Supervised vs. Unsupervised Learning

- **Supervised Learning**
  - Called *classification*
  - Training data (observations, measurement, etc.) are given
  - Training data include class labels predefined
  - Find rules or models of class labels of training data
  - New data are classified based on the rules or models

- **Unsupervised Learning**
  - Called *clustering*
  - No training data are given
  - New data are classified without any training data
Classification vs. Prediction

- **Classification**
  - Training class labels in attributes of a training data set
  - Predicts class labels of a new data set based on the rules or models of class labels of the training data set

- **Prediction**
  - Modeling continuous-valued functions for a data set
  - Predicts unknown or missing values in the data set

### Classification Step 1: Training

#### Training Data

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>Assistant Prof</td>
<td>3</td>
<td>no</td>
</tr>
<tr>
<td>Mary</td>
<td>Assistant Prof</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>Bill</td>
<td>Professor</td>
<td>2</td>
<td>yes</td>
</tr>
<tr>
<td>Jim</td>
<td>Associate Prof</td>
<td>7</td>
<td>yes</td>
</tr>
<tr>
<td>Dave</td>
<td>Assistant Prof</td>
<td>6</td>
<td>no</td>
</tr>
<tr>
<td>Anne</td>
<td>Associate Prof</td>
<td>3</td>
<td>no</td>
</tr>
</tbody>
</table>

#### Classification Algorithms

- IF rank = 'professor'
  - OR years > 6
  - THEN tenured = 'yes'
Classification Step 2: Prediction

NAME | RANK            | YEARS | TENURED
-----|-----------------|-------|--------
Tom   | Assistant Prof  | 2     | no     
Merlisa | Associate Prof  | 7     | no     
George | Professor       | 5     | yes    
Joseph | Assistant Prof  | 7     | yes    

Unseen Data
(Jeff, Professor, 4)

Issues in Classification

- **Accuracy**
  - Training accuracy and prediction accuracy

- **Efficiency**
  - Training time and prediction time

- **Robustness**
  - Handling noise and missing values

- **Scalability**
  - Efficient memory usage in disk-resident databases

- **Interpretability**
  - Understanding of classifying models
Chapters 8 and 9, Classification

- **Decision Tree Induction**
  - **Main Idea**
    - A decision tree is constructed in a top-down recursive divide-and-conquer manner
    - Each non-leaf node represents an attribute, and each leaf node represents a class label
    - Attributes are categorical (if continuous, they are discretized in advance)
    - An attribute is selected for each node based on a heuristic or statistical measure (e.g., information gain, gain index, gini index)
    - The training data are recursively partitioned on the selected attribute at each round
    - The new data are classified by tracing the decision tree
Example of Training Data

- **Training Data Set**

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rate</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>&lt;=30</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>yes</td>
</tr>
</tbody>
</table>

Example of Decision Tree

- **Output Decision Tree for “buys_computer”**

```
age?
  <=30
  student?
    yes
      yes
    no
  >40
    31..40
    credit rating?
      fair
      excellent
      no
      yes
```
Decision Tree Construction

- **Process**
  1. Put all data at the root node
  2. Recursively, select an attribute and partition the data-set into subsets as child nodes, until having a stopping condition

- **Stopping Conditions**
  - If all data samples for a given node in the tree belong to the same class
  - If there are no remaining attributes for further partitioning (majority voting is employed for classifying data in the leaf node)
  - There are no data samples left

---

ID3 Algorithm

- **Main Idea**
  - Attribute selection measure during decision tree construction
    - select the attribute with the highest information gain
  - Let $p_i$ be the probability that an arbitrary record in $D$ belongs to class $C_i$
  - Expected information (entropy):
    \[
    Info(D) = -\sum_{i=1}^{n} p_i \log_2 (p_i)
    \]
  - Information after using an attribute $A$ to split $D$ into $v$ partitions
    \[
    Info_A(D) = \sum_{j=1}^{v} \left( \frac{|D_j|}{|D|} \times Info(D_j) \right)
    \]
  - **Information gain** by branching on attribute $A$
    \[
    Gain(A) = Info(D) - Info_A(D)
    \]
Example of Information Gain

- **Information**
  - 9 “yes”es and 5 “no”s, in buy_computer
  - \( \text{Info}(D) = \)

- **Information after Splitting by “age”**
  - \( \text{Info}_{\text{age}}(D) = \)

- **Information Gain by “age”**
  - \( \text{Gain(\text{age})} = \text{Info}(D) - \text{Info}_{\text{age}}(D) = \)

- **Information Gain by other attributes**
  - \( \text{Gain(\text{income})} = \)
  - \( \text{Gain(\text{student})} = \)
  - \( \text{Gain(\text{credit_rating})} = \)

<table>
<thead>
<tr>
<th>age</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>no</td>
</tr>
<tr>
<td>31~40</td>
<td>yes</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>yes</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>no</td>
</tr>
<tr>
<td>31~40</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>yes</td>
</tr>
<tr>
<td>31~40</td>
<td>yes</td>
</tr>
<tr>
<td>31~40</td>
<td>yes</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>no</td>
</tr>
</tbody>
</table>

C4.5 Algorithm

- **Main Idea**
  - An extension of the ID3 algorithm
  - Information gain measure in ID3 is biased towards attributes with a large number of values
  - Uses gain ratio to overcome the problem (normalizing information gain)
    - select the attribute with the highest gain ratio
  - Split information for normalization of information gain
    \[
    \text{SplitInfo}_A(D) = -\sum_{j=1}^n \frac{|D_j|}{|D|} \times \log_2 \left( \frac{|D_j|}{|D|} \right)
    \]
  - \( \text{Gain ratio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}_A(D)} \)
Example of Gain Ratio

- **Split Information by “age”**
  - $\text{SplitInfo}_{\text{age}}(D) = \begin{array}{c|c}
  \text{age} & \text{buys\_computer} \\
  \hline
  \leq 30 & \text{no} \\
  \leq 30 & \text{no} \\
  31\text{~}40 & \text{yes} \\
  > 40 & \text{yes} \\
  > 40 & \text{no} \\
  31\text{~}40 & \text{yes} \\
  \leq 30 & \text{no} \\
  \leq 30 & \text{yes} \\
  > 40 & \text{yes} \\
  \leq 30 & \text{yes} \\
  31\text{~}40 & \text{yes} \\
  > 40 & \text{no}
  \end{array}$

- **Gain Ratio by “age”**
  - $\text{GainRatio(}\text{age}) = \begin{array}{c}
  \leq 30 & \text{no} \\
  \leq 30 & \text{no} \\
  31\text{~}40 & \text{yes} \\
  > 40 & \text{yes} \\
  > 40 & \text{no} \\
  31\text{~}40 & \text{yes} \\
  \leq 30 & \text{no} \\
  \leq 30 & \text{yes} \\
  > 40 & \text{yes} \\
  \leq 30 & \text{yes} \\
  31\text{~}40 & \text{yes} \\
  > 40 & \text{no}
  \end{array}$

- **Gain Ratio by other attributes**
  - $\text{GainRatio(}\text{income}) = \begin{array}{c}
  \leq 30 & \text{no} \\
  \leq 30 & \text{no} \\
  31\text{~}40 & \text{yes} \\
  > 40 & \text{yes} \\
  > 40 & \text{no} \\
  31\text{~}40 & \text{yes} \\
  \leq 30 & \text{no} \\
  \leq 30 & \text{yes} \\
  > 40 & \text{yes} \\
  \leq 30 & \text{yes} \\
  31\text{~}40 & \text{yes} \\
  > 40 & \text{no}
  \end{array}$
  - $\text{GainRatio(}\text{student}) = \begin{array}{c}
  \leq 30 & \text{no} \\
  \leq 30 & \text{no} \\
  31\text{~}40 & \text{yes} \\
  > 40 & \text{yes} \\
  > 40 & \text{no} \\
  31\text{~}40 & \text{yes} \\
  \leq 30 & \text{no} \\
  \leq 30 & \text{yes} \\
  > 40 & \text{yes} \\
  \leq 30 & \text{yes} \\
  31\text{~}40 & \text{yes} \\
  > 40 & \text{no}
  \end{array}$
  - $\text{GainRatio(}\text{credit\_rating}) = \begin{array}{c}
  \leq 30 & \text{no} \\
  \leq 30 & \text{no} \\
  31\text{~}40 & \text{yes} \\
  > 40 & \text{yes} \\
  > 40 & \text{no} \\
  31\text{~}40 & \text{yes} \\
  \leq 30 & \text{no} \\
  \leq 30 & \text{yes} \\
  > 40 & \text{yes} \\
  \leq 30 & \text{yes} \\
  31\text{~}40 & \text{yes} \\
  > 40 & \text{no}
  \end{array}$

CART

- **CART**
  - Classification and Regression Trees

- **Main Idea**
  - Binary decision tree generation and classification
  - Gini index: a measure of inequality
    \[ \text{Gini}\,(D) = 1 - \sum_{j=1}^{m} p_j^2 \]
  - If a data set $D$ is split on the attribute $A$ into two subsets $D_1$ and $D_2$,
    \[ \text{Gini}_A\,(D) = \frac{|D_1|}{|D|}\text{Gini}\,(D_1) + \frac{|D_2|}{|D|}\text{Gini}\,(D_2) \]
  - Reduction in impurity by the binary split on $A$
    \[ \Delta\text{Gini}(A) = \text{Gini}(D) - \text{Gini}_A\,(D) \]
    - select the attribute with the smallest $\Delta\text{Gini}(A)$
Example of Gini Index

- **Gini Index in “buy_computer”**
  - 9 “yes”es and 5 “no”s, in buy_computer
  - \[ Gini(D) = \]

- **Gini Index after Splitting by “age”**
  - Split D on age into “<=30” and “>30”
  - \[ Gini_{age}(D) = \]

- **Gini Index after Splitting by other attributes**
  - \[ Gini_{income}(D) = \]
  - \[ Gini_{student}(D) = \]
  - \[ Gini_{credit\_rating}(D) = \]

Problems of Attribute Selection

- **Information Gain**
  - Biased towards the attributes with a large number of values

- **Gain Ratio**
  - Biased towards the unbalanced splits in which one partition is much larger than the others

- **Gini Index**
  - Biased to multi-valued attributes
  - Has difficulty when the number of classes is large
### Summary of Decision Tree Induction

**Strength**
- Simple and easy to understand classification rules
- Able to use SQL queries to access databases

**Weakness**
- Applications to continuous attributes – partition the continuous attribute values into a discrete set of intervals
- Overfitting
- Limitation of scalability – restriction of the training data size
  - Scalable algorithms: SLIQ, SPRINT, RainForest

### Overfitting

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

**Two Approaches to Avoid Overfitting**
- Prepruning: Halt tree construction early
  - Stop splitting a node if the result is 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
  - Inefficient
RainForest

Main Idea
- Create AVC-set / AVC-group, which fit in memory, by scanning database

AVC (Attribute-Value, Class-label)
- AVC-set of a attribute X is the projection of the training dataset on X and class labels where counts of individual class labels are aggregated
- AVC-group of a node n is the set of AVC-sets of all predictor attributes at n

Reference

<table>
<thead>
<tr>
<th>age</th>
<th>buy_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>3</td>
</tr>
<tr>
<td>31..40</td>
<td>4</td>
</tr>
<tr>
<td>&gt;40</td>
<td>3</td>
</tr>
</tbody>
</table>

CSI 4352, Introduction to Data Mining

Chapters 8 and 9, Classification

- Decision Tree Induction
  - Bayesian Classification
- k-Nearest Neighbor Learning
- Rule-Based Classification
- Pattern-Based Classification
- Classification Accuracy Measures
Bayesian Classification

- **Main Idea**
  - A statistical classifier: performs probabilistic prediction (i.e., outputs the probability of class membership)
  - Based on the Bayesian Theorem
  - Assumes that the effect of an attribute value on a given class is independent of the values of the other attributes

Bayesian Theorem (1)

- **Components**
  - Let $X$ be a sample data ("evidence"): class label is unknown
  - Let $H$ be a hypothesis that $X$ belongs to class $C$
  - Classification is to determine $P(H|X)$ (called **posterior probability**), the probability that the hypothesis holds given the observed data $X$
  - $P(H)$ (called **prior probability**), the initial probability
    - e.g., $X$ will buy computer regardless of age, income, ...
  - $P(X)$, probability that the sample data is observed
  - $P(X|H)$ (called **likelihood**), 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 old with medium income
Bayesian Theorem (2)

- **Formula**
  - Given training data $X$, posteriori probability of a hypothesis $H$, $P(H|X)$, follows the Bayes’ theorem,
    \[
    P(H|X) = \frac{P(X|H)P(H)}{P(X)}
    \]
    (posterior = likelihood $\times$ prior / 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 $k$ classes
  - Practical difficulty: requires initial knowledge of many probabilities, significant computational cost

Naïve Bayesian Classifier (1)

- **Main Idea**
  - Let $D$ be a training set of data objects and their associated class labels, and each object is represented by an n-D attribute vector $X=(x_1,x_2,...,x_n)$
  - Suppose there are $m$ classes $C_1$, $C_2$, ..., $C_m$
  - Classification is to derive the maximum posterior probability, $P(C_i|X)$
    \[
    P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}
    \]
  - Suppose $P(X)$ is constant for all classes, maximize
    \[
    P(C_i|X) = P(X|C_i)P(C_i)
    \]
Naïve Bayesian Classifier (2)

- **Assumption**
  - Attributes are conditionally independent, i.e., no dependence relationship between attributes
  - \[ P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) = P(x_1 \mid C_i) \times P(x_2 \mid C_i) \times \ldots \times P(x_n \mid C_i) \]
  - Reduces the computation cost
  - If \( x_k \) is categorical, \( P(x_k \mid C_i) \) is the number of objects in \( C_i \) having value \( x_k \) divided by the number of objects of \( C_i \)
  - If \( x_k \) is continuous-valued, \( P(x_k \mid C_i) \) is usually computed based on Gaussian distribution with a mean \( \mu \) and standard deviation \( \sigma \)
  - \[ P(X \mid C_i) = g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]

**Example of Training Data**

- **Training Data Set**

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rate</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>
Bayesian Classification Results

- \( X = (\text{age} \leq 30, \text{income}=\text{medium}, \text{student}=\text{yes}, \text{credit}\_\text{rating}=\text{fair}) \)

- \( P(C_i) \)
  - \( P(\text{buys\_computer} = "\text{yes}" ) = \)
  - \( P(\text{buys\_computer} = "\text{no}" ) = \)

- \( P(X|C_i) \)
  - \( P(\text{age} = "\leq 30" \mid \text{buys\_computer} = "\text{yes}" ) = \)
  - \( P(\text{income} = "\text{medium}" \mid \text{buys\_computer} = "\text{yes}" ) = \)
  - \( P(\text{student} = "\text{yes}" \mid \text{buys\_computer} = "\text{yes}" ) = \)
  - \( P(\text{credit}\_\text{rating} = "\text{fair}" \mid \text{buys\_computer} = "\text{yes}" ) = \)
  - \( P(\text{age} = "\leq 30" \mid \text{buys\_computer} = "\text{no}" ) = \)
  - \( P(\text{income} = "\text{medium}" \mid \text{buys\_computer} = "\text{no}" ) = \)
  - \( P(\text{student} = "\text{yes}" \mid \text{buys\_computer} = "\text{no}" ) = \)
  - \( P(\text{credit}\_\text{rating} = "\text{fair}" \mid \text{buys\_computer} = "\text{no}" ) = \)

- \( P(C_i|X) = P(X|C_i) \times P(C_i) \)
  - \( P(\text{buys\_computer} = "\text{yes}" |X) = \)
  - \( P(\text{buys\_computer} = "\text{no}" |X) = \)

Summary of Naïve Bayesian Classifier

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

- **Weakness**
  - Assumption of conditional independence of attributes
    - Loss of accuracy
  - In practice, dependencies exist between attributes
    - Dealing with dependencies: Bayesian belief networks
Bayesian Belief Networks

- **Main Idea**
  - Represents dependency among attributes by training data in Bayesian networks

- **Bayesian Network**
  - Direct acyclic graph (DAC)

- Conditional probability table

<table>
<thead>
<tr>
<th></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>0.8</td>
<td>0.5</td>
<td>0.7</td>
<td>0.1</td>
</tr>
<tr>
<td>~LC</td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
<td>0.9</td>
</tr>
</tbody>
</table>

---

CSI 4352, Introduction to Data Mining

**Chapters 8 and 9, Classification**

- Decision Tree Induction
- Bayesian Classification
  - k-Nearest Neighbor Learning
- Rule-Based Classification
- Pattern-Based Classification
- Classification Accuracy Measures
**k-Nearest Neighbor Learning (kNN)**

**Main Idea**
- Lazy learning (or, instance-based learning): stores the training data and wait until it is given the data for prediction
  → less time in training but more time in predicting
- All instances (data objects) correspond to points in the n-D space
- The nearest neighbors are defined in terms of a distance function
- The distance function is for numerical or categorical values

**Learning Process**
- Searches the k closest neighbor instances of the unknown instance
- For categorical values, the unknown instance is assigned the most common class among k neighbors
- For numerical values, the unknown instance is assigned the mean of k neighbors

**Distance Functions**

**Numerical Attributes**
- Minkowski distance, \( d = \left( \sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \)
- Euclidean distance when \( p=2 \), and Manhattan distance, when \( p=1 \)

**Binary Attributes**
- If symmetric, \( d = \frac{r + s}{q + r + s + t} \)
- If asymmetric, \( d = \frac{r + s}{q + r + s} \)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>q</td>
<td>r</td>
<td>q+r</td>
</tr>
<tr>
<td>0</td>
<td>s</td>
<td>t</td>
<td>s+t</td>
</tr>
<tr>
<td>sum</td>
<td>q+s</td>
<td>r+t</td>
<td>p</td>
</tr>
</tbody>
</table>

**Categorical Attributes**
- Jaccard coefficient, \( d = \frac{|X \Delta Y|}{|X \cup Y|} = 1 - \frac{|X \cap Y|}{|X \cup Y|} \)
- \( X \Delta Y \): the symmetric difference between \( X \) and \( Y \)
Summary of kNN

➢ **Strength**
  - Robust to noisy data by averaging k neighbors

➢ **Weakness**
  - Distance to neighbors could be dominated by irrelevant attributes
    → Elimination of the least relevant attributes
  - Small k makes sensitive to noise, and large k makes inaccurate
    → Weighting each of the k neighbors according to their distance

---

**Chapters 8 and 9, Classification**

- Decision Tree Induction
- Bayesian Classification
- k-Nearest Neighbor Learning
  ➢ Rule-Based Classification
- Pattern-Based Classification
- Classification Accuracy Measures
Rule-Based Classification

**Main Idea**
- Represents the knowledge in the form of IF-THEN rules
  - e.g., IF age < 30 AND student = yes, THEN buy_computer = yes
  - e.g., IF student = yes AND income = low, THEN buy_computer = no

**Process**
- Training step: generating a set of rules
- Prediction step: classifying a new data by the rules applied
- If more than one rule are triggered, need conflict resolution
  - Attribute size ordering: decreasing order of the number of attributes in the rules
  - Rule-based ordering: decreasing order of rule quality

Rule Extraction from Decision Tree

**Main Idea**
- Each rule can be created by each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction with "AND"
- Rules are mutually exclusive

**Examples**
- IF age = young AND student = yes, THEN buys_computer = yes
- IF age = young AND student = no, THEN buys_computer = no
- IF age = mid-age, THEN buys_computer = yes
- IF age = old AND credit_rating = excellent, THEN buys_computer = yes
- IF age = old AND credit_rating = fair, THEN buys_computer = no
Rule Extraction by Sequential Covering

- **Main Idea**
  - Each rule is learned sequentially

- **Sequential Covering Algorithm**
  1. Learn a rule, and remove the data covered by the rule
  2. Repeat (1) until reaching a termination condition: when there are no more training data, or it does not reach the rule quality threshold
  3. Repeat (1) and (2) for each class

- **Rule Learning**
  - Starts with the most general rule possible, and grows the rule in a general-to-specific manner
  - Adds new attributes into the rule by selecting the one that most improves the rule quality

---

Rule Quality Measures

- **Coverage & Accuracy**
  - $n_{covers} =$ the number of data objects covered by the rule $R$
  - $n_{correct} =$ the number of data objects correctly classified by $R$
  - $\text{coverage}(R) = \frac{n_{covers}}{|D|}$ where $D$ is the training data set
  - $\text{accuracy}(R) = \frac{n_{correct}}{n_{covers}}$

- **FOIL**
  - First Order Inductive Learning (based on the information gain)
  - $pos =$ the number of positive data objects covered by the rule $R$
  - $pos' =$ the number of positive data objects covered by the new rule $R'$
  - $\text{FOIL \_ Gain} = pos' \times \left( \log_2 \frac{pos'}{pos' + \text{neg}} - \log_2 \frac{pos}{pos + \text{neg}} \right)$
CSI 4352, Introduction to Data Mining

Chapters 8 and 9, Classification

- Decision Tree Induction
- Bayesian Classification
- k-Nearest Neighbor Learning
- Rule-Based Classification
  - Pattern-Based Classification
- Classification Accuracy Measures

Pattern-Based Classification

- **Main Idea**
  - Frequent patterns and their corresponding association rules are generated and analyzed for classification
  - Also called associative classification
  - Searches for strong associations between frequent patterns and class labels
  - Each pattern is represented as conjunctions of attribute-value pairs with its support and confidence

- **Methods**
  - CBA (Classification by Association)
  - CMAR (Classification based on Multiple Association Rules)
  - CPAR (Classification based on Predictive Association Rules)
CBA (1)

- **CBA**
  - Classification By Association

- **Main Idea**
  - Mining all possible association rules by their support in the form of 
    
    \[ p_1 \land p_2 \ldots \land p_n \rightarrow \text{A}_{\text{class}} = C \]

  - The right-hand side of the rule is restricted to the class attribute

  - Difference between Association Rule Mining and CBA
    - Association Rule Mining: target is not predetermined
    - CBA: only one predetermined target

  - Building a classifier by arranging the rules according to decreasing precedence of their confidence

CBA (2)

- **Classification Process**
  1. Find all covered CARs from the training data
  2. Classify the test data with the highest confidence CAR
     - If some CARs have the same confidence, use the highest support CAR
     - If some CARs have the same confidence and same support, classify the data with the majority class

- **Reference**
Chapters 8 and 9, Classification

- Decision Tree Induction
- Bayesian Classification
- k-Nearest Neighbor Learning
- Rule-Based Classification
- Pattern-Based Classification
- Classification Accuracy Measures

Evaluation of Classification Methods

- Holdout Method
  - Randomly partitions the given data into a training set and a test set
- Random Sampling
  - Repeats the holdout method k times
  - Estimates the overall accuracy by averaging the accuracy from each round
- k-Fold Cross-Validation
  - Randomly partitions the given data into k mutually exclusive subsets, each approximately equal size
  - Measures accuracy k times using the i-th subset as a test set and the others as a training set
- Leave-One-Out Cross-Validation
  - k-fold cross-validation where k is the total size of data set
  - One sample is left out as a test set for each round
Classifier Accuracy Measures (1)

**Accuracy Measures**

<table>
<thead>
<tr>
<th>Actual class</th>
<th>Predicted class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cᵢ</td>
<td>Cᵢ</td>
</tr>
<tr>
<td>true positive</td>
<td>false negative</td>
</tr>
<tr>
<td>~Cᵢ</td>
<td>true positive</td>
</tr>
<tr>
<td>false positive</td>
<td>true negative</td>
</tr>
</tbody>
</table>

- Sensitivity (true positive rate, recall) = \( \frac{tp}{tp + fn} \)
- Specificity (true negative rate) = \( \frac{tn}{fp + tn} \)
- Positive predictive value (precision) = \( \frac{tp}{tp + fp} \)
- Negative predictive value = \( \frac{tn}{tn + fn} \)
- Accuracy = \( \frac{tp + tn}{total} \) + \( \frac{fp + fn}{total} \)
- Error rate = 1 - accuracy

Classifier Accuracy Measures (2)

**ROC Curve**

- Receiver operating characteristic curve
- A graphic plot of true positive rate (sensitivity) vs. false positive rate (1-specificity)
- A tool to show optimality of a classifier
- The closer to the diagonal line, the less accurate the classifier is
- The area under the ROC curve (AUC) represents the classifier accuracy
Questions?

- Lecture Slides on the Course Website,
  "www ecs baylor edu faculty cho 4352"