

# Data Science

## Lecture 3:

# Classification

Speaker: Hong-Han Shuai (帥宏翰)

Department of Electrical and Computer engineering  
National Chiao Tung University

# Applications

- Automatic Speech Recognition
- Playing Go

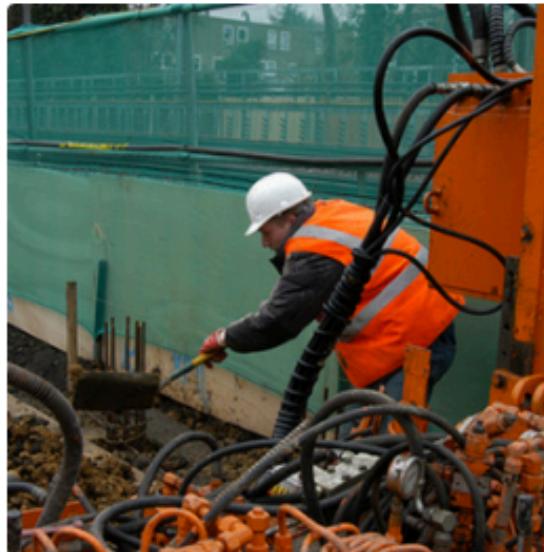


# Applications

- Automatic Image Caption Generation



"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."



"two young girls are playing with lego toy."

# From Learning to Machine Learning

- What's learning?
  - knowledge or skill acquired by instruction or study



- What's machine learning?



# From Learning to Machine Learning

- What's learning?
  - knowledge or skill acquired by instruction or study



- What's machine learning?



# A More Concrete Definition

- Skill: improve the performance measurements (e.g., prediction, 3pts shooting percentage)
- Therefore, machine learning is defined to be the improvements on some performance measurements by computing from data.
- For example,
  - Network data -> **ML** -> Better flow control
  - Signal data -> **ML** -> Faster antenna direction detection
  - Sequential web log data -> **ML** -> Higher accuracy of anomaly detection/efficiency of caching algorithm
  - Stock data -> **ML** -> More money

# The Machine Learning Route

- ML: an **alternative route** to build complicated systems
- Some Use Scenarios
  - when human cannot program the system manually
    - navigating on Mars
  - when human cannot ‘define the solution’ easily
    - speech/visual recognition
  - when needing rapid decisions that humans cannot do
    - high-frequency trading
  - when needing to be user-oriented in a massive scale
    - consumer-targeted marketing

# The Machine Learning Route

- ML: an **alternative route** to build complicated systems
- Some Use Scenarios
  - when human cannot program the system manually
    - navigating on Mars
  - when human cannot ‘define the solution’ easily
    - speech/visual recognition
  - when needing rapid decisions that humans cannot do
    - high-frequency trading
  - when needing to be user-oriented in a massive scale
    - consumer-targeted marketing

# The Machine Learning Route

- ML: an **alternative route** to build complicated systems
- Some Use Scenarios
  - when human cannot program the system manually
    - navigating on Mars
  - when human cannot ‘define the solution’ easily
    - speech/visual recognition
  - when needing rapid decisions that humans cannot do
    - high-frequency trading
  - when needing to be user-oriented in a massive scale
    - consumer-targeted marketing

# The Machine Learning Route

- ML: an **alternative route** to build complicated systems
- Some Use Scenarios
  - when human cannot program the system manually
    - navigating on Mars
  - when human cannot ‘define the solution’ easily
    - speech/visual recognition
  - when needing rapid decisions that humans cannot do
    - high-frequency trading
  - when needing to be user-oriented in a massive scale
    - consumer-targeted marketing

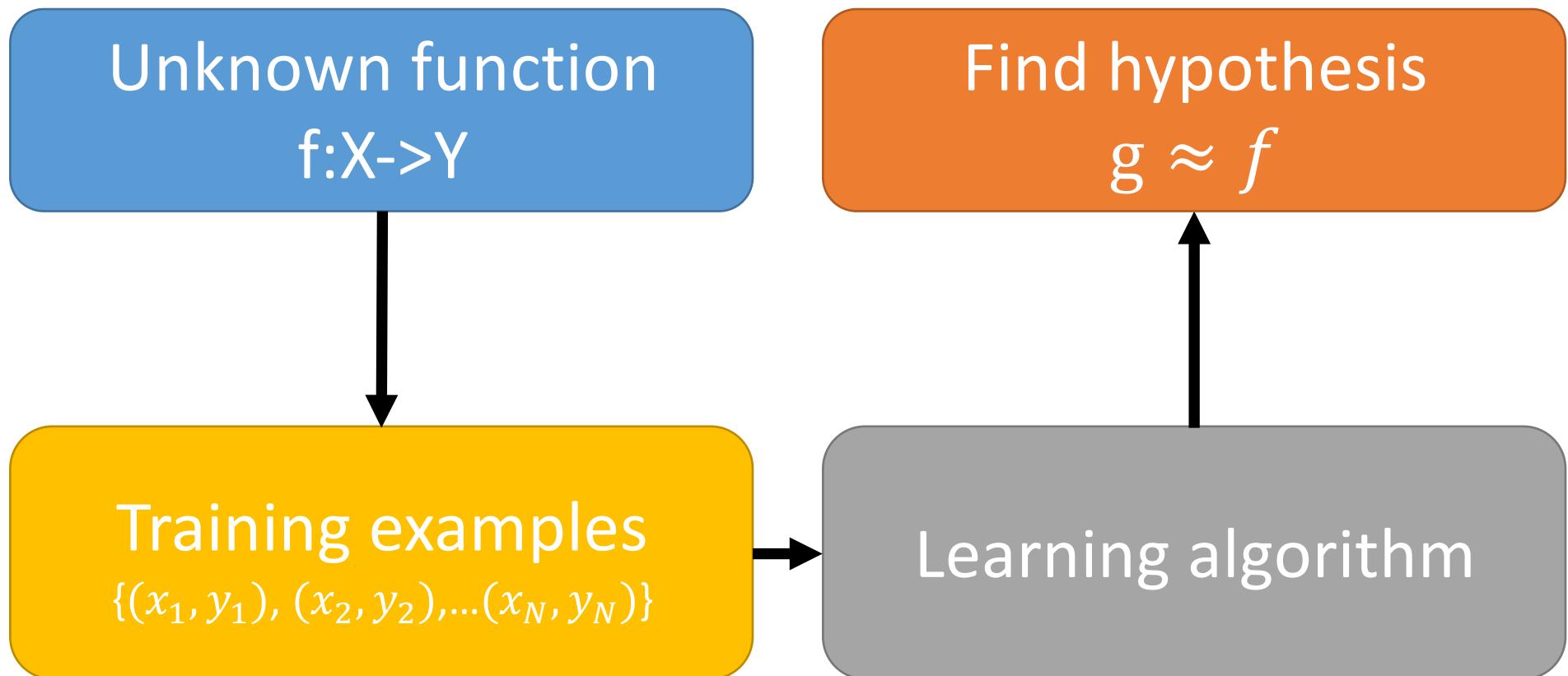
# Key Essence of Machine Learning

- Exists some ‘underlying pattern’ to be learned
  - So performance measure can be improved
- But no programmable (easy) definition
  - So machine learning is required
- Somehow there is data about the pattern
  - So machine learning has inputs to learn from

# Learning Problem Formulation

- Notation
  - **Input:**  $x \in X$ (application)
  - **Output:**  $y \in Y$ (good/bad after approving)
  - **Unknown pattern to be learned can be formulated as a function**
    - $f: X \rightarrow Y$  (ideal function)
  - **Data:**  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$
  - **Hypothesis:**
    - $g: X \rightarrow Y$  (hopefully can be as close to  $f$  as possible)

# Learning Flow



# Learning is to find a function

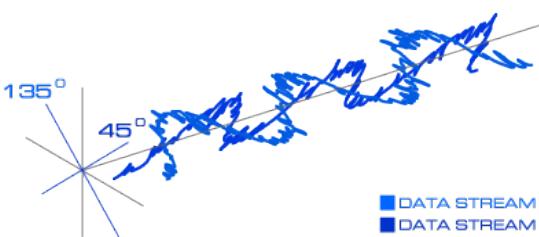
- Speech Recognition

$f($   ) = “我不知道你說什麼”

- Image recognition

$f($   ) = “Seafood”

- Channel estimation

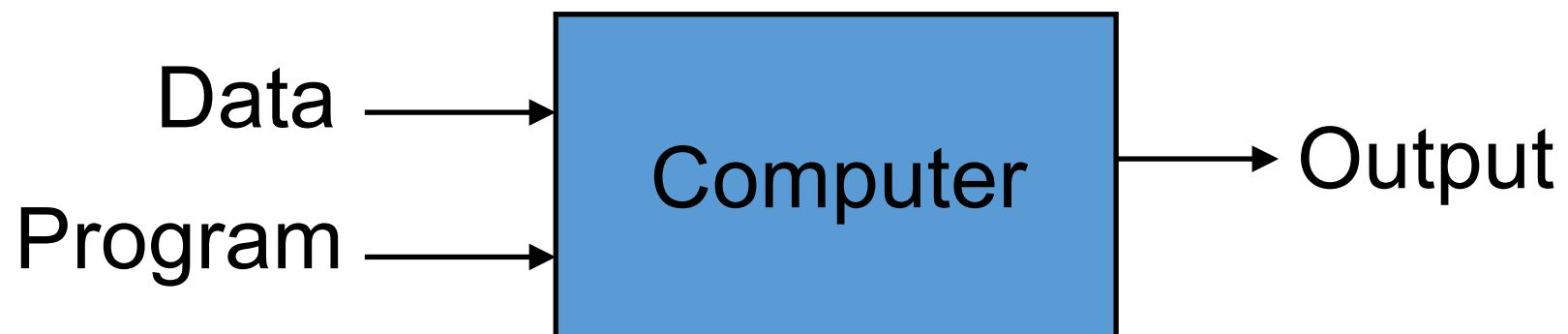
$f($   ) = Channel parameters

[http://cdn1.itpro.co.uk/sites/itpro/files/images/dir\\_176/it\\_photo\\_88225.jpg](http://cdn1.itpro.co.uk/sites/itpro/files/images/dir_176/it_photo_88225.jpg)

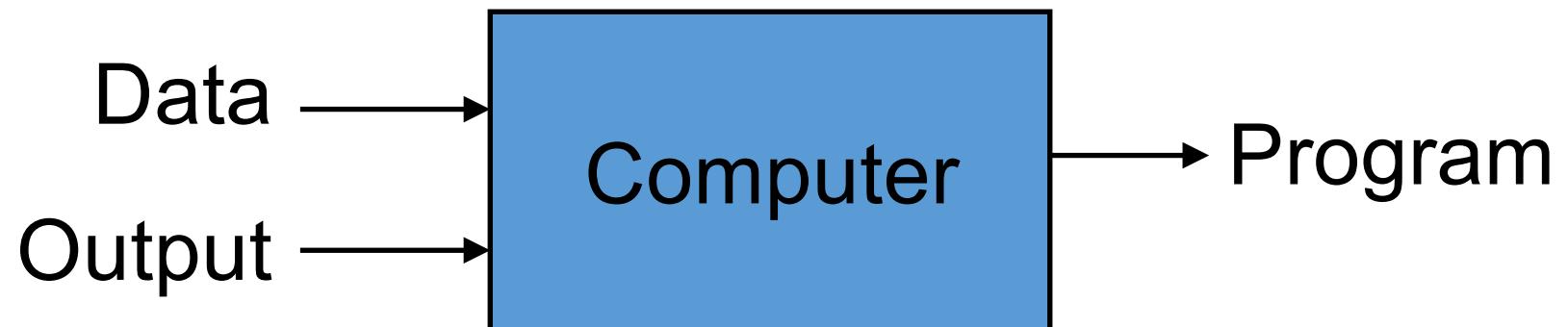
[http://www.berkeleywellness.com/sites/default/files/field/image/ThinkstockPhotos-520490716\\_field\\_img\\_hero\\_988\\_380.jpg](http://www.berkeleywellness.com/sites/default/files/field/image/ThinkstockPhotos-520490716_field_img_hero_988_380.jpg)

<https://telcoantennas.com.au/site/sites/default/files/images/4G-cross-polarisation-low-signal-areas.png>

### Traditional Programming



### Machine Learning



# Magic?

No, more like gardening

- **Seeds** = Algorithms
- **Nutrients** = Data
- **Gardener** = You
- **Plants** = Programs



# Classification

Let's go!

# Classification

- Classification – Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Lazy Learner
- Support Vector Machine
- Summary

# Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by **labels** indicating the class of the **observations**
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class **labels** of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

semi-supervised 有些資料有label有些沒有

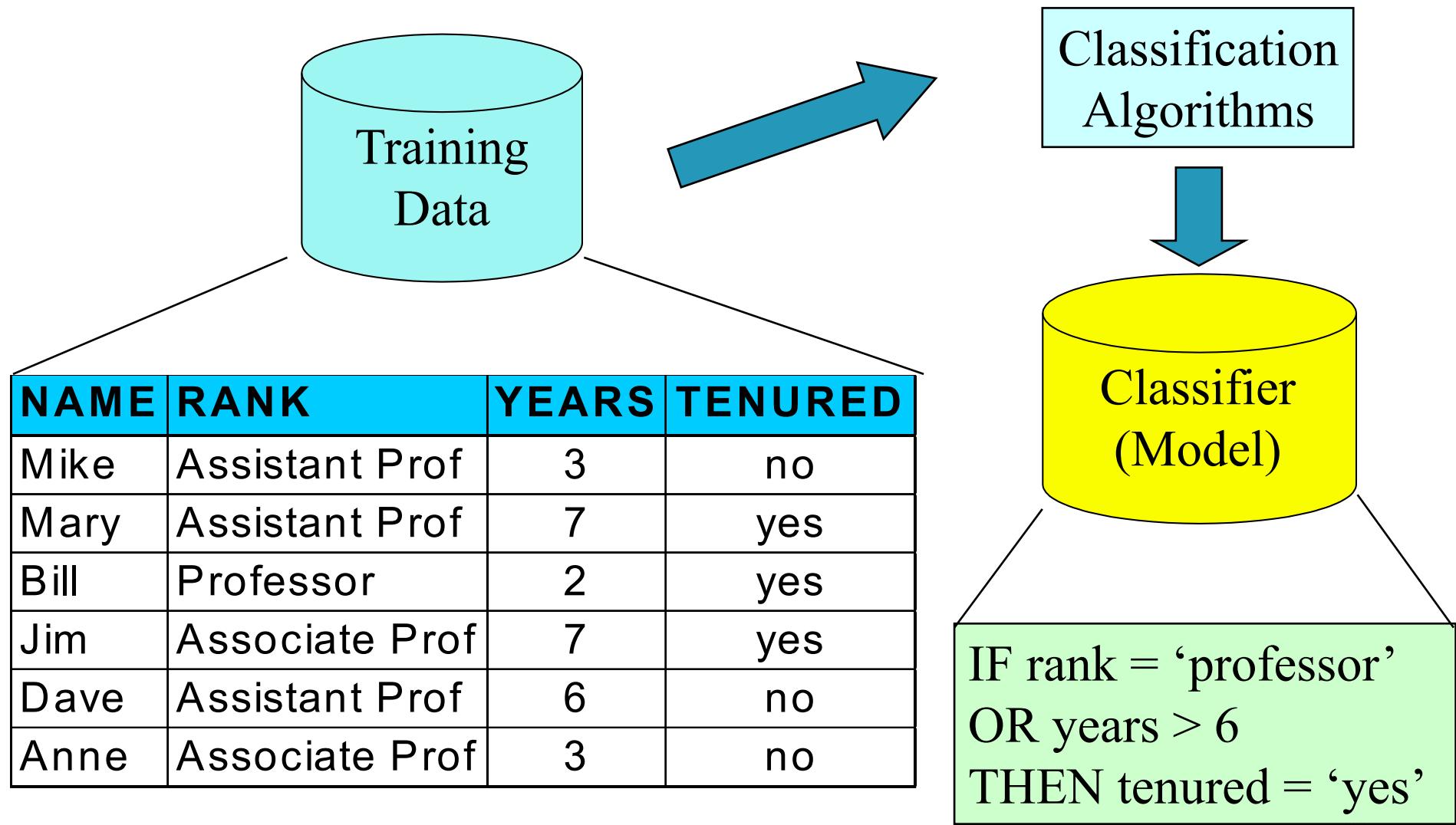
# Prediction Problems: Classification vs. Numeric Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (**class labels**) in a classifying attribute and uses it in classifying new data
- Numeric Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit/loan approval: if an applicant is qualified
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

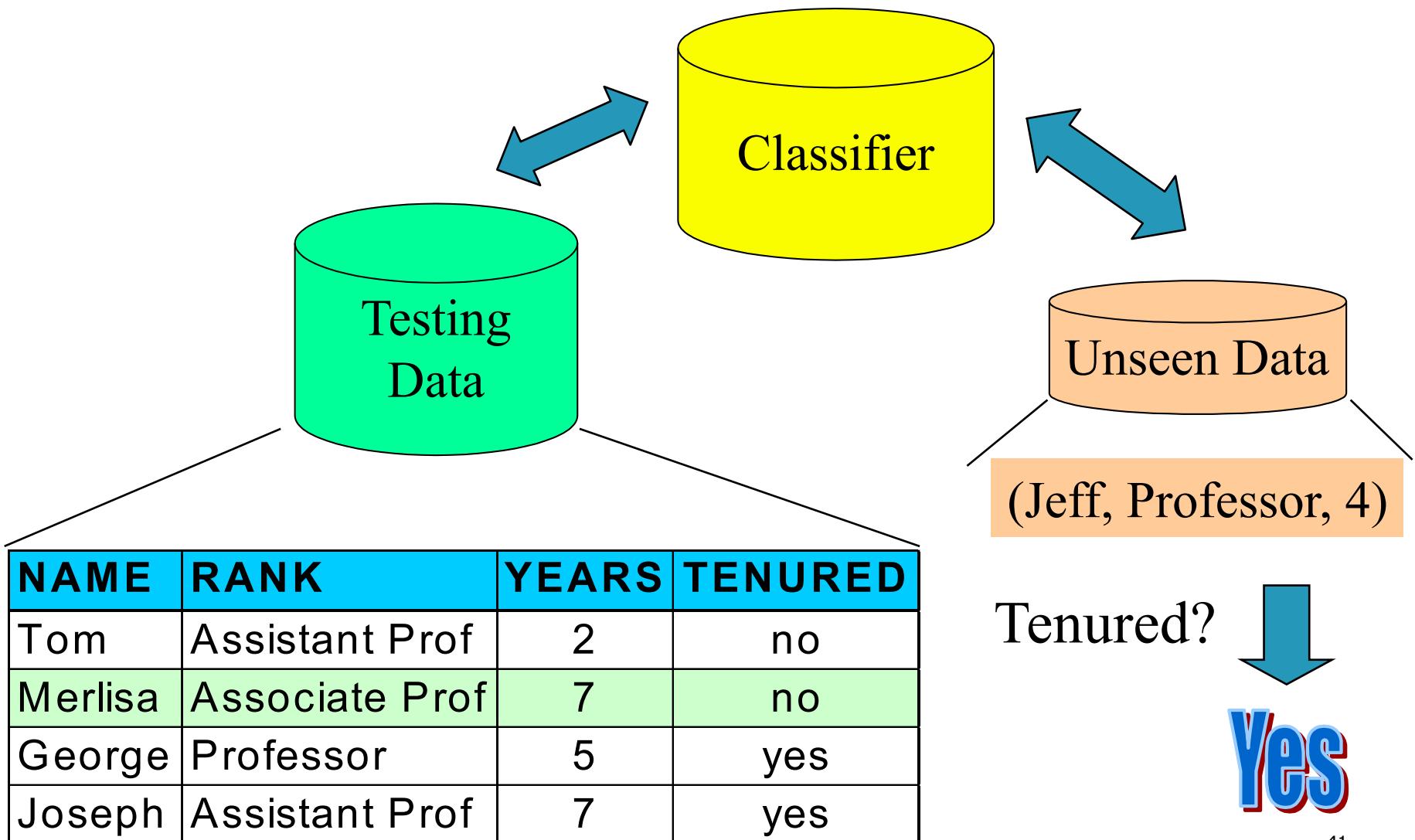
# Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the **class label** attribute
  - The set of tuples used for model construction is **training set**
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set (otherwise overfitting)
    - If the accuracy is acceptable, use the model to **classify new data**
  - Note: If *the test set* is used to select models, it is called **validation (test) set**

# Process (1): Model Construction



## Process (2): Using the Model in Prediction

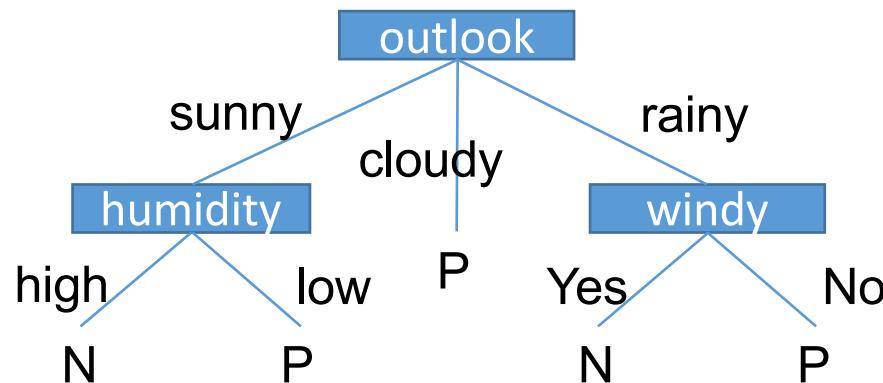


# Classification

- Classification – Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Lazy Learner
- Support Vector Machine
- Summary

# A Decision-Tree Based Classification

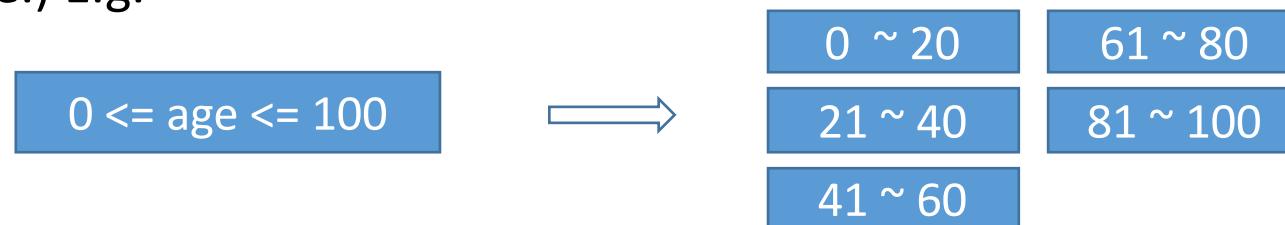
- A decision tree of whether going to play tennis or not:



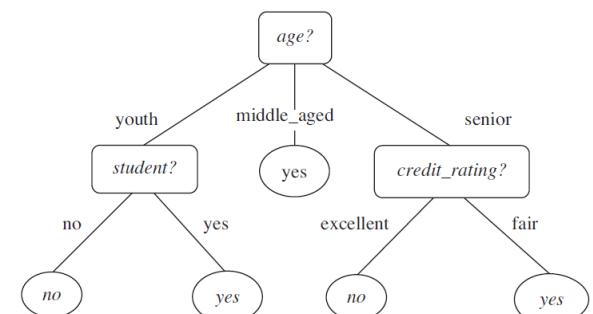
- **ID-3** and its extended version **C4.5** (Quinlan'93):  
A **top-down** decision tree generation algorithm

# Algorithm for Decision Tree Induction

- Basic algorithm (a **greedy algorithm**)
  - Tree is constructed in a **top-down recursive divide-and-conquer manner.**
  - Attributes are categorical.  
(if an attribute is a continuous number, it **needs to be discretized in advance.**) E.g.

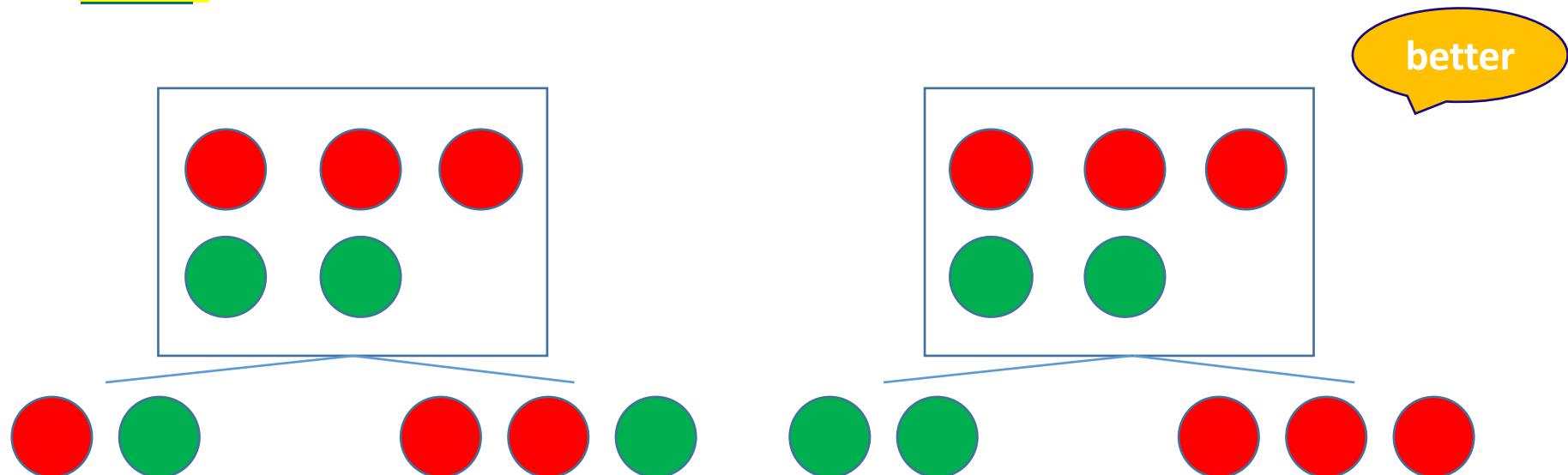


- At start, all the training examples are at the root.
- Examples are **partitioned recursively** based on selected **attributes**.



# Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain): maximizing an information gain measure, i.e., favoring the partitioning which makes the majority of examples belong to a single class.



# Algorithm for Decision Tree Induction

- Basic algorithm (a **greedy algorithm**)  
    啟發式
- Test attributes are selected on the basis of a heuristic or statistical measure (e.g., **information gain**): maximizing an information gain measure, i.e., **favoring the partitioning which makes the majority of examples belong to a single class.**
- Conditions for **stopping partitioning**:
  - All samples for a given node belong to the same class
  - There are **no remaining attributes** for further partitioning – **majority voting** is employed for classifying the leaf
  - There are **no samples left**

# Primary Issues in Tree Construction (1/2)

模型跟數據的擬合度 => 判斷模型好壞

- **Split criterion: Goodness function**

- Used to select the attribute to be split at a tree node during the tree generation phase
- Different algorithms may use different goodness functions:
  - Information gain (used in ID3/C4.5)
  - Gain ratio
  - Gini index (used in CART)

# Primary Issues in Tree Construction (2/2)

- **Branching scheme:**

- Determining the tree branch to which a sample belongs

- Binary vs.  $k$ -ary splitting



- When to stop the further splitting of a node?

e.g. impurity measure

- Labeling rule: a node is labeled as the class to which most samples at the node belongs.

# How to Use a Tree?

- Directly

- Test the attribute value of unknown sample against the tree.
- A path is traced from root to a leaf which holds the label.

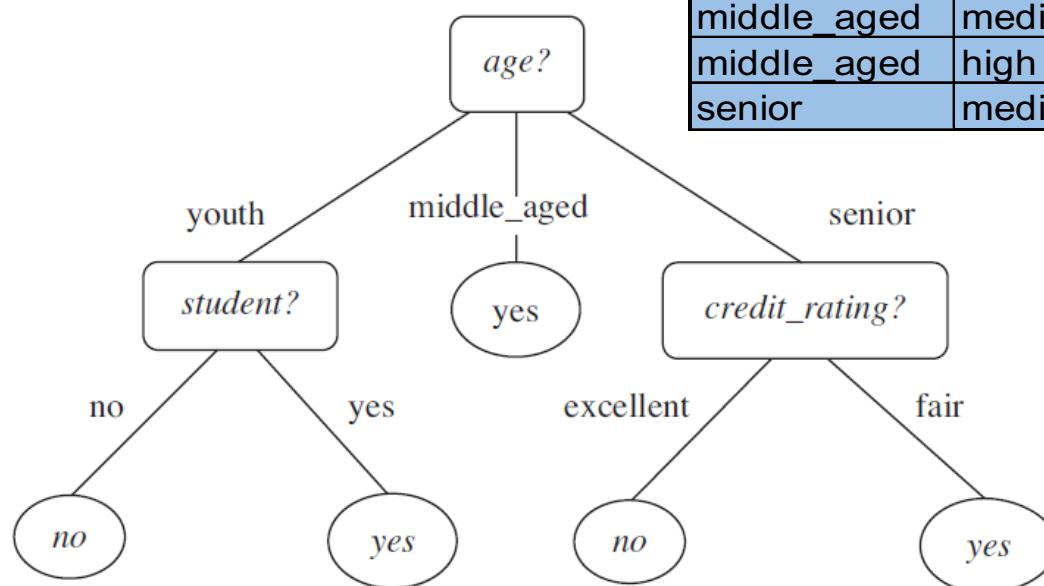
- Indirectly

- Decision tree is converted to classification rules.
- One rule is created for each path from the root to a leaf.
- **IF-THEN** is easier for humans to understand .

# Decision Tree Induction: An Example

- ❑ Training data set:  
Buys\_computer
- ❑ Resulting tree:

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no



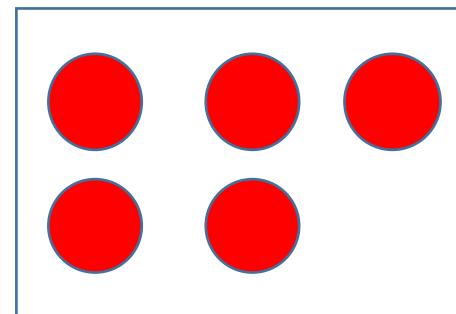
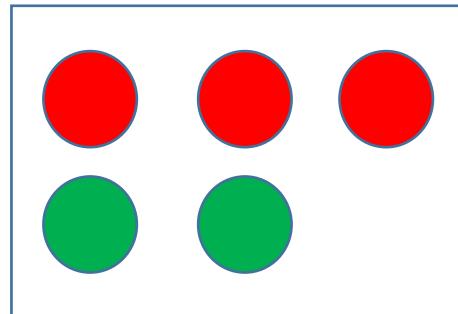
# Expected Information (Entropy)

- Expected information (**entropy**) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^m p_i \log_2(p_i)$$

取 $\log_2$ 為底，代表我需要多少 bits information 才能表示  
當紅色跟綠色同樣大小時候，info = 0

( $p_i$ : probability that a tuple in D belongs to class  $C_i$ ,  $m$ : number of classes)



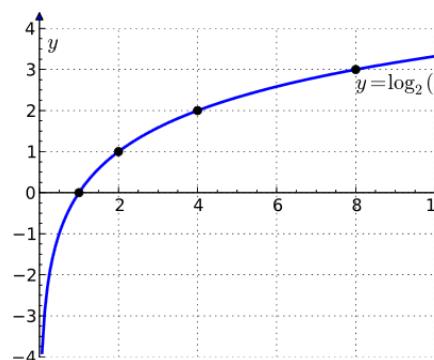
$$Info(D) = I(3,2) = -\frac{3}{5} \log_2\left(\frac{3}{5}\right) - \frac{2}{5} \log_2\left(\frac{2}{5}\right)$$

$$\approx -\frac{3}{5} \times (-0.737) - \frac{2}{5} \times (-1.322)$$

$$\approx 0.971$$

$$Info(D) = I(5,0) = -\frac{5}{5} \log_2\left(\frac{5}{5}\right) - \frac{0}{5} \log_2\left(\frac{0}{5}\right)$$

$$= 0 - 0 = 0$$

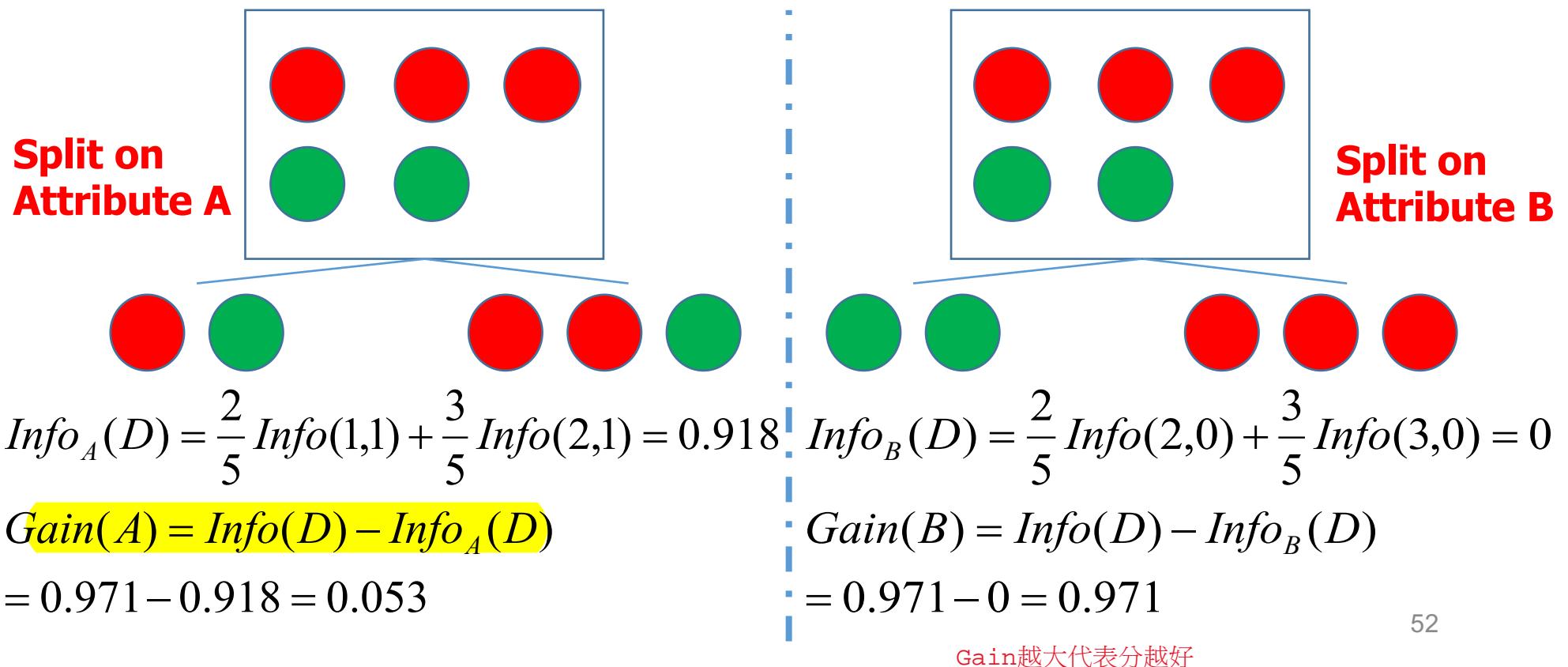


x 動的速度 > logx  
因此  $x \log x$  當 x 趨近於 0  
視為 0

# Expected Information (Entropy)

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times Info(D_j)$$



# Attribute Selection Measure: Information Gain (ID3)

- Select the **attribute** with the **highest information gain**
  - To **minimize # of tests** needed to classify a given tuple
- Let  $p_i$  be the probability that an arbitrary tuple in D belongs to class  $C_i$ , estimated by  $|C_{i,D}|/|D|$
- **Expected information** (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^m p_i \log_2(p_i)$$

- **Information** needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times Info(D_j)$$

- **Information gained** by branching on attribute A

We'd like to maximize  **$Gain(A)$** ,  
i.e., to minimize  **$Info_A(D)$** , the  
information still required to finish  
classifying the tuples

$$Gain(A) = Info(D) - Info_A(D)$$

# Attribute Selection: Information Gain

- Class P: `buys_computer = "yes"`
- Class N: `buys_computer = "no"`

$$Info(D) = I(9,5) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940$$

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) \\ + \frac{5}{14} I(3,2) = 0.694$$

age	$p_i$	$n_i$	$I(p_i, n_i)$
youth	2	3	0.971
middle_aged	4	0	0
senior	3	2	0.971

$\frac{5}{14} I(2,3)$  means “age = youth” has 5 out of 14 samples, with 2 yes’es and 3 no’s.  
Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

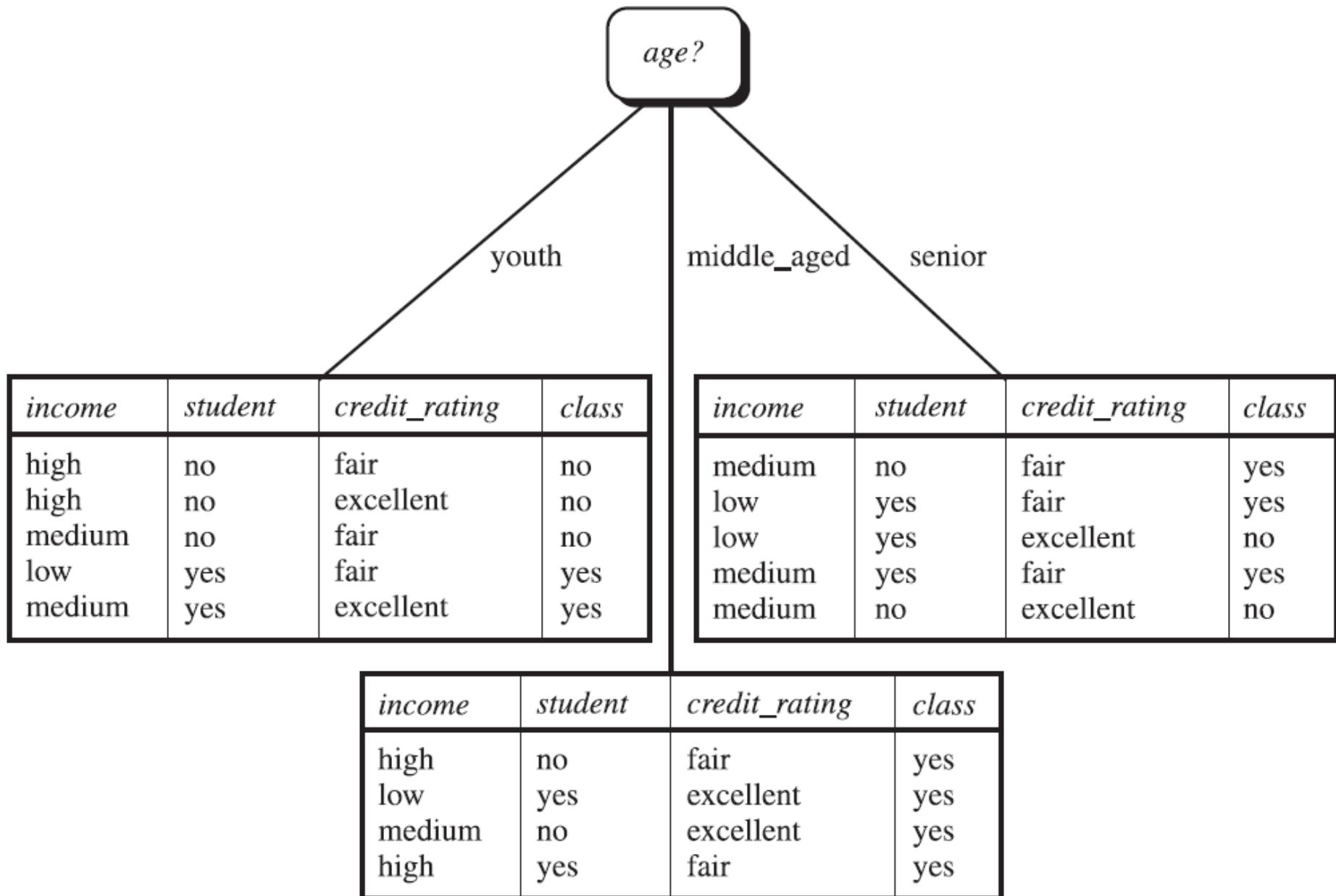
Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no



# Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a **continuous-valued** attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the **midpoint** between each pair of adjacent values is considered as a possible *split point*
    - $(a_i + a_{i+1})/2$  is the **midpoint** between the values of  $a_i$  and  $a_{i+1}$
    - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying  $A \leq \text{split-point}$ , and D2 is the set of tuples in D satisfying  $A > \text{split-point}$

# Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is **biased towards attributes with a large number of values** 純用information gain判斷會遇到unique id problem

- E.g., unique pID -> split on pID results in large number of partitions, each containing just one tuple => **each partition is pure**  
=> information required to classify this partition would be  $Info_{pID}(D)=0$ , i.e., the **information gain is maximal!!**

- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (**normalization** to information gain)

$$SplitInfo_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

- $\text{GainRatio}(A) = \text{Gain}(A)/\text{SplitInfo}(A)$

- E.g.,  $SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$

- $\text{gain\_ratio(income)} = 0.029/1.557 = 0.019$
- The attribute with the **maximum gain ratio** is selected as the splitting attribute

# Gini Index (CART, IBM IntelligentMiner)

- Gini index considers a **binary split** for each attribute
  - To construct a **binary** decision tree
- Consider **A**: discrete-valued attribute
  - with  $v$  distinct values  $\{a_1, a_2, \dots, a_v\}$
- Examine **all possible subsets** that can be formed using values of A
  - Each subset  $S_A$  is considered as a binary test: “ $A \in S_A$ ”
- E.g., if income has three possible values {low, medium, high}
  - $2^3$  possible subsets
    - $\{\text{low, medium, high}\}, \{\text{low, medium}\}, \{\text{low, high}\}, \{\text{medium, high}\}, \{\text{low}\}, \{\text{medium}\}, \{\text{high}\}, \{\}$
    - There are  $(2^v - 2)/2$  possible ways to form two partition of the data **D**, based on a binary split **A**

# Gini Index (CART, IBM IntelligentMiner)

- If a data set  $D$  contains examples from  $m$  classes, Gini index,  $Gini(D)$  is defined as

$$Gini(D) = 1 - \sum_{i=1}^m p_i^2$$

where  $p_i$  is the probability of class  $C_i$  in  $D$ , estimated by  $|C_{i,D}|/|D|$

- If a data set  $D$  is split on A into two subsets  $D_1$  and  $D_2$ , the **Gini index**  $Gini_A(D)$  given the split on A is:

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

- Reduction in Impurity:

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

- The attribute provides the largest reduction in impurity, i.e., maximized  $\Delta Gini(A)$ , is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

# Computation of Gini Index

- E.g., D has 9 tuples in **buys\_computer = “yes”** and 5 in **“no”**

$$Gini(D) = 1 - \left( \frac{9}{14} \right)^2 - \left( \frac{5}{14} \right)^2 = 0.459$$

- Suppose the attribute **income** partitions D into 10 in **D<sub>1</sub>**: {low, medium} and 4 in **D<sub>2</sub>**

$$\begin{aligned} Gini_{income \in \{low, medium\}}(D) &= \frac{10}{14} \left( 1 - \left( \frac{7}{10} \right)^2 - \left( \frac{3}{10} \right)^2 \right) + \frac{4}{14} \left( 1 - \left( \frac{2}{4} \right)^2 - \left( \frac{2}{4} \right)^2 \right) \\ &= \left( \frac{10}{14} \right) Gini(D_1) + \left( \frac{4}{14} \right) Gini(D_2) = 0.443 \\ &= Gini_{income \in \{high\}}(D). \end{aligned}$$

**Gini<sub>{low,high}</sub>** is **0.458**; **Gini<sub>{medium,high}</sub>** is **0.450**. Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index

# Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - **Information gain:**
    - biased towards multivalued attributes
  - **Gain ratio:**
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - **Gini index:**
    - biased to multivalued attributes
    - has difficulty when # of classes is large (high computation overhead)
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

# Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on  $\chi^2$  test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistic: has a close approximation to  $\chi^2$  distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

# Pre - Pruning

- **Based on statistical significance test**
  - Stop growing the tree when there is no statistically significant association between any attribute and the class at a particular node
  - Use all available data for training and apply the statistical test to estimate whether expanding/pruning a node is to produce an improvement beyond the training set
- **Most popular test:** chi-squared test
- $\chi^2 = \text{sum}((O-E)^2 / E)$

Where, O = observed data, E = expected values based on hypothesis.

## Example

- Example : 5 schools have the same test. Total score is 375, individual results are: 50, 93, 67, 78 and 87. Is this distribution significant, or was it just luck? Average is 75.

$$(50-75)^2/75 + (93-75)^2/75 + (67-75)^2/75 + (78-75)^2/75 + (87-75)^2/75 = 15.55$$

This distribution is significant !

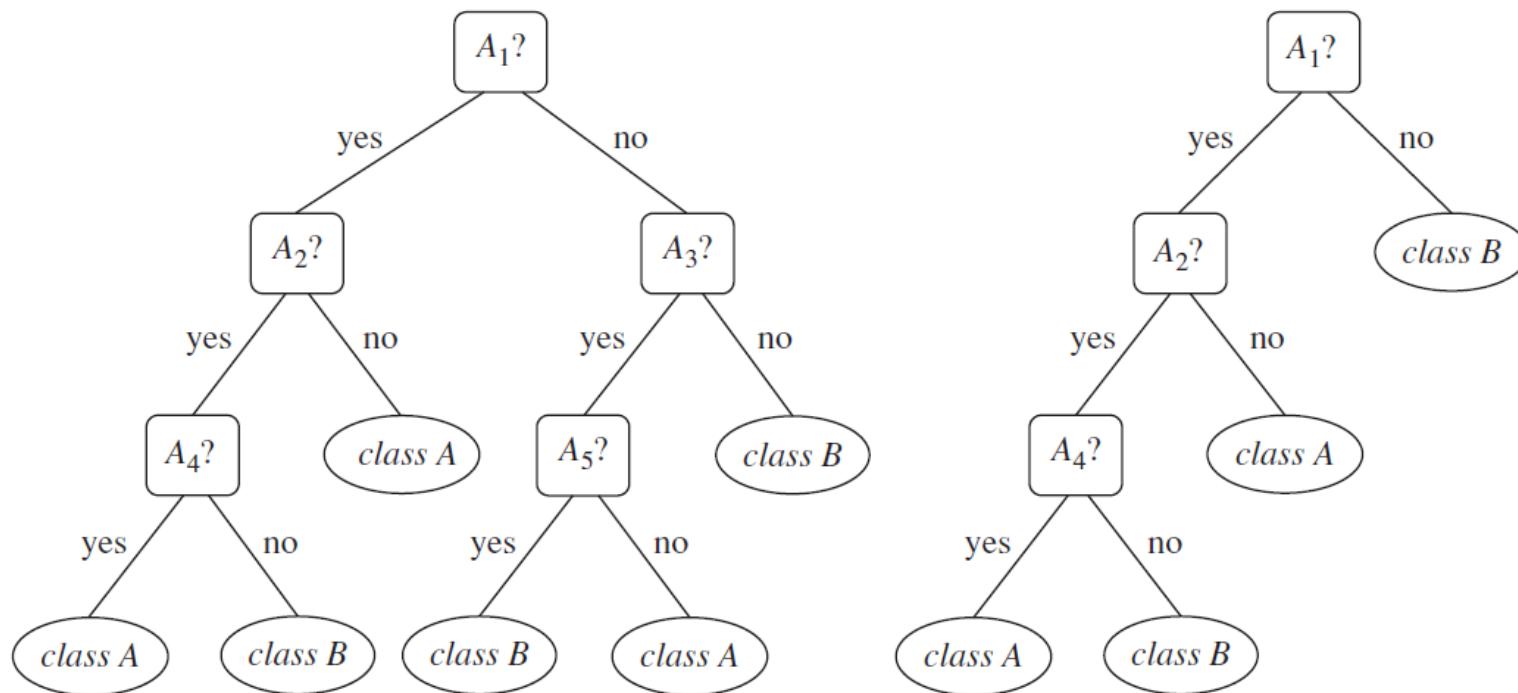
# Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data

## underfitting

# overfitting

- Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples, i.e., lose the ability of generalization



# Overfitting and Tree Pruning

- Two approaches to avoid overfitting
  - Prepruning: *Halt tree construction early*-do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: *Remove branches from a “fully grown” tree*—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the “best pruned tree”

# Enhancements to Basic Decision Tree Induction

- Allow for **continuous-valued attributes**
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- **Handle missing attribute values**
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- **Attribute construction**
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

# Classification

- Classification – Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Lazy Learner
- Support Vector Machine
- Summary

# Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction*, i.e., predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

# Bayes' Theorem: Basics

- Total probability Theorem: 
$$P(B) = \sum_{i=1}^M P(B|A_i)P(A_i)$$
- Bayes' Theorem: 
$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$
  - Let  $\mathbf{X}$  be a data sample (“evidence”): class label is unknown
  - Let  $H$  be a *hypothesis* that  $\mathbf{X}$  belongs to class C
  - Classification is to determine  $P(H|\mathbf{X})$ , (i.e., *posteriori probability*): the probability that the hypothesis holds given the observed data sample  $\mathbf{X}$
  - $P(H)$  (*prior probability*): the initial probability
    - E.g.,  $\mathbf{X}$  will buy computer, regardless of age, income, ...
  - $P(\mathbf{X})$ : probability that sample data is observed
  - $P(\mathbf{X}|H)$  (*likelihood*): the probability of observing the sample  $\mathbf{X}$ , given that the hypothesis holds
    - E.g., Given that  $\mathbf{X}$  will buy computer, the prob. that  $\mathbf{X}$  is 31..40, medium income

# Prediction Based on Bayes' Theorem

- Given training data  $\mathbf{X}$ , *posteriori probability of a hypothesis H*,  $P(H|\mathbf{X})$ , follows the Bayes' theorem

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

- Informally, this can be viewed as  
posteriori = likelihood x prior/evidence
- Predicts  $\mathbf{X}$  belongs to  $C_i$  iff the probability  $P(C_i|\mathbf{X})$  is the highest among all the  $P(C_k|\mathbf{X})$  for all the  $k$  classes
- Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost

# Classification is to Derive the Maximum Posteriori

- Let  $D$  be a training set of tuples and their associated class labels, and each tuple is represented by an  $n$ -D attribute vector  $\mathbf{X} = (x_1, x_2, \dots, x_n)$
- Suppose there are  $m$  classes  $C_1, C_2, \dots, C_m$ .
- Classification is to derive the maximum posteriori, i.e., the maximal  $P(C_i | \mathbf{X})$
- This can be derived from Bayes' theorem

$$P(C_i | \mathbf{X}) = \frac{P(\mathbf{X} | C_i) P(C_i)}{P(\mathbf{X})}$$

- Since  $P(\mathbf{X})$  is constant for all classes, only

$$P(C_i | \mathbf{X}) = P(\mathbf{X} | C_i) P(C_i)$$

needs to be maximized

# Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X} | C_i) = \prod_{k=1}^n P(x_k | C_i) = P(x_1 | C_i) \times P(x_2 | C_i) \times \dots \times P(x_n | C_i)$$

- This greatly reduces the computation cost: Only counts the class distribution
- If  $A_k$  is categorical,  $P(x_k | C_i)$  is the # of tuples in  $C_i$  having value  $x_k$  for  $A_k$  divided by  $|C_i|$  (# of tuples of  $C_i$  in D)
- If  $A_k$  is continuous-valued,  $P(x_k | C_i)$  is usually computed based on Gaussian distribution with a mean  $\mu$  and standard deviation  $\sigma$

and  $P(x_k | C_i)$  is

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(\mathbf{X} | C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

# Naïve Bayes Classifier: Training Dataset

Class:

C1:buys\_computer = 'yes'

C2:buys\_computer = 'no'

Data to be classified:

X = (age = youth,

income = medium,

student = yes

credit\_rating = Fair)

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

We need to maximize  $P(X|C_i)P(C_i)$ , for  $i = 1, 2$

# Naïve Bayes Classifier: An Example

- The prior probability of each class:

$$P(C_i) : P(\text{buys\_computer} = \text{"yes"}) = 9/14 = 0.643$$

$$P(\text{buys\_computer} = \text{"no"}) = 5/14 = 0.357$$

- Compute  $P(X|C_i)$  for each class

$$P(\text{age} = \text{youth} | \text{buys\_computer} = \text{"yes"}) = 2/9 = 0.222$$

$$P(\text{age} = \text{youth} | \text{buys\_computer} = \text{"no"}) = 3/5 = 0.6$$

$$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"yes"}) = 4/9 = 0.444$$

$$P(\text{income} = \text{"medium"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$$

$$P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{student} = \text{"yes"} | \text{buys\_computer} = \text{"no"}) = 1/5 = 0.2$$

$$P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{credit\_rating} = \text{"fair"} | \text{buys\_computer} = \text{"no"}) = 2/5 = 0.4$$

- $X = (\text{age} = \text{youth}, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair})$

$$P(X|C_i) : P(X|\text{buys\_computer} = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$$

$$P(X|\text{buys\_computer} = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$$

$$P(X|C_i) * P(C_i) : P(X|\text{buys\_computer} = \text{"yes"}) * P(\text{buys\_computer} = \text{"yes"}) = 0.028$$

$$P(X|\text{buys\_computer} = \text{"no"}) * P(\text{buys\_computer} = \text{"no"}) = 0.007$$

Therefore,  $X$  belongs to class ("buys\_computer = yes")

# Avoiding the Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be **non-zero**. Otherwise, the predicted prob. **will be zero**

$$P(X | C_i) = \prod_{k=1}^n P(x_k | C_i)$$

- E.g., Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)
- Use **Laplacian correction** (or Laplacian estimator)

- *Adding 1 to each case*

Prob(income = low) = 1/1003

Prob(income = medium) = 991/1003

Prob(income = high) = 11/1003

- The “corrected” prob. estimates are close to their “uncorrected” counterparts

# Naïve Bayes Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: **class conditional independence**, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals, patients: Profile: age, family history, etc.  
Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier

# Classification

- Classification – Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Lazy Learner
- Support Vector Machine
- Summary

# Using IF-THEN Rules for Classification

- Represent the knowledge in the form of **IF-THEN** rules

R: **IF** *age* = youth **AND** *student* = yes **THEN** *buys\_computer* = yes

- Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: *coverage* and *accuracy*

- $n_{covers}$  = # of tuples covered by R
- $n_{correct}$  = # of tuples correctly classified by R

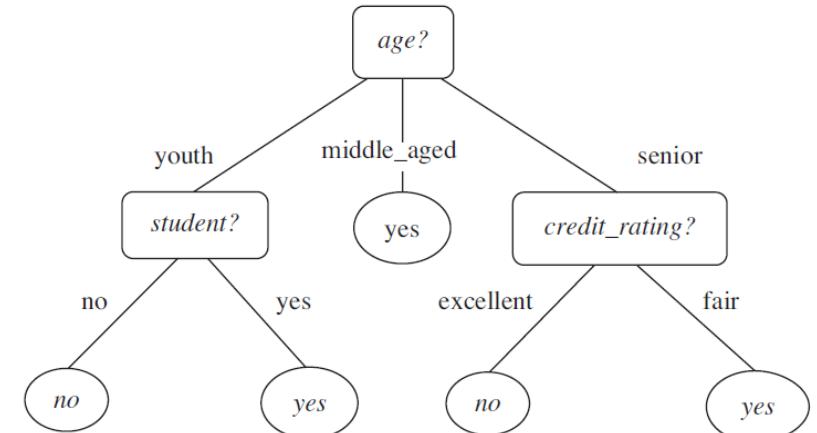
$$\text{coverage}(R) = n_{covers} / |D| \quad /* D: training data set */$$

$$\text{accuracy}(R) = n_{correct} / n_{covers}$$

- If more than one rule are triggered, need **conflict resolution**
  - **Size ordering**: assign the highest priority to the triggering rules that has the “toughest” requirement (i.e., with the **most attribute tests**)
  - **Class-based ordering**: decreasing order of *prevalence* or *misclassification cost per class*
  - **Rule-based ordering (decision list)**: rules are organized into one long priority list, according to some measure of rule quality or by experts

# Rule Extraction from a Decision Tree

- Rules are *easier to understand* than large trees
- One rule is created *for each path* from the root to a leaf
- Each attribute-value pair along a path forms a conjunction:  
the leaf holds the class prediction
- Rules are **mutually exclusive** and **exhaustive**  
No rule conflicts 互相排斥
- Example: Rule extraction from our *buys\_computer* decision-tree



One rule for each attribute-value combination

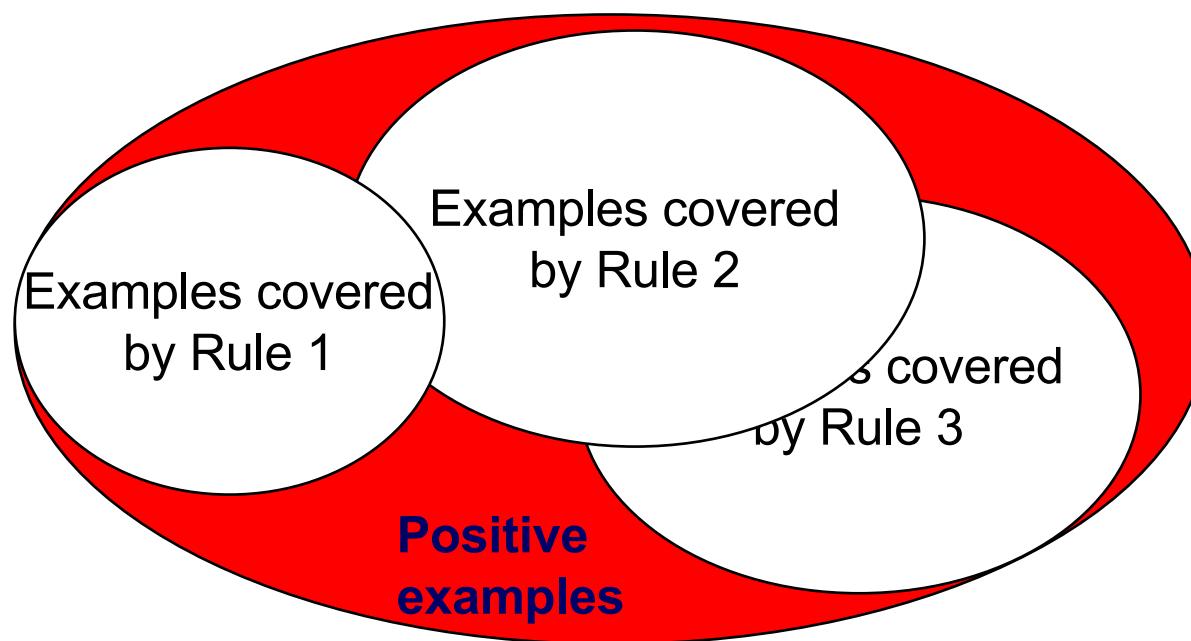
IF <i>age</i> = young AND <i>student</i> = no	THEN <i>buys_computer</i> = no
IF <i>age</i> = young AND <i>student</i> = yes	THEN <i>buys_computer</i> = yes
IF <i>age</i> = mid-age	THEN <i>buys_computer</i> = yes
IF <i>age</i> = old AND <i>credit_rating</i> = excellent	THEN <i>buys_computer</i> = no
IF <i>age</i> = old AND <i>credit_rating</i> = fair	THEN <i>buys_computer</i> = yes

# Rule Induction: Sequential Covering Method

- Sequential covering algorithm =>  
**Extracts rules directly from training data**
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned *sequentially*, each for a given class  $C_i$  **will cover many tuples of  $C_i$  but none (or few) of the tuples of other classes**
- Steps:
  - Rules are learned one at a time  
(i.e., procedure *Learn-One-Rule*, detailed later)
  - Each time a rule is learned, the tuples **covered by the rules** are **removed**
    - Repeat the process on the remaining tuples until *termination condition*, e.g., when no more training examples or when the quality of a rule returned is below a user-specified threshold
  - Comp. w. decision-tree induction:  
Decision-Trees learn a set of rules *simultaneously*

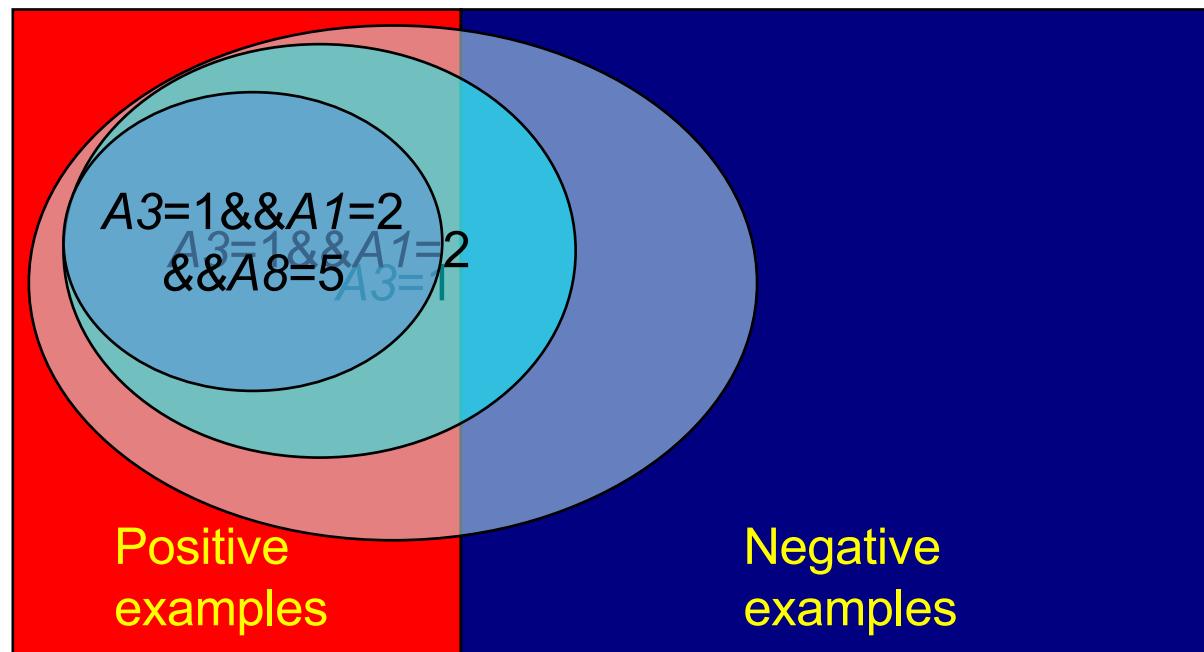
# Sequential Covering Algorithm

```
while (enough target tuples left)
    generate a rule
    remove positive target tuples satisfying this rule
```



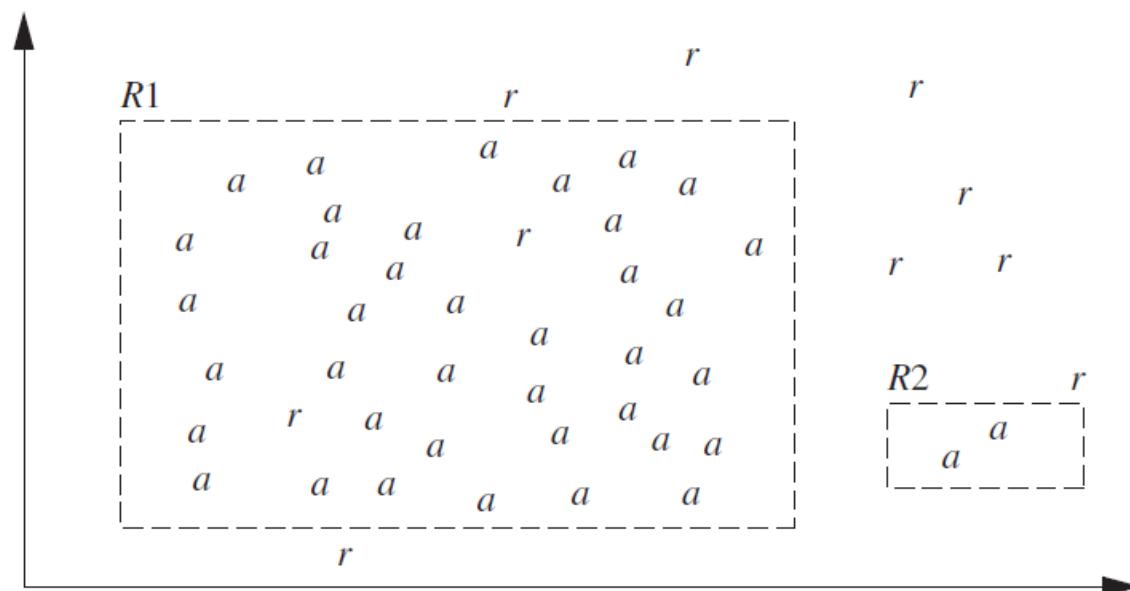
# Rule Generation

- To generate a rule
  - while(true)**
    - find the best predicate  $p$
    - if** foil-gain( $p$ ) > threshold **then** add  $p$  to current rule
    - else** break



# How to *Learn-One-Rule*?

- Start with the *most general rule* possible: condition = empty
- *Adding new attributes* by adopting a greedy depth-first strategy
  - Picks the one that **most improves the rule quality**
- Rule-Quality measures: consider both **coverage** and **accuracy**



Rule R1 has accuracy 95%, while R2 has 100% accuracy.  
However, R2 is not better because of its poor coverage

## How to *Learn-One-Rule*?

- Rule-Quality measures: consider both coverage and accuracy
  - Foil-gain (in FOIL & RIPPER): assesses info\_gain by extending condition  
(**pos'** and **neg'** are the # of positive, negative tuples covered by our new rule **R'**, respectively)

$$FOIL\_Gain = pos' \times (\log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg})$$

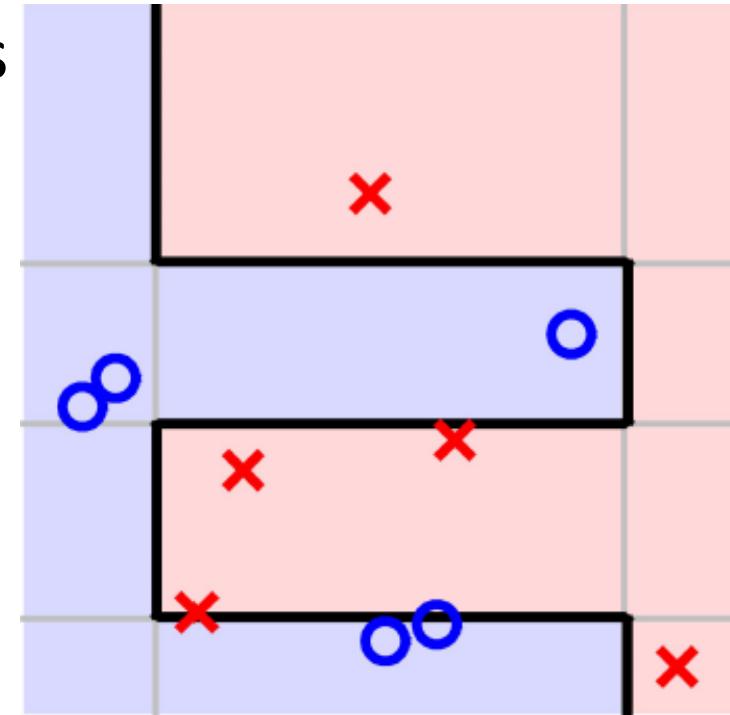
- favors rules that have high accuracy and cover many positive tuples

# Classification

- Classification – Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Lazy Learner
- Support Vector Machine
- Summary

# Motivation

- students: simple hypotheses  $g_t$  (like vertical/horizontal lines)
- (Class): sophisticated hypothesis G (like black curve)
- Teacher: a tactic learning algorithm that **directs the students to focus on key examples**



# Goal of Supervised Learning?

- Minimize the probability of model prediction errors on **future** data
- Two Competing Methodologies
  - Build **one** really good model
    - Traditional approach
  - Build **many** models and average the results
    - Ensemble learning (more recent)

# The Single Model Philosophy

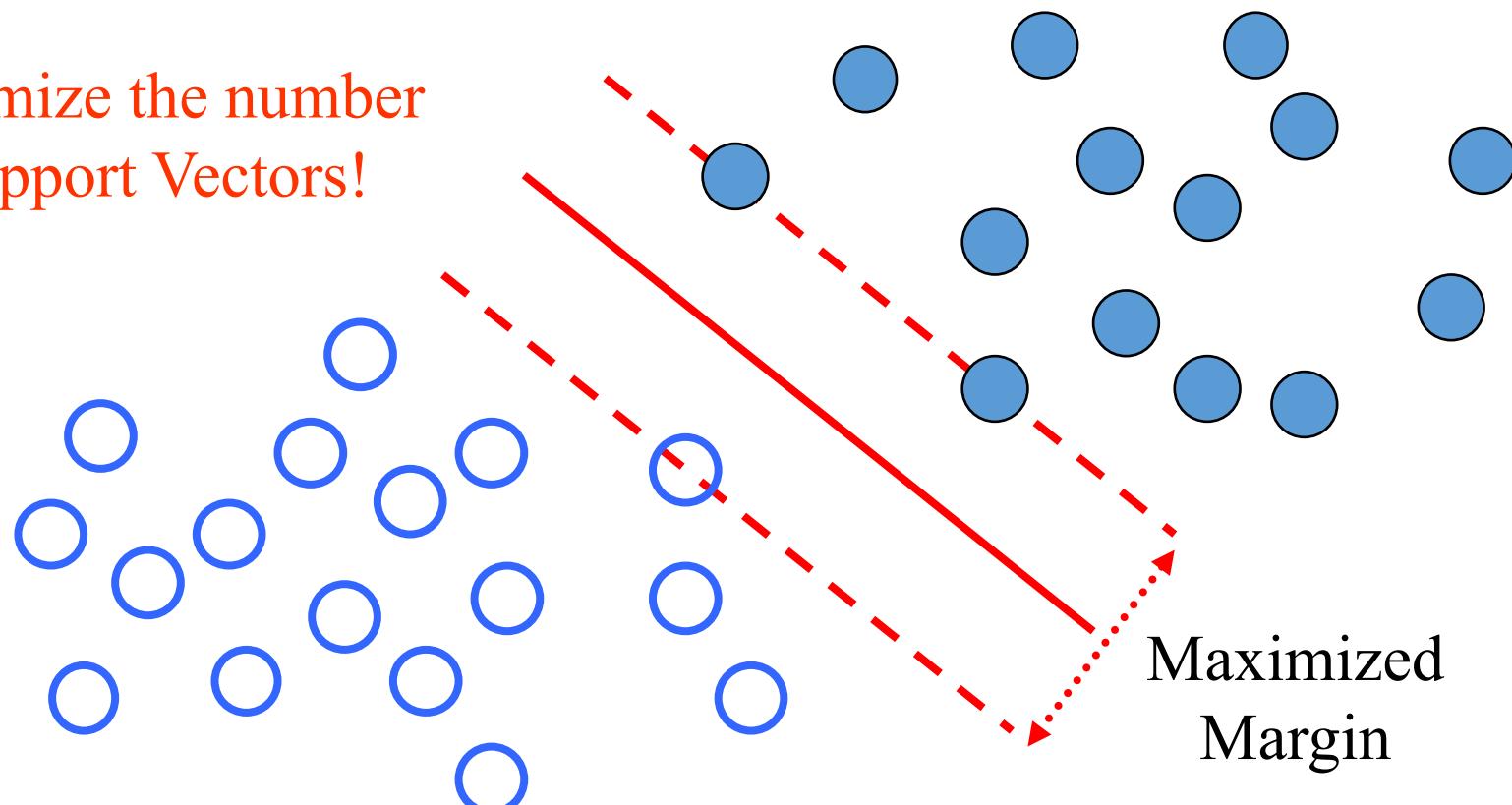
- Motivation: Occam's Razor
  - “one should not increase, beyond what is necessary, the number of entities required to explain anything”
- Infinitely many models can explain any given dataset
  - Might as well pick the smallest one...

# Exact Occam's Razor Models

- Exact approaches find optimal solutions
- Examples:
  - Support Vector Machines
    - Find a model structure that uses the smallest percentage of training data (to explain the rest of it).
  - Bayesian approaches
    - Minimum description length

# How Do Support Vector Machines Define Small?

Minimize the number  
of Support Vectors!



# Approximate Occam's Razor Models

- Approximate solutions use a greedy search approach which is not optimal
- Examples
  - Kernel Projection Pursuit algorithms
    - Find a minimal set of kernel projections
  - Relevance Vector Machines
    - Approximate Bayesian approach
  - Sparse Minimax Probability Machine Classification
    - Find a minimum set of kernels and features

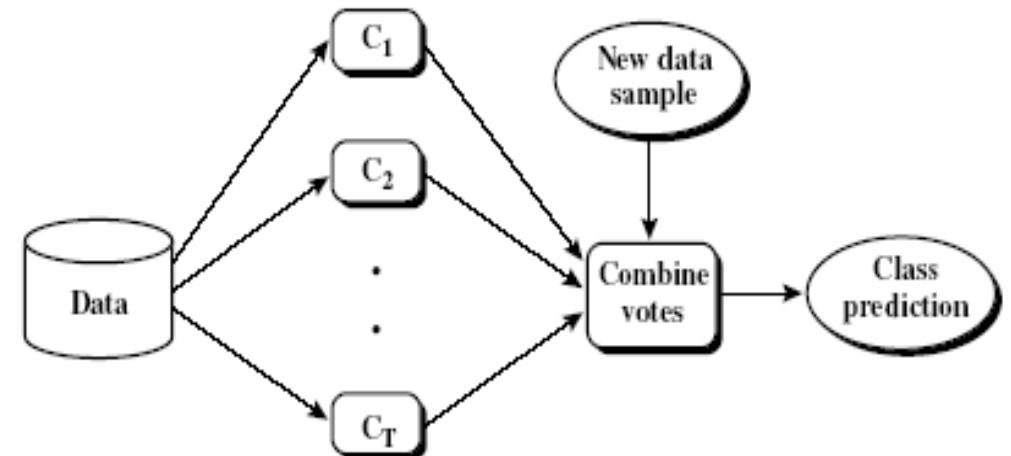
# Other Single Models: Not Necessarily Motivated by Occam's Razor

- Minimax Probability Machine (MPM)
- Trees
  - Greedy approach to sparseness
- Neural Networks
- Nearest Neighbor
- Basis Function Models
  - e.g. Kernel Ridge Regression

# Ensemble Philosophy

- Build many models and combine them
- Only through averaging do we get at the truth!
- It's too difficult (*impossible?*) to build a single model that works best
- Two types of approaches:
  - Models that don't use randomness
  - Models that incorporate randomness

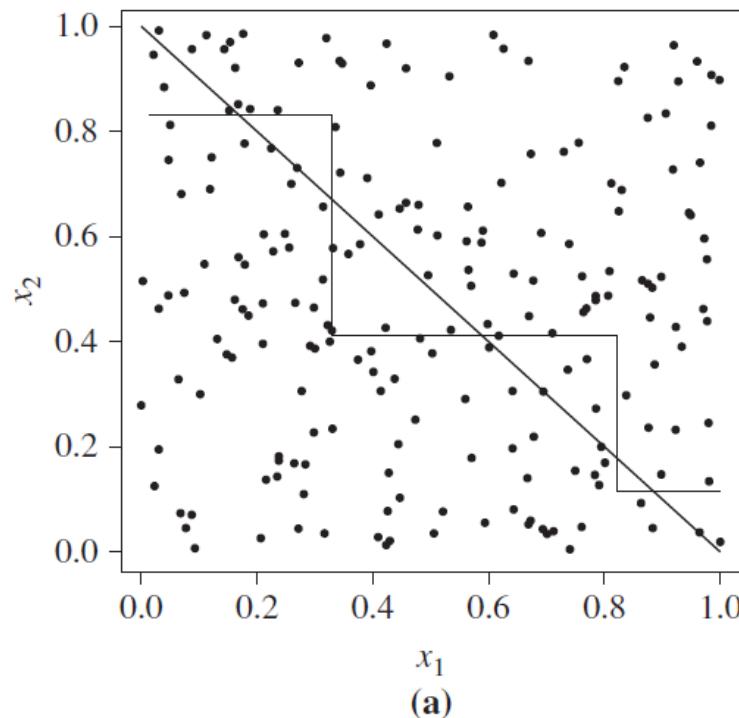
# Ensemble Methods: Increasing the Accuracy



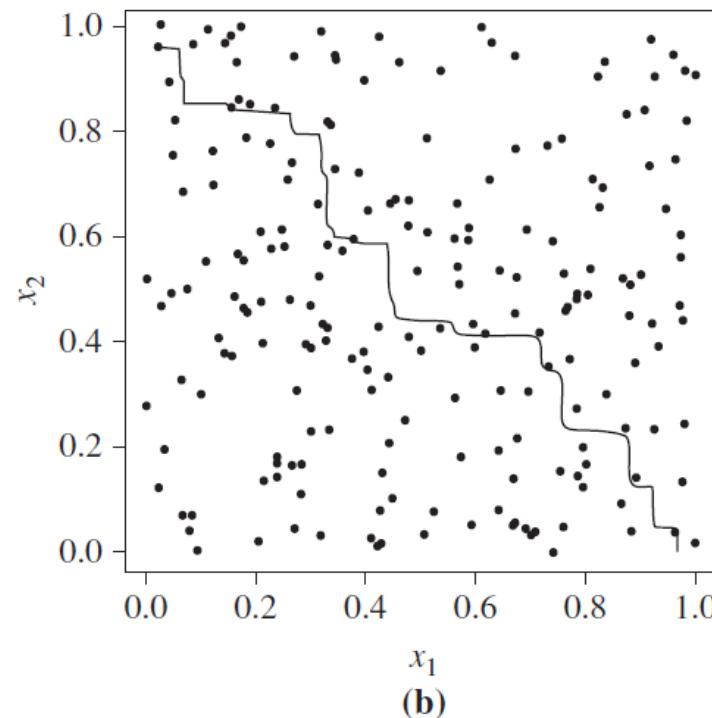
- Ensemble methods
  - Use a **combination of models** to increase accuracy
  - Combine a series of  $k$  learned models,  $M_1, M_2, \dots, M_k$ , with the aim of creating an improved model  $M^*$
  - Ideally, **significant diversity (little correlation)** among the models
- Popular ensemble methods
  - **Bagging:** averaging the prediction over a collection of classifiers
  - **Boosting:** weighted vote with a collection of classifiers
  - **Ensemble:** combining a set of heterogeneous classifiers

# Ensemble methods

- For a two-class problem with attributes  $x_1, x_2$



(a)

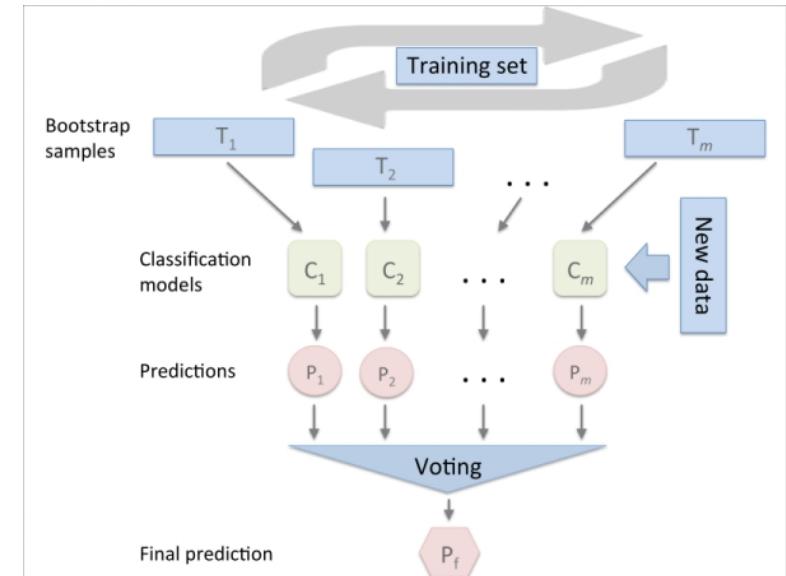


(b)

Decision boundary by (a) a single decision tree and (b) an ensemble of decision trees for a linearly separable problem (i.e., where the actual decision boundary is a straight line). The decision tree struggles with approximating a linear boundary. The decision boundary of the ensemble is closer to the true boundary. *Source:* From Seni and Elder [SE10]. © 2010 Morgan & Claypool Publishers; used with permission.

# Bagging: Bootstrap Aggregation

- Analogy: Diagnosis based on multiple doctors' **majority vote**
- Training
  - Given a set  $D$  of  $d$  tuples, at each iteration  $i$ , a training set  $D_i$  of  $d$  tuples is sampled with replacement from  $D$  (i.e., **bootstrap**)
    - A classifier model  $M_i$  is learned for each training set  $D_i$
- Classification: classify an unknown sample  $X$ 
  - Each classifier  $M_i$  returns its class prediction
  - The bagged classifier  $M^*$  **counts the votes** and assigns the class with the **most votes** to  $X$
- Prediction: can be applied to the prediction of **continuous values** by taking the **average value** of each prediction for a given test tuple
- Accuracy
  - Often significantly better than a single classifier derived from  $D$
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction



# Boosting

- Analogy: Consult **several doctors**, based on a combination of **weighted** diagnoses—**weight assigned based on the previous diagnosis accuracy**
- How boosting works?
  - **Weights** are assigned to each training tuple
  - A series of  $k$  classifiers is iteratively learned
  - After a classifier  $M_i$  is learned, the **weights** are updated to allow the subsequent classifier,  $M_{i+1}$ , to **pay more attention to the training tuples that were misclassified** by  $M_i$
  - The final  **$M^*$  combines the votes** of each individual classifier, where the **weight** of each classifier's vote **is a function of its accuracy**
- Boosting algorithm can be extended for numeric prediction
- Comparing with bagging: Boosting tends to have **greater accuracy**, but it also risks **overfitting** the model to misclassified data

# Adaboost (Freund and Schapire, 1997)

- Given  $d$  class-labeled tuples ( $y_i$  is class label):  $(\mathbf{X}_1, y_1), \dots, (\mathbf{X}_d, y_d)$
- Initially, all the **weights** of tuples are set the same  $(1/d)$
- Generate  $k$  classifiers in  $k$  rounds. At round  $i$ ,
  - Tuples from  $D$  are sampled (with replacement) to form a training set  $D_i$  of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model  $M_i$  is derived from  $D_i$
  - Its error rate is calculated using  $D_i$  as a test set
  - If a tuple is misclassified, its weight is increased, o.w. it is decreased
- Error rate:  $\text{err}(\mathbf{X}_j)$  is the misclassification error of tuple  $\mathbf{X}_j$ . Classifier  $M_i$  error rate is the sum of the weights of the misclassified tuples:

$$\text{error}(M_i) = \sum_j^d w_j \times \text{err}(\mathbf{X}_j)$$

# Adaboost (Freund and Schapire, 1997)

## Weight update

- If tuple in round  $i$  was **correctly** classified, **its weight is multiplied by**

$$\frac{\text{error}(M_i)}{1 - \text{error}(M_i)}$$

where  $M_i$  is the classification model derived from round  $i$ .

- The weights of all tuples will be **normalized**, which can be viewed as increasing the weights of misclassifies tuples

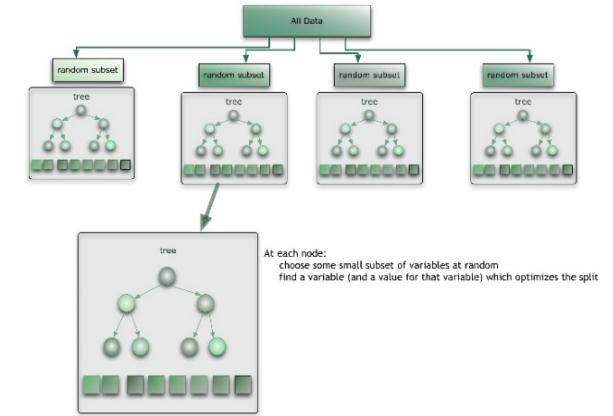
## Prediction

- Adaboost assigns a weight to each classifier's  $M_i$ 's vote
- The weight of classifier  $M_i$ 's vote is:

$$\log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)}$$

- The class with highest sum is returned as the prediction

# Random Forest



- Random Forest:
  - Each classifier in the ensemble is a **decision tree** classifier and is generated using a random selection of **attributes at each node** to determine the split
  - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node,  $F$  attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but **more robust to errors and outliers**
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

# Beyond the Buzz

- Quotes from <http://www.kdnuggets.com/2017/08/deep-learning-not-ai-future.html>
- “*Too many startups and products are named “deep-something”, just as buzzword: **very few** are using DL really*”
- “***Decision Trees*** like XGBoost are not making headlines, but silently beat DL at many Kaggle tabular data competitions.”
- “***Legal nightmare*** of DL decisions, that even if correct, can’t be ***explained*** when legally questioned”
- “*For many tasks, Deep Learning AI is or will become **illegal**, not compliant*”
  - “***GDPR***, about automated decision-making, requires the right to an explanation, and to prevent discriminatory effects based on race, opinions, health, etc.”
  - “***Fines for noncompliance are 4% of global revenue.***”

# Classification of **Class-Imbalanced** Data Sets

- Class-imbalance problem: **Rare positive example but numerous negative ones**, e.g., medical diagnosis, fraud, oil-spill, fault, etc.
- Traditional methods assume a balanced distribution of classes and equal error costs: **not suitable for class-imbalanced data**
- Typical methods for imbalance data in 2-class classification:
  - **Oversampling**: re-sampling of data from positive class
  - **Under-sampling**: randomly eliminate tuples from negative class
  - **Threshold-moving**: moves the decision threshold,  $t$ , so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
  - **Ensemble techniques**: Ensemble multiple classifiers introduced above
- Still difficult for class imbalance problem on multiclass tasks

# Multiclass Classification

- Classification involving more than two classes (i.e.,  $> 2$  Classes)
- Method 1. **One-vs.-all** (OVA): Learn a classifier one at a time
  - Given  $m$  classes, train  $m$  classifiers: one for each class
  - Classifier  $j$ : treat tuples in class  $j$  as *positive* & all others as *negative*
  - To classify a tuple  $X$ , the set of classifiers vote as an ensemble
- Method 2. **All-vs.-all** (AVA): Learn a classifier for each pair of classes
  - Given  $m$  classes, construct  $m(m-1)/2$  binary classifiers
  - A classifier is trained using tuples of the two classes
  - To classify a tuple  $X$ , each classifier votes.  $X$  is assigned to the class with maximal vote
- Comparison
  - All-vs.-all tends to be superior to one-vs.-all
  - Problem: Binary classifier is sensitive to errors, and errors affect vote count

# Classification

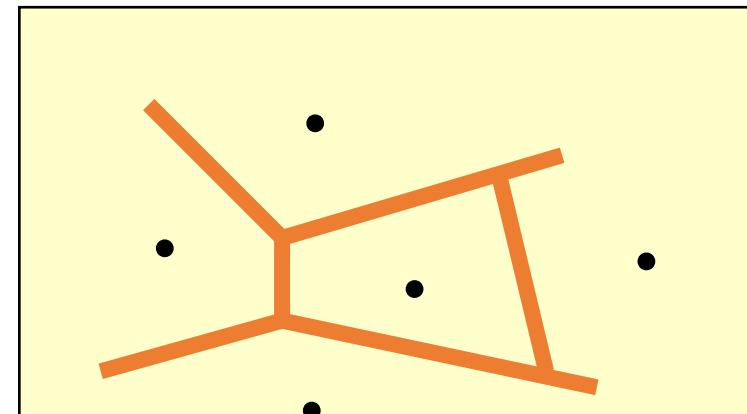
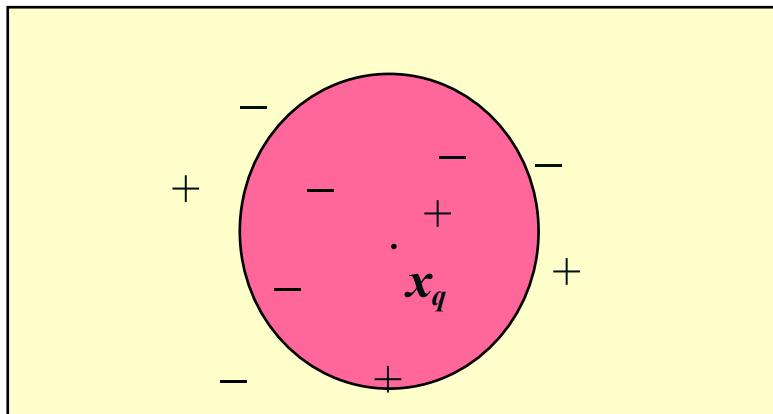
- Classification – Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Lazy Learner
- Support Vector Machine
- Summary

# Lazy vs. Eager Learning

- Lazy vs. eager learning
  - **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - **Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form an implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

# The $k$ -Nearest Neighbor Algorithm

- All instances correspond to points in the n-D space
- The nearest neighbor are defined in terms of Euclidean distance,  $\text{dist}(\mathbf{X}_1, \mathbf{X}_2)$
- Target function could be discrete- or real- valued
- For discrete-valued,  $k$ -NN returns the most common value among the  $k$  training examples nearest to  $x_q$
- Voronoi diagram: the decision surface induced by 1-NN for a typical set of training examples



# Discussion on the $k$ -NN Algorithm

- $k$ -NN for real-valued prediction for a given unknown tuple
  - Returns the mean values of the  $k$  nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the  $k$  neighbors according to their distance to the query  $x_q$ 
    - Give greater weight to closer neighbors
- Robust to noisy data by averaging  $k$ -nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
  - To overcome it, axes stretch or elimination of the least relevant attributes

$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

# Classification

- Classification – Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Lazy Learner
- **Support Vector Machine**
- Summary

# Support Vector Machines

- Please refer to another set of slides

# Model Evaluation and Selection

- Evaluation metrics: How can we measure **accuracy**?  
Other metrics to consider?
- Use **validation test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap
- Comparing classifiers:
  - Cost-benefit analysis and ROC Curves

# Classifier Evaluation Metrics: Confusion Matrix

## Confusion Matrix:

Actual class \ Predicted class	$C_1$	$\neg C_1$
$C_1$	True Positives (TP)	False Negatives (FN)
$\neg C_1$	False Positives (FP)	True Negatives (TN)

## Example of Confusion Matrix:

Actual class \ Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	6954	46	7000
buy_computer = no	412	2588	3000
Total	7366	2634	10000

- Given  $m$  classes, an entry,  $CM_{i,j}$  in a **confusion matrix** indicates # of tuples in class  $i$  that were labeled by the classifier as class  $j$
- May have extra rows/columns to provide totals

# Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

A\P	C	$\neg C$	
C	TP	FN	P
$\neg C$	FP	TN	N
	P'	N'	All

- **Classifier Accuracy**, or recognition rate: percentage of test set tuples that are correctly classified

$$\text{Accuracy} = (\text{TP} + \text{TN})/\text{All}$$

- **Error rate**:  $1 - \text{accuracy}$ , or

$$\text{Error rate} = (\text{FP} + \text{FN})/\text{All}$$

- **Class Imbalance Problem:**
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
- **Sensitivity**: True Positive recognition rate
  - **Sensitivity** =  $\text{TP}/\text{P}$
- **Specificity**: True Negative recognition rate
  - **Specificity** =  $\text{TN}/\text{N}$

# Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision:** exactness – what % of tuples that the classifier labeled as positive are actually positive

$$precision = \frac{TP}{TP + FP}$$

- Does not care how many positive instances are mislabeled as negative, i.e., FN

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P}$$

- But did not tell how many negative instances are mislabeled as positive, i.e., FP

- Perfect score is 1.0
- Inverse relationship between precision & recall
- **F measure ( $F_1$  or F-score):** harmonic mean of precision and recall,

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

- **$F_\beta$ :** weighted measure of precision and recall

- assigns  $\beta$  times as much weight to recall as to precision

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

# Classifier Evaluation Metrics: Example

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	<b>90</b>	<b>210</b>	300	30.00 ( <i>sensitivity</i> )
cancer = no	<b>140</b>	<b>9560</b>	9700	98.56 ( <i>specificity</i> )
Total	230	9770	10000	96.40 ( <i>accuracy</i> )

- $Precision = 90/230 = 39.13\%$        $Recall = 90/300 = 30.00\%$

Measure	Formula
accuracy, recognition rate	$\frac{TP + TN}{P + N}$
error rate, misclassification rate	$\frac{FP + FN}{P + N}$
sensitivity, true positive rate, recall	$\frac{TP}{P}$
specificity, true negative rate	$\frac{TN}{N}$
precision	$\frac{TP}{TP + FP}$
$F$ , $F_1$ , $F$ -score, harmonic mean of precision and recall	$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$
$F_\beta$ , where $\beta$ is a non-negative real number	$\frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$

# Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout  $k$  times, accuracy = avg. of the accuracies obtained
- **Cross-validation ( $k$ -fold, where  $k = 10$  is most popular)**
  - Randomly partition the data into  $k$  *mutually exclusive* subsets, each approximately equal size
  - At  $i$ -th iteration, use  $D_i$  as test set and others as training set
  - Leave-one-out:  $k$  folds where  $k = \#$  of tuples, for small sized data
  - \*Stratified cross-validation\*: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

# Cross-validation visualized

Available Labeled Data

Identify n partitions



# Cross-validation visualized

Available Labeled Data

Identify n partitions



Fold 2



# Cross-validation visualized

Available Labeled Data

Identify n partitions



Fold 3



# Cross-validation visualized

Available Labeled Data

Identify n partitions



# Cross-validation visualized

Available Labeled Data

Identify n partitions



Calculate Average Performance

# Cross-validation visualized

Available Labeled Data

Identify n partitions



Train

Dev

Test

Develop/Tuning set for adjusting  
hyper parameters

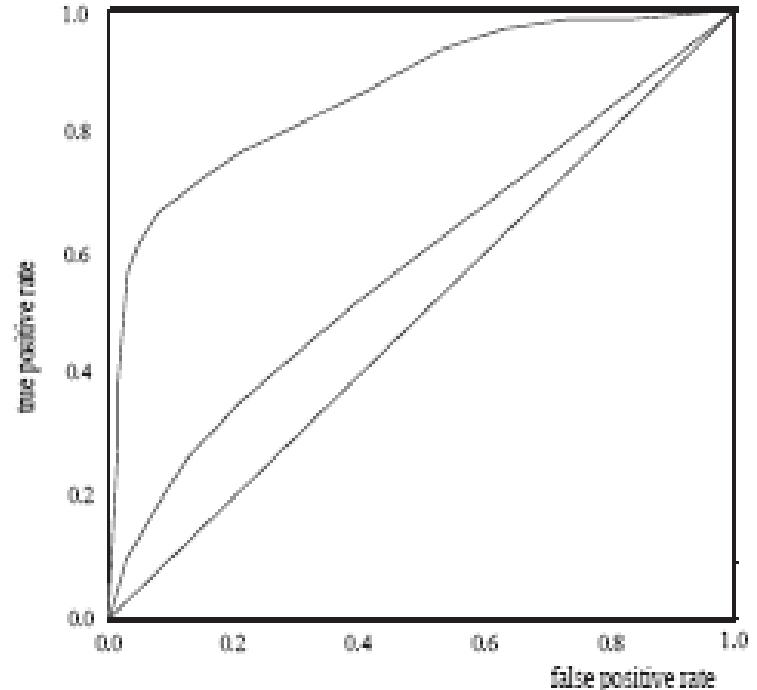
# Evaluating Classifier Accuracy: Bootstrap

- **Bootstrap**
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, and a common one is **.632 bootstrap**
  - A data set with  $d$  tuples is sampled  $d$  times, with replacement, resulting in a training set of  $d$  samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since  $(1 - 1/d)^d \approx e^{-1} = 0.368$ )
  - Repeat the sampling procedure  $k$  times, overall accuracy of the model:

$$Acc(M) = \frac{1}{k} \sum_{i=1}^k (0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set})$$

# Model Selection: ROC Curves

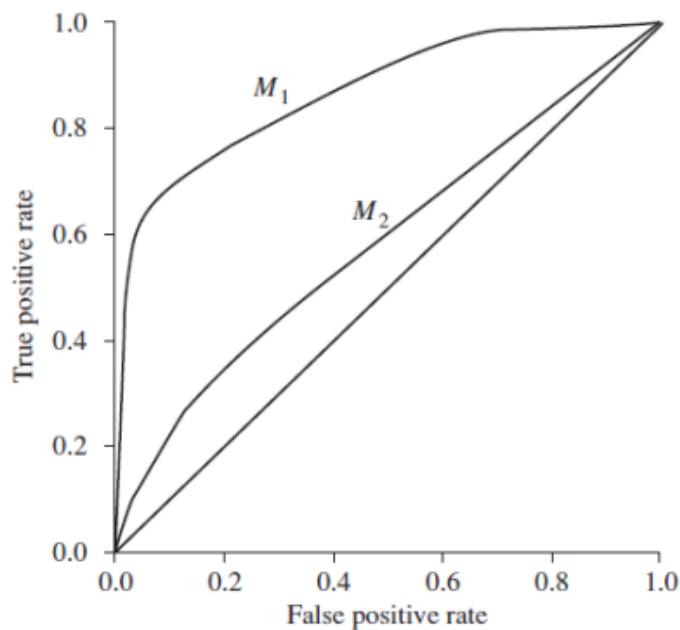
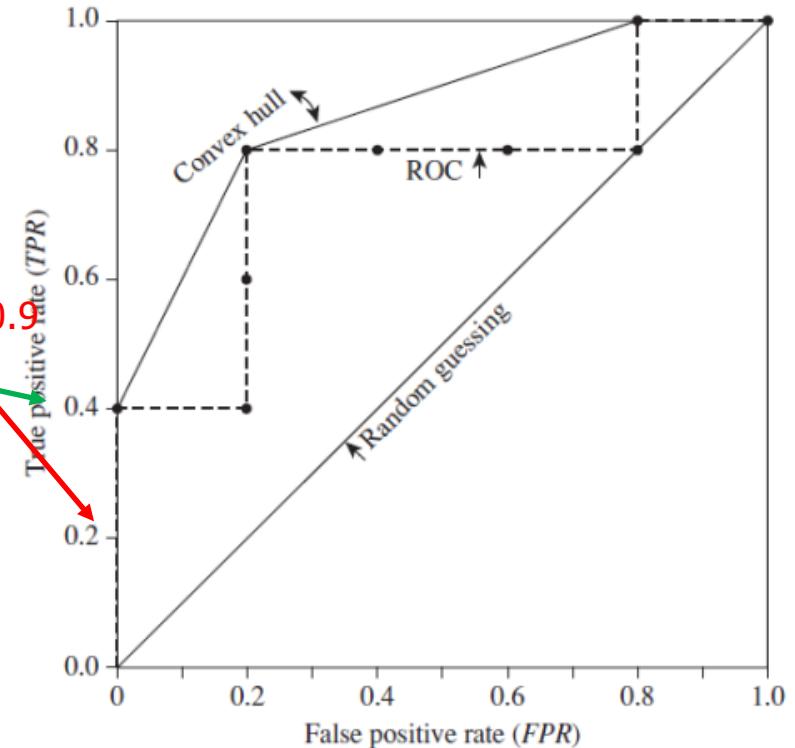
- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Shows the trade-off between the true positive rate and the false positive rate
- The **area** under the ROC curve is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The **closer** to the diagonal line (i.e., the closer the area is to 0.5), the **less** accurate is the model



- **Vertical axis** represents the **true positive** rate
- **Horizontal axis** rep. the **false positive** rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

# ROC curves

Tuple #	Class	Prob.	TP	FP	TN	FN	TPR	FPR
1	P	0.90	1	0	5	4	0.2	0
2	P	0.80	2	0	5	3	0.4	0
3	N	0.70	2	1	4	3	0.4	0.2
4	P	0.60	3	1	4	2	0.6	0.2
5	P	0.55	4	1	4	1	0.8	0.2
6	N	0.54	4	2	3	1	0.8	0.4
7	N	0.53	4	3	2	1	0.8	0.6
8	N	0.51	4	4	1	1	0.8	0.8
9	P	0.50	5	4	1	0	1.0	0.8
10	N	0.40	5	5	0	0	1.0	1.0



# Classification

- Classification – Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Lazy Learner
- Support Vector Machine
- **Summary**

# Issues Affecting Model Selection

- **Accuracy**
  - classifier accuracy: predicting class label
- **Speed**
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- **Robustness:** handling noise and missing values
- **Scalability:** efficiency in disk-resident databases
- **Interpretability**
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

## Summary (I)

- Classification is a form of data analysis that extracts models describing important data classes.
- Effective and scalable methods have been developed for decision tree induction, Naive Bayesian classification, rule-based classification, and many other classification methods.
- Evaluation metrics include: accuracy, sensitivity, specificity, precision, recall,  $F$  measure, and  $F_\beta$  measure.
- Stratified k-fold cross-validation is recommended for accuracy estimation. Bagging and boosting can be used to increase overall accuracy by learning and combining a series of individual models.

## Summary (II)

- Significance tests and ROC curves are useful for model selection.
- There have been numerous comparisons of the different classification methods; the matter remains a research topic
- No single method has been found to be superior over all others for all data sets
- Issues such as accuracy, training time, robustness, scalability, and interpretability must be considered and can involve trade-offs, further complicating the quest for an overall superior method

# References (1)

- C. Apte and S. Weiss. **Data mining with decision trees and decision rules**. Future Generation Computer Systems, 13, 1997
- C. M. Bishop, **Neural Networks for Pattern Recognition**. Oxford University Press, 1995
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. **Classification and Regression Trees**. Wadsworth International Group, 1984
- C. J. C. Burges. **A Tutorial on Support Vector Machines for Pattern Recognition**. *Data Mining and Knowledge Discovery*, 2(2): 121-168, 1998
- P. K. Chan and S. J. Stolfo. **Learning arbiter and combiner trees from partitioned data for scaling machine learning**. KDD'95
- H. Cheng, X. Yan, J. Han, and C.-W. Hsu, **Discriminative Frequent Pattern Analysis for Effective Classification**, ICDE'07
- H. Cheng, X. Yan, J. Han, and P. S. Yu, **Direct Discriminative Pattern Mining for Effective Classification**, ICDE'08
- W. Cohen. **Fast effective rule induction**. ICML'95
- G. Cong, K.-L. Tan, A. K. H. Tung, and X. Xu. **Mining top-k covering rule groups for gene expression data**. SIGMOD'05

# Low-Density Cut based Tree Decomposition for Large-Scale SVM Problems

L. He, H.-H. Shuai, X. Kong, P. S. Yu, Z. Hao, and X. Yang

ICDM, 2014

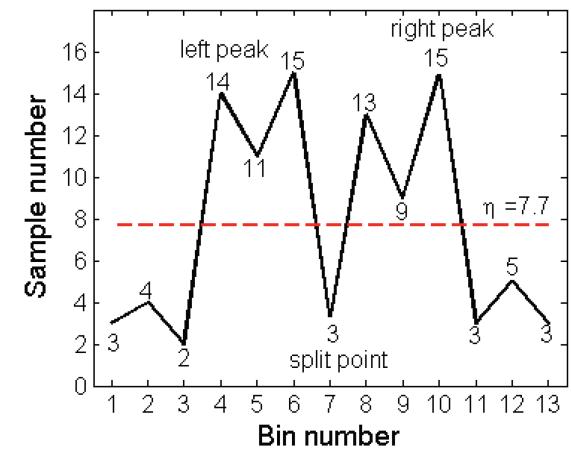
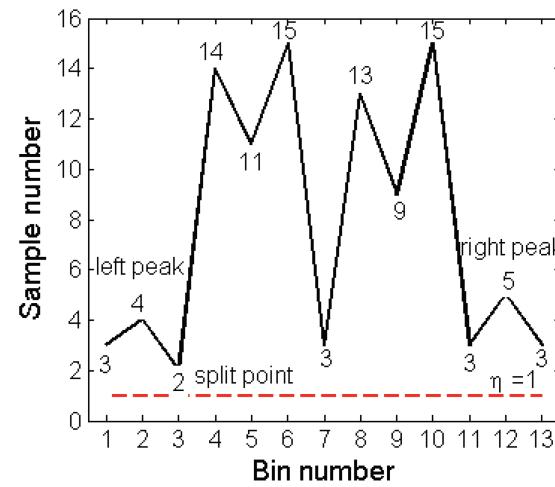
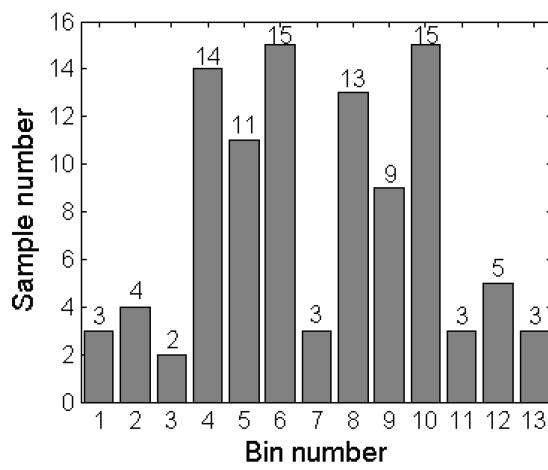


Fig. 1. An illustrative example of splitting point selection with different density threshold  $\eta$ . (a) One-dimensional histogram, (b)  $\eta$  is too small, and (c)  $\eta$  equals to the average density.

# References (2)

- A. J. Dobson. **An Introduction to Generalized Linear Models**. Chapman & Hall, 1990.
- G. Dong and J. Li. **Efficient mining of emerging patterns: Discovering trends and differences**. KDD'99.
- R. O. Duda, P. E. Hart, and D. G. Stork. **Pattern Classification**, 2ed. John Wiley, 2001
- U. M. Fayyad. **Branching on attribute values in decision tree generation**. AAAI'94.
- Y. Freund and R. E. Schapire. **A decision-theoretic generalization of on-line learning and an application to boosting**. J. Computer and System Sciences, 1997.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. **Rainforest: A framework for fast decision tree construction of large datasets**. VLDB'98.
- J. Gehrke, V. Gant, R. Ramakrishnan, and W.-Y. Loh, **BOAT -- Optimistic Decision Tree Construction**. SIGMOD'99.
- T. Hastie, R. Tibshirani, and J. Friedman. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. Springer-Verlag, 2001.
- D. Heckerman, D. Geiger, and D. M. Chickering. **Learning Bayesian networks: The combination of knowledge and statistical data**. Machine Learning, 1995.
- W. Li, J. Han, and J. Pei, **CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules**, ICDM'01.

# References (3)

- T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. **A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms.** Machine Learning, 2000.
- J. Magidson. **The Chaid approach to segmentation modeling: Chi-squared automatic interaction detection.** In R. P. Bagozzi, editor, Advanced Methods of Marketing Research, Blackwell Business, 1994.
- M. Mehta, R. Agrawal, and J. Rissanen. **SLIQ : A fast scalable classifier for data mining.** EDBT'96.
- T. M. Mitchell. **Machine Learning.** McGraw Hill, 1997.
- S. K. Murthy, **Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey,** Data Mining and Knowledge Discovery 2(4): 345-389, 1998
- J. R. Quinlan. **Induction of decision trees.** *Machine Learning*, 1:81-106, 1986.
- J. R. Quinlan and R. M. Cameron-Jones. **FOIL: A midterm report.** ECML'93.
- J. R. Quinlan. **C4.5: Programs for Machine Learning.** Morgan Kaufmann, 1993.
- J. R. Quinlan. **Bagging, boosting, and c4.5.** AAAI'96.

# References (4)

- R. Rastogi and K. Shim. **Public: A decision tree classifier that integrates building and pruning.** VLDB'98.
- J. Shafer, R. Agrawal, and M. Mehta. **SPRINT : A scalable parallel classifier for data mining.** VLDB'96.
- J. W. Shavlik and T. G. Dietterich. **Readings in Machine Learning.** Morgan Kaufmann, 1990.
- P. Tan, M. Steinbach, and V. Kumar. **Introduction to Data Mining.** Addison Wesley, 2005.
- S. M. Weiss and C. A. Kulikowski. **Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems.** Morgan Kaufman, 1991.
- S. M. Weiss and N. Indurkhya. **Predictive Data Mining.** Morgan Kaufmann, 1997.
- I. H. Witten and E. Frank. **Data Mining: Practical Machine Learning Tools and Techniques**, 2ed. Morgan Kaufmann, 2005.
- X. Yin and J. Han. **CPAR: Classification based on predictive association rules.** SDM'03
- H. Yu, J. Yang, and J. Han. **Classifying large data sets using SVM with hierarchical clusters.** KDD'03.