,log_2\frac{4}{1} + 1x\,log_2\frac{1}{4}]$ = 12

# Selected Exercises:4...

(c)The likelihood ratio statistic: $R = 2\sum_{i=1}^{k} f_i \log(f_i/e_i),$

Solution:

The given training set that contains 100 positive examples and 400 negative examples. For each of the following candidate rules,

R2: B ⟶ + (covers 30 positive and 10 negative examples),

For R2, the expected frequency for the positive class is 40x 100/500 = 8 and the expected frequency for the negative class is 40 x400/500 = 32. Therefore, the likelihood ratio for R2 is

R(R2) = $2\,x\,[30x\ log_2\frac{30}{8} + 10x\ log_2\frac{10}{32}]$ = 80.85

# Selected Exercises:4...

(c)The likelihood ratio statistic: $R = 2\sum_{i=1}^{k} f_i \log(f_i/e_i),$

Solution:

The given training set that contains 100 positive examples and 400 negative examples. For each of the following candidate rules,

R3: C ⟶ + (covers 100 positive and 90 negative examples),

For R3, the expected frequency for the positive class is 190 x 100/500 = 38 and the expected frequency for the negative class is 190x 400/500 = 152. Therefore, the likelihood ratio for R3 is

R(R3) = $2\,x\,[100\,x\,log_2\frac{100}{38} + 90x\,log_2\frac{90}{152}]$ = 143.09

Therefore, R3 is the best candidate and R1 is the worst candidate according to the likelihood ratio statistic.

# Selected Exercises:4...

(d) The Laplace measure. $$\text{Laplace} = \frac{f_+ + 1}{n + k},$$

*where , $n$ is the number of examples covered by the rule,*
*$f_+$ is the number of positive examples covered by the rule,*
*$k$ is the total number of classes, and*
*$p_+$ is the prior probability for the positive class.*

**Given:** Consider a training set that contains 100 positive examples and 400 negative examples. For each of the following candidate rules,

R1: A ⟶ + (covers 4 positive and 1 negative examples),

R2: B ⟶ + (covers 30 positive and 10 negative examples),

R3: C ⟶ + (covers 100 positive and 90 negative examples),

**Solution:**

- The Laplace measure of the rules are 71.43% (for R1), 73.81% (for R2), and 52.6% (for R3), respectively. **Therefore, R2 is the best candidate and R3 is the worst candidate according to the Laplace measure.**

4+1/5+2  5/7 = 71.43%
30+1/42 = 73.81%
101/192 = 52.6%

# Selected Exercises:4

e) The m-estimate measure (with k = 2 and p+ = 0.2).

$$\text{m-estimate} = \frac{f_+ + kp_+}{n+k},$$

**Given:** Consider a training set that contains 100 positive examples and 400 negative examples. For each of the following candidate rules,

R1: A ⟶ + (covers 4 positive and 1 negative examples),

R2: B ⟶ + (covers 30 positive and 10 negative examples),

R3: C ⟶ + (covers 100 positive and 90 negative examples),

**Solution:**

The m-estimate measure of the rules are 62.86% (for R1), 73.38% (for R2), and 52.3% (for R3), respectively. Therefore, R2 is the best candidate and R3 is the worst candidate according to the m-estimate measure.

# LECTURE 22

Dr.Vani V

# Selected Exercises:5

Consider the one-dimensional data set shown in below table.

| x | 0.5 | 3.0 | 4.5 | 4.6 | 4.9 | 5.2 | 5.3 | 5.5 | 7.0 | 9.5 |
|---|---|---|---|---|---|---|---|---|---|---|
| y | − | − | + | + | + | − | − | + | − | − |

(a) Classify the data point x = 5.0 according to its 1-, 3-, 5-, and 9-nearest neighbors (using majority vote).

(b) Repeat the previous analysis using the distance-weighted voting approach

**Solution:**

Majority Voting: $y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i, y_i) \in D_z} I(v = y_i)$,

(a)
1-nearest neighbor: +,
3-nearest neighbor: −,
5-nearest neighbor: +,
9-nearest neighbor: −.

Distance-Weighted Voting: $y' = \underset{v}{\operatorname{argmax}} \sum_{(x_i, y_i) \in D_z} w_i \times I(v = y_i)$.

(b)
1-nearest neighbor: +,
3-nearest neighbor: +,
5-nearest neighbor: +,
9-nearest neighbor: +.

$w_i = 1/d(x', x_i)^2$.
$1/(5\text{-}4.9)^2$

(1) v = + ; 100 x 1 = 100 + 25x0 +11.11 x0 = 100
v = - ; 0+25+11.11 = 36.11
Max(+,-) = +

# Example: PEBLS

- PEBLS: Parallel Examplar-Based Learning System (Cost & Salzberg)
  - *Works with both continuous and nominal features*
    - For nominal features, distance between two nominal values is computed using Modified Value Difference Metric (MVDM)
  - *Each record is assigned a weight factor*
  - *Number of nearest neighbor, k = 1*

To deal with categorical attributes

## Example: PEBLS

| Tid | 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 |

Distance between nominal attribute values:

d(Single,Married)
= | 2/4 – 0/4 | + | 2/4 – 4/4 | = 1
d(Single,Divorced)
= | 2/4 – 1/2 | + | 2/4 – 1/2 | = 0
d(Married,Divorced)
= | 0/4 – 1/2 | + | 4/4 – 1/2 | = 1
d(Refund=Yes,Refund=No)
= | 0/3 – 3/7 | + | 3/3 – 4/7 | = 6/7

| Class | Marital Status | | |
|---|---|---|---|
| | Single | Married | Divorced |
| Yes | 2 | 0 | 1 |
| No | 2 | 4 | 1 |

| Class | Refund | |
|---|---|---|
| | Yes | No |
| Yes | 0 | 3 |
| No | 3 | 4 |

$$d(V_1,V_2) = \sum_i \left| \frac{n_{1i}}{n_1} - \frac{n_{2i}}{n_2} \right|$$

where nij is the number of examples from class i with attribute value Vj and nj is the number of examples with attribute value Vj .

# Example: PEBLS

| *Tid* | Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|---|
| X | Yes | Single | 125K | No |
| Y | No | Married | 100K | No |

Distance between record X and record Y:

$$\Delta(X,Y) = w_X w_Y \sum_{i=1}^{d} d(X_i, Y_i)^2$$

where: $w_X = \dfrac{\text{Number of times X is used for prediction}}{\text{Number of times X predicts correctly}}$

$w_X \cong 1$ if X makes accurate prediction most of the time

$w_X > 1$ if X is not reliable for making predictions

# Selected Exercises:6...

Consider the training set for the loan classification problem shown in Figure

Use the Modified Value Difference Metric (MVDM) measure to compute the distance between every pair of attribute values for the **Home Owner and Marital Status attributes.**

| | binary | categorical | continuous | class |
|---|---|---|---|---|
| Tid | Home Owner | Marital Status | Annual Income | Defaulted Borrower |
| 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 |

## Selected Exercises:6

| | binary | categorical | continuous | class |
|---|---|---|---|---|
| Tid | Home Owner | Marital Status | Annual Income | Defaulted Borrower |
| 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 |

| | Marital Status | | |
|---|---|---|---|
| Class | Single | Married | Divorced |
| Yes | 2 | 0 | 1 |
| No | 2 | 4 | 1 |

| | Home Owner | |
|---|---|---|
| Class | Yes | No |
| Yes | 0 | 3 |
| No | 3 | 4 |

$d(\text{Single, Married}) = 1$

$d(\text{Single, Divorced}) = 0$

$d(\text{Married, Divorced}) = 1$

$d(Home\ Owner = yes, Home\ Owner = no) = 6/7$

d(Single,Married) = abs(2/4 – 0/4) + abs(2/4 – 4/4) = 0.5+0.5 = 1

d(Home Owner = yes, Home Owner = no) = 3/7+(3/3-4/7) =3/7+3/7 = 6/7

# References

TEXTBOOKS :

1. Pang-Ning Tan, Vipin Kumar, Michael Steinbach: **Introduction to Data Mining**, Pearson, 2012.
2. Jiawei Han and Micheline Kamber: **Data Mining - Concepts and Techniques**, 3rd Edition, MorganKaufmann Publisher, 2014