Chapter 6

Classification and Prediction

分類與預測
Outlines

1. What is classification? What is prediction?
2. Issues regarding classification and prediction
3. Classification by decision tree (決策樹) induction
4. Bayesian classification (貝氏分類)
5. Rule-based classification
6. Classification by back propagation (倒傳遞類神經網路)
7. Support Vector Machines (SVM) 支援向量機
8. Associative classification
9. Lazy learners (or learning from your neighbors)
10. Other classification methods
11. Prediction
12. Accuracy and error measures
13. Evaluating the Accuracy of a Classifier or Predictor
14. Ensemble methods – Increasing the Accuracy
15. Model selection
16. Summary
Outlines

1. **What is classification? What is prediction?**
2. Issues regarding classification and prediction
3. Classification by decision tree (決策樹) induction
4. Bayesian classification (貝氏分類)
5. Rule-based classification (規則式分類)
6. Classification by Backpropagation (倒傳遞類神經網路)
7. Support Vector Machines (SVM) 支援向量機
8. Associative classification
9. Lazy learners (or learning from your neighbors)
10. Other classification methods
11. Prediction
12. Accuracy and error measures
13. Evaluating the Accuracy of a Classifier or Predictor
14. Ensemble methods – Increasing the Accuracy
15. Model selection
16. Summary
6.1 What is classification? What is prediction?

- **Classification**
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set (訓練集) and the values (class labels) in a classifying attribute and uses it in classifying new data

- **Prediction**
  - models continuous-valued functions, i.e., predicts unknown or missing values

- **Typical applications**
  - Credit approval
  - Target marketing
  - Medical diagnosis
  - Fraud detection (詐欺偵測)
Classification: A Two-Step Process

- Model construction / Learning steps / classifier: describing a set of predetermined classes
  
  [第一步驟：建立一個模型，描述預定資料類集和概念集]
  
  - 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
Classification: A Two-Step Process (cont.)

- **Model usage:** for classifying future or unknown objects
  
  [第二步驟：使用模型，對將來或未知的對象進行分類]

  - 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 (測試集) samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur

  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known
Figure 6.1 The data classification process: (a) **Learning**: Training data are analyzed by a classification algorithm. Here, the class label attribute is **loan decision**, and the learned model or classifier is represented in the form of classification rules.
Figure 6.1 The data classification process: (b) *Classification*: Test data are used to estimate the accuracy of the classification rules. If the accuracy is considered acceptable, the rules can be applied to the classification of new data tuples.
Supervised vs. Unsupervised Learning

- **Supervised learning (classification)** 監督式學習
  - 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)** 非監督式學習
  - 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
- $X$: tuple
- $Y = f(X)$, mapping:
  - classification rules
  - decision trees (e.g., loan application)
  - mathematical formulae

- If we were to use the training set to measure the accuracy of the classifier, this estimate would likely be optimistic, because the classifier tends to overfit the data (i.e., during learning it may incorporate some particular anomalies of the training data that are not present in the general data set overall).
- Therefore, a test set is used, made up of test tuples and their associated class labels.
- These tuples are randomly selected from the general data set.
- They are independent of the training tuples, meaning that they are not used to construct the classifier.
Outlines

1. What is classification? What is prediction?
2. **Issues regarding classification (分類) and prediction (預測)**
3. Classification by decision tree (決策樹) induction
4. Bayesian classification (貝氏分類)
5. Rule-based classification
6. Classification by Backpropagation (倒傳遞類神經網路)
7. Support Vector Machines (SVM, 支援向量機)
8. Associative classification
9. Lazy learners (or learning from your neighbors)
10. Other classification methods
11. Prediction
12. Accuracy and error measures
13. Evaluating the Accuracy of a Classifier or Predictor
14. Ensemble methods – Increasing the Accuracy
15. Model selection
16. Summary
6.2 Issues regarding classification and prediction

6.2.1 Preparing the Data for Classification and Prediction

- 資料預先處理之目的: improve the accuracy, efficiency, and scalability of the classification or prediction process.

- Data cleaning
  - Preprocess data in order to reduce noise and handle missing values

- Relevance analysis
  (attribute subset selection, feature subset selection)
  - Remove the irrelevant or redundant attributes

- Data transformation and reduction
  - Generalize and/or normalize data
6.2.2 Comparing Classification and Prediction Methods

- **Accuracy (正確率)**
  - classifier accuracy: predicting class label
  - predictor accuracy: guessing value of predicted attributes

- **Speed (速度)**
  - computational costs: time to construct the model (training time)
  - computational costs: time to use the model (classification/prediction time)

- **Robustness (穩健性)**
  - handling noise and missing values

- **Scalability (可量度性)**
  - efficiency in large amounts of data

- **Interpretability (可解釋性)**
  - understanding and insight provided by the model

- **Other measures, e.g., goodness of rules, such as decision tree size or compactness (簡潔) of classification rules**
Outlines

1. What is classification? What is prediction?
2. Issues regarding classification and prediction
3. **Classification by decision tree (決策樹) induction**
4. Bayesian classification (貝氏分類)
5. Rule-based classification
6. Classification by Backpropagation (倒傳遞類神經網路)
7. Support Vector Machines (SVM) 支援向量機
8. Associative classification
9. Lazy learners (or learning from your neighbors)
10. Other classification methods
11. Prediction
12. Accuracy and error measures
13. Evaluating the Accuracy of a Classifier or Predictor
14. Ensemble methods – Increasing the Accuracy
15. Model selection
16. Summary
6.3 Classification by Decision Tree Induction

- **Decision tree induction** is the learning of decision trees from class-labeled training tuples.
- A decision tree is a flowchart-like tree structure
  - each internal node (nonleaf node) denotes a test on an attribute (非葉節點:對屬性的測試)
  - each branch represents an outcome of the test (分枝:測試的結果)
  - each leaf node (or terminal node) holds a class label. (葉節點:類別標籤)
  - topmost node in a tree is the root node (根節點).
辨識 圖6.2:

- Root node
- Nonleaf node
- Branch
- Leaf node

圖 6.2 A decision tree for the concept buys computer
6.3.1 Decision Tree Induction

- Quinlan, J.R. – 1979: ID3 (Iterative Dichotomiser) 反覆式二分法
- Quinlan – C4.5
- L. Breiman, J. Friedman, R. Olshen, and C. Stone, 1984 - Classification and Regression Trees (CART) 分類與迴歸樹
- These two cornerstone algorithms (ID3 and CART) spawned a flurry of work on decision tree induction.
Algorithm for Decision Tree Induction

- ID3, C4.5, and CART adopt a greedy (i.e., nonbacktracking) approach in which decision trees are constructed in
  - a top-down (由上而下)
  - recursive (遞迴式)
  - divide-and-conquer (分割與克服) manner.
- Starts with a training set of tuples and their associated class labels.
- The training set is recursively partitioned into smaller subsets as the tree is being built.
- 參考 figure 6.3
Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition $D$.

Input:
- Data partition, $D$, which is a set of training tuples and their associated class labels;
- $attribute_list$, the set of candidate attributes;
- $Attribute_selection_method$, a procedure to determine the splitting criterion that “best” partitions the data tuples into individual classes. This criterion consists of a $splitting_attribute$ and, possibly, either a $split point$ or $splitting subset$.

Output: A decision tree.

Method:

1. create a node $N$;
2. if tuples in $D$ are all of the same class, $C$ then
3. return $N$ as a leaf node labeled with the class $C$;
4. if $attribute_list$ is empty then
5. return $N$ as a leaf node labeled with the majority class in $D$; // majority voting
6. apply $Attribute_selection_method(D, attribute_list)$ to find the “best” $splitting_criterion$;
7. label node $N$ with $splitting_criterion$;
8. if $splitting_attribute$ is discrete-valued and
   multiway splits allowed then // not restricted to binary trees
   $attribute_list ← attribute_list − splitting_attribute$; // remove $splitting_attribute$
9. for each outcome $j$ of $splitting_criterion$
   // partition the tuples and grow subtrees for each partition
10. let $D_j$ be the set of data tuples in $D$ satisfying outcome $j$; // a partition
11. if $D_j$ is empty then
12. attach a leaf labeled with the majority class in $D$ to node $N$;
13. else attach the node returned by $Generate_decision_tree(D_j, attribute_list)$ to node $N$;
14. endfor
15. return $N$;

**Figure 6.3** Basic algorithm for inducing a decision tree from training tuples.
運算法則：產生決策樹。從資料分割訓練值組 \( D \) 中產生決策樹。

輸入:
- 資料分割 \( D \) 為包含類別標籤的訓練值組。
- \( \text{Attribute_list} \) 為候選屬性；
- \( \text{Attribute_selection_method} \) 決定分割條件並將資料值組分割成最好的個別類別，分割條件包含一個分割屬性與一個可能的分割點或分割子集合。

輸出：決策樹。

方法:
1. 產生節點 \( N \)；
2. 如果在 \( D \) 中所有的值組都有相同的類別 \( C \)，則
3. 傳回一個具有類別為 \( C \) 的葉節點 \( N \)；
4. 如果 \( \text{attribute_list} \) 為空則
5. 傳回一個在 \( D \) 中具有多數類別的葉節點 \( N \); // 多數決
6. 套用 \( \text{Attribute_selection_method}(D, \text{attribute_list}) \) 來找出最好的分割條件；
7. 將節點 \( N \) 標示為分割條件；
8. 如果分割屬性為離散值並且允許多元分割則 // 不限於二元分割
9. \( \text{attribute_list} \leftarrow \text{attribute_list} \)-分割屬性； // 移除分割屬性
10. 對於每個分割條件的結果 \( j \) // 分割值組並對每個分割建立子樹
11. 假設 \( D_j \) 為 \( D \) 中滿足結果 \( j \) 的值組集； // 一個分割
12. 如果 \( D_j \) 為空白，則
13. 將一個在 \( D \) 中具有多數別的節點設為節點 \( N \) 葉節點；
14. 否則建 \( \text{Generate_decision_tree}\left(D_j, \text{attribute_list}\right) \) 傳回的節點附加至節點 \( N \)
15. 傳回 \( N \)；
Decision Tree 採用三個參數

- **$D$** 資料集:
  - a data partition.
  - Initially, it is the complete set of training tuples and their associated class labels.

- **Attribute_list** 屬性列:
  - The parameter attribute list is a list of attributes describing the tuples.

- **Attribute_selection_method** 屬性選擇方法:
  - Specifies a heuristic procedure for selecting the attribute that “best” discriminates the given tuples according
  - This procedure employs an attribute selection measure, such as *information gain* or *gini index*. 
Decision Tree Algorithm (1/3)

- [step 1] –
  The tree starts as a single node, \( N \), representing the training tuples in \( D \).

- [step 2 and 3] –
  If the tuples in \( D \) are all of the same class, then node \( N \) becomes a leaf and is labeled with that class.

- [step 4 and 5] –
  Terminating conditions (結束條件). All of the terminating conditions are explained at the end of the algorithm.
[step 6] – Otherwise, the algorithm calls the Attribute selection method to determine the splitting criterion.

- The splitting criterion (分割條件) tells us which attribute to test at node $N$ by determining the “best” way to separate or partition the tuples in $D$ into individual classes.
- The splitting criterion also tells us which branches to grow from node $N$ with respect to the outcomes of the chosen test.
- The splitting criterion indicates the splitting attribute (分割屬性) and may also indicate either a split-point or a splitting subset.
- A partition is $\textit{pure}$ if all of the tuples in it belong to the same class.
Decision Tree (3/3)

- [step 7] –
  The node $N$ is labeled with the splitting criterion, which serves as a test at the node.

- [step 10, 11] –
  A branch is grown from node $N$ for each of the outcomes of the splitting criterion. The tuples in $D$ are partitioned accordingly.
Figure 6.4 Three possibilities for partitioning tuples based on the splitting criterion, shown with examples. Let $A$ be the splitting attribute. (a) If $A$ is discrete-valued, then one branch is grown for each known value of $A$. (b) If $A$ is continuous-valued, then two branches are grown, corresponding to $A \leq \text{split}\_point$ and $A > \text{split}\_point$. (c) If $A$ is discrete-valued and a binary tree must be produced, then the test is of the form $A \in S_A$, where $S_A$ is the splitting subset for $A$. 
There are three possible scenarios, as illustrated in Figure 6.4. Let $A$ be the splitting attribute. $A$ has $v$ distinct values, \{\(a_1\), \(a_2\), \ldots, \(a_v\)\}, based on the training data.

1. **Discrete-valued**: Therefore, it is removed from attribute list (steps 8 to 9).
2. **Continuous-valued**: Two branches are grown from $N$ and labeled according to the above outcomes.
3. **Discrete-valued and a binary tree**:

\[ A \in S_A \]
Terminating conditions

1. All of the tuples in partition \( D \) (represented at node \( N \)) belong to the same class (steps 2 and 3), or

2. There are no remaining attributes on which the tuples may be further partitioned (step 4).
   - In this case, majority voting is employed (step 5).
   - This involves converting node \( N \) into a leaf and labeling it with the most common class in \( D \).
   - Alternatively, the class distribution of the node tuples may be stored.

3. There are no tuples for a given branch, that is, a partition \( D_j \) is empty (step 12). In this case, a leaf is created with the majority class in \( D \) (step 13).
R- tree package

# decision tree
Library(tree)
iris

ir.tree <- tree(Species ~., iris)

summary(ir.tree)

plot(ir.tree); text(ir.tree)
R – tree . Output

- Petal.Length < 2.45
  - Sepal.Length < 5.15
    - versicolor
  - Petal.Length < 4.95
  - Petal.Width < 1.75
    - versicolor
  - virginica
  - Petal.Length < 4.95
    - virginica
  - virginica
6.3.2 Attribute Selection Measures

- An attribute selection measure (屬性選擇法) is a heuristic (啟發式) for selecting the splitting criterion that “best” separates a given data partition, D, of class-labeled training tuples into individual classes.

- If we were to split D into smaller partitions according to the outcomes of the splitting criterion, ideally each partition would be pure.
  - All of the tuples that fall into a given partition would belong to the same class.

- Conceptually, the “best” splitting criterion is the one that most closely results in such a scenario. Attribute selection
This section describes three popular attribute selection measures

1. Information gain 資訊獲得 （例: ID3）
2. Gain ratio 獲利比率 （例: C4.5）
3. Gini index 吉尼指標 （例: CART）
Let $D$, the data partition, be a training set of class-labeled tuples.

Suppose the class label attribute has $m$ distinct values defining $m$ distinct classes, $C_i$ (for $i = 1, \ldots, m$).

Let $C_{i,D}$ be the set of tuples of class $C_i$ in $D$.

Let $|D|$ and $|C_{i,D}|$ denote the number of tuples in $D$ and $C_{i,D}$, respectively.

### Table 6.1

Class-labeled training tuples from the *AllElectronics* customer database.

<table>
<thead>
<tr>
<th>RID</th>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>Class: buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>youth</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no ▲</td>
</tr>
<tr>
<td>2</td>
<td>youth</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no ▲</td>
</tr>
<tr>
<td>3</td>
<td>middle aged</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>senior</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>senior</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>senior</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no ▲</td>
</tr>
<tr>
<td>7</td>
<td>middle aged</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>8</td>
<td>youth</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no ▲</td>
</tr>
<tr>
<td>9</td>
<td>youth</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>10</td>
<td>senior</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>11</td>
<td>youth</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>12</td>
<td>middle aged</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>13</td>
<td>middle aged</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>14</td>
<td>senior</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no ▲</td>
</tr>
</tbody>
</table>

$m=2$

$i=1$, $C_1$=yes, $|C_{1,D}|=9$

$i=2$, $C_2$=no, $|C_{2,D}|=5$
Attributes Selection Measures –
[方法1] Information gain (ID3)

- Select the attribute with the highest information gain.
- This attribute minimizes the information needed to classify the tuples in the resulting partitions and reflects the least randomness or “impurity” in these partitions.
- Such an approach minimizes the expected number of tests needed to classify a given tuple and guarantees that a simple (but not necessarily the simplest) tree is found.
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*, 熵) needed to classify a tuple in $D$:
  \[
  Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)
  \]  
  \[\text{(6.1)}\]

- **Information** needed (after using A to split $D$ into $v$ partitions) to classify $D$:
  \[
  Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)
  \]  
  \[\text{(6.2)}\]

- **Information gained** by branching on attribute $A$
  \[
  Gain(A) = Info(D) - Info_A(D)
  \]  
  \[\text{(6.3)}\]
Example 6.1
Induction of a decision tree using information gain

Table 6.1  Class-labeled training tuples from the AllElectronics customer database.

<table>
<thead>
<tr>
<th>RID</th>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>Class: buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>youth</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>2</td>
<td>youth</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>3</td>
<td>middle_aged</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>senior</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>senior</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>senior</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>7</td>
<td>middle_aged</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>8</td>
<td>youth</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>9</td>
<td>youth</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>10</td>
<td>senior</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>11</td>
<td>youth</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>12</td>
<td>middle_aged</td>
<td>medium</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>13</td>
<td>middle_aged</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>14</td>
<td>senior</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

Because age has the highest information gain among the attributes, it is selected as the splitting attribute.

結果參考圖 6.5

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

\[ Gain(age) = Info(D) - Info_{age}(D) = 0.246 \]

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

\[ Gain(income) = 0.029 \]

\[ Gain(student) = 0.151 \]

\[ Gain(credit\_rating) = 0.048 \]
Figure 6.5 The attribute *age* has the highest information gain and therefore becomes the splitting attribute at the root node of the decision tree. Branches are grown for each outcome of *age*. The tuples are shown partitioned accordingly.
Computing Information-Gain for Continuous-Value 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:
  - \(D_1\) is the set of tuples in D satisfying \(A \leq\) split-point, and \(D_2\) is the set of tuples in D satisfying \(A >\) split-point
Attributes Selection Measures –
[方法2] Gain ratio (C4.5)

- The information gain measure is biased toward tests with many outcomes.
  - It prefers to select attributes having a large number of values.
- For example, consider an attribute that acts as a unique identifier, such as product ID.
  - A split on product ID would result in a large number of partitions (as many as there are values), each one containing just one tuple.
  - Because each partition is pure, the information required to classify data set \( D \) based on this partitioning would be \( \text{Info}_{product\_ID}(D) = 0 \).
  - Therefore, the information gained by partitioning on this attribute is maximal.
  - Clearly, such a partitioning is useless for classification.
Gain Ratio

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

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

(6.5)

- It differs from information gain, which measures the information with respect to classification that is acquired based on the same partitioning.

- information gain: 以類別相同為基礎, gain ratio: 以屬性相同分割為基礎

\[
GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}
\]  

(6.6)

- The attribute with the maximum gain ratio is selected as the splitting attribute
Example 6.2
Computation of gain ratio for the attribute income

- A test on income splits the data of Table 6.1 into three partitions, namely low, medium, and high, containing four, six, and four tuples, respectively.
- To compute the gain ratio of income, we first use Equation (6.5) to obtain

Table 6.1  Class-labeled training tuples from the AllElectronics customer database.

<table>
<thead>
<tr>
<th>RID</th>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>Class: buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>youth</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no ▲</td>
</tr>
<tr>
<td>2</td>
<td>youth</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no ▲</td>
</tr>
<tr>
<td>3</td>
<td>middle_aged</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>□ yes</td>
</tr>
<tr>
<td>4</td>
<td>senior</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>□ yes</td>
</tr>
<tr>
<td>5</td>
<td>senior</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>□ yes</td>
</tr>
<tr>
<td>6</td>
<td>senior</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no ▲</td>
</tr>
<tr>
<td>7</td>
<td>middle_aged</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>□ yes</td>
</tr>
<tr>
<td>8</td>
<td>youth</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no ▲</td>
</tr>
<tr>
<td>9</td>
<td>youth</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>□ yes</td>
</tr>
<tr>
<td>10</td>
<td>senior</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>□ yes</td>
</tr>
<tr>
<td>11</td>
<td>youth</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>□ yes</td>
</tr>
<tr>
<td>12</td>
<td>middle_aged</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>□ yes</td>
</tr>
<tr>
<td>13</td>
<td>middle_aged</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>□ yes</td>
</tr>
<tr>
<td>14</td>
<td>senior</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no ▲</td>
</tr>
</tbody>
</table>

$\text{Gain}(\text{income}) = 0.029$

$\text{GainRatio}(\text{income}) = \frac{0.029}{1.557} = 0.019$
Attributes Selection Measures –

[方法3] Gini index (CART, IBM IntelligentMiner)

- If a data set $D$ contains examples from $n$ classes, gini index, $gini(D)$ is defined as
  
  $$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$  \hspace{1cm} (6.7)

  where $p_j$ is the relative frequency of class $i$ in $D$

- If a data set $D$ is split on $A$ into two subsets $D_1$ and $D_2$, the gini index $gini(D)$ is defined as

  $$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$  \hspace{1cm} (6.8)

- Reducing in Impurity:

  $$\Delta gini(A) = gini(D) - gini_A(D)$$  \hspace{1cm} (6.9)

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (need to enumerate all the possible splitting points for each attribute)
Example 6.3
Induction of a decision tree using gini index

- Ex. 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 D₁: {low, medium} and 4 in D₂: {high}

\[
\begin{align*}
Gini_{\text{income } \in \{\text{low, medium}\}}(D) \\
= \frac{10}{14} Gini(D₁) + \frac{4}{14} Gini(D₂) \\
= \frac{10}{14} \left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14} \left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right) \\
= 0.450 \cdot 0.443 \\
= Gini_{\text{income } \in \{\text{high}\}}(D).
\end{align*}
\]
Example 6.3
Induction of a decision tree using gini index (cont.)

- Similarly, the Gini index values for splits on the remaining subsets are:
  - \( \{\text{low, high}\} \) or \( \{\text{medium}\} \) : 0.458
  - \( \{\text{medium, high}\} \) or \( \{\text{low}\} \) : 0.456
  - Therefore, the best binary split for attribute \textit{income} is on \( \{\text{low, medium}\} \) or \( \{\text{high}\} \) : 0.443
  - \textit{student} : Gini index of 0.367; \textit{credit_rating} : Gini index of 0.429

- The attribute \textit{age} and splitting subset \( \{\text{youth, senior}\} \) therefore give the minimum gini index overall, with a reduction in impurity of 0.459-0.357 = 0.102. The binary split \textit{age} \( \text{IN} \) \( \{\text{youth, senior}\} \) results in the maximum reduction in impurity of the tuples in \( D \) and is returned as the splitting criterion.

- The Gini index has selected \textit{income} instead of \textit{age}(範例 6.1) at the root node.
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
    - 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-statistics: 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 多變量分割 (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
6.3.3 Tree Pruning

- **Overfitting**: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples

- 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 in Large Databases

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why decision tree induction in data mining?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods
6.3.4 Scalability and Decision Tree Induction

- **SLIQ** (EDBT’96 — Mehta et al.)
  - Builds an index for each attribute and only class list and the current attribute list reside in memory

- **SPRINT** (VLDB’96 — J. Shafer et al.)
  - Constructs an attribute list data structure

- **PUBLIC** (VLDB’98 — Rastogi & Shim)
  - Integrates tree splitting and tree pruning: stop growing the tree earlier

- **RainForest** (VLDB’98 — Gehrke, Ramakrishnan & Ganti)
  - Builds an AVC-list (attribute, value, class label)

- **BOAT** (PODS’99 — Gehrke, Ganti, Ramakrishnan & Loh)
  - Uses bootstrapping to create several small samples
Scalability Framework for RainForest

- Separates the scalability aspects from the criteria that determine the quality of the tree
- Builds an AVC-list: AVC (Attribute, Value, Class_label)
- AVC-set (of an attribute $X$)
  - Projection of training dataset onto the attribute $X$ and class label where counts of individual class label are aggregated
- AVC-group (of a node $n$)
  - Set of AVC-sets of all predictor attributes at the node $n$
Rainforest: Training Set and Its AVC Sets

### Training Examples

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31…40</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>31…40</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

### AVC-set on Age

<table>
<thead>
<tr>
<th>Age</th>
<th>Buy_Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>yes</td>
</tr>
<tr>
<td>31..40</td>
<td>no</td>
</tr>
</tbody>
</table>

### AVC-set on income

<table>
<thead>
<tr>
<th>income</th>
<th>Buy_Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>high</td>
<td>2</td>
</tr>
<tr>
<td>medium</td>
<td>2</td>
</tr>
<tr>
<td>low</td>
<td>1</td>
</tr>
</tbody>
</table>

### AVC-set on Student

<table>
<thead>
<tr>
<th>student</th>
<th>Buy_Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>6</td>
</tr>
<tr>
<td>no</td>
<td>3</td>
</tr>
</tbody>
</table>

### AVC-set on credit_rating

<table>
<thead>
<tr>
<th>Credit rating</th>
<th>Buy_Computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>6</td>
</tr>
<tr>
<td>fair</td>
<td>2</td>
</tr>
<tr>
<td>excellent</td>
<td>3</td>
</tr>
</tbody>
</table>
Data Cube-Based Decision-Tree Induction

- Integration of generalization with decision-tree induction (Kamber et al.’97)
- Classification at primitive concept levels
  - E.g., precise temperature, humidity, outlook, etc.
  - Low-level concepts, scattered classes, bushy classification-trees
  - Semantic interpretation problems
- Cube-based multi-level classification
  - Relevance analysis at multi-levels
  - Information-gain analysis with dimension + level
**BOAT**  
(Bootstrapped Optimistic Algorithm for Tree Construction)

- Use a statistical technique called *bootstrapping* to create several smaller samples (subsets), each fits in memory
- Each subset is used to create a tree, resulting in several trees
- These trees are examined and used to construct a new tree $T'$
  - It turns out that $T'$ is very close to the tree that would be generated using the whole data set together
- Adv: requires only two scans of DB, an incremental alg.
Outlines

1. What is classification? What is prediction?
2. Issues regarding classification and prediction
3. Classification by decision tree (決策樹) induction
4. **Bayesian classification (貝氏分類)**
5. Rule-based classification
6. Classification by Backpropagation (倒傳遞類神經網路)
7. Support Vector Machines (SVM) 支援向量機
8. Associative classification
9. Lazy learners (or learning from your neighbors)
10. Other classification methods
11. Prediction
12. Accuracy and error measures
13. Evaluating the Accuracy of a Classifier or Predictor
14. Ensemble methods – Increasing the Accuracy
15. Model selection
16. Summary
6.4 Bayesian Classification

- A statistical classifier: (統計分類法) performs **probabilistic prediction**, i.e., predicts class membership probabilities.
- **Foundation:**
  Based on Bayes’ Theorem (貝氏定理).
- **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 [棘手的], they can provide a standard of optimal decision making against which other methods can be measured.
6.4.1 Bayesian Theorem

- Let $X$ be a data sample (data tuple) ("evidence"): class label is unknown.
- Let $H$ be a hypothesis that $X$ belongs to class C.
- Classification is to determine $P(H|X)$, the probability that the hypothesis holds given the observed data sample $X$.
- $P(H|X)$ is a posterior probability.
  
  e.g., Suppose our world of data tuples is confined to customers described by the attributes age and income, respectively, and that $X$ is a 35-year-old customer with an income of $40,000. Suppose that $H$ is the hypothesis that our customer will buy a computer. Then $P(H|X)$ reflects the probability that customer $X$ will buy a computer given that we know the customer’s age and income.
6.4.1 Bayesian Theorem

- \( P(H) \): *prior probability* 事前機率, the initial probability
  - e.g., \( X \) will buy computer, regardless of age, income, …
- \( P(X) \): probability that sample data is observed
- The posterior probability, \( P(H|X) \), is based on more information (e.g., customer information) than the prior probability, \( P(H) \), which is independent of \( X \).
- \( P(X|H) \): *posteriori probability* 事後機率, the probability of observing the sample \( X \), given that the hypothesis holds
  - E.g., Given that \( X \) will buy computer, the probability that \( X \) is 31..40, medium income.
Bayesian Theorem (cont.)

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

\[
P(H | X) = \frac{P(X | H)P(H)}{P(X)}
\]  

(6.10)

- Informally, this can be written as

\[
\text{posteriori} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}
\]

- Predicts $X$ belongs to $C_i$ iff the probability $P(C_i|X)$ is the highest among all the $P(C_k|X)$ for all the $k$ classes

- Practical difficulty: require initial knowledge of many probabilities, significant computational cost
6.4.2 Naïve Bayesian Classifier

1. Let $D$ be a training set of tuples and their associated class labels, and each tuple is represented by an $n$-D attribute vector $X = (x_1, x_2, \ldots, x_n)$

2. Suppose there are $m$ classes $C_1, C_2, \ldots, C_m$. Classification is to derive the maximum posteriori (最大事後機率), conditioned on $X$. i.e., the maximal $P(C_i|X)$

$$P(C_i | X) > P(C_j | X) \text{ for } 1 \leq j \leq m, j \neq i.$$ 

This can be derived from Bayes’ theorem

$$P(C_i | X) = \frac{P(X|C_i)P(C_i)}{P(X)} \quad (6.11)$$

3. Since $P(X)$ is constant for all classes, only $P(X|C_i)P(C_i)$ needs to be maximized
If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, \( P(C_1) = P(C_2) = \ldots = P(C_m) \), and we would therefore maximize \( P(X|C_i) \).

Otherwise, we maximize \( P(X|C_i)P(C_i) \).

Note that the class prior probabilities may be estimated by \( P(C_i) = |C_{i,D}| / |D| \), where \( |C_{i,D}| \) is the number of training tuples of class \( C_i \) in \( D \).
Given data sets with many attributes, it would be extremely computationally expensive to compute $P(X|C_i)$.

- In order to reduce computation in evaluating $P(X|C_i)$, the naive assumption of class conditional independence is made.
- This presumes that the values of the attributes are conditionally independent of one another, given the class label of the tuple (i.e., that there are no dependence relationships among the attributes). Thus,

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

\[
= P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i).
\]
Derivation of Naïve Bayes Classifier

• Recall that here \( x_k \) refers to the value of attribute \( A_k \) for tuple \( X \).

• For each attribute, we look at whether the attribute is categorical or continuous-valued. For instance, to compute \( P(X|C_i) \), we consider the following:

(a) If \( A_k \) is categorical, then \( P(x_k|C_i) \) is the number of tuples of class \( C_i \) in \( D \) having the value \( x_k \) for \( A_k \), divided by \( |C_{i,D}| \), the number of tuples of class \( C_i \) in \( D \).

(b) If \( A_k \) is continuous-valued, then we need to do a bit more work, but the calculation is pretty straightforward. A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean \( \mu \) and standard deviation \( \sigma \), defined by

\[
g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad (6.13)
\]

so that

\[
P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}). \quad (6.14)
\]
For example, let $X = (35, 40,000)$, where $A_1$ and $A_2$ are the attributes \textit{age} and \textit{income}, respectively. Let the class label attribute be \textit{buys computer}. The associated class label for $X$ is \textit{yes} (i.e., $\textit{buys computer} = \textit{yes}$).

Let's suppose that \textit{age} has not been discretized and therefore exists as a continuous-valued attribute. Suppose that from the training set, we find that customers in $D$ who buy a computer are 38±12 years of age. In other words, for attribute \textit{age} and this class, we have $\mu = 38$ years and $\sigma = 12$.

We can plug these quantities, along with $x_1 = 35$ for our tuple $X$ into Equation (6.13) in order to estimate $P(\text{age} = 35 | \text{buys computer} = \text{yes})$.

\begin{verbatim}
R 程式碼: > dnorm(35, mean=38, sd=12) [1] 0.03222234 > dnorm(35, 38, 12) [1] 0.03222234
\end{verbatim}
5. In order to predict the class label of \( X \), \( (X|C_i)P(C_i) \) is evaluated for each class \( C_i \).

- The classifier predicts that the class label of tuple \( X \) is the class \( C_i \) if and only if:

\[
P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \quad \text{for} \ 1 \leq j \leq m, \ j \neq i. \tag{6.15}
\]

- In other words, the predicted class label is the class \( C_i \) for which \( P(X|C_i)P(C_i) \) is the maximum [找最大值].
Naïve Bayesian Classifier: Training Dataset

Class:

$C_1$: buys_computer = ‘yes’

$C_2$: buys_computer = ‘no’

Data sample:

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

<table>
<thead>
<tr>
<th>RID</th>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>Class: buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>youth</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no ▲</td>
</tr>
<tr>
<td>2</td>
<td>youth</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no ▲</td>
</tr>
<tr>
<td>3</td>
<td>middle_aged</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>senior</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>senior</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>senior</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no ▲</td>
</tr>
<tr>
<td>7</td>
<td>middle_aged</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>8</td>
<td>youth</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no ▲</td>
</tr>
<tr>
<td>9</td>
<td>youth</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>10</td>
<td>senior</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>11</td>
<td>youth</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>12</td>
<td>middle_aged</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>13</td>
<td>middle_aged</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>14</td>
<td>senior</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no ▲</td>
</tr>
</tbody>
</table>
Example 6.4 Naïve Bayesian Classifier

We need to maximize $P(X|C_i)P(C_i)$, for $i = 1, 2$. $P(C_i)$, the prior probability of each class, can be computed based on the training tuples:

$P(buys\text{-}computer = yes) = 9/14 = 0.643$
$P(buys\text{-}computer = no) = 5/14 = 0.357$

To compute $P(X|C_i)$, for $i = 1, 2$, we compute the following conditional probabilities:

$P(\text{age} = \text{youth} | \text{buys\text{-}computer} = \text{yes}) = 2/9 = 0.222$
$P(\text{age} = \text{youth} | \text{buys\text{-}computer} = \text{no}) = 3/5 = 0.600$
$P(\text{income} = \text{medium} | \text{buys\text{-}computer} = \text{yes}) = 4/9 = 0.444$
$P(\text{income} = \text{medium} | \text{buys\text{-}computer} = \text{no}) = 2/5 = 0.400$
$P(\text{student} = \text{yes} | \text{buys\text{-}computer} = \text{yes}) = 6/9 = 0.667$
$P(\text{student} = \text{yes} | \text{buys\text{-}computer} = \text{no}) = 1/5 = 0.200$
$P(\text{credit\_rating} = \text{fair} | \text{buys\text{-}computer} = \text{yes}) = 6/9 = 0.667$
$P(\text{credit\_rating} = \text{fair} | \text{buys\text{-}computer} = \text{no}) = 2/5 = 0.400$
Example 6.4 Naïve Bayesian Classifier (cont.)

Using the above probabilities, we obtain

\[
P(X|\text{buys\_computer} = \text{yes}) = P(\text{age} = \text{youth} | \text{buys\_computer} = \text{yes}) \times \\
P(\text{income} = \text{medium} | \text{buys\_computer} = \text{yes}) \times \\
P(\text{student} = \text{yes} | \text{buys\_computer} = \text{yes}) \times \\
P(\text{credit\_rating} = \text{fair} | \text{buys\_computer} = \text{yes}) \\
= 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044.
\]

Similarly,

\[
P(X|\text{buys\_computer} = \text{no}) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019.
\]

To find the class, \( C_i \), that maximizes \( P(X|C_i)P(C_i) \), we compute

\[
P(X|\text{buys\_computer} = \text{yes})P(\text{buys\_computer} = \text{yes}) = 0.044 \times 0.643 = 0.028 \\
P(X|\text{buys\_computer} = \text{no})P(\text{buys\_computer} = \text{no}) = 0.019 \times 0.357 = 0.007
\]

Therefore, the naïve Bayesian classifier predicts \( \text{buys\_computer} = \text{yes} \) for tuple \( X \).

選取較大者, 0.028[yes] > 0.007[no],所以選取 yes
Avoiding the zero - Probability Problem

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

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

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10),

- Use Laplacian correction 拉普拉斯修正 (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 Bayesian 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 Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks
6.4.3 Bayesian Belief Networks

- The naïve Bayesian classifier makes the assumption of class conditional independence, that is, given the class label of a tuple, the values of the attributes are assumed to be conditionally independent of one another.
- Bayesian belief network allows a subset of the variables conditionally independent
- A graphical model of causal relationships
  - Represents dependency among the variables
  - Gives a specification of joint probability distribution

Nodes: random variables
Links: dependency
- X and Y are the parents of Z, and Y is the parent (immediate predecessor) of P
- No dependency between Z and P
- Has no loops or cycles
Bayesian Belief Network: An Example

- LungCancer 受到 FamilyHistory 與 Smoker之影響。
- *PositiveXRay* is independent of whether the patient has a family history of lung cancer or is a smoker, given that we know the patient has lung cancer.
- In other words, once we know the outcome of the variable LungCancer, then the variables FamilyHistory and Smoker do not provide any additional information regarding PositiveXRay.
- The arcs also show that the variable LungCancer is conditionally independent of Emphysema, given its parents, FamilyHistory and Smoker.
Bayesian Belief Network: An Example (cont.)

The **conditional probability table (CPT)** for variable LungCancer:

\[
\begin{align*}
\text{P(LungCancer} &= \text{yes} \mid \text{FamilyHistory} = \text{yes}, \text{Smoker} = \text{yes}) = 0.8 \\
\text{P(LungCancer} &= \text{no} \mid \text{FamilyHistory} = \text{no}, \text{Smoker} = \text{no}) = 0.9 \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>( \text{FH, S} )</th>
<th>( \text{FH, } \text{S} )</th>
<th>( \text{FH, S} )</th>
<th>( \text{FH, S} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{LC} )</td>
<td>0.8</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>( \sim \text{LC} )</td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
</tr>
</tbody>
</table>

CPT shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of X, from CPT:
Let $X = (x_1, \ldots, x_n)$ be a data tuple described by the variables or attributes $Y_1, \ldots, Y_n$, respectively. Recall that each variable is conditionally independent of its nondescendants in the network graph, given its parents. This allows the network to provide a complete representation of the existing joint probability distribution with the following equation:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|\text{Parents}(Y_i)),$$

(6.16)

where $P(x_1, \ldots, x_n)$ is the probability of a particular combination of values of $X$, and the values for $P(x_i|\text{Parents}(Y_i))$ correspond to the entries in the CPT for $Y_i$. 


6.4.4 Training Bayesian Networks

- Several scenarios:
  - Given both the network structure and all variables observable: *learn only the CPTs*
  - Network structure known, some hidden variables: *gradient descent* (greedy hill-climbing) method, analogous to neural network learning
  - Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
  - Unknown structure, all hidden variables: No good algorithms known for this purpose

- Ref. D. Heckerman: Bayesian networks for data mining
Outlines

1. What is classification? What is prediction?
2. Issues regarding classification and prediction
3. Classification by decision tree (決策樹) induction
4. Bayesian classification (貝氏分類)
5. **Rule-based classification** (規則式分類)
6. Classification by Backpropagation (倒傳遞類神經網路)
7. Support Vector Machines (SVM) 支援向量機
8. Associative classification
9. Lazy learners (or learning from your neighbors)
10. Other classification methods
11. Prediction
12. Accuracy and error measures
13. Evaluating the Accuracy of a Classifier or Predictor
14. Ensemble methods – Increasing the Accuracy
15. Model selection
16. Summary
6.5.1 Using IF-THEN Rules for Classification

- Represent the knowledge in the form of IF-THEN rules
  
  $$R: \text{IF } \text{age} = \text{youth} \ \text{AND} \ \text{student} = \text{yes} \ \text{THEN} \ \text{buys\_computer} = \text{yes}$$

  - Rule antecedent/precondition vs. rule consequent

- Assessment of a rule: coverage and accuracy
  
  - $$n_{\text{covers}} = \# \text{ of tuples covered by } R$$
  - $$n_{\text{correct}} = \# \text{ of tuples correctly classified by } R$$
  
  $$\text{coverage}(R) = \frac{n_{\text{covers}}}{|D|} \quad /* \ D: \text{training data set} */$$
  
  $$\text{accuracy}(R) = \frac{n_{\text{correct}}}{n_{\text{covers}}}$$

- If more than one rule is 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 test)
  
  - 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
6.5.2 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
- Example: Rule extraction from our *buys_computer* decision-tree

IF $age = \text{young}$ AND $student = \text{no}$ THEN $buys\_computer = \text{no}$

IF $age = \text{young}$ AND $student = \text{yes}$ THEN $buys\_computer = \text{yes}$

IF $age = \text{mid-age}$ THEN $buys\_computer = \text{yes}$

IF $age = \text{old}$ AND $credit\_rating = \text{excellent}$ THEN $buys\_computer = \text{yes}$

IF $age = \text{young}$ AND $credit\_rating = \text{fair}$ THEN $buys\_computer = \text{no}$
Rule Extraction from the Training Data

- 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
  - Each time a rule is learned, the tuples covered by the rules are removed
  - The process repeats on the remaining tuples unless *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: learning a set of rules *simultaneously*
How to Learn-One-Rule?

- Star 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
  - Foil-gain (in FOIL & RIPPER): assesses info_gain by extending condition
    \[
    FOIL\_Gain = pos \times (\log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg})
    \]
    It favors rules that have high accuracy and cover many positive tuples
- Rule pruning based on an independent set of test tuples
  \[
  FOIL\_Prune(R) = \frac{pos - neg}{pos + neg}
  \]
  Pos/neg are # of positive/negative tuples covered by R.
  If \( FOIL\_Prune \) is higher for the pruned version of R, prune R
Outlines

1. What is classification? What is prediction?
2. Issues regarding classification and prediction
3. Classification by decision tree (決策樹) induction
4. Bayesian classification (貝氏分類)
5. Rule-based classification (規則式分類)
6. **Classification by Backpropagation** (倒傳遞類神經網路)
7. Support Vector Machines (SVM) 支援向量機
8. Associative classification
9. Lazy learners (or learning from your neighbors)
10. Other classification methods
11. Prediction
12. Accuracy and error measures
13. Evaluating the Accuracy of a Classifier or Predictor
14. Ensemble methods – Increasing the Accuracy
15. Model selection
16. Summary
6.6 Classification by Backpropagation

- Backpropagation is a **neural network** learning algorithm.
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons.
- A neural network: A set of connected input/output units where each connection has a **weight** associated with it.
- During the learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of the input tuples.
- Also referred to as **connectionist learning** due to the connections between units.
Neural Network as a Classifier

- **Weakness**
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g., the network topology or "structure."
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of "hidden units" in the network

- **Strength**
  - High tolerance to noisy data
  - Ability to classify untrained patterns
  - Well-suited for continuous-valued inputs and outputs
  - Successful on a wide array of real-world data
  - Algorithms are inherently parallel
  - Techniques have recently been developed for the extraction of rules from trained neural networks
6.6.1 A Multi-Layer Feed-Forward Neural Network

- The *backpropagation algorithm* performs learning on a multilayer feed-forward neural network.
- It iteratively learns a set of *weights* for prediction of the class label of tuples.
- A *multilayer feed-forward neural network* consists of an input layer, one or more hidden layers, and an output layer.
Figure 6.15 multilayer feed-forward neural network

輸入層: 4個屬性
隱藏層
輸出層: 2個輸出

Weight 權重

The number of hidden layers is arbitrary.

Weight 權重
The units in the input layer are called *input units*. The units in the hidden layers and output layer are sometimes referred to as *neurodes*, due to their symbolic biological basis, or as output units.

The multilayer neural network shown in Figure 6.15 has two layers of output units. Therefore, we say that it is a *two-layer neural network*.

Similarly, a network containing two hidden layers is called a *three-layer neural network*, and so on.

The network is *feed-forward* in that none of the weights cycles back to an input unit or to an output unit of a previous layer. It is fully connected in that each unit provides input to each unit in the next forward layer.
• A hidden or output layer unit $j$: The inputs to unit $j$ are outputs from the previous layer.
• These are multiplied by their corresponding weights in order to form a weighted sum, which is added to the bias associated with unit $j$.
• A **nonlinear activation function** 非線性激發函數 is applied to the net input.
• For ease of explanation, the inputs to unit $j$ are labeled $y_1, y_2, \ldots, y_n$. If unit $j$ were in the first hidden layer, then these inputs would correspond to the input tuple $(x_1, x_2, \ldots, x_n)$. 

**Figure 6.17**
6.6.2 Defining a Network Topology

- Before training can begin, the user must decide on the network topology by specifying the number of units in the input layer, the number of hidden layers (if more than one), the number of units in each hidden layer, and the number of units in the output layer.

- **Normalizing** (標準化-第2章) the input values for each attribute measured in the training tuples will help speed up the learning phase. Typically, input values are normalized so as to fall between 0 and 1.

- Discrete-valued attributes may be encoded such that there is one input unit per domain value. For example, if an attribute $A$ has three possible or known values, namely $\{a_0, a_1, a_2\}$, then we may assign three input units to represent $A$. That is, we may have, say, $I_0, I_1, I_2$ as input units. Each unit is initialized to 0. If $A=a_0$, then $I_0$ is set to 1. If $A=a_1$, $I_1$ is set to 1, and so on.
There are no clear rules as to the “best” number of hidden layer units.

Network design is a trial-and-error process and may affect the accuracy of the resulting trained network.

The initial values of the weights may also affect the resulting accuracy. Once a network has been trained and its accuracy is not considered acceptable, it is common to repeat the training process with a different network topology or a different set of initial weights.

Cross-validation techniques for accuracy estimation (described in Section 6.13) can be used to help decide when an acceptable network has been found.

A number of automated techniques have been proposed that search for a “good” network structure. These typically use a hill-climbing approach that starts with an initial structure that is selectively modified.
6.6.3 Backpropagation 倒傳遞

- Backpropagation learns by iteratively processing a data set of training tuples, comparing the network’s prediction for each tuple with the actual *known target value*.
- The target value: a. classification problems (for 類別標籤) b. continuous value (for 預測).
- For each training tuple, the *weights* are modified so as to *minimize the mean squared error* between the network’s prediction and the actual target value.
- These modifications are made in the “backwards” direction, that is, from the output layer, through each hidden layer down to the first hidden layer (hence the name *backpropagation*).
- Although it is not guaranteed, in general the weights will eventually converge, and the learning process stops.
Figure 6.16 Backpropagation algorithm

(1) Initialize all weights and biases in network;
(2) while terminating condition is not satisfied {
(3) for each training tuple $X$ in $D$ {
(4) // Propagate the inputs forward:
(5) for each input layer unit $j$ {
(6) $O_j = I_j$; // output of an input unit is its actual input value
(7) for each hidden or output layer unit $j$ {
(8) $I_j = \sum_i w_{ij} O_i + \theta_j$; //compute the net input of unit $j$ with respect to the previous layer, $i$
(9) $O_j = \frac{1}{1+e^{-I_j}}$; // compute the output of each unit $j$
(10) // Backpropagate the errors:
(11) for each unit $j$ in the output layer
(12) $Err_j = O_j(1-O_j)(T_j-O_j)$; // compute the error
(13) for each unit $j$ in the hidden layers, from the last to the first hidden layer
(14) $Err_j = O_j(1-O_j) \sum_k Err_k w_{jk}$; // compute the error with respect to the next higher layer, $k$
(15) for each weight $w_{ij}$ in network {
(16) $\Delta w_{ij} = (l)Err_j O_i$; // weight increment
(17) $w_{ij} = w_{ij} + \Delta w_{ij}$; // weight update
(18) for each bias $\theta_j$ in network {
(19) $\Delta \theta_j = (l)Err_j$; // bias increment
(20) $\theta_j = \theta_j + \Delta \theta_j$; // bias update
(21) $}$

$D$, a data set consisting of the training tuples and their associated target values; $l$, the learning rate; network, a multilayer feed-forward network.
• **Initialize the weights:**
  The weights in the network are initialized to small random numbers (e.g., ranging from -1 to 1 or -0.5 to 0.5).

• **Propagate the inputs forward:**
  - First, the training tuple is fed to the input layer of the network. The inputs pass through the input units, unchanged. That is, for an input unit, \( j \), its output, \( O_j \), is equal to its input value, \( I_j \).
  - Next, the net input and output of each unit in the hidden and output layers are computed.

• Given a unit \( j \) in a hidden or output layer, the net input, \( I_j \), to unit \( j \) is

\[
I_j = \sum_i w_{ij} O_i + \theta_j
\]  

(6.24)

- \( w_{ij} \) is the weight of the connection from unit \( i \) in the previous layer to unit \( j \);
- \( O_i \) is the output of unit \( i \) from the previous layer; and \( \theta_j \) is the bias of the unit. The bias acts as a threshold in that it serves to vary the activity of the unit.
Each unit in the hidden and output layers takes its net input and then applies an activation function to it, as illustrated in Figure 6.17. The function symbolizes the activation of the neuron represented by the unit. The logistic, or sigmoid, function is used.

Given the net input $I_j$ to unit $j$, then $O_j$, the output of unit $j$, is computed as

$$O_j = \frac{1}{1 + e^{-I_j}}$$ (6.25)

This function is also referred to as a *squashing function*, because it maps a large input domain onto the smaller range of 0 to 1. The logistic function is nonlinear and differentiable, allowing the backpropagation algorithm to model classification problems that are linearly inseparable.
Backpropagation and Interpretability

- Efficiency of backpropagation: Each epoch (one iteration through the training set) takes $O(|D| \times w)$, with $|D|$ tuples and $w$ weights, but the number of epochs can be exponential to $n$, the number of inputs, in the worst case.

- Rule extraction from networks: network pruning
  - Simplify the network structure by removing weighted links that have the least effect on the trained network.
  - Then perform link, unit, or activation value clustering.
  - The set of input and activation values are studied to derive rules describing the relationship between the input and hidden unit layers.

- Sensitivity analysis: assess the impact that a given input variable has on a network output. The knowledge gained from this analysis can be represented in rules.
Outlines

1. What is classification? What is prediction?
2. Issues regarding classification and prediction
3. Classification by decision tree (決策樹) induction
4. Bayesian classification (貝氏分類)
5. Rule-based classification (規則式分類)
6. Classification by Backpropagation (倒傳遞類神經網路)
7. **Support Vector Machines (SVM)** 支援向量機
8. Associative classification
9. Lazy learners (or learning from your neighbors)
10. Other classification methods
11. Prediction
12. Accuracy and error measures
13. Evaluating the Accuracy of a Classifier or Predictor
14. Ensemble methods – Increasing the Accuracy
15. Model selection
16. Summary
6.7 SVM - Support Vector Machines

- A new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyperplane (i.e., “decision boundary”)
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using support vectors (“essential” training tuples) and margins (defined by the support vectors)
SVM—History and Applications

- Vapnik and colleagues (1992)—groundwork from Vapnik & Chervonenkis’ statistical learning theory in 1960s
- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- Used both for classification and prediction
- Applications:
  - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests
SVM—General Philosophy

Small Margin

Large Margin

Support Vectors
SVM—Margins and Support Vectors

Fig. 6-21

Let data D be \((X_1, y_1), \ldots, (X_{|D|}, y_{|D|})\), where \(X_i\) is the set of training tuples associated with the class labels \(y_i\).

There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data).

SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH).
SVM—Linearly Separable

- A separating hyperplane can be written as
  \[ W \cdot X + b = 0 \]
  where \( W = \{w_1, w_2, \ldots, w_n\} \) is a weight vector and \( b \) a scalar (bias)

- For 2-D it can be written as
  \[ w_0 + w_1 x_1 + w_2 x_2 = 0 \]

- The hyperplane defining the sides of the margin:
  \[ H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1 \quad \text{for } y_i = +1, \text{ and} \]
  \[ H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1 \quad \text{for } y_i = -1 \]

- Any training tuples that fall on hyperplanes \( H_1 \) or \( H_2 \) (i.e., the sides defining the margin) are support vectors

- This becomes a constrained (convex) quadratic optimization problem:
  Quadratic objective function and linear constraints \( \rightarrow \) Quadratic Programming (QP) \( \rightarrow \) Lagrangian multipliers
Why Is SVM Effective on High Dimensional Data?

- The complexity of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data.
- The support vectors are the essential or critical training examples —they lie closest to the decision boundary (MMH).
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found.
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality.
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high.
SVM—Linearly Inseparable

- Transform the original input data into a higher dimensional space

Example 6.8 Nonlinear transformation of original input data into a higher dimensional space. Consider the following example. A 3D input vector $\mathbf{X} = (x_1, x_2, x_3)$ is mapped into a 6D space $\mathbf{Z}$ using the mappings $\phi_1(X) = x_1$, $\phi_2(X) = x_2$, $\phi_3(X) = x_3$, $\phi_4(X) = (x_1)^2$, $\phi_5(X) = x_1x_2$, and $\phi_6(X) = x_1x_3$. A decision hyperplane in the new space is $d(Z) = \mathbf{WZ} + b$, where $\mathbf{W}$ and $\mathbf{Z}$ are vectors. This is linear. We solve for $\mathbf{W}$ and $b$ and then substitute back so that we see that the linear decision hyperplane in the new $\mathbf{Z}$ space corresponds to a nonlinear second order polynomial in the original 3-D input space,

$$d(Z) = w_1x_1 + w_2x_2 + w_3x_3 + w_4(x_1)^2 + w_5x_1x_2 + w_6x_1x_3 + b$$

$$= w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5 + w_6z_6 + b$$

- Search for a linear separating hyperplane in the new space
SVM—Kernel functions

- Instead of computing the dot product on the transformed data tuples, it is mathematically equivalent to instead applying a kernel function \( K(\mathbf{X}_i, \mathbf{X}_j) \) to the original data, i.e., \( K(\mathbf{X}_i, \mathbf{X}_j) = \Phi(\mathbf{X}_i) \Phi(\mathbf{X}_j) \)

- Typical Kernel Functions
  
  **Polynomial kernel of degree \( h \):** \( K(\mathbf{X}_i, \mathbf{X}_j) = (\mathbf{X}_i \cdot \mathbf{X}_j + 1)^h \)

  **Gaussian radial basis function kernel:** \( K(\mathbf{X}_i, \mathbf{X}_j) = e^{-\|\mathbf{X}_i - \mathbf{X}_j\|^2/2\sigma^2} \)

  **Sigmoid kernel:** \( K(\mathbf{X}_i, \mathbf{X}_j) = \tanh(\kappa \mathbf{X}_i \cdot \mathbf{X}_j - \delta) \)

- SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional user parameters)
Scaling SVM by Hierarchical Micro-Clustering

- SVM is not scalable to the number of data objects in terms of training time and memory usage
- “Classifying Large Datasets Using SVMs with Hierarchical Clusters Problem” by Hwanjo Yu, Jiong Yang, Jiawei Han, KDD’03
- CB-SVM (Clustering-Based SVM)
  - Given limited amount of system resources (e.g., memory), maximize the SVM performance in terms of accuracy and the training speed
  - Use micro-clustering to effectively reduce the number of points to be considered
  - At deriving support vectors, de-cluster micro-clusters near “candidate vector” to ensure high classification accuracy
CB-SVM: Clustering-Based SVM

- Training data sets may not even fit in memory
- Read the data set once (minimizing disk access)
  - Construct a statistical summary of the data (i.e., hierarchical clusters) given a limited amount of memory
  - The statistical summary maximizes the benefit of learning SVM
- The summary plays a role in indexing SVMs
- Essence of Micro-clustering (Hierarchical indexing structure)
  - Use micro-cluster hierarchical indexing structure
    - provide finer samples closer to the boundary and coarser samples farther from the boundary
  - Selective de-clustering to ensure high accuracy
CF-Tree: Hierarchical Micro-cluster
CB-SVM Algorithm: Outline

- Construct two CF-trees from positive and negative data sets independently
  - Need one scan of the data set
- Train an SVM from the centroids of the root entries
- De-cluster the entries near the boundary into the next level
  - The children entries de-clustered from the parent entries are accumulated into the training set with the non-declustered parent entries
- Train an SVM again from the centroids of the entries in the training set
- Repeat until nothing is accumulated
Selective Declustering

- CF tree is a suitable base structure for selective declustering
- De-cluster only the cluster $E_i$ such that
  - $D_i - R_i < D_s$, where $D_i$ is the distance from the boundary to the center point of $E_i$ and $R_i$ is the radius of $E_i$
  - Decluster only the cluster whose subclusters have possibilities to be the support cluster of the boundary
    - “Support cluster”: The cluster whose centroid is a support vector
Experiment on Synthetic Dataset

Figure 6: Synthetic data set in a two-dimensional space. ‘↑’: positive data; ‘↓’: negative data
# Experiment on a Large Data Set

<table>
<thead>
<tr>
<th>S-Rate</th>
<th># of data</th>
<th># of errors</th>
<th>T-Time</th>
<th>S-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0001%</td>
<td>23</td>
<td>6425</td>
<td>0.000114</td>
<td>822.97</td>
</tr>
<tr>
<td>0.001%</td>
<td>226</td>
<td>2413</td>
<td>0.000972</td>
<td>825.40</td>
</tr>
<tr>
<td>0.01%</td>
<td>2333</td>
<td>1132</td>
<td>0.03</td>
<td>828.61</td>
</tr>
<tr>
<td>0.1%</td>
<td>23273</td>
<td>1012</td>
<td>6.287</td>
<td>835.87</td>
</tr>
<tr>
<td>1%</td>
<td>230380</td>
<td>1015</td>
<td>1192.793</td>
<td>838.92</td>
</tr>
<tr>
<td>5%</td>
<td>1151714</td>
<td>1020</td>
<td>20705.4</td>
<td>842.92</td>
</tr>
<tr>
<td>ASVM</td>
<td>307</td>
<td>865</td>
<td></td>
<td>54872.213</td>
</tr>
<tr>
<td>CB-SVM</td>
<td>2893</td>
<td>876</td>
<td></td>
<td>2528.213</td>
</tr>
</tbody>
</table>

Table 4: Performance results on the very large data set (# of training data = 23066169, # of testing data = 233890). S-Rate: sampling rate; T-Time: training time; S-Time: sampling time; ASVM: selective sampling
<table>
<thead>
<tr>
<th>SVM</th>
<th>Neural Network</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relatively new concept</td>
<td>Relatively old</td>
</tr>
<tr>
<td>Deterministic algorithm</td>
<td>Nondeterministic algorithm</td>
</tr>
<tr>
<td>Nice Generalization properties</td>
<td>Generalizes well but doesn’t have strong</td>
</tr>
<tr>
<td></td>
<td>mathematical foundation</td>
</tr>
<tr>
<td>Hard to learn – learned in batch mode</td>
<td>Can easily be learned in incremental fashion</td>
</tr>
<tr>
<td>using quadratic programming techniques</td>
<td></td>
</tr>
<tr>
<td>Using kernels can learn very complex</td>
<td>To learn complex functions—use multilayer</td>
</tr>
<tr>
<td>functions</td>
<td>perceptron (not that trivial)</td>
</tr>
</tbody>
</table>
SVM Related Links

- SVM Website
  - http://www.kernel-machines.org/

- Representative implementations
  - LIBSVM: an efficient implementation of SVM, multi-class classifications, nu-SVM, one-class SVM, including also various interfaces with java, python, etc.
  - SVM-light: simpler but performance is not better than LIBSVM, support only binary classification and only C language
  - SVM-torch: another recent implementation also written in C.
SVM—Introduction Literature

- “Statistical Learning Theory” by Vapnik: extremely hard to understand, containing many errors too.

  - Better than the Vapnik’s book, but still written too hard for introduction, and the examples are so not-intuitive

- The book “An Introduction to Support Vector Machines” by N. Cristianini and J. Shawe-Taylor
  - Also written hard for introduction, but the explanation about the mercer’s theorem is better than above literatures

- The neural network book by Haykins
  - Contains one nice chapter of SVM introduction
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary
Associative Classification

- Associative classification
  - Association rules are generated and analyzed for use in classification
  - Search for strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels
  - Classification: Based on evaluating a set of rules in the form of
    \[ P_1 \land P_2 \land \ldots \land P_l \rightarrow \text{“A_{class} = C”} \] (conf, sup)
- Why effective?
  - It explores highly confident associations among multiple attributes and may overcome some constraints introduced by decision-tree induction, which considers only one attribute at a time
  - In many studies, associative classification has been found to be more accurate than some traditional classification methods, such as C4.5
Typical Associative Classification Methods

- **CBA** (Classification By Association: Liu, Hsu & Ma, KDD’98)
  - Mine association possible rules in the form of
    - Cond-set (a set of attribute-value pairs) → class label
  - Build classifier: Organize rules according to decreasing precedence based on confidence and then support

- **CMAR** (Classification based on Multiple Association Rules: Li, Han, Pei, ICDM’01)
  - Classification: Statistical analysis on multiple rules

- **CPAR** (Classification based on Predictive Association Rules: Yin & Han, SDM’03)
  - Generation of predictive rules (FOIL-like analysis)
  - High efficiency, accuracy similar to CMAR

- **RCBT** (Mining top-k covering rule groups for gene expression data, Cong et al. SIGMOD’05)
  - Explore high-dimensional classification, using top-k rule groups
  - Achieve high classification accuracy and high run-time efficiency
A Closer Look at CMAR

- CMAR (Classification based on Multiple Association Rules: Li, Han, Pei, ICDM’01)
- Efficiency: Uses an enhanced FP-tree that maintains the distribution of class labels among tuples satisfying each frequent itemset
- Rule pruning whenever a rule is inserted into the tree
  - Given two rules, $R_1$ and $R_2$, if the antecedent of $R_1$ is more general than that of $R_2$ and $\text{conf}(R_1) \geq \text{conf}(R_2)$, then $R_2$ is pruned
  - Prunes rules for which the rule antecedent and class are not positively correlated, based on a $\chi^2$ test of statistical significance
- Classification based on generated/pruned rules
  - If only one rule satisfies tuple $X$, assign the class label of the rule
  - If a rule set $S$ satisfies $X$, CMAR
    - divides $S$ into groups according to class labels
    - uses a weighted $\chi^2$ measure to find the strongest group of rules, based on the statistical correlation of rules within a group
    - assigns $X$ the class label of the strongest group
Associative Classification May Achieve High Accuracy and Efficiency (Cong et al. SIGMOD05)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RCBT</th>
<th>CBA</th>
<th>IRG Classifier</th>
<th>C4.5 family</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>single tree</td>
<td>bagging</td>
</tr>
<tr>
<td>AML/ALL (ALL)</td>
<td>91.18%</td>
<td>91.18%</td>
<td>64.71%</td>
<td>91.18%</td>
<td>91.18%</td>
</tr>
<tr>
<td>Lung Cancer(LC)</td>
<td>97.99%</td>
<td>81.88%</td>
<td>89.93%</td>
<td>81.88%</td>
<td>96.64%</td>
</tr>
<tr>
<td>Ovarian Cancer(OC)</td>
<td>97.67%</td>
<td>93.02%</td>
<td>-</td>
<td>97.67%</td>
<td>97.67%</td>
</tr>
<tr>
<td>Prostate Cancer(PC)</td>
<td>97.06%</td>
<td>82.35%</td>
<td>88.24%</td>
<td>26.47%</td>
<td>26.47%</td>
</tr>
<tr>
<td>Average Accuracy</td>
<td>95.98%</td>
<td>87.11%</td>
<td>80.96%</td>
<td>74.3%</td>
<td>77.99%</td>
</tr>
</tbody>
</table>

Table 2: Classification Results

(a) ALL-AML leukemia
(b) Lung Cancer
(c) Ovarian Cancer
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- 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 set, 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 its implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space
Lazy Learner: Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified

- Typical approaches
  - $k$-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference
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, $\text{dist}(X_1, X_2)$
- 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 $x_q$
- Vonoroi 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
  - $w \equiv \frac{1}{d(x_q, x_i)^2}$
- 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
Case-Based Reasoning (CBR)

- **CBR**: Uses a database of problem solutions to solve new problems
- Store **symbolic description** (tuples or cases)—not points in a Euclidean space
- **Applications**: Customer-service (product-related diagnosis), legal ruling
- **Methodology**
  - Instances represented by rich symbolic descriptions (e.g., function graphs)
  - Search for similar cases, multiple retrieved cases may be combined
  - Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- **Challenges**
  - Find a good similarity metric
  - Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary
Genetic Algorithms (GA)

- Genetic Algorithm: based on an analogy to biological evolution
- An initial **population** is created consisting of randomly generated rules
  - Each rule is represented by a string of bits
  - E.g., if $A_1$ and $\neg A_2$ then $C_2$ can be encoded as 100
  - If an attribute has $k > 2$ values, $k$ bits can be used
- Based on the notion of survival of the **fittest**, a new population is formed to consist of the fittest rules and their offsprings
- The fitness of a rule is represented by its *classification accuracy* on a set of training examples
- Offsprings are generated by *crossover* and *mutation*
- The process continues until a population $P$ evolves *when each rule in $P$ satisfies a prespecified threshold*
- Slow but easily parallelizable
Rough Set Approach

- Rough sets are used to **approximately or “roughly” define equivalent classes**

- A rough set for a given class $C$ is approximated by two sets: a **lower approximation** (certain to be in $C$) and an **upper approximation** (cannot be described as not belonging to $C$)

- Finding the minimal subsets (**reducts**) of attributes for feature reduction is NP-hard but a **discernibility matrix** (which stores the differences between attribute values for each pair of data tuples) is used to reduce the computation intensity
Fuzzy Set Approaches

- Fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership (such as using fuzzy membership graph).
- Attribute values are converted to fuzzy values:
  - e.g., income is mapped into the discrete categories \{low, medium, high\} with fuzzy values calculated.
- For a given new sample, more than one fuzzy value may apply.
- Each applicable rule contributes a vote for membership in the categories.
- Typically, the truth values for each predicted category are summed, and these sums are combined.
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary
What Is Prediction?

- (Numerical) prediction is similar to classification
  - construct a model
  - use model to predict continuous or ordered value for a given input
- Prediction is different from classification
  - Classification refers to predict categorical class label
  - Prediction models continuous-valued functions
- Major method for prediction: regression
  - model the relationship between one or more independent or predictor variables and a dependent or response variable
- Regression analysis
  - Linear and multiple regression
  - Non-linear regression
  - Other regression methods: generalized linear model, Poisson regression, log-linear models, regression trees
Linear Regression

- **Linear regression**: involves a response variable $y$ and a single predictor variable $x$
  \[ y = w_0 + w_1 x \]
  where $w_0$ (y-intercept) and $w_1$ (slope) are regression coefficients

- **Method of least squares**: estimates the best-fitting straight line
  \[
  w_1 = \frac{\sum_{i=1}^{\|D\|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{\|D\|}(x_i - \bar{x})^2} \quad w_0 = \bar{y} - w_1 \bar{x}
  \]

- **Multiple linear regression**: involves more than one predictor variable
  - Training data is of the form $(X_1, y_1)$, $(X_2, y_2)$, …, $(X_{\|D\|}, y_{\|D\|})$
  - Ex. For 2-D data, we may have: $y = w_0 + w_1 x_1 + w_2 x_2$
  - Solvable by extension of least square method or using SAS, S-Plus
  - Many nonlinear functions can be transformed into the above
Nonlinear Regression

- Some nonlinear models can be modeled by a polynomial function
- A polynomial regression model can be transformed into linear regression model. For example,
  \[ y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 \]
  convertible to linear with new variables: \( x_2 = x^2, x_3 = x^3 \)
  \[ y = w_0 + w_1 x + w_2 x_2 + w_3 x_3 \]
- Other functions, such as power function, can also be transformed to linear model
- Some models are intractable nonlinear (e.g., sum of exponential terms)
  - possible to obtain least square estimates through extensive calculation on more complex formulae
Other Regression-Based Models

- **Generalized linear model:**
  - Foundation on which linear regression can be applied to modeling categorical response variables
  - Variance of $y$ is a function of the mean value of $y$, not a constant
  - **Logistic regression:** models the prob. of some event occurring as a linear function of a set of predictor variables
  - **Poisson regression:** models the data that exhibit a Poisson distribution

- **Log-linear models:** (for categorical data)
  - Approximate discrete multidimensional prob. distributions
  - Also useful for data compression and smoothing

- **Regression trees and model trees**
  - Trees to predict continuous values rather than class labels
Regression Trees and Model Trees

- Regression tree: proposed in CART system (Breiman et al. 1984)
  - CART: Classification And Regression Trees
  - Each leaf stores a continuous-valued prediction
  - It is the average value of the predicted attribute for the training tuples that reach the leaf

- Model tree: proposed by Quinlan (1992)
  - Each leaf holds a regression model—a multivariate linear equation for the predicted attribute
  - A more general case than regression tree

- Regression and model trees tend to be more accurate than linear regression when the data are not represented well by a simple linear model
Predictive Modeling in Multidimensional Databases

- Predictive modeling: Predict data values or construct generalized linear models based on the database data
- One can only predict value ranges or category distributions
- Method outline:
  - Minimal generalization
  - Attribute relevance analysis
  - Generalized linear model construction
  - Prediction
- Determine the major factors which influence the prediction
  - Data relevance analysis: uncertainty measurement, entropy analysis, expert judgement, etc.
- Multi-level prediction: drill-down and roll-up analysis
Prediction: Numerical Data

Relevance Analysis

Profit:

<table>
<thead>
<tr>
<th>Range</th>
<th>Graph Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>-365.00~480.00</td>
<td>Red</td>
</tr>
<tr>
<td>805.00~1000.00</td>
<td>Blue</td>
</tr>
<tr>
<td>1260.00~6005.00</td>
<td>Magenta</td>
</tr>
<tr>
<td>490.00~905.00</td>
<td>Green</td>
</tr>
<tr>
<td>1130.00~1260.00</td>
<td>Yellow</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>#</th>
<th>Predictive Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sale_Price</td>
<td>$4950.00~91950.00</td>
</tr>
<tr>
<td>2</td>
<td>Channel</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Cost_of_Goods_Sold</td>
<td>$3800.00~85000.00</td>
</tr>
<tr>
<td>4</td>
<td>Advertising_Cost</td>
<td>$0.00~1715000</td>
</tr>
<tr>
<td>5</td>
<td>Average_Sales_Area</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>(1130.000~4230.000)</td>
<td></td>
</tr>
</tbody>
</table>

For Help, press F1
Prediction: Categorical Data
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary
Classifier Accuracy Measures

<table>
<thead>
<tr>
<th>classes</th>
<th>buy_computer = yes</th>
<th>buy_computer = no</th>
<th>total</th>
<th>recognition(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>buy_computer = yes</td>
<td>6954</td>
<td>46</td>
<td>7000</td>
<td>99.34</td>
</tr>
<tr>
<td>buy_computer = no</td>
<td>412</td>
<td>2588</td>
<td>3000</td>
<td>86.27</td>
</tr>
<tr>
<td>total</td>
<td>7366</td>
<td>2634</td>
<td>10000</td>
<td>95.52</td>
</tr>
</tbody>
</table>

- **Accuracy of a classifier** $M$, $\text{acc}(M)$: percentage of test set tuples that are correctly classified by the model $M$
  - Error rate (misclassification rate) of $M = 1 - \text{acc}(M)$
  - Given $m$ classes, $C_{M_{i,j}}$, an entry in a **confusion matrix**, indicates # of tuples in class $i$ that are labeled by the classifier as class $j$
- **Alternative accuracy measures** (e.g., for cancer diagnosis)
  - Sensitivity = $t$-pos/$pos$  /* true positive recognition rate */
  - Specificity = $t$-neg/$neg$  /* true negative recognition rate */
  - Precision = $t$-pos/($t$-pos + $f$-pos)
  - Accuracy = Sensitivity * $pos/($pos + neg) + Specificity * neg/($pos + neg)
- This model can also be used for cost-benefit analysis
Predictor Error Measures

- Measure predictor accuracy: measure how far off the predicted value is from the actual known value

- **Loss function**: measures the error betw. $y_i$ and the predicted value $y_i'$
  - Absolute error: $|y_i - y_i'|$
  - Squared error: $(y_i - y_i')^2$

- Test error (generalization error): the average loss over the test set
  - Mean absolute error: $\frac{\sum_{i=1}^{d}|y_i - y_i'|}{d}$
  - Relative absolute error: $\frac{\sum_{i=1}^{d}|y_i - y_i'|}{\sum_{i=1}^{d}|y_i - \bar{y}|}$
  - Mean squared error: $\frac{\sum_{i=1}^{d}(y_i - y_i')^2}{d}$
  - Relative squared error: $\frac{\sum_{i=1}^{d}(y_i - y_i')^2}{\sum_{i=1}^{d}(y_i - \bar{y})^2}$

The mean squared-error exaggerates the presence of outliers

Popularly use (square) root mean-square error, similarly, root relative squared error
Evaluating the Accuracy of a Classifier or Predictor (I)

- **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*<sub>i</sub> 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
Evaluating the Accuracy of a Classifier or Predictor (II)

- **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**
  - Suppose we are given a data set of \( d \) tuples. The data set 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 will end up in the bootstrap, and the remaining 36.8% will 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:

\[
\text{acc}(M) = \sum_{i=1}^{k} (0.632 \times \text{acc}(M_i)_{\text{test\_set}} + 0.368 \times \text{acc}(M_i)_{\text{train\_set}})
\]
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary
Ensemble Methods: Increasing the Accuracy

- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of $k$ learned models, $M_1, M_2, \ldots, M_k$, with the aim of creating an improved model $M^*$

- 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
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 significant 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.
- The boosting algorithm can be extended for the prediction of continuous values.
- Comparing with bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data.
Adaboost (Freund and Schapire, 1997)

- Given a set of $d$ class-labeled tuples, $(X_1, y_1), \ldots, (X_d, y_d)$
- 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: $\text{err}(X_j)$ is the misclassification error of tuple $X_j$. Classifier $M_i$ error rate is the sum of the weights of the misclassified tuples:

$$\text{error}(M_i) = \sum_{j=1}^{d} w_j \times \text{err}(X_j)$$

- The weight of classifier $M_i$’s vote is

$$\log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)}$$
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary
Model Selection: ROC Curves

- ROC (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- 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 represents the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary
Summary (I)

- Classification and prediction are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends.

- Effective and scalable methods have been developed for decision trees induction, Naive Bayesian classification, Bayesian belief network, rule-based classifier, Backpropagation, Support Vector Machine (SVM), associative classification, nearest neighbor classifiers, and case-based reasoning, and other classification methods such as genetic algorithms, rough set and fuzzy set approaches.

- Linear, nonlinear, and generalized linear models of regression can be used for prediction. Many nonlinear problems can be converted to linear problems by performing transformations on the predictor variables. Regression trees and model trees are also used for prediction.
Summary (II)

- **Stratified k-fold cross-validation** is a recommended method for accuracy estimation. *Bagging* and *boosting* can be used to increase overall accuracy by learning and combining a series of individual models.

- **Significance tests** and **ROC curves** are useful for model selection.

- There have been numerous **comparisons of the different classification and prediction methods**, and 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, interpretability, and scalability must be considered and can involve trade-offs, further complicating the quest for an overall superior method.