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
## Motivating Example – Fruit Identification

<table>
<thead>
<tr>
<th>Skin</th>
<th>Color</th>
<th>Size</th>
<th>Flesh</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hairy</td>
<td>Brown</td>
<td>Large</td>
<td>Hard</td>
<td>safe</td>
</tr>
<tr>
<td>Hairy</td>
<td>Green</td>
<td>Large</td>
<td>Hard</td>
<td>Safe</td>
</tr>
<tr>
<td>Smooth</td>
<td>Red</td>
<td>Large</td>
<td>Soft</td>
<td>Dangerous</td>
</tr>
<tr>
<td>Hairy</td>
<td>Green</td>
<td>Large</td>
<td>Soft</td>
<td>Safe</td>
</tr>
<tr>
<td>Smooth</td>
<td>Red</td>
<td>Small</td>
<td>Hard</td>
<td>Dangerous</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
Prediction Problems: Classification vs. Numeric Prediction

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

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

- **Typical applications**
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is
Classification—A Two-Step Process

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

- **Model usage**: for classifying future or unknown objects
  - **Estimate accuracy** of the model
    - The known label of test sample is compared with the classified result from the model
    - **Accuracy** rate is the percentage of test set samples that are correctly classified by the model
    - **Test set** is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to **classify new data**

- **Note**: If *the test set* is used to select models, it is called **validation (test) set**
**Process (1): Model Construction**

**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’
Process (2): Using the Model in Prediction

<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>Merlisa</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
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
Decision Tree Induction: An Example

- Training data set: Buys_computer
- Resulting tree:

<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>
Algorithm for Decision Tree Induction

- ID3 (Iterative Dichotomiser), C4.5, by Quinlan
- CART (Classification and Regression Trees)
- 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 Measures

- Idea: select attribute that partition samples into homogeneous groups
- Measures
  - Information gain (ID3)
  - Gain ratio (C4.5)
  - Gini index (CART)
  - Variance reduction for continuous target variable (CART)
Brief Review of Entropy

- Entropy (Information Theory)
  - A measure of uncertainty associated with a random variable
  - Calculation: For a discrete random variable $Y$ taking $m$ distinct values $\{y_1, \ldots, y_m\}$,
    - $H(Y) = -\sum_{i=1}^{m} p_i \log(p_i)$, where $p_i = P(Y = y_i)$
  - Interpretation:
    - Higher entropy $\Rightarrow$ higher uncertainty
    - Lower entropy $\Rightarrow$ lower uncertainty

- Conditional Entropy
  - $H(Y|X) = \sum_x p(x)H(Y|X = x)$
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 $D$:
  $$\text{Info}(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$
- Information needed (after using $A$ to split $D$ into $v$ partitions) to classify $D$:
  $$\text{Info}_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \text{Info}(D_j)$$
- Information gained by branching on attribute $A$
  $$\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$$
Attribute Selection: Information Gain

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

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

<table>
<thead>
<tr>
<th>age</th>
<th>( p_i )</th>
<th>( n_i )</th>
<th>( I(p_i, n_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>2</td>
<td>3</td>
<td>0.971</td>
</tr>
<tr>
<td>31...40</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&gt;40</td>
<td>3</td>
<td>2</td>
<td>0.971</td>
</tr>
</tbody>
</table>

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

\[
\frac{5}{14} I(2,3)
\]

means “age <=30” has 5 out of 14 samples, with 2 yes’es and 3 no’s. Hence

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

Similarly,

\[
Gain(income) = 0.029
\]

\[
Gain(student) = 0.151
\]

\[
Gain(credit\_rating) = 0.048
\]
Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - \((a_i+a_{i+1})/2\) is the midpoint between the values of \(a_i\) and \(a_{i+1}\)
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying \(A \leq \text{split-point}\), and D2 is the set of tuples in D satisfying \(A > \text{split-point}\)
Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

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

- GainRatio(A) = Gain(A)/SplitInfo(A)

Ex.

\[
SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) - \frac{6}{14} \times \log_2 \left( \frac{6}{14} \right) - \frac{4}{14} \times \log_2 \left( \frac{4}{14} \right) = 1.557
\]

- gain_ratio(income) = 0.029/1.557 = 0.019

- The attribute with the maximum gain ratio is selected as the splitting attribute
Gini Index (CART, IBM IntelligentMiner)

- If a data set \( D \) contains examples from \( n \) classes, gini index, \( gini(D) \) is defined as
  \[
gini(D) = 1 - \sum_{j=1}^{n} p_j^2
\]
  where \( p_j \) is the relative frequency of class \( j \) in \( D \)

- If a data set \( D \) is split on \( A \) into two subsets \( D_1 \) and \( D_2 \), the gini index \( gini(D) \) is defined as
  \[
gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)
\]

- Reduction in Impurity:
  \[
  \Delta gini(A) = gini(D) - gini_A(D)
\]

- The attribute provides the smallest \( gini_{split}(D) \) (or the largest reduction in impurity) is chosen to split the node (need to enumerate all the possible splitting points for each attribute)
**Computation of Gini Index**

- Ex. D has 9 tuples in `bought_computer = “yes”` and 5 in “no”
  \[
  gini(D) = 1 - \left( \frac{9}{14} \right)^2 - \left( \frac{5}{14} \right)^2 = 0.459
  \]

- Suppose the attribute `income` partitions D into 10 in \( D_1 \): \{low, medium\} and 4 in \( D_2 \)
  \[
  gini_{income \in \{low,medium\}}(D) = \left( \frac{10}{14} \right) Gini(D_1) + \left( \frac{4}{14} \right) Gini(D_2)
  = \frac{10}{14} \left( 1 - \left( \frac{7}{10} \right)^2 - \left( \frac{3}{10} \right)^2 \right) + \frac{4}{14} \left( 1 - \left( \frac{2}{4} \right)^2 - \left( \frac{2}{4} \right)^2 \right)
  = 0.443
  = Gini_{income \in \{high\}}(D).
  \]

\( Gini_{\{low,high\}} \) is 0.458; \( Gini_{\{medium,high\}} \) is 0.450. Thus, split on the \{low,medium\} (and \{high\}) since it has the lowest Gini index
Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - **Information gain:**
    - biased towards multivalued attributes
  - **Gain ratio:**
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - **Gini index:**
    - biased to multivalued attributes
    - tends to favor tests that result in equal-sized partitions and purity in both partitions
Other Attribute Selection Measures

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

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies and noises
  - Poor accuracy for unseen sample
- Underfitting: when model is too simple, both training and test errors are large
Tree Pruning

Two approaches to avoid overfitting

- Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
  - Difficult to choose an appropriate threshold
- Postpruning: Remove branches from a “fully grown” tree
  - Use a set of data different from the training data to decide which is the “best pruned tree”
  - Occam's razor: prefers smaller decision trees (simpler theories) over larger ones
- Random forest (ensemble methods discussed later)
Scalable Decision Tree Induction Methods

- 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
  - **SLIQ** (EDBT’96 — Mehta et al.)
    - Builds an index for each attribute and only class list and the current attribute list reside in memory
  - **SPRINT** (VLDB’96 — J. Shafer et al.)
    - Constructs an attribute list data structure
  - **PUBLIC** (VLDB’98 — Rastogi & Shim)
    - Integrates tree splitting and tree pruning: stop growing the tree earlier
  - **RainForest** (VLDB’98 — Gehrke, Ramakrishnan & Ganti)
    - Builds an AVC-list (attribute, value, class label)
  - **BOAT** (PODS’99 — Gehrke, Ganti, Ramakrishnan & Loh)
    - Uses bootstrapping to create several small samples
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
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.
Bayes’ Theorem: Basics

- Total probability Theorem: \[ P(B) = \sum_{i=1}^{M} P(B|A_i)P(A_i) \]
- Bayes’ Theorem: \[ P(H|X) = \frac{P(X|H)P(H)}{P(X)} \]

Informally, this can be viewed as: posteriori = likelihood x prior/evidence

- 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 \( P(H|X) \), (i.e., posteriori probability): 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
- \( P(X|H) \) (likelihood): the probability of observing the sample \( X \), given that the hypothesis holds
  - E.g., Given \( X \) will buy computer, the prob. that \( X \) is age 31..40, medium income
Bayes’ theorem: Cookie Example

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

- Red jar: 10 chocolate + 30 plain
- Yellow jar: 20 chocolate + 20 plain
- Pick a jar, and then pick a cookie
- If it’s a plain cookie, what’s the probability the cookie is picked out of red jar?
Classification Is to Derive the Maximum Posteriori

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

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

needs to be maximized
- Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost
Naïve Bayes Classifier

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

- 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 continuous-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

\[
g(x_k, \mu_{C_i}, \sigma_{C_i})
\]
Naïve Bayes Classifier: Training Dataset

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

Data to be classified:
X = (age <=30, Income = medium, Student = yes, Credit_rating = Fair)
Naïve Bayes Classifier: An Example

- **P(C_i):**  
  \[ \text{P(buys\_computer = “yes”) } = \frac{9}{14} = 0.643 \]  
  \[ \text{P(buys\_computer = “no”) } = \frac{5}{14} = 0.357 \]  

- Compute **P(X|C_i)** for each class
  
  - \[ \text{P(age = “<=30” | buys\_computer = “yes”) } = \frac{2}{9} = 0.222 \]  
  - \[ \text{P(age = “<=30” | buys\_computer = “no”) } = \frac{3}{5} = 0.6 \]  
  - \[ \text{P(income = “medium” | buys\_computer = “yes”) } = \frac{4}{9} = 0.444 \]  
  - \[ \text{P(income = “medium” | buys\_computer = “no”) } = \frac{2}{5} = 0.4 \]  
  - \[ \text{P(student = “yes” | buys\_computer = “yes”) } = \frac{6}{9} = 0.667 \]  
  - \[ \text{P(student = “yes” | buys\_computer = “no”) } = \frac{1}{5} = 0.2 \]  
  - \[ \text{P(credit\_rating = “fair” | buys\_computer = “yes”) } = \frac{6}{9} = 0.667 \]  
  - \[ \text{P(credit\_rating = “fair” | buys\_computer = “no”) } = \frac{2}{5} = 0.4 \]  

- **X** = (age <= 30, income = medium, student = yes, credit\_rating = fair)
  
  \[ \text{P(X|C_i) : } P(X|\text{buys\_computer = “yes”}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044 \]  
  \[ \text{P(X|\text{buys\_computer = “no”}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019} \]  

\[ \text{P(X|C_i)\times P(C_i) : } P(X|\text{buys\_computer = “yes”}) \times P(\text{buys\_computer = “yes”}) = 0.028 \]  
\[ P(X|\text{buys\_computer = “no”}) \times P(\text{buys\_computer = “no”}) = 0.007 \]  

Therefore, **X** belongs to class (“buys\_computer = yes”)
Avoiding the Zero-Probability Problem

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

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

- 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 Bayes Classifier
  - How to deal with these dependencies? Bayesian Belief Networks (Chapter 9)
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
Using IF-THEN Rules for Classification

- Represent the knowledge in the form of IF-THEN rules
  
  R: IF age = youth AND student = yes  THEN buys_computer = yes

  - Rule antecedent/precondition vs. rule consequent

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

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

- Rules are *easier to understand* than large trees
- One rule is created *for each path* from the root to a leaf
- Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive

Example: Rule extraction from our *buys_computer* decision-tree

IF age = young AND student = no  THEN buys_computer = no
IF age = young AND student = yes  THEN buys_computer = yes
IF age = mid-age            THEN buys_computer = yes
IF age = old AND credit_rating = excellent  THEN buys_computer = no
IF age = old AND credit_rating = fair  THEN buys_computer = yes
Rule Induction: Sequential Covering Method

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

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

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

  **else** break

<table>
<thead>
<tr>
<th>Positive examples</th>
<th>Negative examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A3 = 1 &amp;&amp; A1 = 2$</td>
<td>$A3 = 1 &amp;&amp; A1 = 2$</td>
</tr>
<tr>
<td>$&amp;&amp; A8 = 5$</td>
<td>$&amp;&amp; A8 = 5$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How to Learn-One-Rule?

- Start with the most general rule possible: condition = empty
- Adding new attributes by adopting a greedy depth-first strategy
  - Picks the one that most improves the rule quality
- Rule-Quality measures: consider both coverage and accuracy
  - Foil-gain (in FOIL & RIPPER): assesses info_gain by extending condition
    \[
    \text{FOIL\_Gain} = \text{pos}' \times (\log_2 \left( \frac{\text{pos}'}{\text{pos}' + \text{neg}'} \right) - \log_2 \left( \frac{\text{pos}}{\text{pos} + \text{neg}} \right))
    \]
  - favors rules that have high accuracy and cover many positive tuples
- Rule pruning based on an independent set of test tuples
  \[
  \text{FOIL\_Prune}(R) = \frac{\text{pos} - \text{neg}}{\text{pos} + \text{neg}}
  \]

Pos/neg are # of positive/negative tuples covered by R.
If \text{FOIL\_Prune} is higher for the pruned version of R, prune R
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
Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?

- Use **validation test set** of class-labeled tuples instead of training set when assessing accuracy

- Methods for estimating a classifier’s accuracy:
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap

- Comparing classifiers:
  - 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>$C_1$</th>
<th>$\neg C_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$</td>
<td>True Positives (TP)</td>
<td>False Negatives (FN)</td>
</tr>
<tr>
<td>$\neg C_1$</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

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>¬C</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>TP</td>
<td>FN</td>
<td>P</td>
</tr>
<tr>
<td>¬C</td>
<td>FP</td>
<td>TN</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>P’</td>
<td>N’</td>
<td>All</td>
</tr>
</tbody>
</table>

- **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
  \[
  \text{Error rate} = \frac{FP + FN}{\text{All}}
  \]
Classifier Evaluation Metrics: Sensitivity and Specificity

<table>
<thead>
<tr>
<th>A/P</th>
<th>C</th>
<th>¬C</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>¬C</td>
<td>FP</td>
<td>TN</td>
</tr>
<tr>
<td>P’</td>
<td>N’</td>
<td>All</td>
</tr>
</tbody>
</table>

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

- **Sensitivity:** True Positive recognition rate
  - Sensitivity = TP/P

- **Specificity:** True Negative recognition rate
  - Specificity = TN/N
Classifier Evaluation Metrics: 
Precision and Recall, and F-measures

- **Precision**: positive predictive value (exactness)
  
  \[
  \text{precision} = \frac{TP}{TP + FP}
  \]

- **Recall (sensitivity)**: true positive recognition rate (completeness)
  
  \[
  \text{recall} = \frac{TP}{TP + FN}
  \]

- **F measure (F₁ or F-score)**: harmonic mean of precision and recall,
  
  \[
  F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
  \]

- **F_β**: weighted measure of precision and recall
  - assigns \( \beta \) times weight to recall as to precision
  
  \[
  F_\beta = \frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}
  \]

<table>
<thead>
<tr>
<th>A \ P</th>
<th>C</th>
<th>¬C</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>¬C</td>
<td>FP</td>
<td>TN</td>
</tr>
<tr>
<td>P'</td>
<td>N'</td>
<td>All</td>
</tr>
</tbody>
</table>
Classifier Evaluation Metrics: Example

- Precision = $\frac{90}{230} = 39.13\%$
- Recall = $\frac{90}{300} = 30.00\%$

<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>
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 given data for 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 the **.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 $\overline{err}(M_2)$
- 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
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
    \[ \overline{err(M_1)} \] and \[ \overline{err(M_2)} \]
  - **t-test** computes **t-statistic** with $k-1$ degrees of freedom:
    \[
t = \frac{\overline{err(M_1)} - \overline{err(M_2)}}{\sqrt{\frac{var(M_1 - M_2)}{k}}}
    \]
    where
    \[
    var(M_1 - M_2) = \frac{1}{k} \sum_{i=1}^{k} \left[ err(M_1)_i - err(M_2)_i - (\overline{err(M_1)} - \overline{err(M_2)}) \right]^2
    \]

- 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 B: t-Distribution Critical Values

<table>
<thead>
<tr>
<th>df</th>
<th>.25</th>
<th>.20</th>
<th>.15</th>
<th>.10</th>
<th>.05</th>
<th>.025</th>
<th>.01</th>
<th>.005</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.000</td>
<td>1.376</td>
<td>1.963</td>
<td>2.329</td>
<td>2.841</td>
<td>3.355</td>
<td>6.314</td>
<td>12.71</td>
</tr>
<tr>
<td>9</td>
<td>10.34</td>
<td>3.586</td>
<td>4.676</td>
<td>4.890</td>
<td>5.204</td>
<td>6.150</td>
<td>6.314</td>
<td>12.71</td>
</tr>
</tbody>
</table>

Confidence level $C$
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 binary classification models
- Shows the trade-off between the true positive rate and the false positive rate
  - Y: true positive rate
  - X: false positive rate
- Perfect classification and line of no-discrimination
- The area under the ROC curve (Area Under Curve, AUC) is a measure of the accuracy of the model
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
- Often improved accuracy
Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy

- How boosting works?
  - **Weights** are assigned to each training tuple
  - A series of k classifiers is iteratively learned
  - After a classifier $M_i$ is learned, the weights are updated to allow the subsequent classifier, $M_{i+1}$, to **pay more attention to the training tuples that were misclassified** by $M_i$
  - The final $M^*$ **combines the votes** of each individual classifier, where the weight of each classifier's vote is a function of its accuracy

- Boosting algorithm can be extended for numeric prediction

- Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data
Adaboost (Freund and Schapire, 1997)

- Given 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: $err(X_j)$ is the misclassification error of tuple $X_j$. Classifier $M_i$ error rate is the sum of the weights of the misclassified tuples:
  \[
  error(M_i) = \sum_{j=1}^{d} w_j \times err(X_j)
  \]
- The weight of classifier $M_i$’s vote is
  \[
  \log \frac{1-error(M_i)}{error(M_i)}
  \]
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
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
Classification is a form of data analysis that extracts models describing important data classes.

Effective and scalable methods have been developed for decision tree induction, Naive Bayesian classification, rule-based classification, and many other classification methods.

Evaluation metrics include: accuracy, sensitivity, specificity, precision, recall, $F$ measure, and $F_\beta$ measure.

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

- Significance tests and ROC curves are useful for model selection.
- There have been numerous comparisons of the different classification methods; the matter remains a research topic.
- No single method has been found to be superior over all others for all data sets.
- Issues such as accuracy, training time, robustness, scalability, and interpretability must be considered and can involve trade-offs, further complicating the quest for an overall superior method.
References (1)

- C. M. Bishop, **Neural Networks for Pattern Recognition.** Oxford University Press, 1995
- P. K. Chan and S. J. Stolfo. **Learning arbiter and combiner trees from partitioned data for scaling machine learning.** KDD'95
- H. Cheng, X. Yan, J. Han, and C.-W. Hsu, **Discriminative Frequent Pattern Analysis for Effective Classification**, ICDE'07
- H. Cheng, X. Yan, J. Han, and P. S. Yu, **Direct Discriminative Pattern Mining for Effective Classification**, ICDE'08
- W. Cohen. **Fast effective rule induction.** ICML'95
- G. Cong, K.-L. Tan, A. K. H. Tung, and X. Xu. **Mining top-k covering rule groups for gene expression data.** SIGMOD'05
References (2)

- W. Li, J. Han, and J. Pei, *CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules*, ICDM'01.
References (3)


- J. R. Quinlan. **Bagging, boosting, and c4.5.** AAAI'96.
R. Rastogi and K. Shim. **Public: A decision tree classifier that integrates building and pruning.** VLDB’98.

J. Shafer, R. Agrawal, and M. Mehta. **SPRINT: A scalable parallel classifier for data mining.** VLDB’96.


P. Tan, M. Steinbach, and V. Kumar. **Introduction to Data Mining.** Addison Wesley, 2005.


X. Yin and J. Han. **CPAR: Classification based on predictive association rules.** SDM'03

H. Yu, J. Yang, and J. Han. **Classifying large data sets using SVM with hierarchical clusters.** KDD'03.