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	
		Class=Yes	Class=No
ACTUAL CLASS	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	
		Class=Yes	Class=No
ACTUAL CLASS	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 :

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

- 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

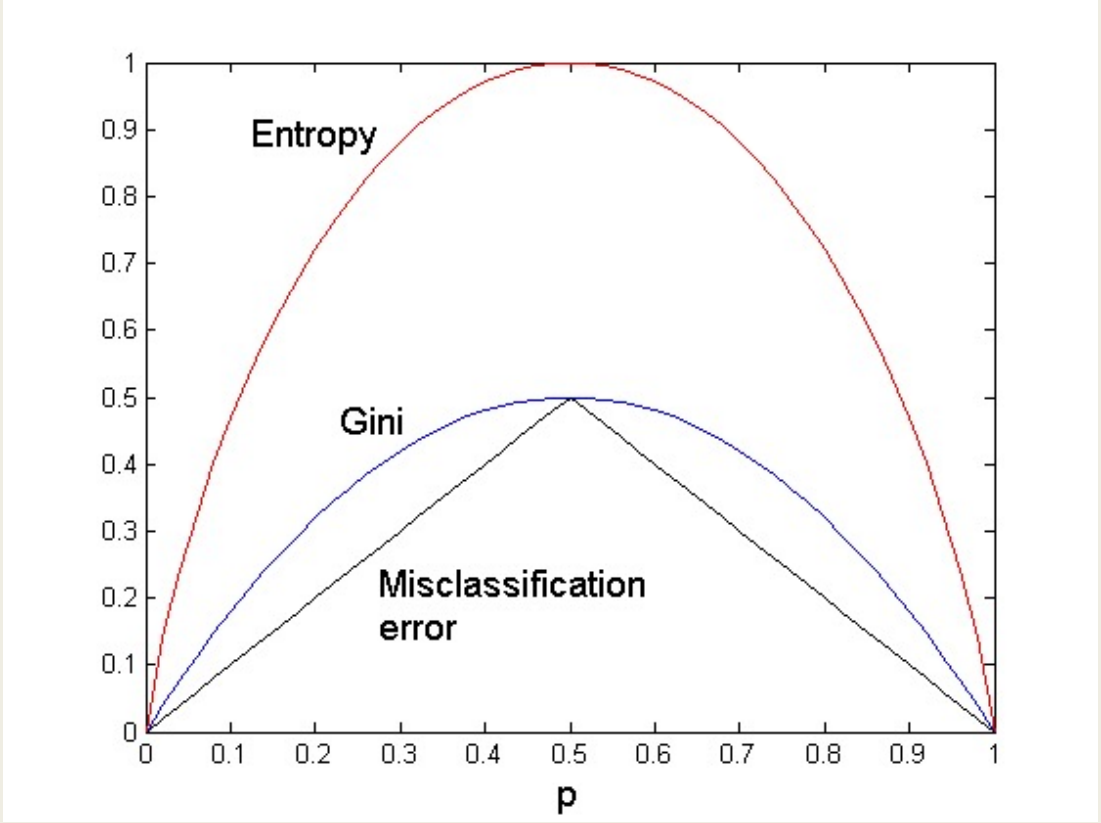
Comparison among Splitting Criteria

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)
 1: if stopping_cond(E,F) = true then
 2:    leaf = createNode().
 3:    leaf.label = Classify(E).
 4:    return leaf.
 5: else
 6:    root = createNode().
 7:    root.test_cond = find_best_split(E, F).
 8:    let V = {v|v is a possible outcome of root.test_cond }.
 9:    for each v ∈ V do
10:       E_v = {e | root.test_cond(e) = v and e ∈ E}.
11:       child = TreeGrowth(E_v, F).
12:       add child as descendent of root and label the edge (root → child) as v.
13:    end for
14: end if
15: 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

$$leaf.label = \underset{i}{\text{argmax}}\ p(i|t),$$

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.

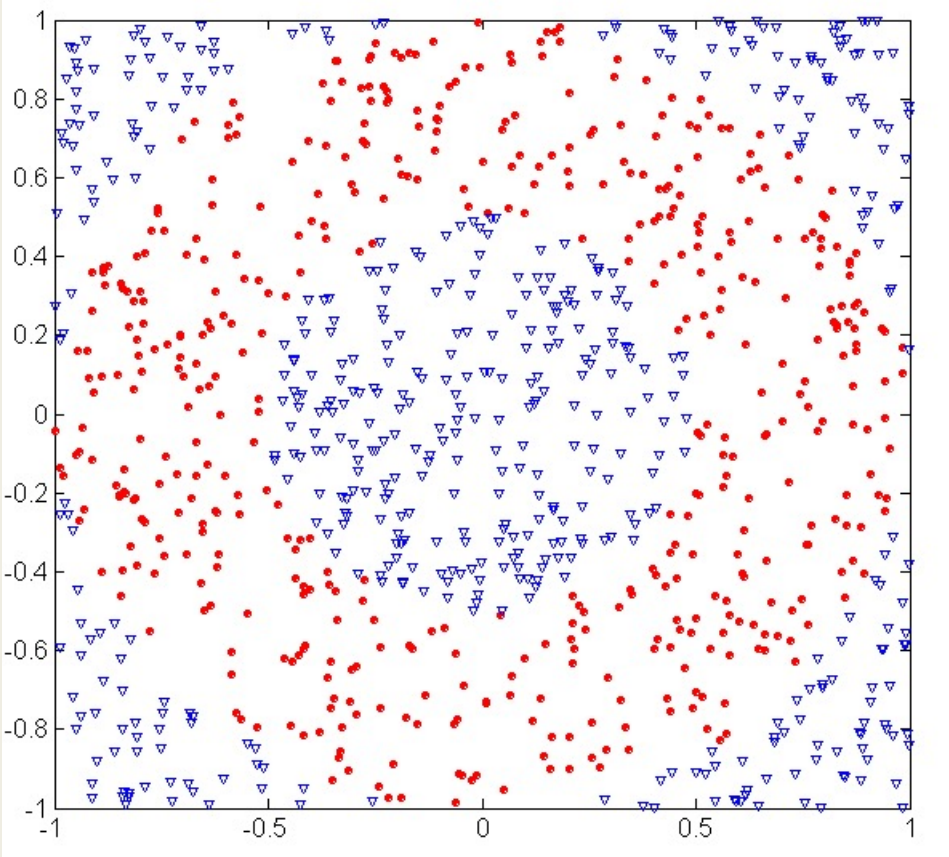
Algorithm for Decision Tree Induction

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:

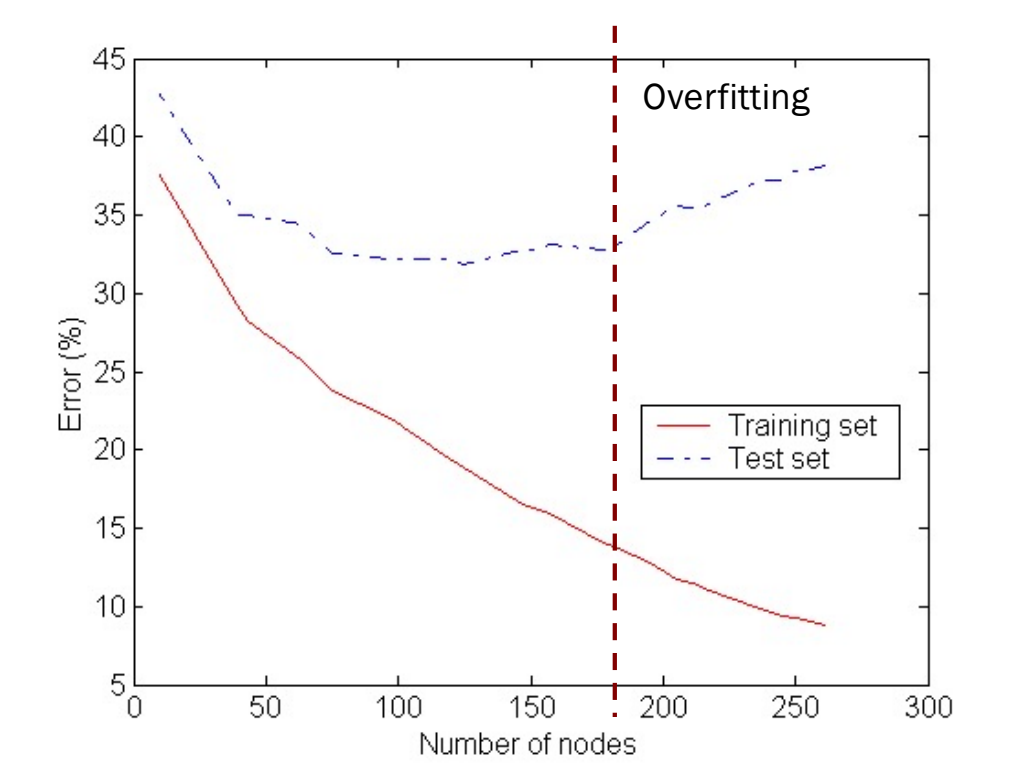
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

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

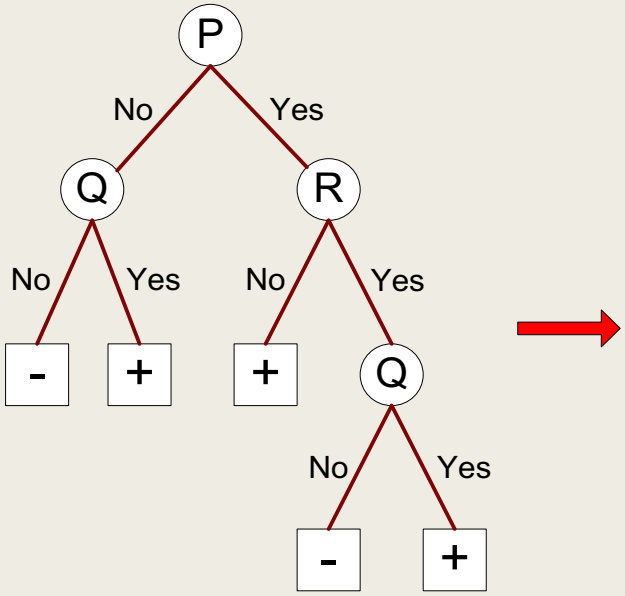
(Refund=No, Marital Status={Married}) ==> No

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

Indirect Method: C4.5rules

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

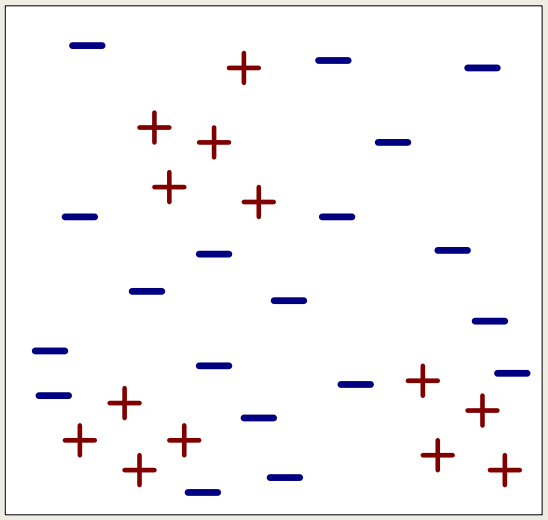
LECTURE 20

Dr.Vani V

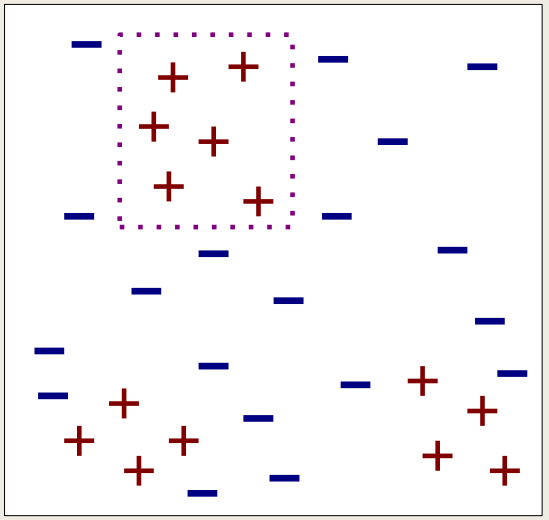
Direct Method: Sequential Covering

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

Example of Sequential Covering

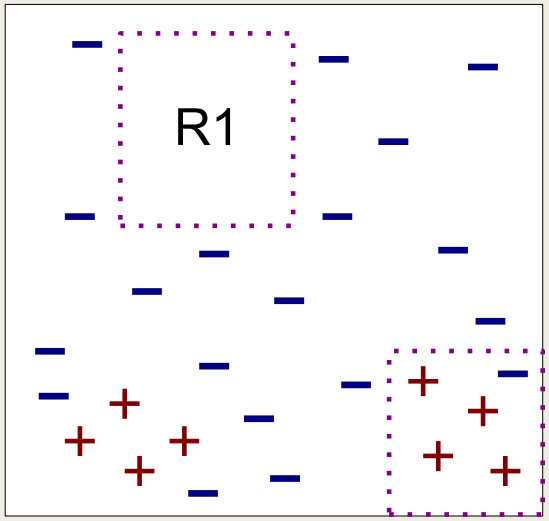


(i) Original Data

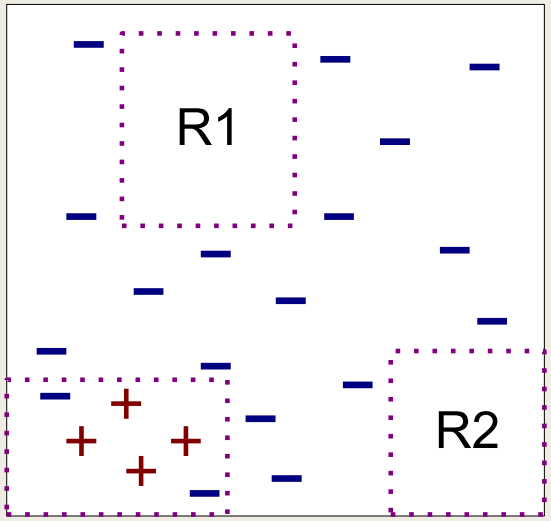


(ii) Step 1

Example of Sequential Covering…



(iii) Step 2



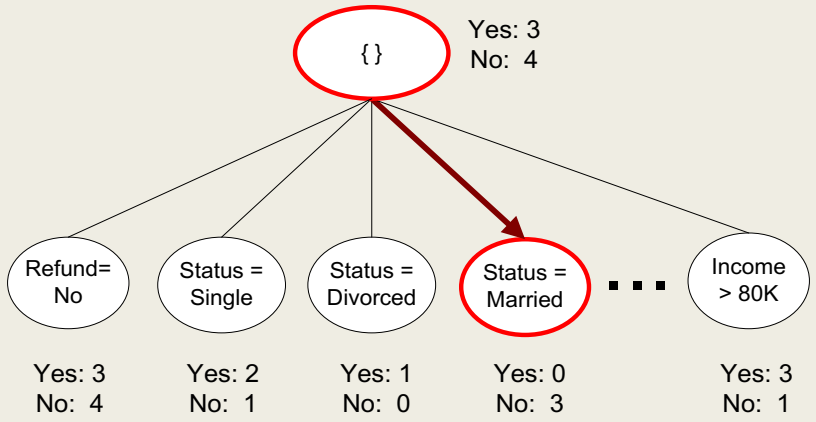
(iv) Step 3

Aspects of Sequential Covering

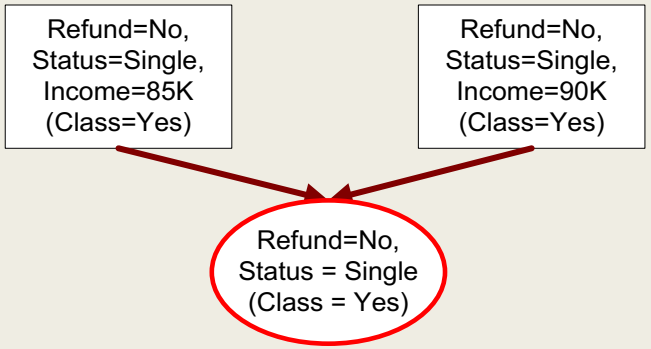
- Rule Growing
- Instance Elimination
- Rule Evaluation
- Stopping Criterion
- Rule Pruning

Rule Growing

- Two common strategies



(a) General-to-specific



(b) Specific-to-general

Rule 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

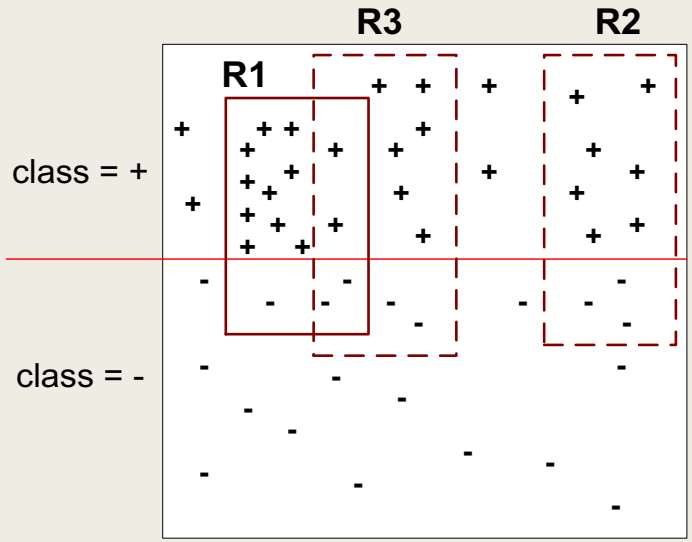
      n0: number of negative instances covered by R0

      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*



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 instances covered by rule

k : Number of classes

p : Prior probability

Stopping Criterion and Rule Pruning

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

- Rule Pruning
  - *Similar to post-pruning of decision trees*
  - *Reduced Error Pruning:*
    - Remove one of the conjuncts in the rule
    - Compare error rate on validation set before and after pruning
    - If error improves, prune the conjunct

Summary of Direct Method

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

Direct Method: RIPPER

- 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

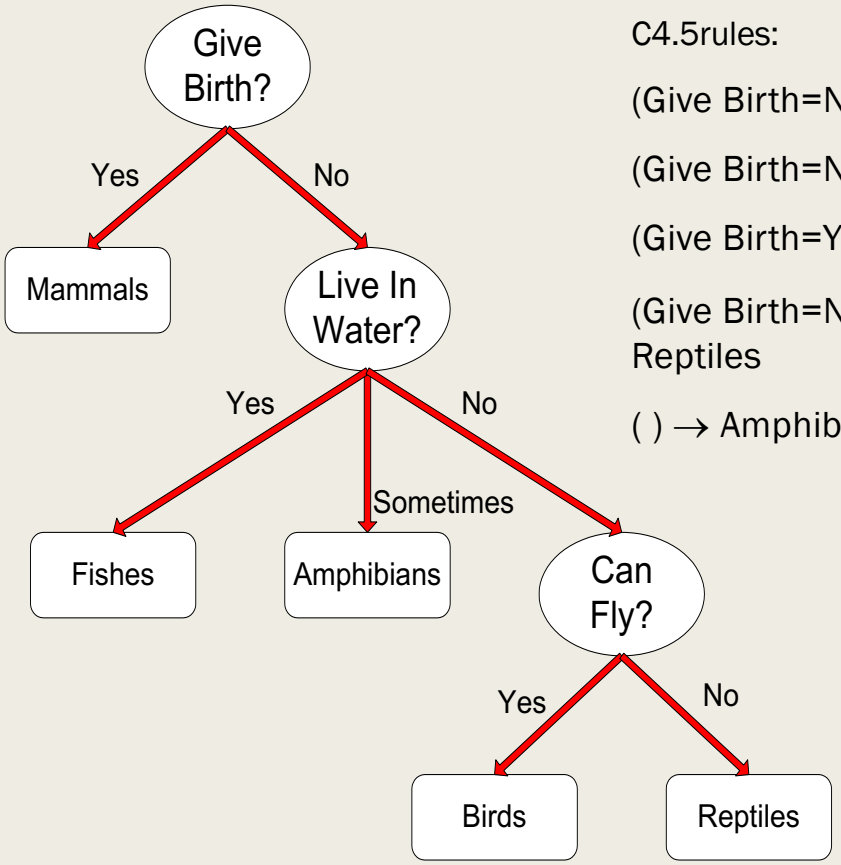
Direct Method: RIPPER

- 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

C4.5 versus C4.5rules versus RIPPER

C4.5 and C4.5rules:

			PREDICTED CLASS			
		Amphibians	Fishes	Reptiles	Birds	Mammals
ACTUAL CLASS	Amphibians	2	0	0	0	0
	Fishes	0	2	0	0	1
	Reptiles	1	0	3	0	0
	Birds	1	0	0	3	0
	Mammals	0	0	1	0	6

RIPPER:

			PREDICTED CLASS			
		Amphibians	Fishes	Reptiles	Birds	Mammals
ACTUAL CLASS	Amphibians	0	0	0	0	2
	Fishes	0	3	0	0	0
	Reptiles	0	0	3	0	1
	Birds	0	0	1	2	1
	Mammals	0	2	1	0	4

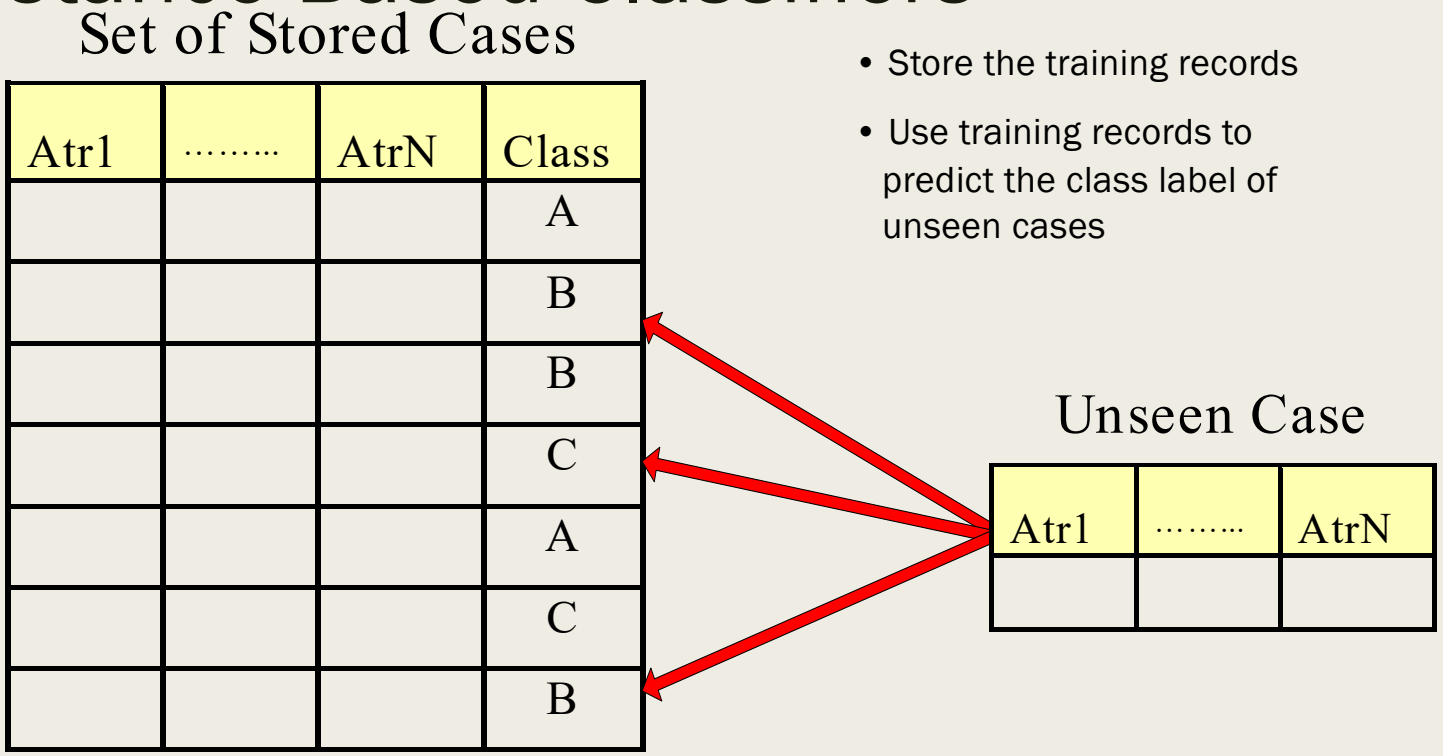
Advantages of Rule-Based Classifiers

- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees

LECTURE 21

Dr.Vani V

Instance-Based Classifiers



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

Instance Based Classifiers

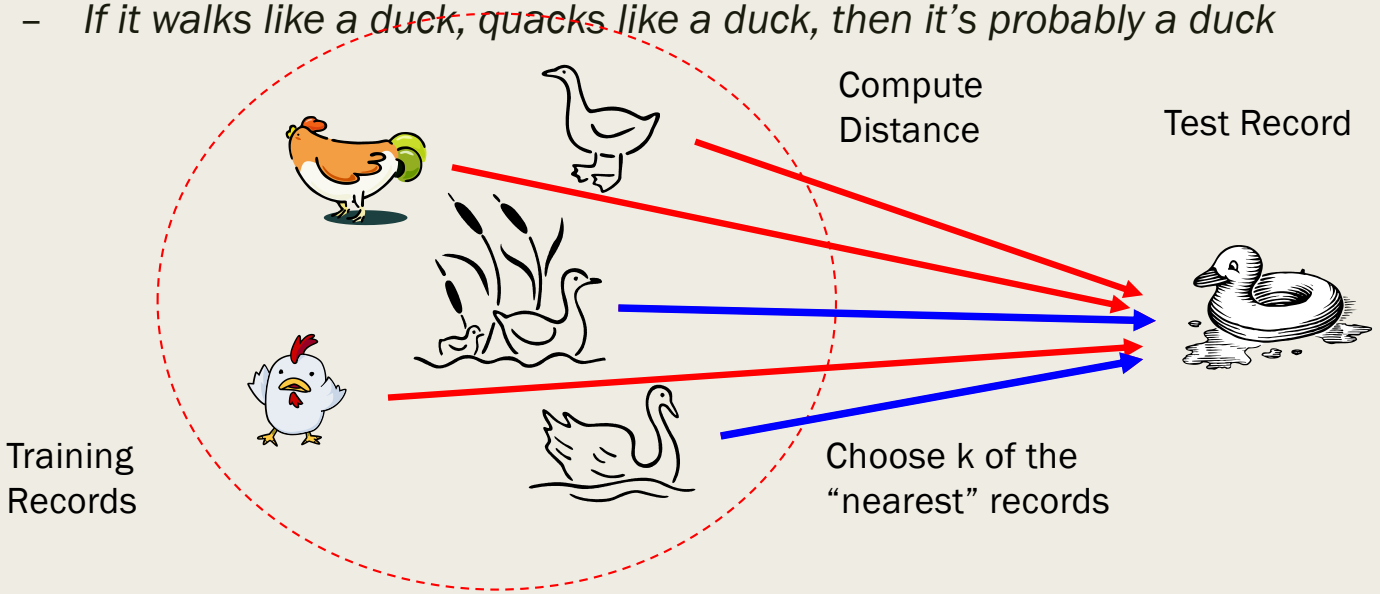
- Examples:
  - *Rote-learner*
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - *Nearest neighbor*
    - Uses k “closest” points (nearest neighbors) for performing classification

K- Nearest Neighbors Classification

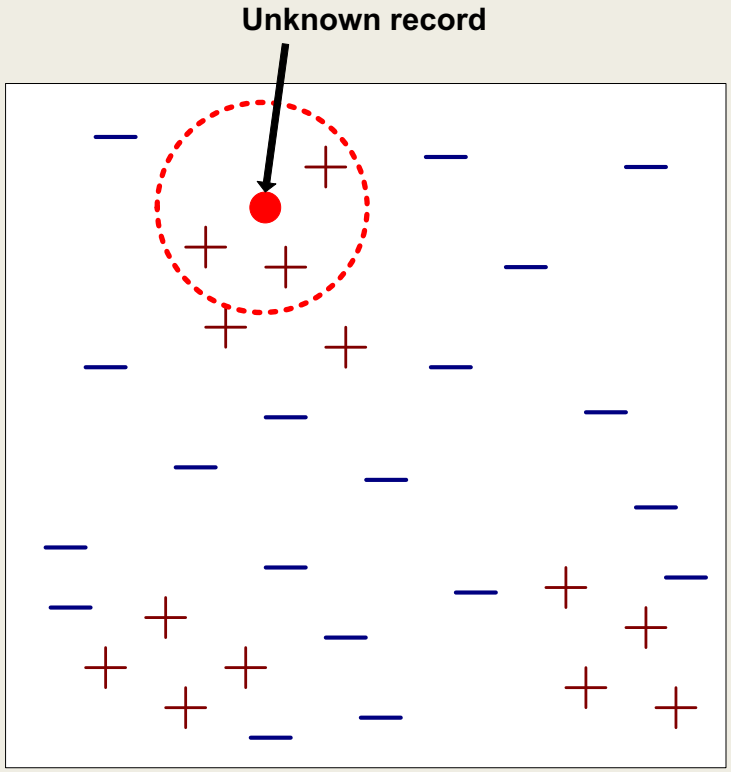
- **K Nearest Neighbors** is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., Distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.
- Also, 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*



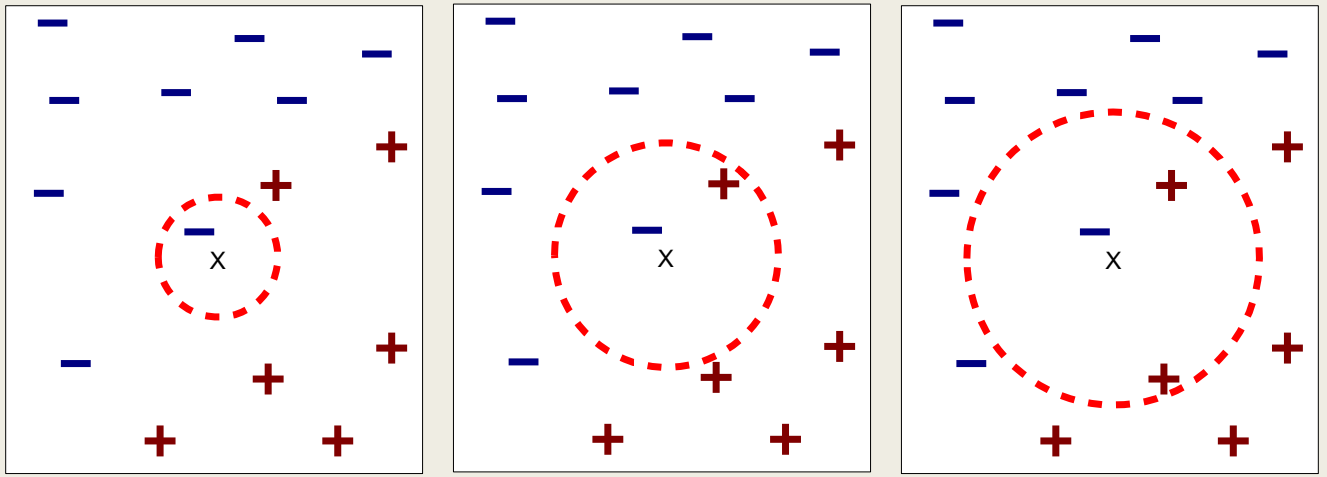
Nearest-Neighbor Classifiers



- 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

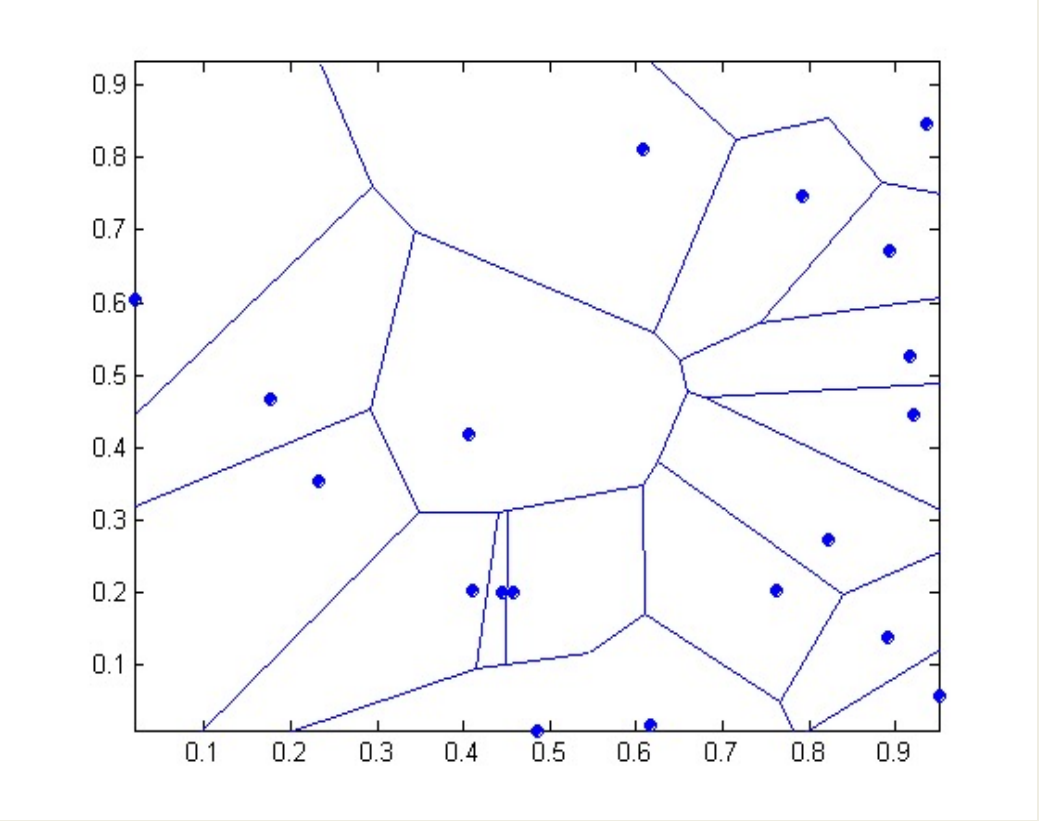


(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

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

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

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

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

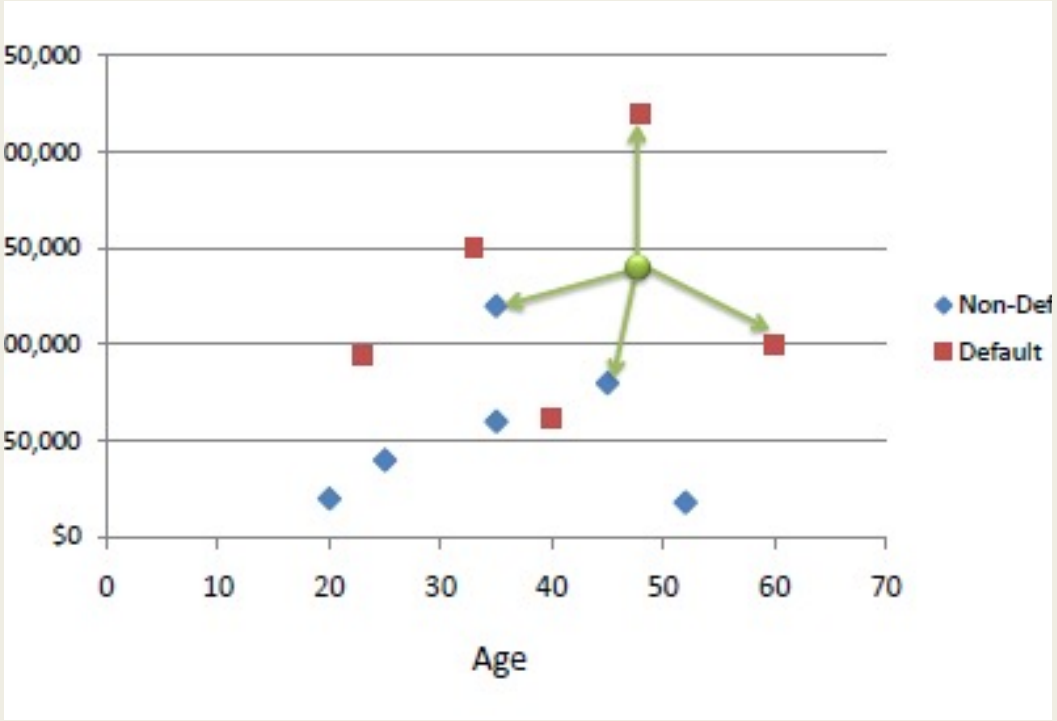
$$x = y \Rightarrow D = 0$$

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

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.

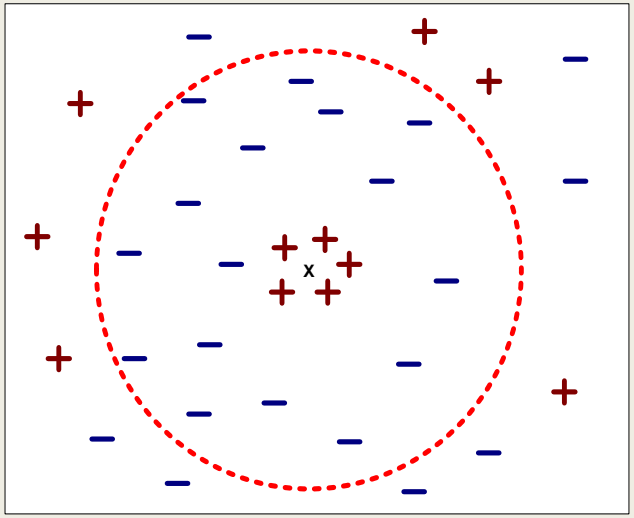
Age	Loan	Default	Distance
0.125	0.11	N	0.7652
0.375	0.21	N	0.5200
0.625	0.31	N	0.3160
0	0.01	N	0.9245
0.375	0.50	N	0.3428
0.8	0.00	N	0.6220
0.075	0.38	Y	0.6669
0.5	0.22	Y	0.4437
1	0.41	Y	0.3650
0.7	1.00	Y	0.3861
0.325	0.65	Y	0.3771
0.7	**0.61**	**?**	

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*



Nearest neighbor Classification…

- k-NN classifiers are lazy learners
  - *It does not build models explicitly*
  - *Unlike eager learners such as decision tree induction and rule-based systems*
  - *Classifying unknown records are relatively expensive*

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

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