Classification and Prediction

- What is classification? What is regression?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian Classification
- Classification based on concepts from association rule mining
- Other Classification Methods
- Prediction
- Classification accuracy
- Summary
Classification vs. Prediction

- **Classification:**
  - predicts categorical class labels
  - 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

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

- **Typical Applications**
  - credit approval
  - target marketing
  - medical diagnosis
  - treatment effectiveness analysis

Why Classification? A motivating application

- **Credit approval**
  - A bank wants to classify its customers based on whether they are expected to pay back their approved loans
  - The history of past customers is used to train the classifier
  - The classifier provides rules, which identify potentially reliable future customers
  - Classification rule:
    - If age = “31...40” and income = high then credit_rating = excellent
  - Future customers
    - Paul: age = 35, income = high ⇒ excellent credit rating
    - John: age = 20, income = medium ⇒ fair credit rating
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: 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 samples is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise overfitting will occur

Classification Process (1): Model Construction

<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>

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

Accuracy = ?

<table>
<thead>
<tr>
<th>NAME</th>
<th>RANK</th>
<th>YEARS</th>
<th>TENURED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom</td>
<td>Assistant Prof</td>
<td>2</td>
<td>no</td>
</tr>
<tr>
<td>Mellisa</td>
<td>Associate Prof</td>
<td>7</td>
<td>no</td>
</tr>
<tr>
<td>George</td>
<td>Professor</td>
<td>5</td>
<td>yes</td>
</tr>
<tr>
<td>Joseph</td>
<td>Assistant Prof</td>
<td>7</td>
<td>yes</td>
</tr>
</tbody>
</table>

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
Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian Classification
- Classification based on concepts from association rule mining
- Other Classification Methods
- Prediction
- Classification accuracy
- Summary

Issues regarding classification and prediction (1): 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
    - numerical attribute income ⇒ categorical {low, medium, high}
    - normalize all numerical attributes to [0,1]
Issues regarding classification and prediction (2): Evaluating Classification Methods

- Predictive **accuracy**
- **Speed**
  - time to construct the model
  - time to use the model
- **Robustness**
  - handling noise and missing values
- **Scalability**
  - efficiency in disk-resident databases
- **Interpretability**:
  - understanding and insight provided by the model
- **Goodness of rules (quality)**
  - decision tree size
  - compactness of classification rules

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- **Classification by decision tree induction**
- Bayesian Classification
- Classification based on concepts from association rule mining
- Other Classification Methods
- Prediction
- Classification accuracy
- Summary
Classification by Decision Tree Induction

- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution

- Decision tree generation consists of two phases
  - **Tree construction**
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - **Tree pruning**
    - Identify and remove branches that reflect noise or outliers

- Use of decision tree: Classifying an unknown sample
  - Test the attribute values of the sample against the decision tree

Training Dataset

This follows an example from Quinlan’s ID3

<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”

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)
  - Samples 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
Algorithm for Decision Tree Induction (pseudocode)

Algorithm GenDecTree(Sample S, Attlist A)
1. create a node N
2. If all samples are of the same class C then label N with C; terminate;
3. If A is empty then label N with the most common class C in S (majority voting); terminate;
4. Select a ∈ A, with the highest information gain; Label N with a;
5. For each value v of a:
   a. Grow a branch from N with condition a=v;
   b. Let S_v be the subset of samples in S with a=v;
   c. If S_v is empty then attach a leaf labeled with the most common class in S;
   d. Else attach the node generated by GenDecTree(S_v, A-a)

Attribute Selection Measure

- **Information gain** (ID3/C4.5)
  - All attributes are assumed to be categorical
  - Can be modified for continuous-valued attributes

- **Gini index** (IBM IntelligentMiner)
  - All attributes are assumed continuous-valued
  - Assume there exist several possible split values for each attribute
  - May need other tools, such as clustering, to get the possible split values
  - Can be modified for categorical attributes
Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Assume there are two classes, \( P \) and \( N \)
  - Let the set of examples \( S \) contain \( p \) elements of class \( P \) and \( n \) elements of class \( N \)
  - The amount of information, needed to decide if an arbitrary example in \( S \) belongs to \( P \) or \( N \) is defined as
    \[
    I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}
    \]

Information Gain in Decision Tree Induction

- Assume that using attribute \( A \) a set \( S \) will be partitioned into sets \( \{S_1, S_2, \ldots, S_v\} \)
  - If \( S_i \) contains \( p_i \) examples of \( P \) and \( n_i \) examples of \( N \), the entropy, or the expected information needed to classify objects in all subtrees \( S_i \) is
    \[
    E(A) = \sum_{i=1}^v \frac{p_i + n_i}{p + n} I(p_i, n_i)
    \]
- The encoding information that would be gained by branching on \( A \)
  \[
  Gain(A) = I(p,n) - E(A)
  \]
Training Dataset

This follows an example from Quinlan’s ID3

<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>

Attribute Selection by Information Gain Computation

- Class P: buys_computer = “yes”
- Class N: buys_computer = “no”
- \( I(p, n) = I(9, 5) = 0.940 \)
- Compute the entropy for \( age \):

\[
E(age) = \frac{5}{14} I(2, 3) + \frac{4}{14} I(4, 0) + \frac{5}{14} I(3, 2) = 0.69
\]

Hence

\[
Gain(age) = I(p, n) - E(age)
\]

Similarly

\[
Gain(income) = 0.029
\]

\[
Gain(student) = 0.151
\]

\[
Gain(credit_rating) = 0.048
\]
Splitting the samples using *age*

Output: A Decision Tree for "*buys_computer*"
**Gini Index (IBM IntelligentMiner)**

- If a data set $T$ contains examples from $n$ classes, gini index, $gini(T)$ is defined as:
  \[
gini(T) = 1 - \sum_{j=1}^{n} p_j^2
\]

  where $p_j$ is the relative frequency of class $j$ in $T$.

- If a data set $T$ is split into two subsets $T_1$ and $T_2$ with sizes $N_1$ and $N_2$ respectively, the gini index of the split data contains examples from $n$ classes, the gini index $gini(T)$ is defined as:
  \[
gini_{\text{split}}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)
\]

- The attribute which provides the smallest $gini_{\text{split}}(T)$ is chosen to split the node (we need to enumerate all possible splitting points for each attribute).

---

**Avoid Overfitting in Classification**

- The generated tree may **overfit** the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Result is in 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”
Approaches to Determine the Final Tree Size

- Separate training (2/3) and testing (1/3) sets
- Use cross validation
- Use all the data for training
  - but apply a statistical test (e.g., chi-square) to estimate whether expanding or pruning a node may improve the entire distribution
- Use minimum description length (MDL) principle:
  - halting growth of the tree when the encoding is minimized

Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
- 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 node holds the class prediction
- Rules are easy for humans to understand

Example

IF age = "<=30" AND student = "no" THEN buys_computer = "no"
IF age = "<=30" AND student = "yes" THEN buys_computer = "yes"
IF age = "31...40" THEN buys_computer = "yes"
IF age = ">40" AND credit_rating = "excellent" THEN buys_computer = "yes"
IF age = ">40" AND credit_rating = "fair" THEN buys_computer = "no"
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
    - E.g., \( A = \{1, \ldots, v\} \) is split to \( A \leq V \) and \( A > V \) for \( v-1 \) positions of \( V \)
- 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
Scalable Decision Tree Induction Methods in Data Mining Studies

- **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)
  - separates the scalability aspects from the criteria that determine the quality of the tree
  - builds an AVC-list (attribute, value, class label)