UNIT III - CLASSIFICATION

18CS54 – DATA MINING

Outline

- Recap
- Classification
  - *Definition*
  - *Illustrating Classification Task*
  - *Classification Techniques*
- Decision Tree
  - *Decision Tree Induction*
  - *Hunt's Algorithm*
  - *Measure of Node Impurity*
    - GINI
    - Entropy
    - Misclassification Error
  - *CART , SLIQ, SPRINT*
  - *C4.5*
- Rule based Classifiers
- Nearest Neighbor classifiers

LECTURE 18

Dr.Vani V

Model Evaluation

- Metrics for Performance Evaluation
  - *How to evaluate the performance of a model?*

Metrics for Performance Evaluation

- Focus on the predictive capability of a model
  - *Rather than how fast it takes to classify or build models, scalability, etc.*
- Confusion Matrix:

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Metrics for Performance Evaluation…

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

LECTURE 19

Dr.Vani V

Splitting Based on INFO...

- Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;

n_i is number of records in partition i

– *Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)*

– *Used in ID3 and C4.5*

– *Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.*

Splitting Based on INFO...

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \qquad SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node, p is split into k partitions

n_i is the number of records in partition i

- *Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!*
- *Used in C4.5*
- *Designed to overcome the disadvantage of Information Gain*

Splitting Criteria based on Classification Error

- Classification error at a node t :

$$\boxed{Error(t) = 1 - \max_{i} P(i \mid t)}$$

- Measures misclassification error made by a node.
  - Maximum (1 - 1/n_c) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

Examples for Computing Error

$$Error(t) = 1 - \max_i P(i \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Splitting Criteria

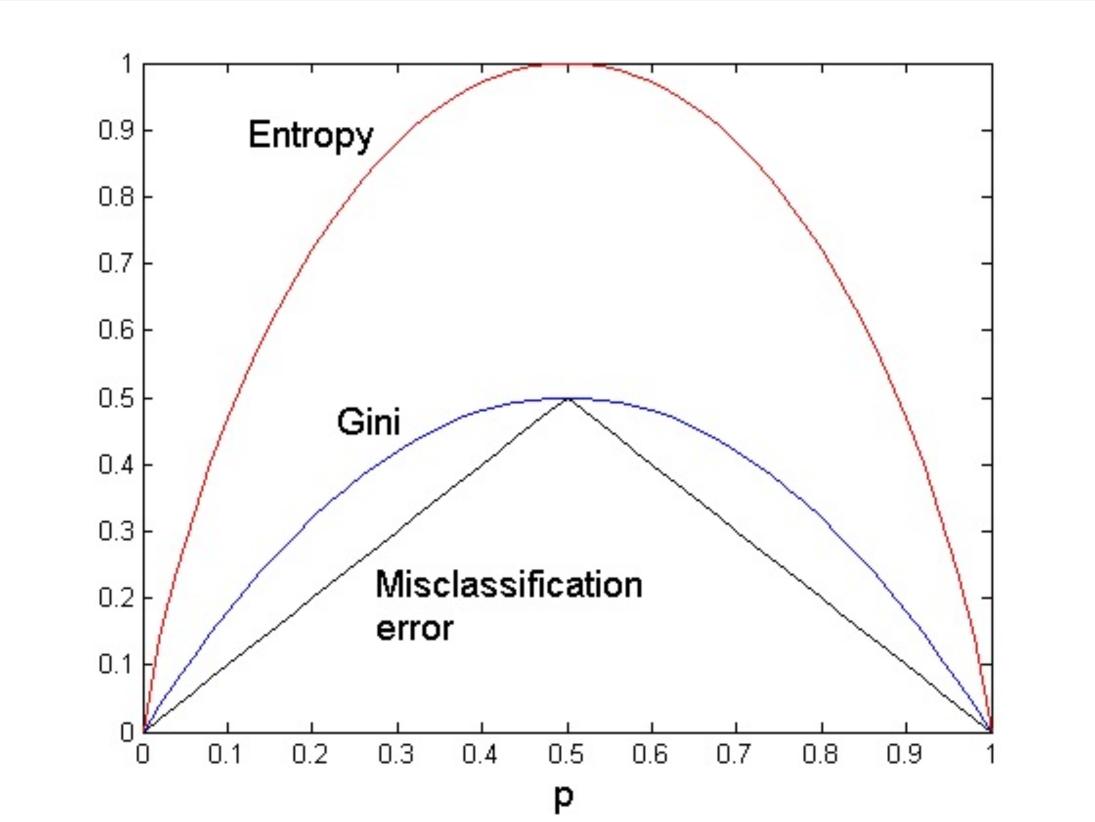
For a 2-class problem:

compares the values of the impurity measures for binary classification problems.

p refers to the fraction of records that belong to one of the two classes.

Observe that all three measures attain their maximum value when the class distribution is uniform (i.e., when p = 0.5).

The minimum values for the measures are attained when all the records belong to the same class (i.e., when p equals 0 or 1).



Tree Induction

- Greedy strategy.
  - *Split the records based on an attribute test that optimizes certain criterion.*

- Issues
  - *Determine how to split the records*
    - How to specify the attribute test condition?
    - How to determine the best split?
  - *Determine when to stop splitting*

Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
- Early termination

Decision Tree Based Classification

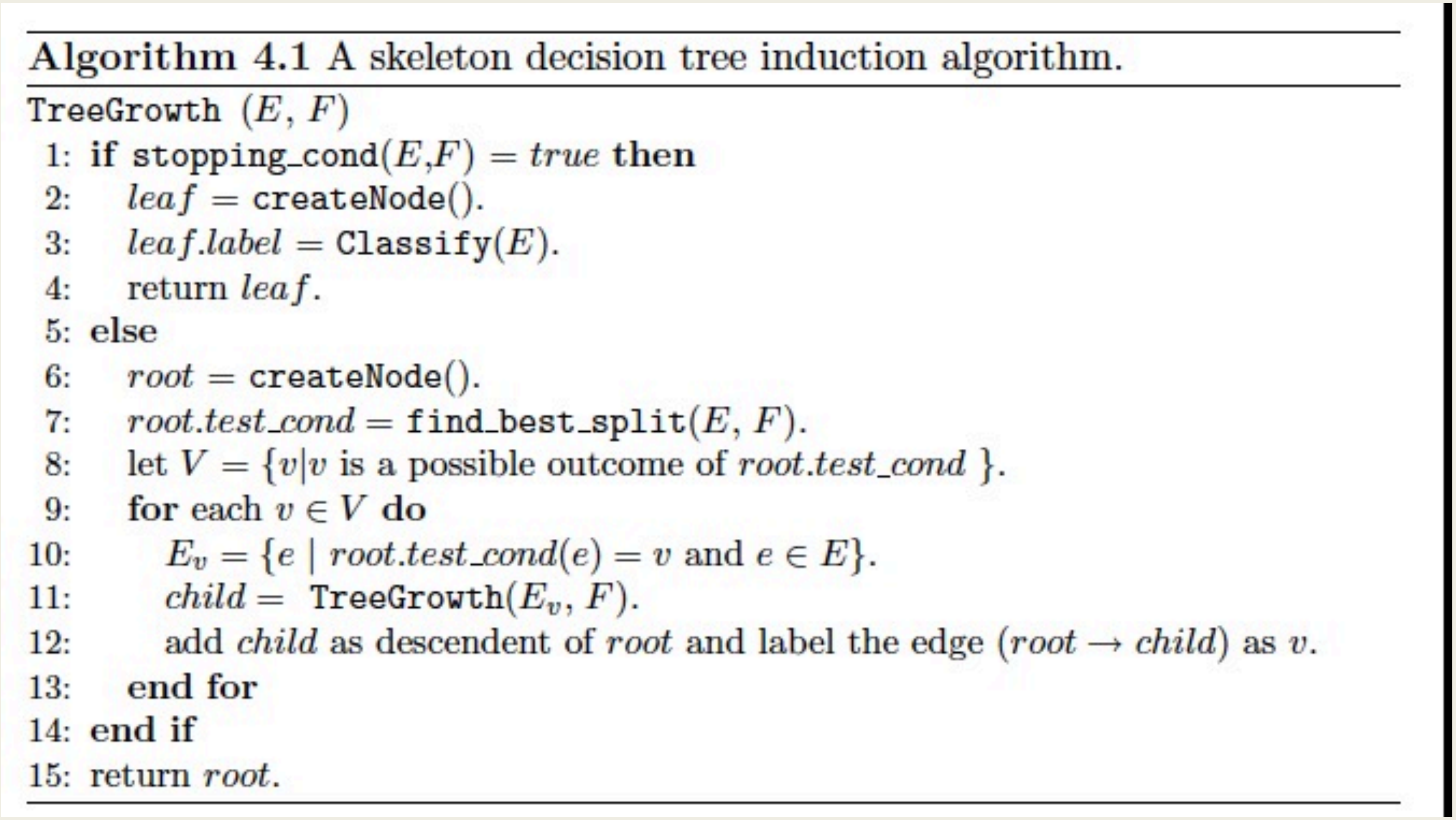
- Advantages:
  - *Inexpensive to construct*
  - *Extremely fast at classifying unknown records*
  - *Easy to interpret for small-sized trees*
  - *Accuracy is comparable to other classification techniques for many simple data sets*

Example: C4.5

- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - *Needs out-of-core sorting.*

- You can download the software from: http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz

Algorithm for Decision Tree Induction



Algorithm 4.1 A skeleton decision tree induction algorithm.

```
TreeGrowth $(E, F)$
if stopping_cond$(E,F) = true$ then
    $leaf$ = createNode().
    $leaf.label$ = Classify$(E)$.
    return $leaf$.
else
    $root$ = createNode().
    $root.test\_cond$ = find_best_split$(E, F)$.
    let $V = \{v | v$ is a possible outcome of $root.test\_cond\}$.
    for each $v \in V$ do
        $E_v = \{e \mid root.test\_cond(e) = v$ and $e \in E\}$.
        $child$ = TreeGrowth$(E_v, F)$.
        add $child$ as descendent of $root$ and label the edge $(root \rightarrow child)$ as $v$.
    end for
end if
return $root$.
```

Algorithm for Decision Tree Induction

- The input to this algorithm consists of the training records E and the attribute set F.
- The algorithm works by recursively selecting the best attribute to split the data (Step 7) and expanding the leaf nodes of the tree (Steps 11 and 12) until the stopping criterion is met (Step 1).

1. The **createNode()** function extends the decision tree by creating a new node. A node in the decision tree has either a test condition, denoted as **node.test cond**, or a class label, denoted as **node.label.**

Algorithm for Decision Tree Induction

2. The find best **split() function** determines which attribute should be selected as the test condition for splitting the training records. As previously noted, the choice of test condition depends on which impurity measure is used to determine the goodness of a split. Some widely used measures include **entropy, the Gini index, and the χ2 statistic**.

3. The **classify()** function determines the class label to be assigned to a leaf node. For each leaf node t, let p(i|t) denote the fraction of training records from class i associated with the node t. In most cases, the leaf node is assigned to the class that has the majority number of training records

Algorithm for Decision Tree Induction

$$leaf.label = \operatorname*{argmax}_{i} \; p(i|t),$$

where the argmax operator returns the argument i that maximizes the expression p(i|t). Besides providing the information needed to determine the class label of a leaf node, the fraction p(i|t) can also be used to estimate the probability that a record assigned to the leaf node t belongs to class i.

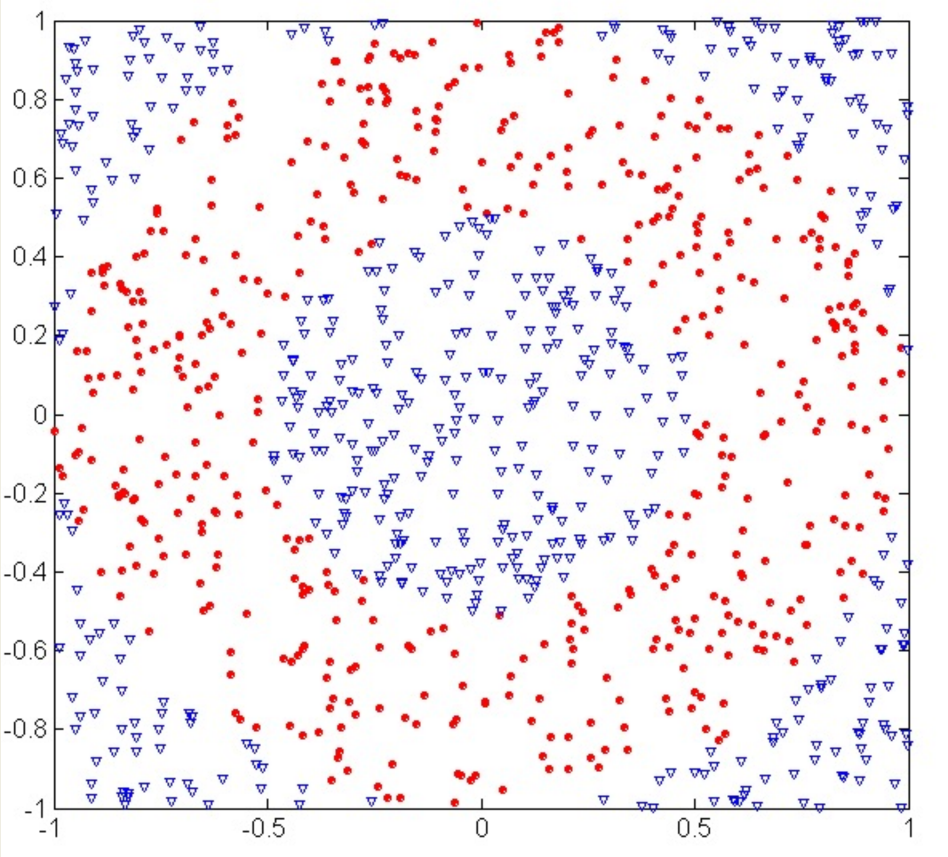
Algorithm for Decision Tree Induction

4. The **stopping cond()** function is used to terminate the tree-growing process by testing whether all the records have either the same class label or the same attribute values. Another way to terminate the recursive function is to test whether the number of records have fallen below some minimum threshold.

Practical Issues of Classification

- Underfitting and Overfitting
- Missing Values
- Costs of Classification

Underfitting and Overfitting (Example)



500 circular and 500 triangular data points.

Circular points:

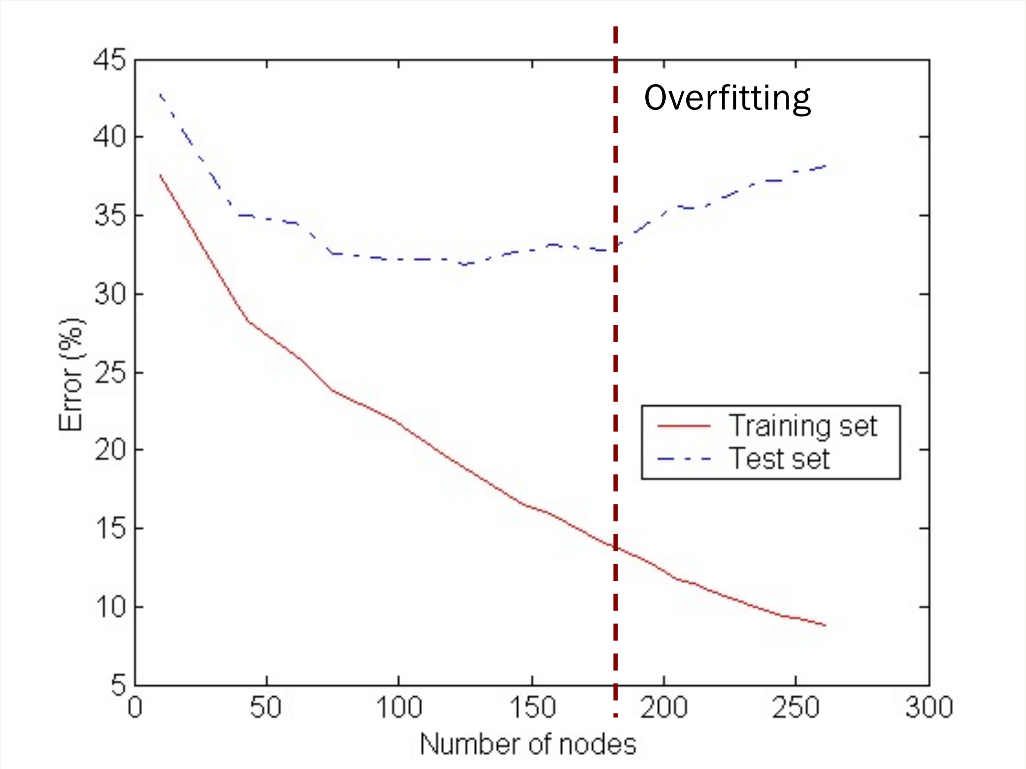
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

$\text{sqrt}(x_1^2+x_2^2) > 0.5$ or

$\text{sqrt}(x_1^2+x_2^2) < 1$

Underfitting and Overfitting



Underfitting: when model is too simple, both training and test errors are large

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary

- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records

- Need new ways for estimating errors

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

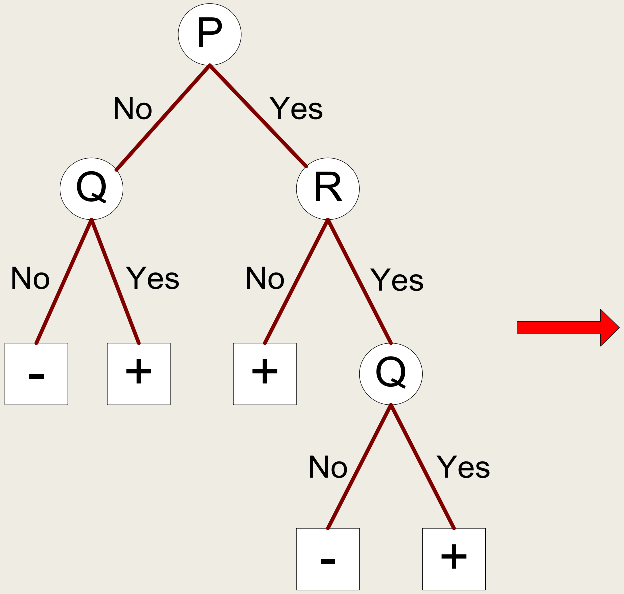
(Refund=No, Marital Status={Single,Divorced}, Taxable Income>80K) ==> Yes

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

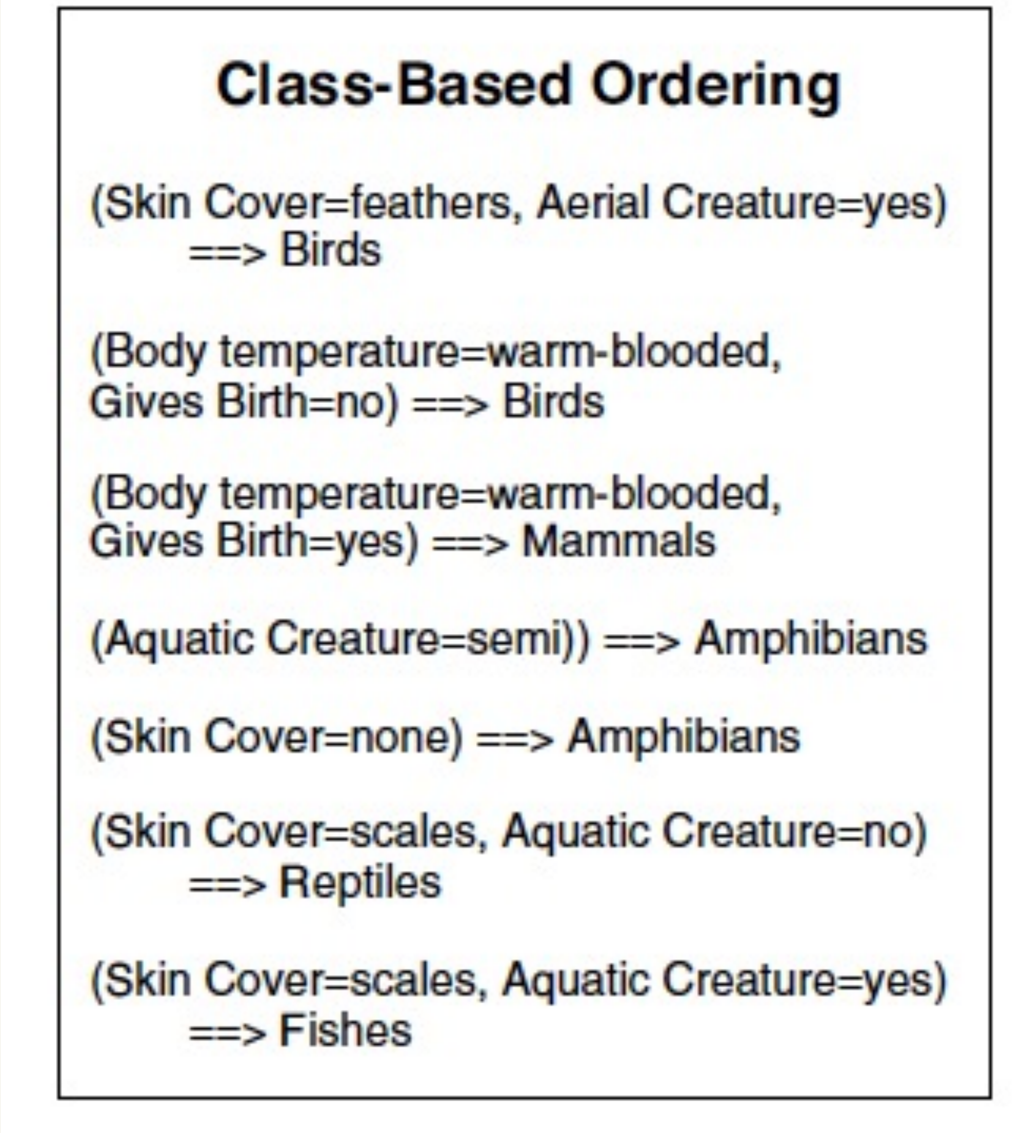
- Instead of ordering the rules, order subsets of rules (class ordering)
  - *Each subset is a collection of rules with the same rule consequent (class)*
  - *Compute description length of each subset*
    - Description length = L(error) + g L(model)
    - g is a parameter that considers the presence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

LECTURE 20

Dr.Vani V

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

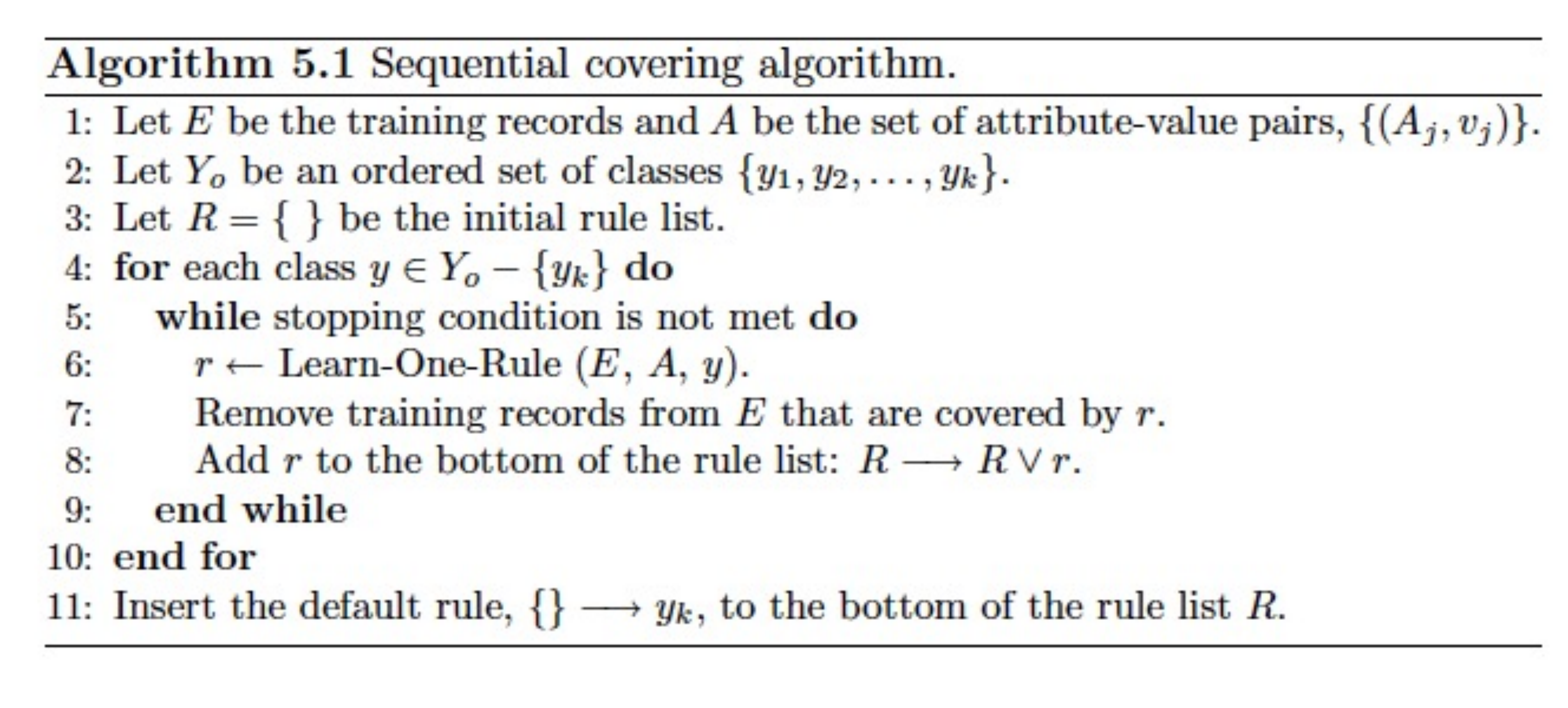
Direct Method: Sequential Covering…

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.



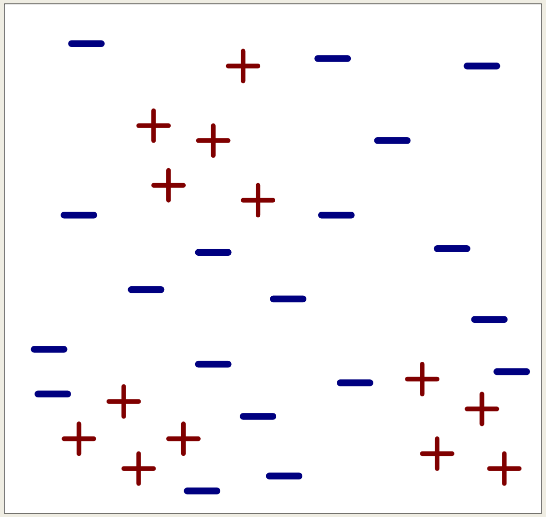
Algorithm 5.1 Sequential covering algorithm.

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

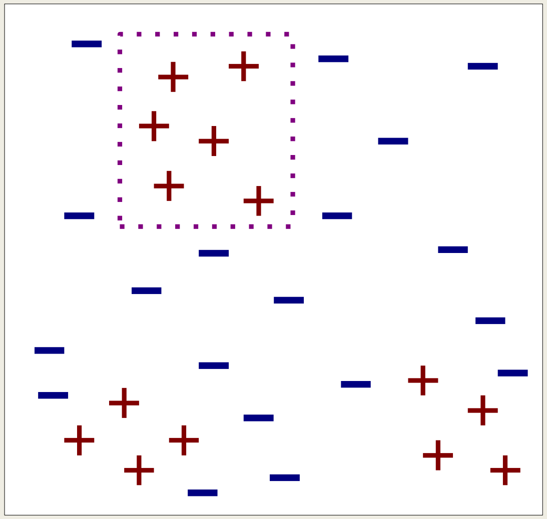
Sequential Covering Algorithm…

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…

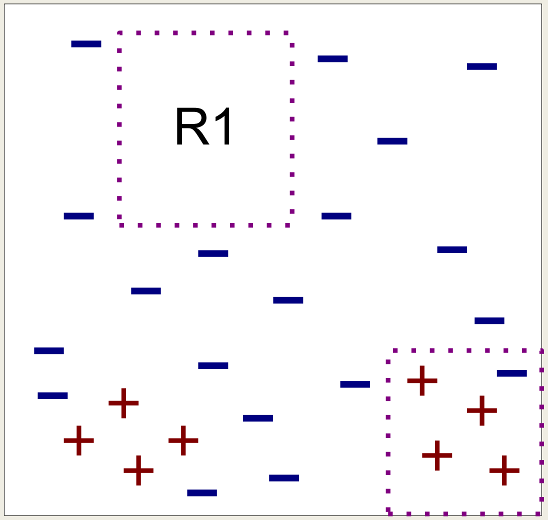


(i) Original Data

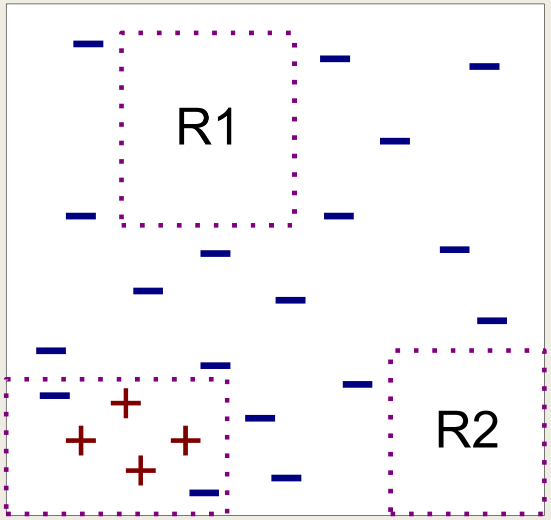


(ii) Step 1

Example of Sequential Covering



(iii) Step 2



(iv) Step 3

Learn-One Rule Function

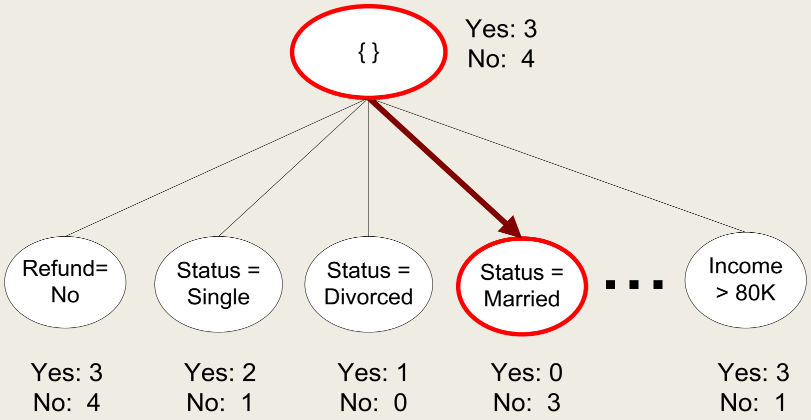
- The **objective of the Learn-One-Rule function is to extract a classification rule that covers many of the positive examples** and none (or very few) of the negative examples in the training set.
- Finding an optimal rule is computationally expensive given the exponential size of the search space.
- 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

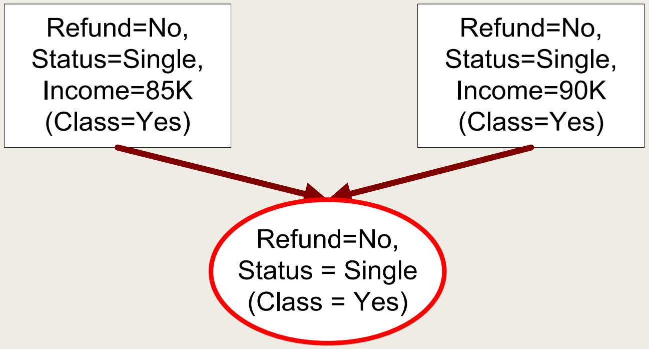
- Rule Growing
- Instance Elimination
- Rule Evaluation
- Stopping Criterion
- Rule Pruning

Rule Growing

- Two common strategies



(a) General-to-specific



(b) Specific-to-general

Rule Evaluation…

- Metrics:
  - *Accuracy* $= \dfrac{n_c}{n}$
  - *Laplace* $= \dfrac{n_c + 1}{n + k}$
  - *M-estimate* $= \dfrac{n_c + kp}{n + k}$

*n : N*umber 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

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

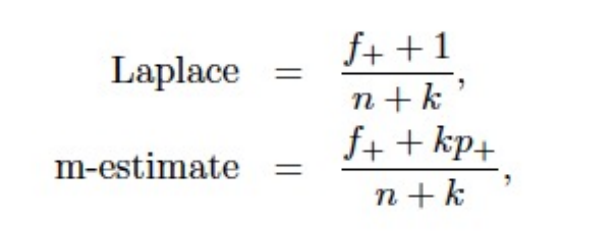
- This statistic therefore suggests that **r1 is a better rule than r2.**

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.**



$$\begin{aligned}\text{Laplace} &= \frac{f_+ + 1}{n + k},\\ \text{m-estimate} &= \frac{f_+ + kp_+}{n + k},\end{aligned}$$

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%

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:

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

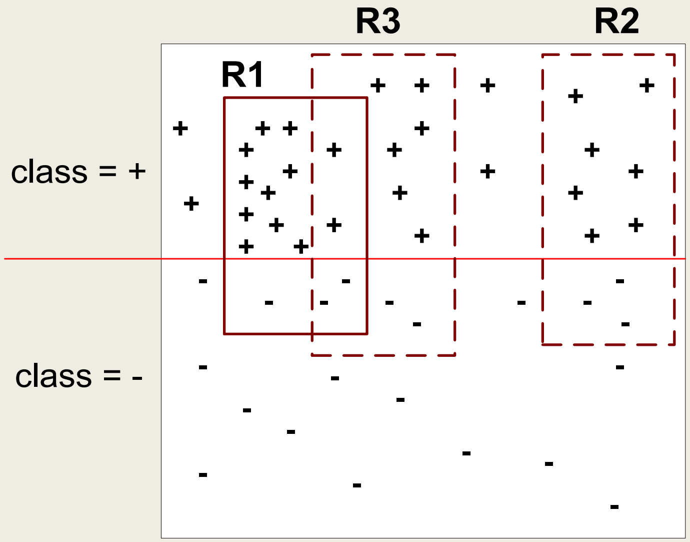
Rule Growing (Examples)

- 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

LECTURE 21

Dr.Vani V

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

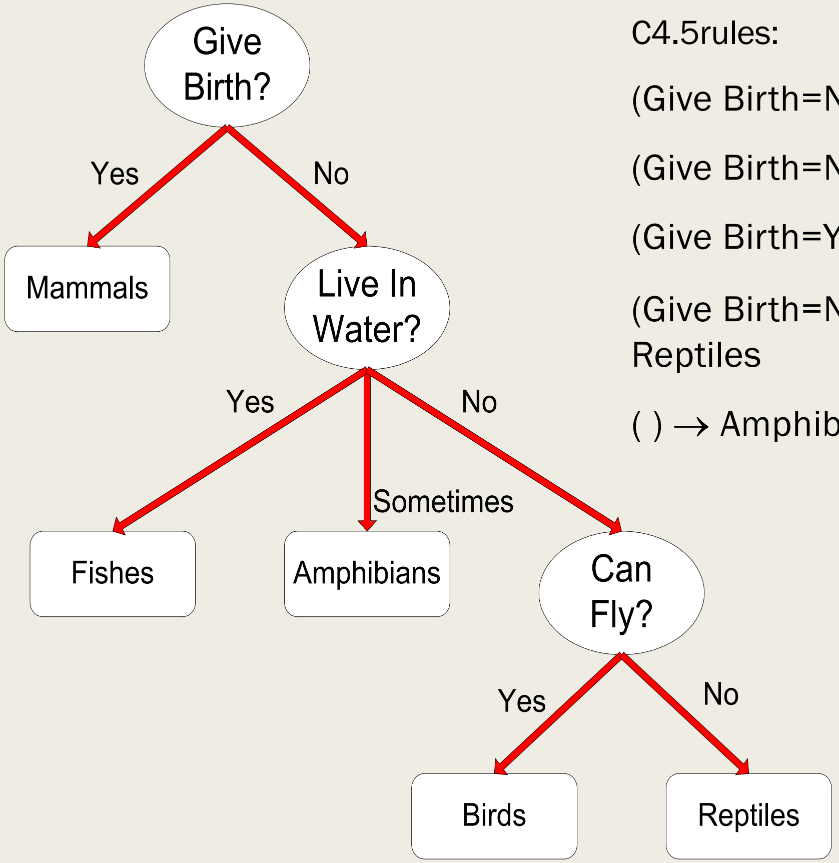
Direct Method: RIPPER

- 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*

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

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

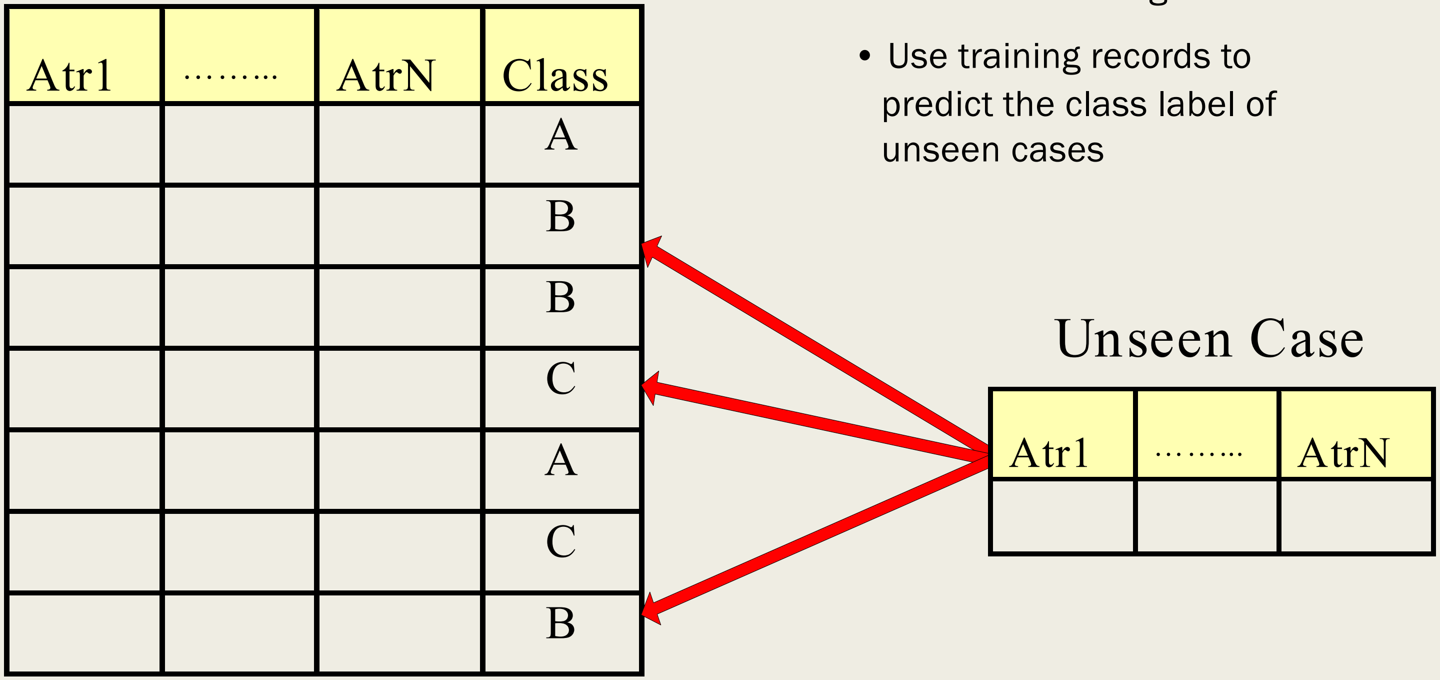
Characteristics of Rule-Based Classifiers

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.**

Instance-Based Classifiers

- Store the training records
- Use training records to predict the class label of unseen cases



Set of Stored Cases

Atr1	………	AtrN	Class
			A
			B
			B
			C
			A
			C
			B

Unseen Case

Atr1	………	AtrN

Instance Based Classifiers

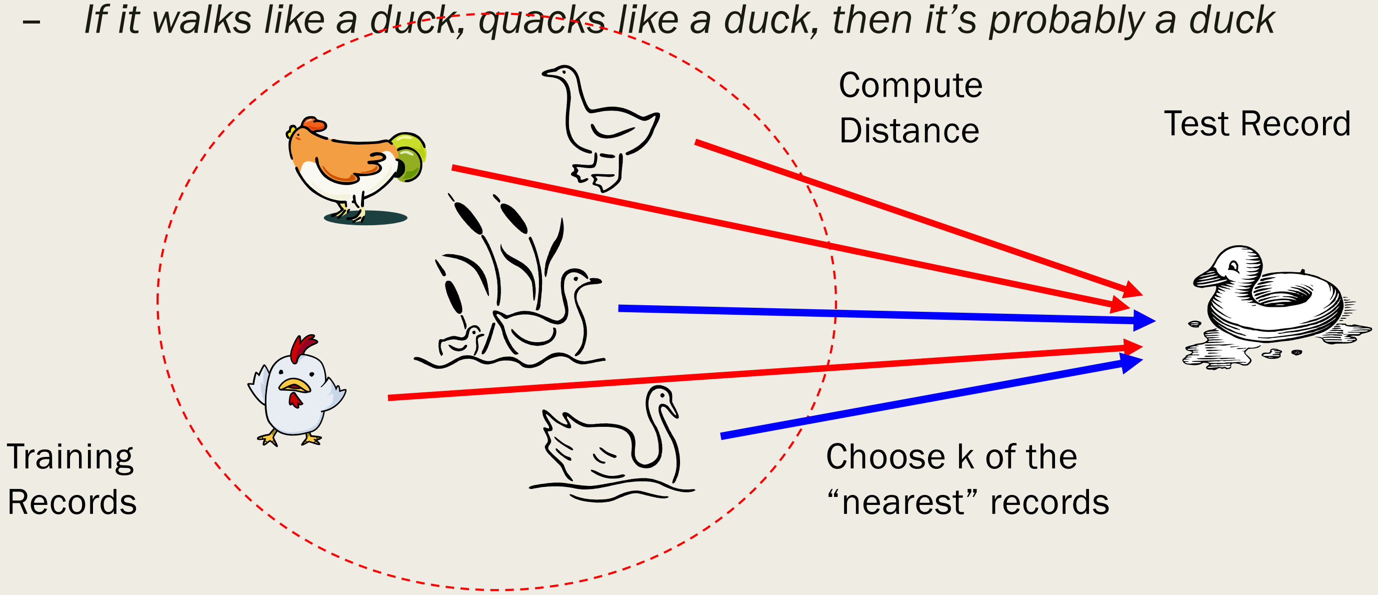
- Examples:
  - *Rote-learner*
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - *Nearest neighbor*
    - Uses k “closest” points (nearest neighbors) for performing classification

K- Nearest Neighbors Classification

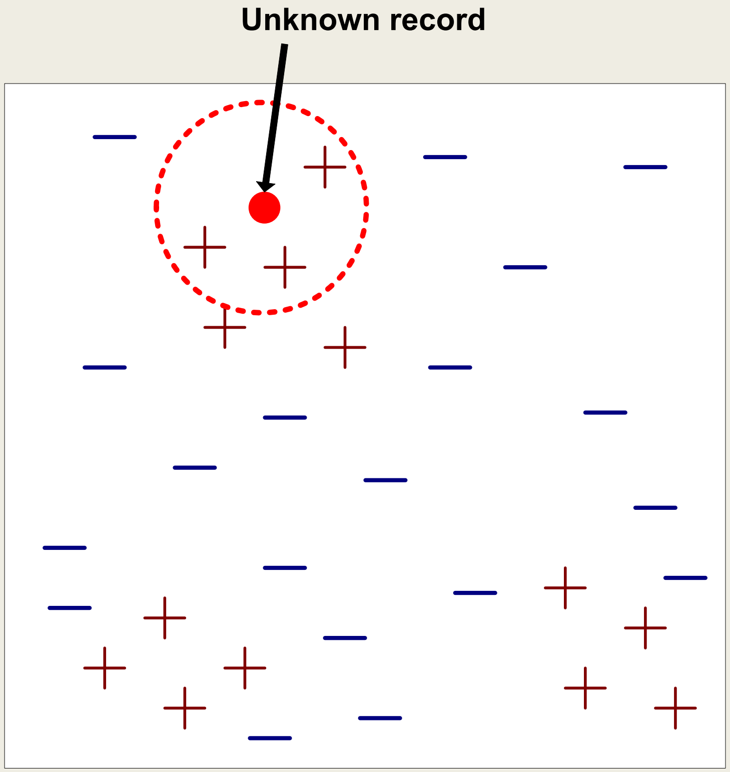
- **K Nearest Neighbors** is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., Distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.
- Also, KNN is classified under **instance based learning** which sometimes referred as **lazy learning .** This is because the process is delayed until a new instance must be classified.

Nearest Neighbor Classifiers

- Basic idea:
  - *If it walks like a duck, quacks like a duck, then it's probably a duck*



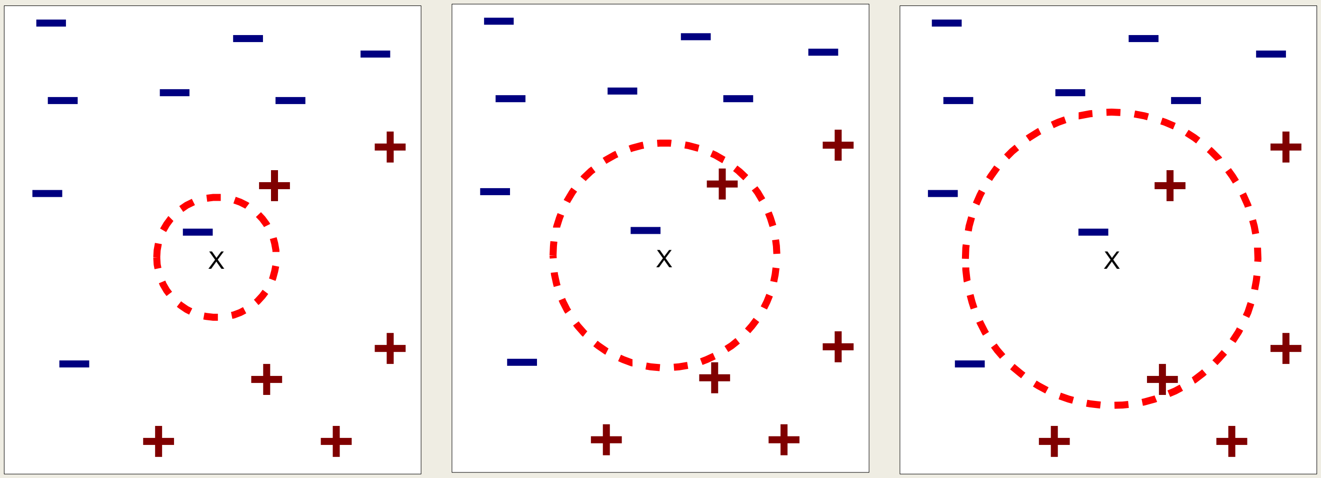
Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor

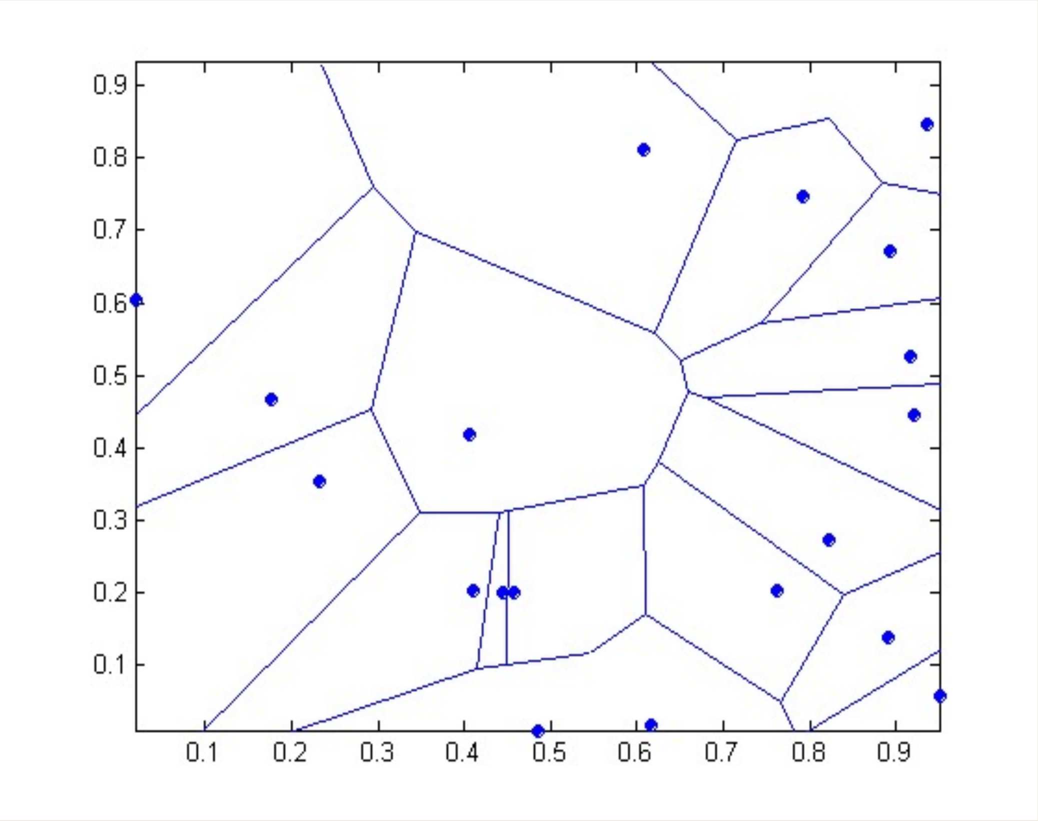


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

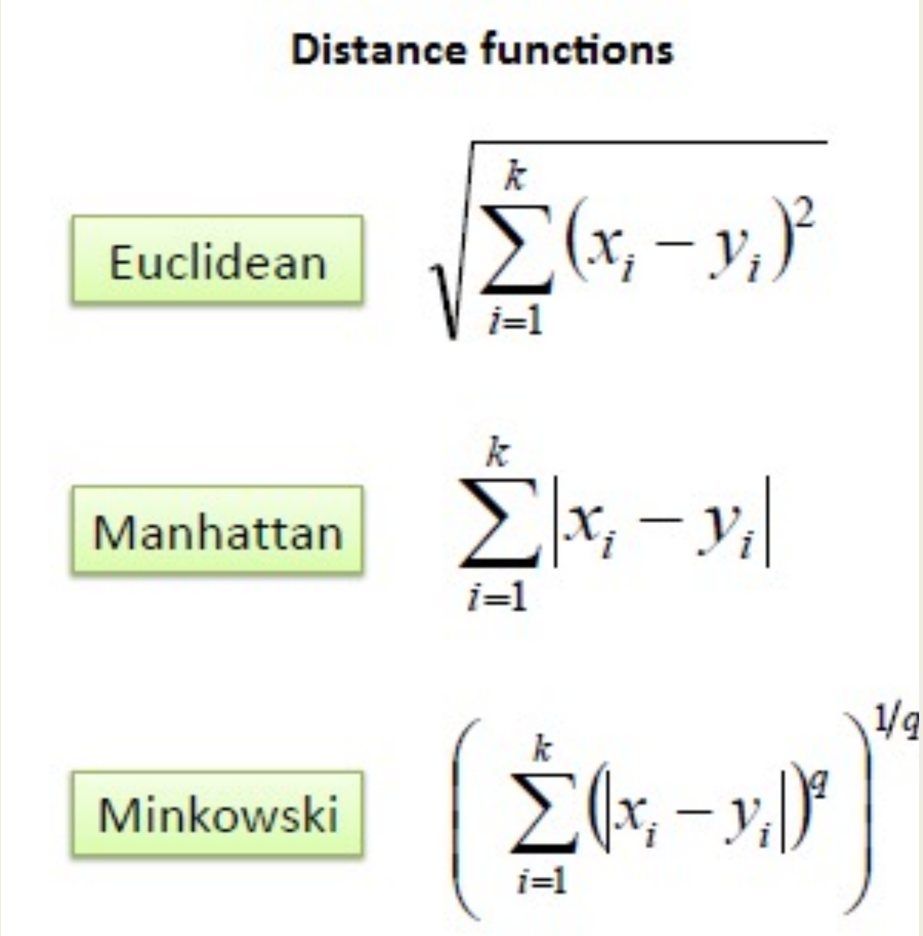
1 nearest-neighbor

Voronoi Diagram



KNN Algorithm

- A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor.



Distance functions

Euclidean

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^{k}\left|x_i - y_i\right|$$

Minkowski

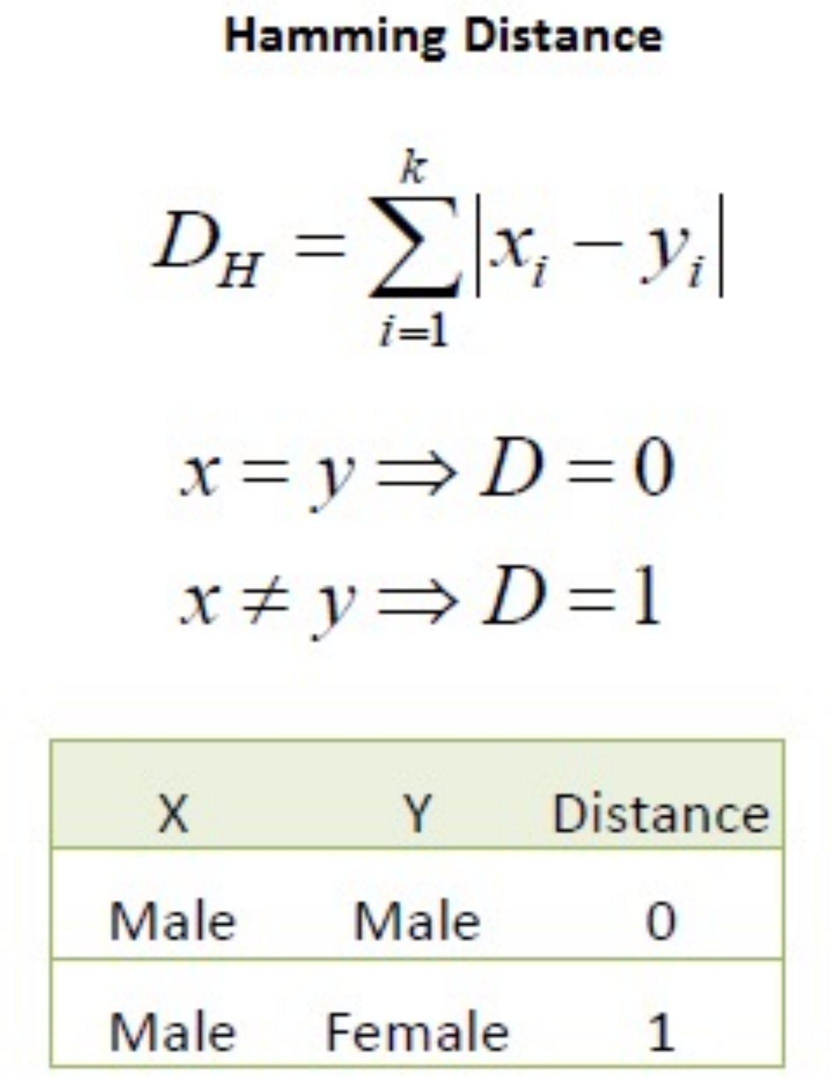
$$\left(\sum_{i=1}^{k}(\left|x_i - y_i\right|)^q\right)^{1/q}$$

KNN Algorithm

- It should also be noted that all three distance measures are only valid for continuous variables.
- In the instance of categorical variables, the hamming distance must be used.
- It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

KNN Algorithm

- Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee. Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.



Hamming Distance

$$D_H = \sum_{i=1}^{k} \left| x_i - y_i \right|$$

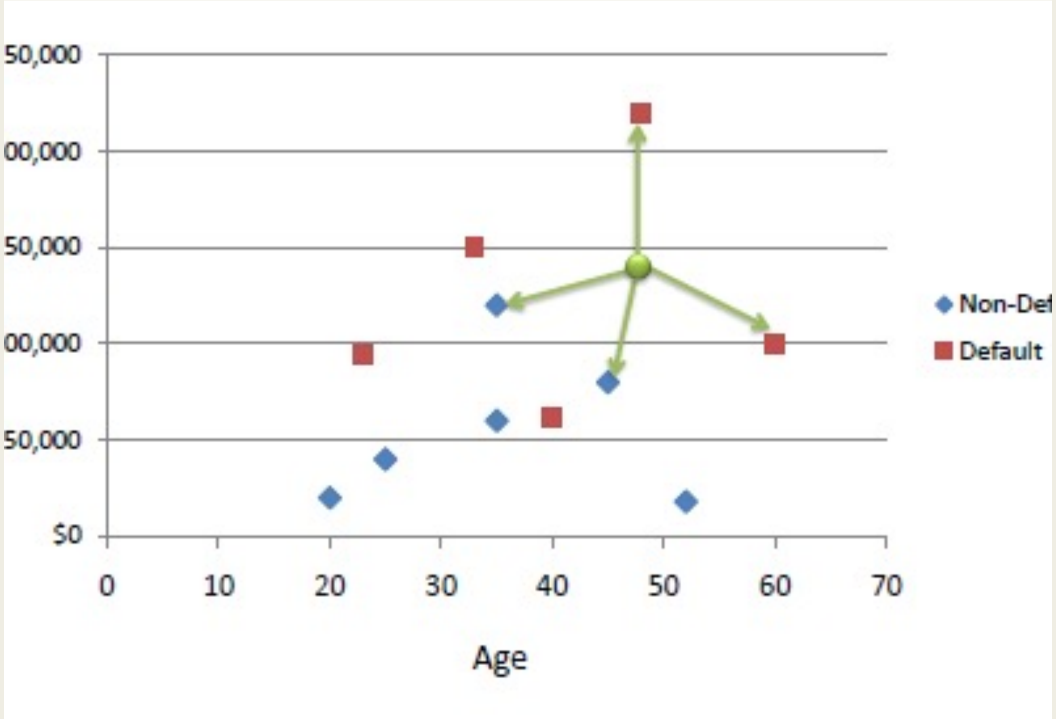
$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

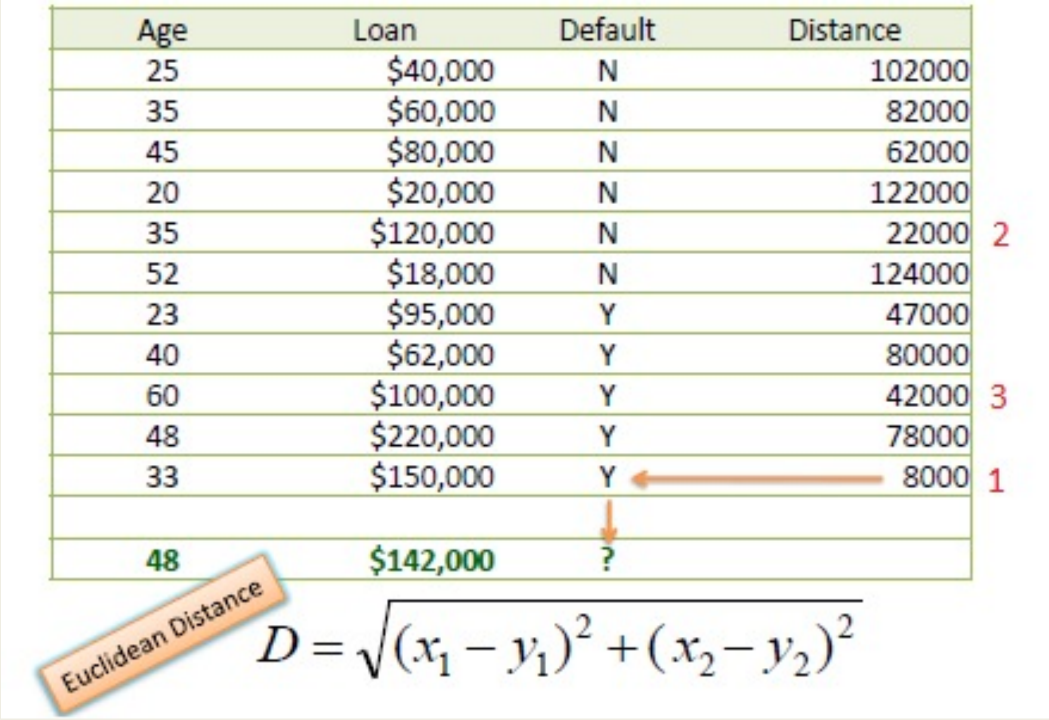
Example

- Consider the following data concerning credit default. Age and Loan are two numerical variables (predictors) and Default is the target.



Example

- We can now use the training set to classify an unknown case (Age=48 and Loan=$142,000) using Euclidean distance. If K=1 then the nearest neighbor is the last case in the training set with Default=Y.
- D = Sqrt[(48-33)^2 + (142000-150000)^2] = 8000.01 >> Default=Y



Age	Loan	Default	Distance	
25	$40,000	N	102000	
35	$60,000	N	82000	
45	$80,000	N	62000	
20	$20,000	N	122000	
35	$120,000	N	22000	2
52	$18,000	N	124000	
23	$95,000	Y	47000	
40	$62,000	Y	80000	
60	$100,000	Y	42000	3
48	$220,000	Y	78000	
33	$150,000	Y	8000	1
48	**$142,000**	**?**		

Euclidean Distance

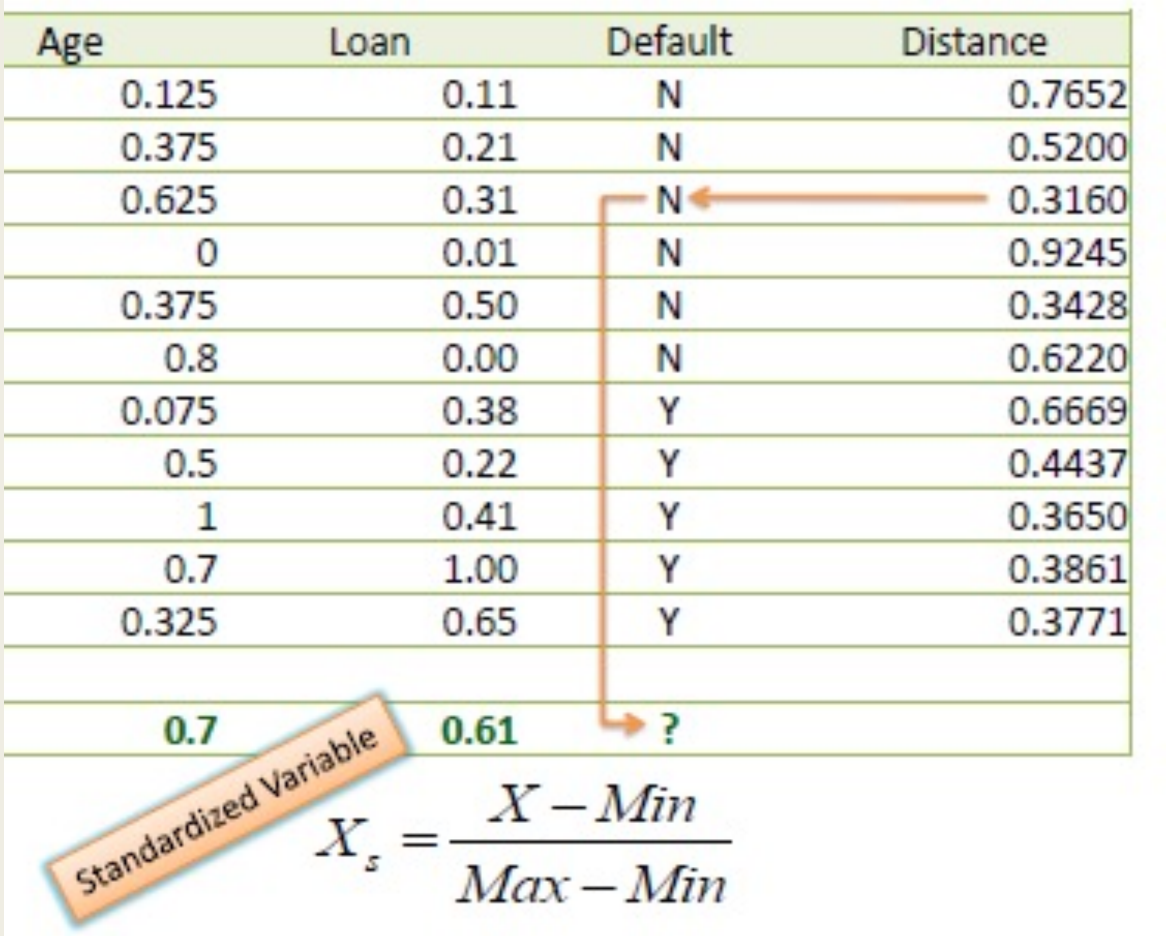
$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

- With K=3, there are two Default=Y and one Default=N out of three closest neighbors. The prediction for the unknown case is again Default=Y.

Example

Standardized distance

- One major drawback in calculating distance measures directly from the training set is in the case where variables have different measurement scales or there is a mixture of numerical and categorical variables.
- For example, if one variable is based on annual income in dollars, and the other is based on age in years then income will have a much higher influence on the distance calculated. One solution is to standardize the training set as shown aside.
- Using the standardized distance on the same training set, the unknown case returned a different neighbor which is not a good sign of robustness.



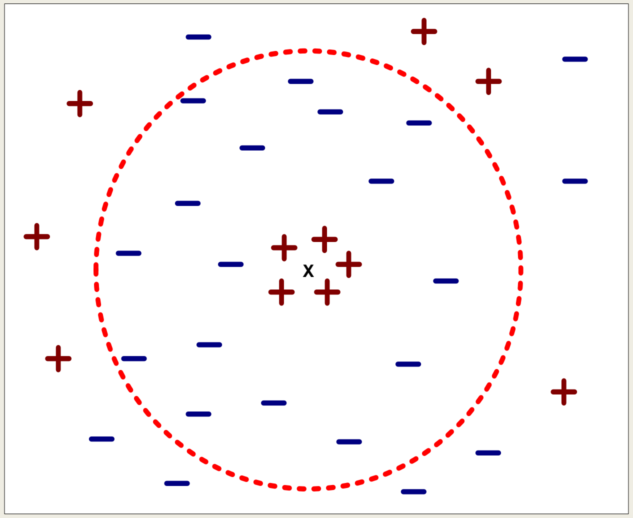
Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	**0.61**	**?**	

Standardized Variable

$$X_s = \frac{X - Min}{Max - Min}$$

Nearest Neighbor Classification…

- Choosing the value of k:
  - *If k is too small, sensitive to noise points*
  - *If k is too large, neighborhood may include points from other classes*



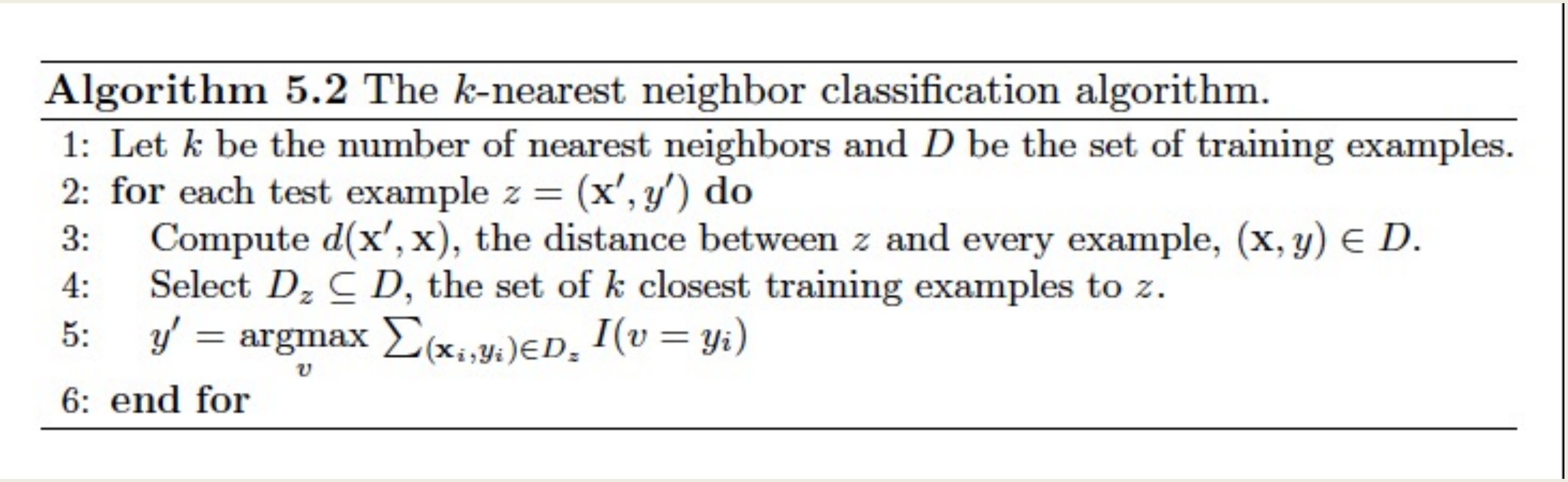
LECTURE 22

Dr.Vani V

KNN Algorithm Summary…

- A high-level summary of the nearest-neighbor classification method is given in next slide
- The algorithm computes the distance (or similarity) between each test example $z = (x' \ , y')$ and all the training examples $(x, y) \in D$ to determine its nearest-neighbor list, Dz.
- This computation can be costly if the number of training examples is large. However, efficient indexing techniques are available to reduce the amount of computations needed to find the nearest neighbors of a test example.

KNN Algorithm Summary…

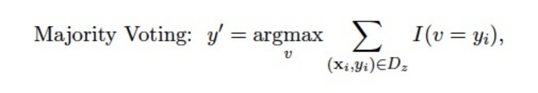


Algorithm 5.2 The k-nearest neighbor classification algorithm.

```
Let $k$ be the number of nearest neighbors and $D$ be the set of training examples.
for each test example $z = (\mathbf{x}', y')$ do
    Compute $d(\mathbf{x}', \mathbf{x})$, the distance between $z$ and every example, $(\mathbf{x}, y) \in D$.
    Select $D_z \subseteq D$, the set of $k$ closest training examples to $z$.
    $y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$
end for
```

KNN Algorithm Summary…

- Once the nearest-neighbor list is obtained, the test example is classified based on the majority class of its nearest neighbors:



$$\text{Majority Voting:}\quad y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i),$$

where v is a class label, y_i is the class label for one of the nearest neighbors, and $I(.)$ is an indicator function that returns the value 1 if its argument is true and 0 otherwise.

KNN Algorithm Summary…

- In the majority voting approach, every neighbor has the same impact on the classification.
- One way to reduce the impact of k is to weight the influence of each nearest neighbor x_i according to its distance: $w_i = 1/d(x^{'}, x_i)^2$.
- As a result, training examples that are located far away from z have a weaker impact on the classification compared to those that are located close to z.
- Using the distance-weighted voting scheme, the class label can be determined as follows:

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

Characteristics of Nearest-Neighbor Classifiers

- The characteristics of the nearest-neighbor classifier are summarized below:

1. Nearest-neighbor classification is part of a more general technique **known as instance-based learning**, which **uses specific training instances** to make predictions without having to maintain an abstraction (or model) derived from data.

Instance-based learning algorithms **require a proximity measure** to determine the similarity or distance between instances and a classification function that returns the predicted class of a test instance based on its proximity to other instances.

Characteristics of Nearest-Neighbor Classifiers

- The characteristics of the nearest-neighbor classifier are summarized below:

2. Lazy learners such as nearest-neighbor classifiers do not require model building.

Classifying a test example can be quite expensive because the proximity values need to be computed individually between the test and training examples. In contrast, eager learners such as decision tree induction and rule-based systems often spend the bulk of their computing resources for model building. Once a model has been built, classifying a test example is extremely fast.

Characteristics of Nearest-Neighbor Classifiers

- The characteristics of the nearest-neighbor classifier are summarized below:

3. Nearest-neighbor classifiers make their predictions based on local information, whereas decision tree and rule-based classifiers attempt to find a global model that fits the entire input space**. Because the classification decisions are made locally, nearest-neighbor classifiers (with small values of k) are quite susceptible to noise.**

Characteristics of Nearest-Neighbor Classifiers

- The characteristics of the nearest-neighbor classifier are summarized below:

4. Nearest-neighbor classifiers can produce arbitrarily shaped decision boundaries. Such boundaries provide a more flexible model representation compared to decision tree and rule-based classifiers that are often constrained to rectilinear decision boundaries.

The decision boundaries of nearest-neighbor classifiers also have high variability because they depend on the composition of training examples. **Increasing the number of nearest neighbors may reduce such variability.**

Characteristics of Nearest-Neighbor Classifiers

- The characteristics of the nearest-neighbor classifier are summarized below:

5 Nearest-neighbor classifiers can produce wrong predictions unless the appropriate proximity measure and data preprocessing steps are taken.

For example, to classify a group of people based on attributes such as height (measured in meters) and weight (measured in pounds). The height attribute has a low variability, ranging from 1.5 m to 1.85 m, whereas the weight attribute may vary from 90 lb. to 250 lb. If the scale of the attributes are not taken into consideration, the proximity measure may be dominated by differences in the weights of a person.

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*

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

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

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

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

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

Selected Exercises: 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 −⟶ Value = Low

r2: Mileage = Low −⟶ Value = High

r3: Air Conditioner = Working, Engine = Good −⟶ Value = High

r4: Air Conditioner = Working, Engine = Bad −⟶ Value = Low

r5: Air Conditioner = Broken −⟶ 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?

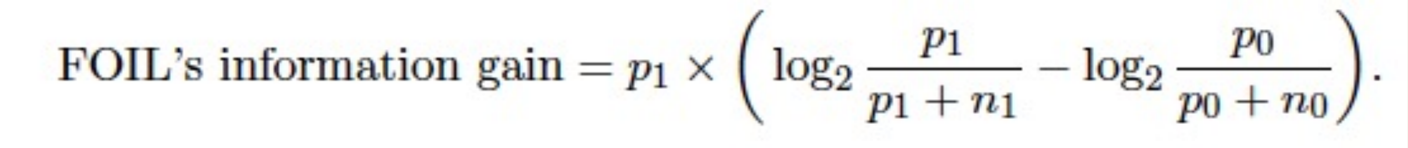
No, because every instance is guaranteed to trigger at least one rule.

Selected Exercises:2…

<u>Given:</u>

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.

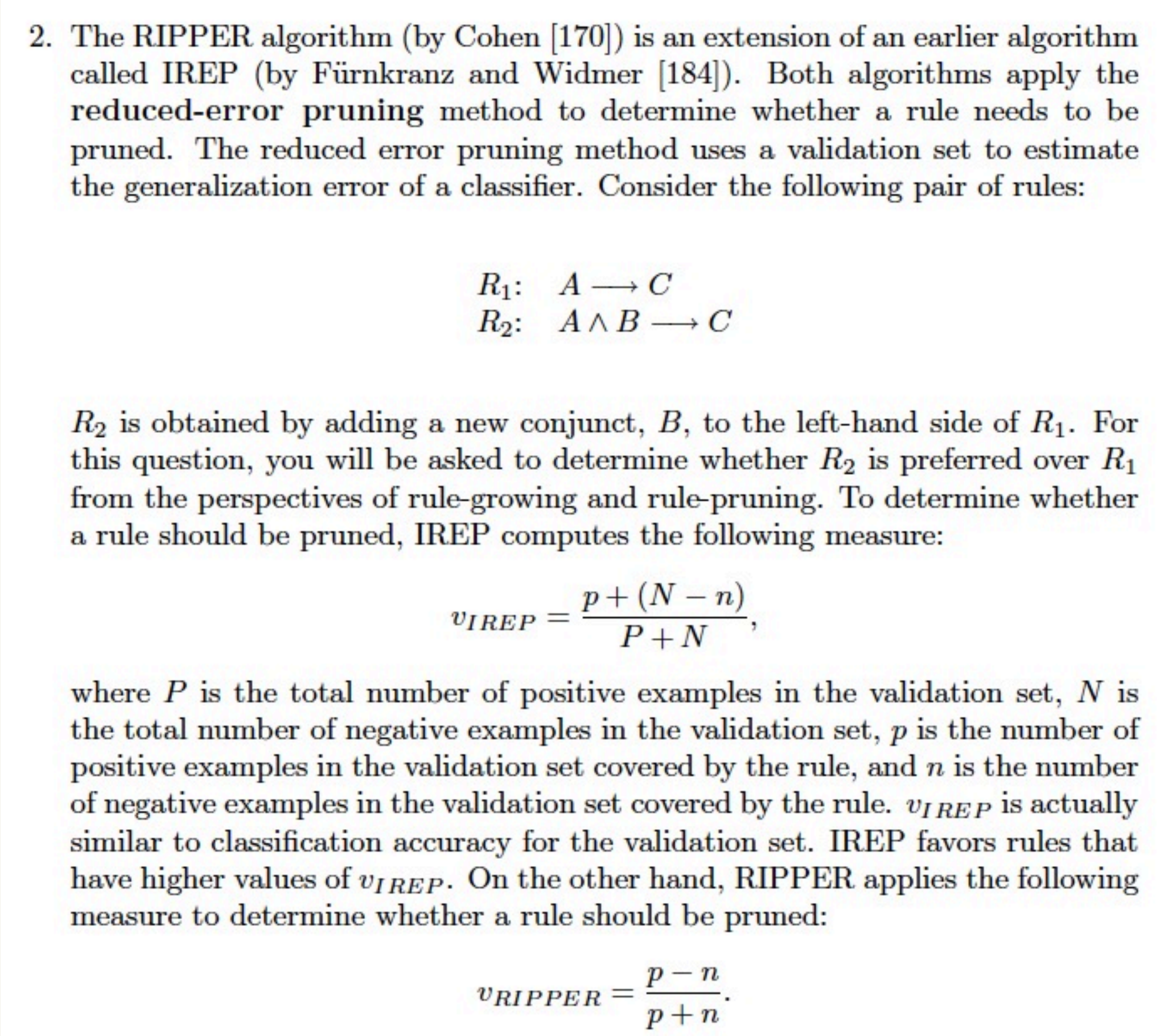
<u>Solution:</u>



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

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:

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



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:

$$\begin{aligned} R_1&: \quad A \longrightarrow C \\ R_2&: \quad A \wedge B \longrightarrow C \end{aligned}$$

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,

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

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

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

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

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

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

C4.5rules 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.

(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 Ø →+. 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

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

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

$$Gain(R2) = 57.2$$

$$Gain(R3) = 139.6$$

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

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) = $2\ x\ [4x\ log_2\frac{4}{1} + 1\text{x}\ log_2\frac{1}{4}]$ = 12

Selected Exercises:4…

(c)The likelihood ratio statistic:

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

<u>Solution:</u>

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) = $2\ x\ [30x\ log_2 \frac{30}{8} + 10\text{x}\ log_2 \frac{10}{32}]$ = 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) = $2\ x\ [100\ x\ log_2 \frac{100}{38} + 90\text{x}\ log_2 \frac{90}{152}]$ = 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.**

Selected Exercises:4

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

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

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),

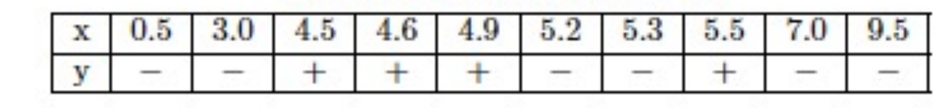
R3: C ⟶ + (covers 100 positive and 90 negative examples),

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.

Selected Exercises:5

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

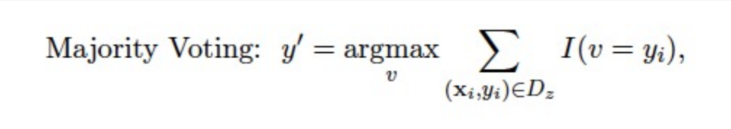


x	0.5	3.0	4.5	4.6	4.9	5.2	5.3	5.5	7.0	9.5
y	−	−	+	+	+	−	−	+	−	−

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

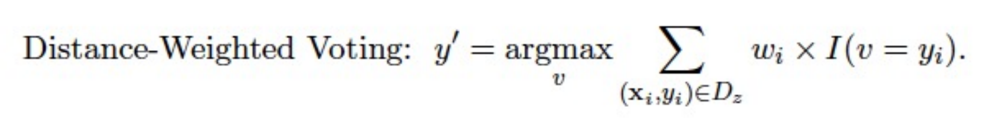
(b) Repeat the previous analysis using the distance-weighted voting approach

Solution:



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

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



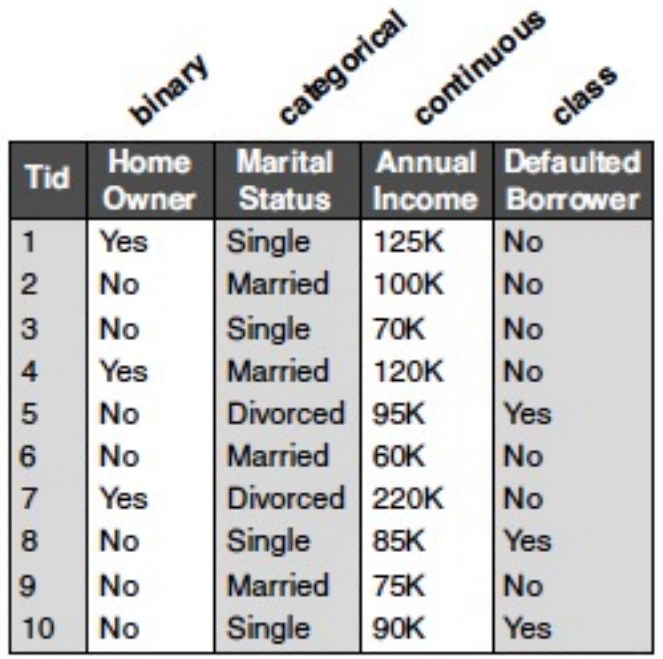
Distance-Weighted Voting: $y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} w_i \times I(v = y_i).$

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

Selected Exercises:6...

Consider the training set for the loan classification problem shown in Figure 5.9.

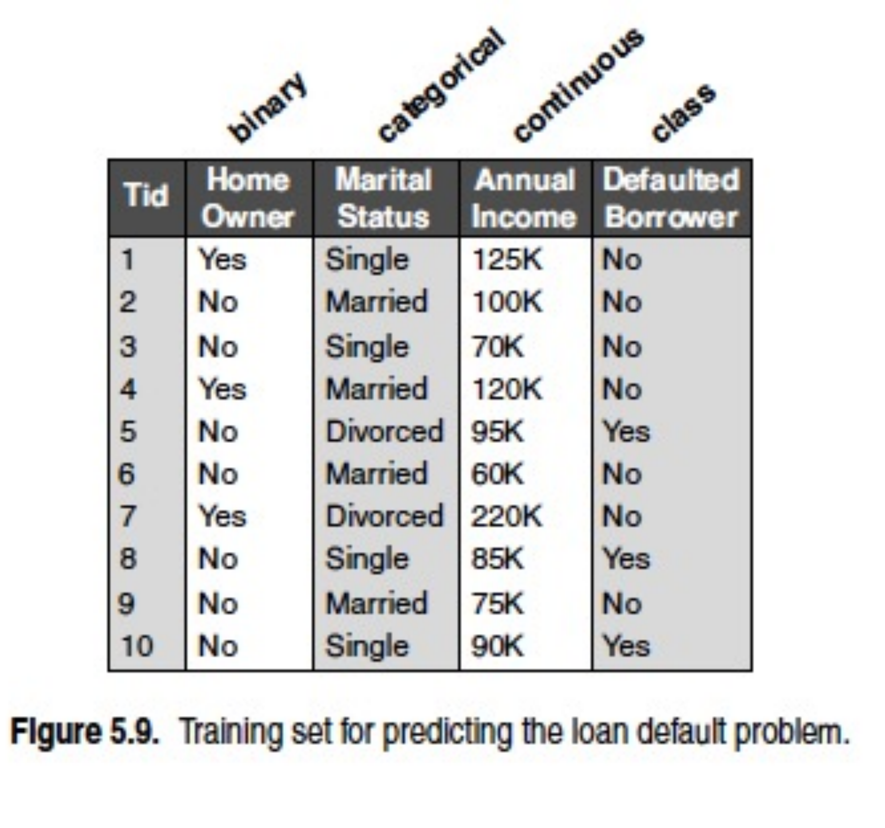
Use the MVDM measure to compute the distance between every pair of attribute values for the **Home Owner and Marital Status attributes.**


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

Figure 5.9. Training set for predicting the loan default problem.

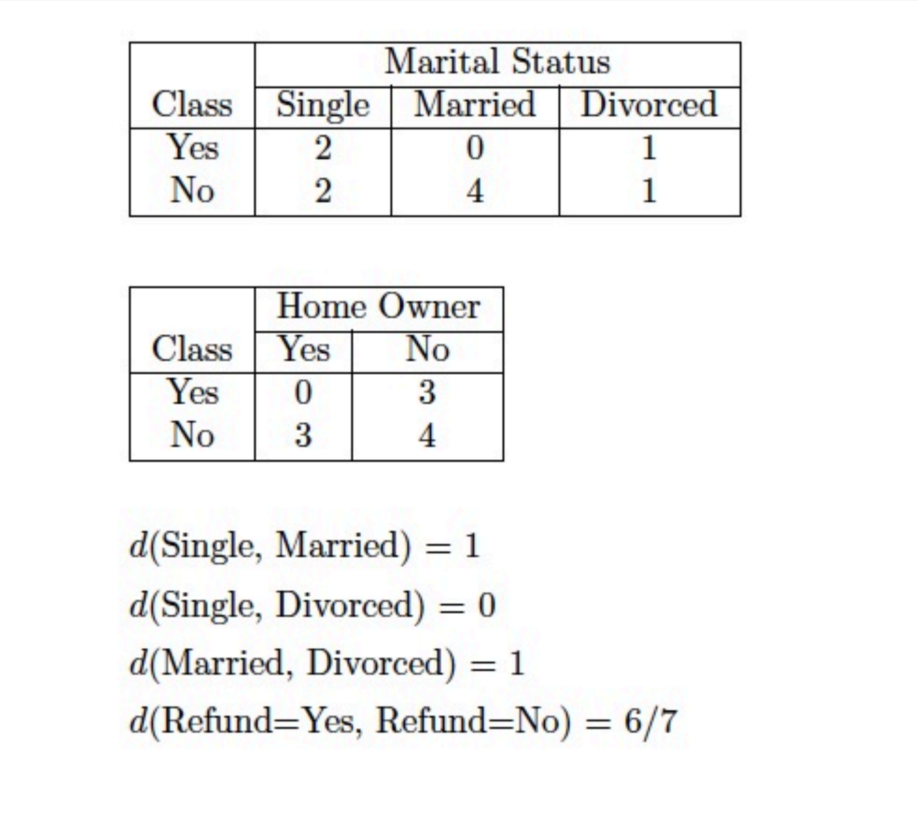
Selected Exercises:6



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

Figure 5.9. Training set for predicting the loan default problem.



	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

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

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

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

$d(\text{Refund=Yes, Refund=No}) = 6/7$

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