Roadmap

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

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
**Bayes Theorem**

- Given a hypothesis $H$ and data $X$ which bears on the hypothesis:
  \[ P(H | X) = \frac{P(X | H)P(H)}{P(X)} \]

- $P(H)$: independent probability of $H$: *prior probability*
- $P(X)$: independent probability of $X$
- $P(X | H)$: conditional probability of $X$ given $H$: *likelihood*
- $P(H | X)$: conditional probability of $H$ given $X$: *posterior probability*

---

**Bayes Theorem: Basics**

- Let $X$ be a data sample ("evidence"): class label is unknown
- Let $H$ be a hypothesis that $X$ belongs to class $C$
- $P(H)$ (*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 that $X$ will buy computer, the prob. that $X$ is 31..40, medium income
- Classification is to determine the max $P(H | X)$ (*posteriori probability*), the probability that the hypothesis holds given the observed data sample $X$, over all the possible $H$ (over all class labels)
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 $\mathbf{X} = (x_1, x_2, \ldots, x_n)$
  - $x_k$ is the value of the $k$-th attribute ($A_k$) of data tuple $\mathbf{X}$

- Suppose there are $m$ classes $C_1, C_2, \ldots, C_m$.
- 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 needs to be maximized

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

Derivation of Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

\[
P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)
\]

- 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 continuous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean $\mu$ and standard deviation $\sigma$

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

thus, $P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$
Naïve Bayesian Classifier: Training Dataset

<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>

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

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

Naïve Bayesian Classifier: An Example

- P(C_1): P(buys_computer = “yes”) = 9/14 = 0.643
  P(buys_computer = “no”) = 5/14 = 0.357

- Compute P(X|C) 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_1): 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_1)*P(C_1): 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”)
Implementation Details

- We want to find the class, \( i \), that maximizes the following probability:

\[
P(C_i | X) = P(X | C_i) P(C_i)
\]

where

\[
P(X | C_i) = \prod_{k=1}^{n} P(x_k | C_i) = P(x_1 | C_i) \times P(x_2 | C_i) \times \ldots \times P(x_n | C_i)
\]

- what happens when we multiply all those probabilities?
  - each one of these numbers is between 0 and 1
  - possible underflow!
Implementation Details

- We want to find the class, $i$, that maximizes the following probability:

$$P(C_i | X) = P(X | C_i) P(C_i),$$

where

$$P(X | C_i) = \prod_{k=1}^{n} P(x_k | C_i) = P(x_1 | C_i) \times P(x_2 | C_i) \times \ldots \times P(x_n | C_i)$$

- what happens when we multiply all those probabilities?
  - each one of these numbers is between 0 and 1
  - possible underflow!

- solution
  - first compute the log of each probability
  - then convert product to sumation ( $\log(xy) = \log(x) + \log(y)$ )

Avoiding the 0-Probability Problem

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

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

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

- Use Laplacian correction (or Laplacian estimator)
  - Adding 1 to each case
    - Prob(income = low) = 1/1003
    - Prob(income = medium) = 991/1003
    - Prob(income = high) = 11/1003
  - The “corrected” prob. estimates are close to their “uncorrected” counterparts
Naïve Bayesian Classifier: Comments

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

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

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

Bayesian Belief Networks

- Bayesian belief network allows a *subset* of the variables be 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

The conditional probability table (CPT) for variable LungCancer:

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

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

Bayesian Belief Networks

Derivation of the probability of a particular combination of values of \( X \), from CPT:

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

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
Roadmap

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

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
  \[ n_{\text{covers}} = \text{# of tuples covered by } R \]
  \[ n_{\text{correct}} = \text{# of tuples correctly classified by } R \]
  \[ \text{coverage}(R) = \frac{n_{\text{covers}}}{|D|} \quad (*: \text{training data set}) \]
  \[ \text{accuracy}(R) = \frac{n_{\text{correct}}}{n_{\text{covers}}} \]
- If more than one rule is triggered, need conflict resolution
  - Size ordering: assign the highest priority to the triggering rules that has the “toughest” requirement (i.e., with the most attribute test)
  - Class-based ordering: decreasing order of prevalence or misclassification cost per class
  - Rule-based ordering (decision list): rules are organized into one long priority list, according to some measure of rule quality or by experts
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 = yes
  IF age = young AND credit_rating = fair THEN buys_computer = no

Rule Extraction from the Training Data

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

- Star with the most general rule possible: condition = empty
- Adding new attributes by adopting a greedy depth-first strategy
  - Picks the one that most improves the rule quality
- Rule-Quality measures: consider both coverage and accuracy
  - Foil-gain (in FOIL & RIPPER): assesses info_gain by extending condition
    \[ FOIL\_Gain = \text{pos} \times (\log_2 \frac{\text{pos}}{\text{pos} + \text{neg}} - \log_2 \frac{\text{pos} + \text{neg}}{\text{pos} + \text{neg}}) \]
    It favors rules that have high accuracy and cover many positive tuples
- Rule pruning based on an independent set of test tuples
  \[ FOIL\_Prune(R) = \frac{\text{pos} - \text{neg}}{\text{pos} + \text{neg}} \]
  Pos/neg are # of positive/negative tuples covered by R.
  If \( FOIL\_Prune \) is higher for the pruned version of R, prune R

Roadmap

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary
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 \( \mathbf{x} \) is mapped into variable \( y \) by means of the scalar product and a nonlinear function mapping

\[
\text{For Example} \quad y = \text{sign} \left( \sum_{i=0}^{n} w_i x_i + \mu_k \right)
\]

A Multi-Layer Feed-Forward Neural Network

\[
\begin{align*}
\text{Err}_j &= O_j (1 - O_j) \sum_k \text{Err}_k w_{jk} \\
\theta_j &= \theta_j + (1) \text{Err}_j \\
w_{ij} &= w_{ij} + (1) \text{Err}_j O_i \\
\text{Err}_j &= O_j (1 - O_j) (T_j - O_j)
\end{align*}
\]

\[
\begin{align*}
O_j &= \frac{1}{1 + e^{-T_j}} \\
T_j &= \sum_i w_{ij} O_i + \theta_j
\end{align*}
\]
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 # 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.
Roadmap

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

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

Linear Classifiers

\[ f(x, w, b) = \text{sign}(w \cdot x + b) \]

- \( \alpha \)
- \( x \)
- \( y \text{est} \)
- \( w \cdot x + b > 0 \)
- \( w \cdot x + b < 0 \)
- \( w \cdot x + b = 0 \)

How would you classify this data?

How would you classify this data?
Linear Classifiers

\[ f(x, w, b) = \text{sign}(w \cdot x + b) \]

• denotes +1
• denotes -1

How would you classify this data?
Linear Classifiers

\[ f(x, w, b) = \text{sign}(w \cdot x + b) \]

- denotes +1
- denotes -1

Any of these would be fine...

..but which is best?

How would you classify this data?

Misclassified to +1 class
Define the **margin** of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

**Maximum Margin**

1. Maximizing the margin is good.
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.

**Support Vectors** are those datapoints that the margin pushes up against.

This is the simplest kind of SVM (Called an LSVM)
Linear SVM Mathematically

- What we know:
  - $w \cdot x^+ + b = +1 \ (1)$
  - $w \cdot x + b = -1 \ (2)$
  - $x^+ = x + kw \ (3)$
  - $M = |x^+ - x| \ (4)$

from (1) and (3):

$w (x + kw) + b = +1 \rightarrow$

$w \cdot x + b + kww = +1 \rightarrow$

using (2):

$-1 + kww = +1 \rightarrow$

$k = 2/ww$
Linear SVM Mathematically

- What we know:
  - \( w \cdot x^+ + b = +1 \) (1)
  - \( w \cdot x^- + b = -1 \) (2)
  - \( x^+ = x^- + kw \) (3)
  - \( M = |x^+ - x^-| \) (4)
  - \( k = \frac{2}{w'w} \) (5)

From (4), and (3):
\[
M = \frac{2}{\sqrt{ww'}}
\]

Goal: 1) Correctly classify all training data

\[
\begin{align*}
wx_i + b & \geq 1 \quad \text{if } y_i = +1 \\
wx_i + b & \leq 1 \quad \text{if } y_i = -1 \\
y_i(wx_i + b) & \geq 1 \quad \text{for all } i
\end{align*}
\]

2) Maximize the Margin

\[
M = \frac{2}{\|w\|}
\]

same as minimize
\[
\frac{1}{2} w'w
\]

We can formulate a Quadratic Optimization Problem and solve for \( w \) and \( b \)

Minimize \( \Phi(w) = \frac{1}{2} w'w \)

subject to \( y_i(wx_i + b) \geq 1 \quad \forall i \)
Solving the Optimization Problem

Find $w$ and $b$ such that
\[ \Phi(w) = \frac{1}{2} w^T w \] is minimized;
and for all \( \{ (x_i, y_i) \} \): \( y_i (w^T x_i + b) \geq 1 \)

- Need to optimize a quadratic function subject to linear constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.
- The solution involves constructing a dual problem where a Lagrange multiplier $\alpha_i$ is associated with every constraint in the primary problem:

\[
\text{Find } \alpha_1, \ldots, \alpha_N \text{ such that }
\begin{align*}
Q(\alpha) &= \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j x_i^T x_j \\
(1) \sum \alpha_i y_i &= 0 \\
(2) \alpha_i &\geq 0 \text{ for all } \alpha_i
\end{align*}
\]

The Optimization Problem Solution

- The solution has the form:

\[
w = \sum \alpha_i y_i x_i \\
b = y_k - w^T x_k \text{ for any } x_k \text{ such that } \alpha_k \neq 0
\]

- Each non-zero $\alpha_i$ indicates that corresponding $x_i$ is a support vector.
- Then the classifying function will have the form:

\[
f(x) = \sum \alpha_i y_i x_i^T x + b
\]

- Notice that it relies on an inner product between the test point $x$ and the support vectors $x_i$—we will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $x_i^T x_j$ between all pairs of training points.
Hard Margin: So far we require all data points be classified correctly
- No training error

What if the training set is noisy?
- Solution 1: use very powerful kernels

Soft Margin Classification

Slack variables $\xi_i$ can be added to allow misclassification of difficult or noisy examples.

What should our quadratic optimization criterion be?

Minimize

$$\frac{1}{2} w . w + C \sum_{k=1}^{R} \xi_k$$
Hard Margin v.s. Soft Margin

- The old formulation:

Find $w$ and $b$ such that:

$\Phi(w) = \frac{1}{2} w^T w$ is minimized and for all $\{(x_i, y_i)\}$

$y_i (w^T x_i + b) \geq 1$

- The new formulation incorporating slack variables:

Find $w$ and $b$ such that:

$\Phi(w) = \frac{1}{2} w^T w + C \sum \xi_i$ is minimized and for all $\{(x_i, y_i)\}$

$y_i (w^T x_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all $i$

- Parameter $C$ can be viewed as a way to control overfitting.

Linear SVMs: Overview

- The classifier is a *separating hyperplane*.
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points $x_i$ are support vectors with non-zero Lagrangian multipliers $\alpha_i$.
- Both in the dual formulation of the problem and in the solution training points appear only inside dot products:

Find $\alpha_1…\alpha_N$ such that

$Q(\alpha) = \Sigma \alpha_i - \frac{1}{2} \Sigma \Sigma \alpha_i \alpha_j y_i y_j x_i^T x_j$ is maximized and

(1) $\Sigma \alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all $\alpha_i$

$f(x) = \Sigma \alpha_i y_i x_i^T x + b$
Non-linear SVMs

- Datasets that are linearly separable with some noise work out great:

But what are we going to do if the dataset is just too hard?
Non-linear SVMs

- Datasets that are linearly separable with some noise work out great:

- But what are we going to do if the dataset is just too hard?

- How about… mapping data to a higher-dimensional space?

Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:
Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:

$$\Phi: x \rightarrow \phi(x)$$

Examples of Kernel Functions

- Linear: $$K(x_i, x_j) = x_i^T x_j$$
- Polynomial of power $$\rho$$: $$K(x_i, x_j) = (1 + x_i^T x_j)^\rho$$
- Gaussian (radial-basis function network):
  $$K(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right)$$
- Sigmoid: $$K(x_i, x_j) = \tanh(\beta_0 x_i^T x_j + \beta_1)$$
Non-linear SVMs Mathematically

- Dual problem formulation:

Find \( \alpha_1 \ldots \alpha_N \) such that

\[
Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j K(x_i, x_j)
\]

is maximized and

1. \( \sum \alpha_i = 0 \)
2. \( \alpha_i \geq 0 \) for all \( \alpha_i \)

- The solution is:

\[
f(x) = \sum \alpha_i y_i K(x_i, x) + b
\]

- Optimization techniques for finding \( \alpha_i \)'s remain the same!

Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space

- It does not need to represent the space explicitly, simply by defining a kernel function

- The kernel function plays the role of the dot product in the feature space.
Properties of SVM

- Flexibility in choosing a similarity function
- Sparseness of solution when dealing with large data sets
  - only support vectors are used to specify the separating hyperplane
- Ability to handle large feature spaces
  - complexity does not depend on the dimensionality of the feature space
- Overfitting can be controlled by soft margin approach
- Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution
- Feature Selection

Weakness of SVM

- It is sensitive to noise
  - A relatively small number of mislabeled examples can dramatically decrease the performance
- It only considers two classes
  - how to do multi-class classification with SVM?
    - Answer:
      1) with output arity m, learn m SVM’s
        - SVM 1 learns “Output==1” vs “Output != 1”
        - SVM 2 learns “Output==2” vs “Output != 2”
        - ...
        - SVM m learns “Output==m” vs “Output != m”
      2)To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.
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.

Scaling SVM by Hierarchical Micro-Clustering

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

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

CF-Tree: Hierarchical Micro-cluster
CB-SVM Algorithm: Outline

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

Selective Declustering

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

- “Statistical Learning Theory” by Vapnik: extremely hard to understand, containing many errors too.
  - Better than the Vapnik’s book, but still written too hard for introduction, and the examples are so not-intuitive
- The book “An Introduction to Support Vector Machines” by N. Cristianini and J. Shawe-Taylor
  - Also written hard for introduction, but the explanation about the mercer’s theorem is better than above literatures
- The neural network book by Haykins
  - Contains one nice chapter of SVM introduction

Additional Resources

- An excellent tutorial on VC-dimension and Support Vector Machines:

- The VC/SRM/SVM Bible:

http://www.kernel-machines.org/
### Roadmap

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

### Classifier Accuracy Measures

- Accuracy of a classifier $M$, $\text{acc}(M)$: percentage of test-set tuples that are correctly classified by the model $M$
  - Error rate (misclassification rate) of $M = 1 - \text{acc}(M)$
  - Given $m$ classes, $CM_{i,j}$ an entry in a [confusion matrix](https://en.wikipedia.org/wiki/Confusion_matrix), indicates # of tuples in class $i$ that are labeled by the classifier as class $j$

<table>
<thead>
<tr>
<th>Actual class</th>
<th>Predicted class</th>
<th>$\text{buy_computer = yes}$</th>
<th>$\text{buy_computer = no}$</th>
<th>Total</th>
<th>Acc(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{buy_computer = yes}$</td>
<td>6954</td>
<td>46</td>
<td>7000</td>
<td>99.34</td>
<td></td>
</tr>
<tr>
<td>$\text{buy_computer = no}$</td>
<td>412</td>
<td>2588</td>
<td>3000</td>
<td>86.27</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>7366</td>
<td>2634</td>
<td>10000</td>
<td>95.52</td>
<td></td>
</tr>
</tbody>
</table>
Classifier Accuracy Measures

- Alternative accuracy measures (e.g., for cancer diagnosis)
  
  sensitivity = \frac{t\text{-}pos}{pos} /* true positive recognition rate */
  
  specificity = \frac{t\text{-}neg}{neg} /* true negative recognition rate */
  
  precision = \frac{t\text{-}pos}{t\text{-}pos + f\text{-}pos}
  
  accuracy = \frac{sensitivity \cdot pos}{pos + neg} + \frac{specificity \cdot neg}{pos + neg}

- This model can also be used for cost-benefit analysis

<table>
<thead>
<tr>
<th>actual class</th>
<th>predicted class</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>True positive</td>
</tr>
<tr>
<td>C2</td>
<td>False negative</td>
</tr>
<tr>
<td>C2</td>
<td>False positive</td>
</tr>
<tr>
<td>C1</td>
<td>True negative</td>
</tr>
</tbody>
</table>

Evaluating the Accuracy of a Classifier or Predictor (I)

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

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

- **Bootstrap**
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
  - Several bootstrap methods, and a common one is **.632 bootstrap**
    - Suppose we are given a data set of *d* tuples. The data set is sampled *d* times, with replacement, resulting in a training set of *d* samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data will end up in the bootstrap, and the remaining 36.8% will form the test set. (Prob(not select tuple *t*) = 1-1/*d*, for a sample of size *d*: (1 – 1/*d*)^*d* ≈ e^-1 = 0.368)
    - Repeat the sampling procedure *k* times, overall accuracy of the model:
      
      \[
      acc(M) = \sum_{i=1}^{k} (0.632 \times acc(M_i)_{train} + 0.368 \times acc(M_i)_{test})
      \]

**Roadmap**

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

- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, $M_1, M_2, \ldots, M_k$, with the aim of creating an improved model $M^*$
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

Bagging: Bootstrap Aggregation

- Analogy: Diagnosis based on multiple doctors’ majority vote
- Training
  - Given a set $D$ of $d$ tuples, at each iteration $i$, a training set $D_i$ of $d$ tuples is sampled with replacement from $D$ (i.e., bootstrap)
  - A classifier model $M_i$ is learned for each training set $D_i$
- Classification: classify an unknown sample $X$
  - Each classifier $M_i$ returns its class prediction
  - The bagged classifier $M^*$ counts the votes and assigns the class with the most votes to $X$
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significant better than a single classifier derived from $D$
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction
Boosting

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

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

- The boosting algorithm can be extended for the prediction of continuous values

- Comparing with bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data

Adaboost (Freund and Schapire, 1997)

- Given a set of $d$ class-labeled tuples, $(X_1, y_1), \ldots, (X_d, y_d)$
- Initially, all the weights of tuples are set the same $(1/d)$
- Generate $k$ classifiers in $k$ rounds. At round $i$,
  - Tuples from $D$ are sampled (with replacement) to form a training set $D_i$ of the same size
  - Each tuple’s chance of being selected is based on its weight
  - A classification model $M_i$ is derived from $D_i$
  - Its error rate is calculated using $D_i$ as a test set
  - If a tuple is misclassified, its weight is increased, o.w. it is decreased

- Error rate: $\text{err}(X_i)$ is the misclassification error of tuple $X_i$. 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
  \[
  \log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)}
  \]
Roadmap

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

Model Selection: ROC Curves

- ROC (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model

- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0
**Roadmap**

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

**Summary (I)**

- **Classification** and **prediction** are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends.
- Effective and scalable methods have been developed for **decision trees induction**, Naive Bayesian classification, Bayesian belief network, rule-based classifier, Backpropagation, Support Vector Machine (SVM), associative classification, nearest neighbor classifiers, and **case-based reasoning**, and other classification methods such as genetic algorithms, rough set and fuzzy set approaches.
- **Linear, nonlinear, and generalized linear models of regression** can be used for **prediction**. Many nonlinear problems can be converted to linear problems by performing transformations on the predictor variables. **Regression trees** and **model trees** are also used for prediction.
Summary (II)

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

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

- There have been numerous **comparisons of the different classification and prediction methods**, and the matter remains a research topic.

- No single method has been found to be superior over all others for all data sets.

- Issues such as accuracy, training time, robustness, interpretability, and scalability must be considered and can involve trade-offs, further complicating the quest for an overall superior method.

References (1)


References (2)

- U. M. Fayyad. Branching on attribute values in decision tree generation. AAAI'94.
- B. Liu, W. Hsu, and Y. Ma. Integrating Classification and Association Rule. KDD'98.
- W. Li, J. Han, and J. Pei, CMAR: Accurate and Efficient Classification Based on Multiple Class-Assocation Rules, ICDM'01.

References (3)

References (4)

- 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.

Data Mining for Knowledge Management 210