Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Lazy learners (or learning from your neighbors)
- Prediction
- Model Evaluation and Selection
- Ensemble methods
Classification vs. Prediction

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

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

- **Typical applications**
  - Credit approval
  - Target marketing
  - Medical diagnosis
  - Fraud detection
Classification—A Two-Step Process

- **Model construction**: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the **class label attribute**
  - The set of tuples used for model construction is **training set**
  - The model is represented as classification rules, decision trees, or mathematical formulae

- **Model usage**: for classifying future or unknown objects
  - **Estimate accuracy** of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to **classify data** tuples whose class labels are not known
Process (1): Model Construction

IF rank = ‘professor’ OR years > 6 THEN tenured = ‘yes’
Process (2): Using the Model in Prediction

Testing Data

Classifier

Unseen Data

(Jeff, Professor, 4)

Tenured?

Yes
Supervised vs. Unsupervised Learning

- **Supervised learning (classification)**
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- **Unsupervised learning (clustering)**
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Lazy learners (or learning from your neighbors)
- Prediction
- Model Evaluation and Selection
- Ensemble methods
Issues: Data Preparation

- Data cleaning
  - Preprocess data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes
- Data transformation
  - Generalize and/or normalize data
Issues: Evaluating Classification Methods

- Accuracy
  - classifier accuracy: predicting class label
  - predictor accuracy: guessing value of predicted attributes
- Speed
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in coupling more data
- Interpretability
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Lazy learners (or learning from your neighbors)
- Prediction
- Accuracy and error measures
- Ensemble methods
## Decision Tree Induction: Training Dataset

This follows an example of Quinlan’s ID3 (Playing Tennis)

<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>
Output: A Decision Tree for “buys_computer”

- age?
  - <=30
    - student?
      - no
      - yes
  - 31..40
    - yes
  - >40
    - credit rating?
      - excellent
        - no
      - fair
        - yes
Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left
Attribute Selection Measure: Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Let $p_i$ be the probability that an arbitrary tuple in $D$ belongs to class $C_i$, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in $D$:
  
  $$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

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

- Information gained by branching on attribute A
  
  $$Gain(A) = Info(D) - Info_A(D)$$
Attribute Selection: Information Gain

- Class P: buys_computer = “yes”
- Class N: buys_computer = “no”

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

\[
\text{Entropy}([9+,5-])
\]

\[
\begin{array}{|c|c|c|c|}
\hline
\text{age} & p_i & n_i & I(p_i, n_i) \\
\hline
\leq 30 & 2 & 3 & 0.971 \\
31...40 & 4 & 0 & 0 \\
>40 & 3 & 2 & 0.971 \\
\hline
\end{array}
\]

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

\[
\text{Gain}(\text{income}) = 0.029
\]

\[
\text{Gain}(\text{student}) = 0.151
\]

\[
\text{Gain}(\text{credit\_rating}) = 0.048
\]

\[
I(2,3) \text{ means “age } \leq 30\text{” has 5 out of 14 samples, with 2 yes’es and 3 no’s. Hence}
\]

\[
\text{Gain}(\text{age}) = \text{Info}(D) - \text{Info}_{\text{age}}(D) = 0.246
\]
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:
  - D1 is the set of tuples in D satisfying \(A \leq \text{split-point}\), and
  - D2 is the set of tuples in D satisfying \(A > \text{split-point}\)
Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values.
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain).
  \[
  \text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}(A)}
  \]

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

- Ex. \[
\text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}(A)}
\]

\[
\text{SplitInfo}_A(D) = -\frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) - \frac{6}{14} \times \log_2 \left( \frac{6}{14} \right) - \frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) = 0.926
\]

- gain\_ratio(income) = 0.029/0.926 = 0.031

- The attribute with the maximum gain ratio is selected as the splitting attribute.
Overfitting and 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
Example:
Decision Tree ID3 for “Play Tennis”
## Training Examples

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cold</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Decision Tree for PlayTennis

Outlook
  Sunny
  Humidity
    High No
    Normal Yes
  Overcast
  Rain
    Wind
      Strong No
      Weak Yes
Decision Tree for “Play Tennis”

- Each internal node tests an attribute
- Each branch corresponds to an attribute value node
- Each leaf node assigns a classification
Decision Tree for “Play Tennis”

Outlook
- Sunny
  - Temperature
    - Hot
  - Humidity
    - High
      - Wind
        - Weak
          - PlayTennis: No
  - Overcast
    - PlayTennis: Yes
  - Rain
    - Wind
      - Strong
        - No
      - Weak
        - Yes

Humidity
- High
  - PlayTennis: No
- Normal
  - Yes

Wind
- Strong
  - No
- Weak
  - Yes
Decision Tree for Conjunction

Outlook=Sunny \land Wind=Weak
Decision Tree for Disjunction

Outlook=Sunny ∨ Wind=Weak

Outlook

- Sunny
  - Yes
- Overcast
- Rain
  - Wind
    - Strong
    - No
    - Weak
      - Yes
      - No
    - Strong
      - No
    - Weak
      - Yes
Decision Tree for XOR

Outlook=Sunny  XOR Wind=Weak

1. Outlook
   - Sunny
   - Overcast
   - Rain

2. Wind
   - Strong
   - Weak

3. Strong
   - Yes
   - No

4. Weak
   - No
   - Yes

5. Wind
   - Strong
   - Weak

6. Strong
   - Yes
   - No

7. Weak
   - Yes
- decision trees represent disjunctions of conjunctions

\[(\text{Outlook}=\text{Sunny} \land \text{Humidity}=\text{Normal})\]
\[\lor (\text{Outlook}=\text{Overcast})\]
\[\lor (\text{Outlook}=\text{Rain} \land \text{Wind}=\text{Weak})\]
Top-Down Induction of Decision Trees ID3

1. A ← the “best” decision attribute for next node
2. Assign A as decision attribute for node
3. For each value of A create new descendant
4. Sort training examples to leaf node according to the attribute value of the branch
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.
Which Attribute is "best"?

[A_1=?]

[29+, 35-]

True

False

[21+, 5-]

[8+, 30-]

[A_2=?]

[29+, 35-]

True

False

[18+, 33-]

[11+, 2-]
Information Gain

- Gain(D, A): expected reduction in entropy due to sorting D on attribute A

$$\text{Gain}(D, A) = \text{Entropy}(D) - \sum_{v \in \text{values}(A)} \frac{|D_v|}{|D|} \text{Entropy}(D_v)$$

Entropy([29+, 35-]) = $-\frac{29}{64} \log_2 \frac{29}{64} - \frac{35}{64} \log_2 \frac{35}{64}$

$= 0.99$
Information Gain

\[
\text{Entropy([21+, 5-])} = 0.71 \\
\text{Entropy([8+, 30-])} = 0.74 \\
\text{Gain}(S,A_1) = \text{Entropy}(S) \\
\quad - \frac{26}{64} \times \text{Entropy([21+, 5-])} \\
\quad - \frac{38}{64} \times \text{Entropy([8+, 30-])} \\
= 0.27
\]

\[
\text{Entropy([18+, 33-])} = 0.94 \\
\text{Entropy([8+, 30-])} = 0.62 \\
\text{Gain}(S,A_2) = \text{Entropy}(S) \\
\quad - \frac{51}{64} \times \text{Entropy([18+, 33-])} \\
\quad - \frac{13}{64} \times \text{Entropy([11+, 2-])} \\
= 0.12
\]

0.27 > 0.12 \Rightarrow A_1 \text{ is best}

[29+, 35-]  
\[ A_1=? \]

True  
\[ [21+, 5-] \]

False  
\[ [8+, 30-] \]

[29+, 35-]  
\[ A_2=? \]

True  
\[ [18+, 33-] \]

False  
\[ [11+, 2-] \]
# Training Examples

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cold</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Selecting the First Attribute

Humidity

- High: [3+, 4-]
  - E=0.985

- Normal: [6+, 1-]
  - E=0.592

Gain(D,Humidity) = 0.940 - (7/14)*0.985 - (7/14)*0.592 = 0.151

Wind

- Weak: [7+, 2-]
  - E=0.764

- Strong: [3+, 2-]
  - E=0.971

Gain(D,Wind) = 0.940 - (9/14)*0.764 - (5/14)*0.971 = 0.102
Selecting the First Attribute

D=[9+,5-]
E=0.940

Outlook

Sunny
[2+, 3-]
E=0.971

Overcast
[4+, 0]
E=0.0

Rain
[3+, 2-]
E=0.971

Gain(D,Outlook)
=0.940-(5/14)*0.971
-(4/14)*0.0 – (5/14)*0.0971
=0.247
Selecting the Next Attribute

[D1,D2,...,D14] [9+,5-]

Outlook

Sunny

Overcast

Rain

Dsunny=[D1,D2,D8,D9,D11] [2+,3-]

[D3,D7,D12,D13] [4+,0-]

Yes

[D4,D5,D6,D10,D14] [3+,2-]

Gain(Dsunny, Humidity)=0.970-(3/5)0.0 – 2/5(0.0) = 0.970
Gain(Dsunny, Temp.)=0.970-(2/5)0.0 –2/5(1.0)-(1/5)0.0 = 0.570
Gain(Dsunny, Wind)=0.970= -(2/5)1.0 – 3/5(0.918) = 0.019
Selecting the Next Attribute

Outlook

Sunny
Humidity
High
No [D1,D2]
Normal
Yes [D8,D9,D11]

Overcast

Rain

Humidity

[D3,D7,D12,D13]

Yes

[D4,D5,D6,D10,D14] [3+,2-]

? 

Gain(D_{\text{rain}}, \text{Humidity}) = ... 
Gain(D_{\text{rain}}, \text{Temp.}) = ... 
Gain(D_{\text{rain}}, \text{Wind}) = ...
Final ID3 Decision Tree

Outlook

Sunny

- Humidity
  - High
    - No: [D1,D2]
  - Normal
    - Yes: [D8,D9,D11]

Overcast

Rain

- Wind
  - Strong
    - No: [D6,D14]
  - Weak
    - Yes: [D4,D5,D10]
Chapter 6. Classification and Prediction

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

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

- Let $X$ be a data sample (“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 $X$
- $P(H)$ (prior probability), the initial probability
  - E.g., $X$ will buy computer, regardless of age, income, ...
- $P(X)$: probability that sample data is observed
- $P(X|H)$ (posteriori probability), the probability of observing the sample $X$, given that the hypothesis holds
  - E.g., Given that $X$ will buy computer, the prob. that $X$ is 31..40, medium income
Bayesian Theorem

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

$$P(H | \mathbf{x}) = \frac{P(\mathbf{x}|H)P(H)}{P(\mathbf{x})}$$

- Informally, this can be written as

  posteriori = likelihood x prior/evidence

- Predicts $\mathbf{x}$ belongs to $C_2$ iff the probability $P(C_i|\mathbf{x})$ is the highest among all the $P(C_k|\mathbf{x})$ for all the $k$ classes

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

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

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

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

\[
P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)
\]
Derivation of Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):
  \[
P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)
  \]

- This greatly reduces the computation cost: Only counts the class distribution

- If \( A_k \) is categorical, \( P(x_k|C_i) \) is the \# of tuples in \( C_i \) having value \( x_k \) for \( A_k \) divided by \( |C_i, D| \) (\# of tuples of \( C_i \) in \( D \))

- If \( A_k \) is continous-valued, \( P(x_k|C_i) \) is usually computed based on Gaussian distribution with a mean \( \mu \) and standard deviation \( \sigma \)
  \[
g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}
  \]

  and \( P(x_k|C_i) \) is
  \[
P(X|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})
  \]
Naïve Bayes Classifier: Training Dataset

Class:
C1:buys_computer = ‘yes’
C2:buys_computer = ‘no’

Data sample
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

<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>
Naïve Bayes Classifier: An Example

- **P(C_i):**
  - P(buys_computer = “yes”) = 9/14 = 0.643
  - P(buys_computer = “no”) = 5/14 = 0.357

- Compute P(X|C_i) for each class
  - P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222
  - P(age = “<=30” | buys_computer = “no”) = 3/5 = 0.6
  - P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444
  - P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4
  - P(student = “yes” | buys_computer = “yes”) = 6/9 = 0.667
  - P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2
  - P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667
  - P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

- **X = (age <= 30, income = medium, student = yes, credit_rating = fair)**

  - P(X | C_i) : P(X|buys_computer = “yes”) = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
  - P(X|buys_computer = “no”) = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

  - P(X | C_i) * P(C_i) : P(X|buys_computer = “yes”) * P(buys_computer = “yes”) = 0.028
  - P(X|buys_computer = “no”) * P(buys_computer = “no”) = 0.007

  Therefore, X belongs to class (“buys_computer = yes”)
Avoiding the 0-Probability Problem

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

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

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income=medium (990), and income = high (10),
- Use Laplacian correction (or Laplacian estimator)
  - Adding 1 to each case
    - Prob(income = low) = 1/1003
    - Prob(income = medium) = 991/1003
    - Prob(income = high) = 11/1003
  - The “corrected” prob. estimates are close to their “uncorrected” counterparts
Naïve Bayes Classifier: Comments

- **Advantages**
  - Easy to implement
  - Good results obtained in most of the cases

- **Disadvantages**
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.
      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier

- How to deal with these dependencies?
  - Bayesian Belief Networks
Bayesian Belief Networks

- 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 of P
- No dependency between Z and P
- Has no loops or cycles
Bayesian Belief Network: An Example

Bayesian Belief Networks

The conditional probability table (CPT) for variable LungCancer:

<table>
<thead>
<tr>
<th></th>
<th>LC</th>
<th>~LC</th>
<th>(FH, S)</th>
<th>(FH, ~S)</th>
<th>(~FH, S)</th>
<th>(~FH, ~S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>~LC</td>
<td>0.2</td>
<td>0.5</td>
<td>0.5</td>
<td>0.7</td>
<td>0.3</td>
<td>0.9</td>
</tr>
<tr>
<td>LC</td>
<td>0.8</td>
<td>0.2</td>
<td>0.1</td>
<td>0.9</td>
<td>0.5</td>
<td>0.5</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:

$$P(x_1,\ldots,x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(Y_i))$$
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
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Lazy learners (or learning from your neighbors)
- Prediction
- Model Evaluation and Selection
- Ensemble methods
Classification: A Mathematical Mapping

- Classification:
  - predicts categorical class labels
  - E.g., Personal homepage classification
    - \(x_i = (x_1, x_2, x_3, \ldots), y_i = +1 \text{ or } -1\)
    - \(x_1: \# \text{ of a word “homepage”}\)
    - \(x_2: \# \text{ of a word “welcome”}\)
  - Mathematically
    - \(x \in X = \mathbb{R}^n, y \in Y = \{+1, -1\}\)
    - We want a function \(f: X \rightarrow Y\)
Linear Classification

- Binary Classification problem
- The data above the red line belongs to class ‘x’
- The data below red line belongs to class ‘o’
- Examples: SVM, Perceptron, Probabilistic Classifiers
Discriminative Classifiers

Advantages
- prediction accuracy is generally high
  - As compared to Bayesian methods – in general
- robust, works when training examples contain errors
- fast evaluation of the learned target function
  - Bayesian networks are normally slow

Criticism
- long training time
- difficult to understand the learned function (weights)
  - Bayesian networks can be used easily for pattern discovery
- not easy to incorporate domain knowledge
  - Easy in the form of priors on the data or distributions
Classification by Backpropagation

- Backpropagation: 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
A Neuron (= a perceptron)

The \( n \)-dimensional input vector \( x \) is mapped into variable \( y \) by means of the scalar product and a nonlinear function mapping

\[
\sum w_i x_i + \mu_k
\]

For Example

\[
y = \text{sign} \left( \sum_{i=0}^{n} w_i x_i + \mu_k \right)
\]
A Multi-Layer Feed-Forward Neural Network

Output vector

Output layer

Hidden layer

Input layer

Input vector: \( X \)

\[
\text{Err}_j = O_j (1 - O_j) \sum_k \text{Err}_k w_{jk}
\]

\[
\theta_j = \theta_j + (l) \text{Err}_j
\]

\[
w_{ij} = w_{ij} + (l) \text{Err}_j O_i
\]

\[
\text{Err}_j = O_j (1 - O_j) (T_j - O_j)
\]

\[
O_j = \frac{1}{1 + e^{-I_j}}
\]

\[
I_j = \sum_i w_{ij} O_i + \theta_j
\]
How A Multi-Layer Neural Network Works?

- The **inputs** to the network correspond to the attributes measured for each training tuple.
- Inputs are fed simultaneously into the units making up the **input layer**.
- They are then weighted and fed simultaneously to a **hidden layer**.
- The number of hidden layers is arbitrary, although usually only one.
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction.
- 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.
- From a statistical point of view, networks perform **nonlinear regression**: Given enough hidden units and enough training samples, they can closely approximate any function.
Defining a Network Topology

- First decide the **network topology**: # of units in the **input layer**, # of **hidden layers** (if > 1), # of units in each **hidden layer**, and # of units in the **output layer**

- Normalizing the input values for each attribute measured in the training tuples to [0.0—1.0]

- One **input** unit per domain value, each initialized to 0

- **Output**, if for classification and more than two classes, one output unit per class is used

- Once a network has been trained and its accuracy is **unacceptable**, repeat the training process with a **different network topology** or a **different set of initial weights**
Backpropagation

- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to minimize the mean squared error between the network's prediction and the actual target value
- Modifications are made in the “backwards” direction: from the output layer, through each hidden layer down to the first hidden layer, hence “backpropagation”
- Steps
  - Initialize weights (to small random #s) and biases in the network
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)
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.
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Prediction
- Model Evaluation and Selection
- Ensemble methods
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

- Class 1, $y = +1$ (buys_computer = "yes")
- Class 2, $y = -1$ (buys_computer = "no")
SVM—When Data Is Linearly Separable

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
  \[ \mathbf{W} \cdot \mathbf{X} + b = 0 \]
  where \( \mathbf{W} = \{w_1, w_2, \ldots, w_n\} \) 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:
  \[ H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1 \quad \text{for} \quad y_i = +1, \quad \text{and} \]
  \[ H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1 \quad \text{for} \quad y_i = -1 \]

- 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 $X = (x_1, x_2, x_3)$ is mapped into a 6D space $Z$ using the mappings $\phi_1(X) = x_1, \phi_2(X) = x_2, \phi_3(X) = x_3, \phi_4(X) = (x_1)^2, \phi_5(X) = x_1 x_2$, and $\phi_6(X) = x_1 x_3$. A decision hyperplane in the new space is $d(Z) = WZ + b$, where $W$ and $Z$ are vectors. This is linear. We solve for $W$ and $b$ and then substitute back so that we see that the linear decision hyperplane in the new $(Z)$ space corresponds to a nonlinear second order polynomial in the original 3-D input space,

$$d(Z) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 (x_1)^2 + w_5 x_1 x_2 + w_6 x_1 x_3 + b$$

$$= w_1 z_1 + w_2 z_2 + w_3 z_3 + w_4 z_4 + w_5 z_5 + w_6 z_6 + b$$

- 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 $K(X_i, X_j)$ to the original data, i.e., $K(X_i, X_j) = \Phi(X_i) \Phi(X_j)$.

- Typical Kernel Functions

  Polynomial kernel of degree $h$: $K(X_i, X_j) = (X_i \cdot X_j + 1)^h$

  Gaussian radial basis function kernel: $K(X_i, X_j) = e^{-\|X_i - X_j\|^2/2\sigma^2}$

  Sigmoid kernel: $K(X_i, X_j) = \tanh(\kappa X_i \cdot X_j - \delta)$

- SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional user parameters).
SVM vs. Neural Network

- SVM
  - Relatively new concept
  - Deterministic algorithm
  - Nice Generalization properties
  - Hard to learn – learned in batch mode using quadratic programming techniques
  - Using kernels can learn very complex functions

- Neural Network
  - Relatively old
  - Nondeterministic algorithm
  - Generalizes well but doesn’t have strong mathematical foundation
  - Can easily be learned in incremental fashion
  - To learn complex functions—use multilayer perceptron (not that trivial)
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Lazy learners (or learning from your neighbors)
- Prediction
- Model Evaluation and Selection
- Ensemble methods
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 \textit{k}-Nearest Neighbor Algorithm

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

- $k$-NN for real-valued prediction for a given unknown tuple
  - Returns the mean values of the $k$ nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the $k$ neighbors according to their distance to the query $x_q$
    - Give greater weight to closer neighbors
  - Robust to noisy data by averaging $k$-nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
  - To overcome it, axes stretch or elimination of the least relevant attributes
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Lazy learners (or learning from your neighbors)
- Prediction
- Model Evaluation and Selection
- Ensemble methods
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
  \[
  w_1 = \frac{\sum_{i=1}^{\|D\|} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{\|D\|} (x_i - \bar{x})^2}
  \]
  \[
  w_0 = \bar{y} - w_1 \bar{x}
  \]

- **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 probability 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 probability 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
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Lazy learners (or learning from your neighbors)
- Prediction
- Model Evaluation and Selection
- Ensemble methods
Model Evaluation and Selection

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

## Confusion Matrix:

<table>
<thead>
<tr>
<th>Actual class \ Predicted class</th>
<th>C₁</th>
<th>¬ C₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>C₁</td>
<td>True Positives (TP)</td>
<td>False Negatives (FN)</td>
</tr>
<tr>
<td>¬ C₁</td>
<td>False Positives (FP)</td>
<td>True Negatives (TN)</td>
</tr>
</tbody>
</table>

## Example of Confusion Matrix:

<table>
<thead>
<tr>
<th>Actual class \ Predicted class</th>
<th>buy_computer = yes</th>
<th>buy_computer = no</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>buy_computer = yes</td>
<td>6954</td>
<td>46</td>
<td>7000</td>
</tr>
<tr>
<td>buy_computer = no</td>
<td>412</td>
<td>2588</td>
<td>3000</td>
</tr>
<tr>
<td>Total</td>
<td>7366</td>
<td>2634</td>
<td>10000</td>
</tr>
</tbody>
</table>

- Given \( m \) classes, an entry, \( CM_{ij} \) in a confusion matrix indicates # of tuples in class \( i \) that were labeled by the classifier as class \( j \)
- May have extra rows/columns to provide totals
Classifier Evaluation Metrics:
Accuracy, Error Rate, Sensitivity and Specificity

- **Classifier Accuracy**, or recognition rate: percentage of test set tuples that are correctly classified
  
  \[
  \text{Accuracy} = \frac{TP + TN}{\text{All}}
  \]

- **Error rate**: \(1 - \text{accuracy}\), or
  
  \[
  \text{Error rate} = \frac{FP + FN}{\text{All}}
  \]

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class

- **Sensitivity**: True Positive recognition rate
  - \(\text{Sensitivity} = \frac{TP}{P}\)

- **Specificity**: True Negative recognition rate
  - \(\text{Specificity} = \frac{TN}{N}\)

\[
\begin{array}{|c|c|c|c|}
\hline
A \text{\slash} P & C & \neg C \\
\hline
C & TP & FN & P \\
\hline
\neg C & FP & TN & N \\
\hline
P' & N' & \text{All} \\
\hline
\end{array}
\]
Classifier Evaluation Metrics: Precision and Recall, and F-measures

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

\[
\text{precision} = \frac{TP}{TP + FP}
\]

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

\[
\text{recall} = \frac{TP}{TP + FN}
\]

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

\[
F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

- **$F_\beta$**: weighted measure of precision and recall
  - assigns $\beta$ times as much weight to recall as to precision

\[
F_\beta = \frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}
\]
## Classifier Evaluation Metrics: Example

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

- *Precision* = $90/230 = 39.13\%$
- *Recall* = $90/300 = 30.00\%$
Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout $k$ times, accuracy = avg. of the accuracies obtained

- **Cross-validation** ($k$-fold, where $k = 10$ is most popular)
  - Randomly partition the data into $k$ mutually exclusive subsets, each approximately equal size
  - At $i$-th iteration, use $D_i$ as test set and others as training set
  - **Leave-one-out**: $k$ folds where $k = \#$ of tuples, for small sized data
  - *Stratified cross-validation*: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data
Evaluating Classifier Accuracy: Bootstrap

- **Bootstrap**
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
  - Several bootstrap methods, and a common one is **.632 bootstrap**
    - A data set with \(d\) tuples is sampled \(d\) times, with replacement, resulting in a training set of \(d\) samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since \((1 – 1/d)^d \approx e^{-1} = 0.368\))
    - Repeat the sampling procedure \(k\) times, overall accuracy of the model:
      \[
      Acc(M) = \frac{1}{k} \sum_{i=1}^{k} (0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set})
      \]
Estimating Confidence Intervals: Classifier Models $M_1$ vs. $M_2$

- Suppose we have 2 classifiers, $M_1$ and $M_2$, which one is better?
- Use 10-fold cross-validation to obtain $\overline{err}(M_1)$ and $\overline{err}(M_2)$.
- These mean error rates are just estimates of error on the true population of future data cases.
- What if the difference between the 2 error rates is just attributed to chance?
  - Use a test of statistical significance.
  - Obtain confidence limits for our error estimates.
Estimating Confidence Intervals: Null Hypothesis

- Perform 10-fold cross-validation
- Assume samples follow a t distribution with $k-1$ degrees of freedom (here, $k=10$)
- Use t-test (or Student’s t-test)

Null Hypothesis: $M_1$ & $M_2$ are the same

If we can reject null hypothesis, then

- we conclude that the difference between $M_1$ & $M_2$ is statistically significant
- Chose model with lower error rate
Estimating Confidence Intervals: $t$-test

- If only 1 test set available: **pairwise comparison**
  - For $i^{th}$ round of 10-fold cross-validation, the same cross partitioning is used to obtain $err(M_1)_i$ and $err(M_2)_i$
  - Average over 10 rounds to get $\bar{err}(M_1)$ and $\bar{err}(M_2)$

- $t$-test computes **t-statistic** with $k-1$ degrees of freedom:
  \[
  t = \frac{\bar{err}(M_1) - \bar{err}(M_2)}{\sqrt{\frac{var(M_1 - M_2)}{k}}} 
  \]
  where
  \[
  var(M_1 - M_2) = \frac{1}{k} \sum_{i=1}^{k} \left[ err(M_1)_i - err(M_2)_i - (\bar{err}(M_1) - \bar{err}(M_2)) \right]^2
  \]

- If two test sets available: use **non-paired t-test**
  \[
  var(M_1 - M_2) = \sqrt{\frac{var(M_1)}{k_1} + \frac{var(M_2)}{k_2}},
  \]
  where $k_1$ & $k_2$ are # of cross-validation samples used for $M_1$ & $M_2$, resp.
Estimating Confidence Intervals: Table for t-distribution

- Symmetric
- Significance level, e.g., \( \text{sig} = 0.05 \) or 5% means \( M_1 \) & \( M_2 \) are significantly different for 95% of population
- Confidence limit, \( z = \text{sig}/2 \)

<table>
<thead>
<tr>
<th>df</th>
<th>.05</th>
<th>.025</th>
<th>.01</th>
<th>.005</th>
<th>.0025</th>
<th>.001</th>
<th>.0005</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00</td>
<td>1.645</td>
<td>2.326</td>
<td>3.078</td>
<td>3.747</td>
<td>4.303</td>
<td>4.998</td>
</tr>
<tr>
<td>2</td>
<td>1.62</td>
<td>1.980</td>
<td>2.571</td>
<td>3.143</td>
<td>3.747</td>
<td>4.303</td>
<td>4.998</td>
</tr>
<tr>
<td>3</td>
<td>1.82</td>
<td>2.365</td>
<td>2.920</td>
<td>3.499</td>
<td>4.303</td>
<td>4.998</td>
<td>5.577</td>
</tr>
<tr>
<td>4</td>
<td>2.06</td>
<td>2.632</td>
<td>3.143</td>
<td>3.747</td>
<td>4.303</td>
<td>4.998</td>
<td>5.642</td>
</tr>
<tr>
<td>5</td>
<td>2.23</td>
<td>2.841</td>
<td>3.319</td>
<td>3.966</td>
<td>4.303</td>
<td>4.998</td>
<td>5.708</td>
</tr>
<tr>
<td>6</td>
<td>2.36</td>
<td>2.960</td>
<td>3.447</td>
<td>4.144</td>
<td>4.303</td>
<td>4.998</td>
<td>5.758</td>
</tr>
<tr>
<td>7</td>
<td>2.45</td>
<td>3.055</td>
<td>3.537</td>
<td>4.275</td>
<td>4.303</td>
<td>4.998</td>
<td>5.788</td>
</tr>
<tr>
<td>8</td>
<td>2.53</td>
<td>3.134</td>
<td>3.607</td>
<td>4.382</td>
<td>4.303</td>
<td>4.998</td>
<td>5.805</td>
</tr>
<tr>
<td>9</td>
<td>2.57</td>
<td>3.177</td>
<td>3.652</td>
<td>4.438</td>
<td>4.303</td>
<td>4.998</td>
<td>5.817</td>
</tr>
<tr>
<td>10</td>
<td>2.60</td>
<td>3.208</td>
<td>3.686</td>
<td>4.479</td>
<td>4.303</td>
<td>4.998</td>
<td>5.828</td>
</tr>
<tr>
<td>11</td>
<td>2.63</td>
<td>3.233</td>
<td>3.710</td>
<td>4.500</td>
<td>4.303</td>
<td>4.998</td>
<td>5.838</td>
</tr>
<tr>
<td>12</td>
<td>2.65</td>
<td>3.251</td>
<td>3.727</td>
<td>4.508</td>
<td>4.303</td>
<td>4.998</td>
<td>5.845</td>
</tr>
<tr>
<td>13</td>
<td>2.67</td>
<td>3.265</td>
<td>3.739</td>
<td>4.516</td>
<td>4.303</td>
<td>4.998</td>
<td>5.851</td>
</tr>
<tr>
<td>14</td>
<td>2.69</td>
<td>3.277</td>
<td>3.747</td>
<td>4.522</td>
<td>4.303</td>
<td>4.998</td>
<td>5.856</td>
</tr>
<tr>
<td>15</td>
<td>2.70</td>
<td>3.287</td>
<td>3.753</td>
<td>4.527</td>
<td>4.303</td>
<td>4.998</td>
<td>5.860</td>
</tr>
<tr>
<td>16</td>
<td>2.71</td>
<td>3.295</td>
<td>3.758</td>
<td>4.531</td>
<td>4.303</td>
<td>4.998</td>
<td>5.864</td>
</tr>
<tr>
<td>17</td>
<td>2.72</td>
<td>3.302</td>
<td>3.762</td>
<td>4.535</td>
<td>4.303</td>
<td>4.998</td>
<td>5.867</td>
</tr>
<tr>
<td>18</td>
<td>2.73</td>
<td>3.308</td>
<td>3.765</td>
<td>4.539</td>
<td>4.303</td>
<td>4.998</td>
<td>5.870</td>
</tr>
<tr>
<td>19</td>
<td>2.74</td>
<td>3.314</td>
<td>3.768</td>
<td>4.542</td>
<td>4.303</td>
<td>4.998</td>
<td>5.872</td>
</tr>
<tr>
<td>20</td>
<td>2.75</td>
<td>3.319</td>
<td>3.770</td>
<td>4.545</td>
<td>4.303</td>
<td>4.998</td>
<td>5.874</td>
</tr>
<tr>
<td>21</td>
<td>2.75</td>
<td>3.324</td>
<td>3.772</td>
<td>4.548</td>
<td>4.303</td>
<td>4.998</td>
<td>5.876</td>
</tr>
<tr>
<td>22</td>
<td>2.76</td>
<td>3.328</td>
<td>3.774</td>
<td>4.551</td>
<td>4.303</td>
<td>4.998</td>
<td>5.878</td>
</tr>
<tr>
<td>23</td>
<td>2.76</td>
<td>3.332</td>
<td>3.776</td>
<td>4.553</td>
<td>4.303</td>
<td>4.998</td>
<td>5.879</td>
</tr>
<tr>
<td>24</td>
<td>2.76</td>
<td>3.335</td>
<td>3.778</td>
<td>4.556</td>
<td>4.303</td>
<td>4.998</td>
<td>5.881</td>
</tr>
<tr>
<td>30</td>
<td>2.80</td>
<td>3.355</td>
<td>3.794</td>
<td>4.573</td>
<td>4.303</td>
<td>4.998</td>
<td>5.893</td>
</tr>
<tr>
<td>100</td>
<td>2.80</td>
<td>3.392</td>
<td>3.838</td>
<td>4.604</td>
<td>4.303</td>
<td>4.998</td>
<td>5.934</td>
</tr>
</tbody>
</table>

Confidence level \( C \)

- 50% 60% 70% 80% 90% 95% 99% 99.5% 99.8% 99.9%
Estimating Confidence Intervals: Statistical Significance

- Are $M_1$ & $M_2$ significantly different?
  - Compute $t$. Select *significance level* (e.g. $\text{sig} = 5\%$)
  - Consult table for t-distribution: Find *t value* corresponding to *k-1 degrees of freedom* (here, 9)
  - t-distribution is symmetric: typically upper % points of distribution shown → look up value for *confidence limit* $z=\text{sig}/2$ (here, 0.025)
  - If $t > z$ or $t < -z$, then $t$ value lies in rejection region:
    - *Reject null hypothesis* that mean error rates of $M_1$ & $M_2$ are same
    - Conclude: *statistically significant* difference between $M_1$ & $M_2$
  - Otherwise, conclude that any difference is *chance*
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
Issues Affecting Model Selection

- **Accuracy**
  - classifier accuracy: predicting class label

- **Speed**
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)

- **Robustness**: handling noise and missing values

- **Scalability**: efficiency in disk-resident databases

- **Interpretability**
  - understanding and insight provided by the model

- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules
Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- 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

- Boosting algorithm can be extended for numeric prediction

- 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:
  \[
  \text{error}(M_i) = \sum_{j} w_j \times \text{err}(X_j)
  \]
- The weight of classifier \( M_i \)’s vote is
  \[
  \log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)}
  \]
Random Forest (Breiman 2001)

- Random Forest:
  - Each classifier in the ensemble is a *decision tree* classifier and is
generated using a random selection of attributes at each node to
determine the split
  - During classification, each tree votes and the most popular class is
returned

- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node, F
attributes as candidates for the split at the node. The CART methodology
is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*): Creates new attributes (or
features) that are a linear combination of the existing attributes (reduces
the correlation between individual classifiers)

- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each
split, and faster than bagging or boosting
Classification of Class-Imbalanced Data Sets

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

- http://www.cs.uiuc.edu/homes/hanj/bk2/