

# **Data Mining:**

---

# **Concepts and Techniques**

**(3<sup>rd</sup> ed.)**

## **— Chapter 9 —**

## **Classification: Advanced Methods**


Jiawei Han, Micheline Kamber, and Jian Pei  
University of Illinois at Urbana-Champaign &  
Simon Fraser University

©2010 Han, Kamber & Pei. All rights reserved.



# Chapter 9. Classification: Advanced Methods

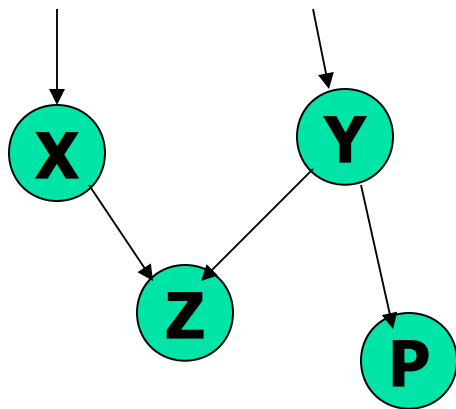
---

- Bayesian Belief Networks 
- Classification by Backpropagation
- Support Vector Machines
- Classification by Using Frequent Patterns
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods
- Additional Topics Regarding Classification
- Summary



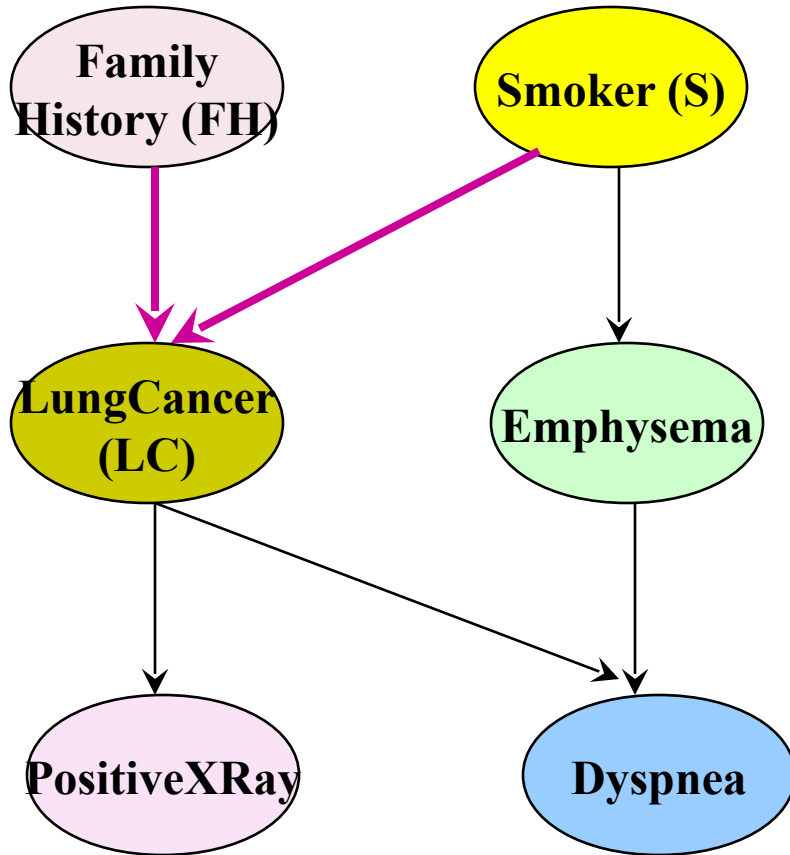
# Bayesian Belief Networks

- **Bayesian belief networks** (also known as **Bayesian networks, probabilistic networks**): allow *class conditional independencies* between *subsets* of variables
- A (*directed acyclic*) 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/cycles

# Bayesian Belief Network: An Example



**CPT: Conditional Probability Table**  
for variable LungCancer:

	(FH, S)	(FH, ~S)	(~FH, S)	(~FH, ~S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of  $\mathbf{X}$ , from CPT:

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i \mid \text{Parents}(Y_i))$$

**Bayesian Belief Network**

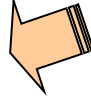
# Training Bayesian Networks: Several Scenarios

---

- Scenario 1: Given both the network structure and all variables observable: *compute only the CPT entries*
- Scenario 2: Network structure known, some variables hidden: *gradient descent* (greedy hill-climbing) method, i.e., search for a solution along the steepest descent of a criterion function
  - Weights are initialized to random probability values
  - At each iteration, it moves towards what appears to be the best solution at the moment, w.o. backtracking
  - Weights are updated at each iteration & converge to local optimum
- Scenario 3: Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
- Scenario 4: Unknown structure, all hidden variables: No good algorithms known for this purpose
- D. Heckerman. [A Tutorial on Learning with Bayesian Networks](#). In *Learning in Graphical Models*, M. Jordan, ed.. MIT Press, 1999.

# Chapter 9. Classification: Advanced Methods

---

- Bayesian Belief Networks
- Classification by Backpropagation 
- Support Vector Machines
- Classification by Using Frequent Patterns
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods
- Additional Topics Regarding Classification
- 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

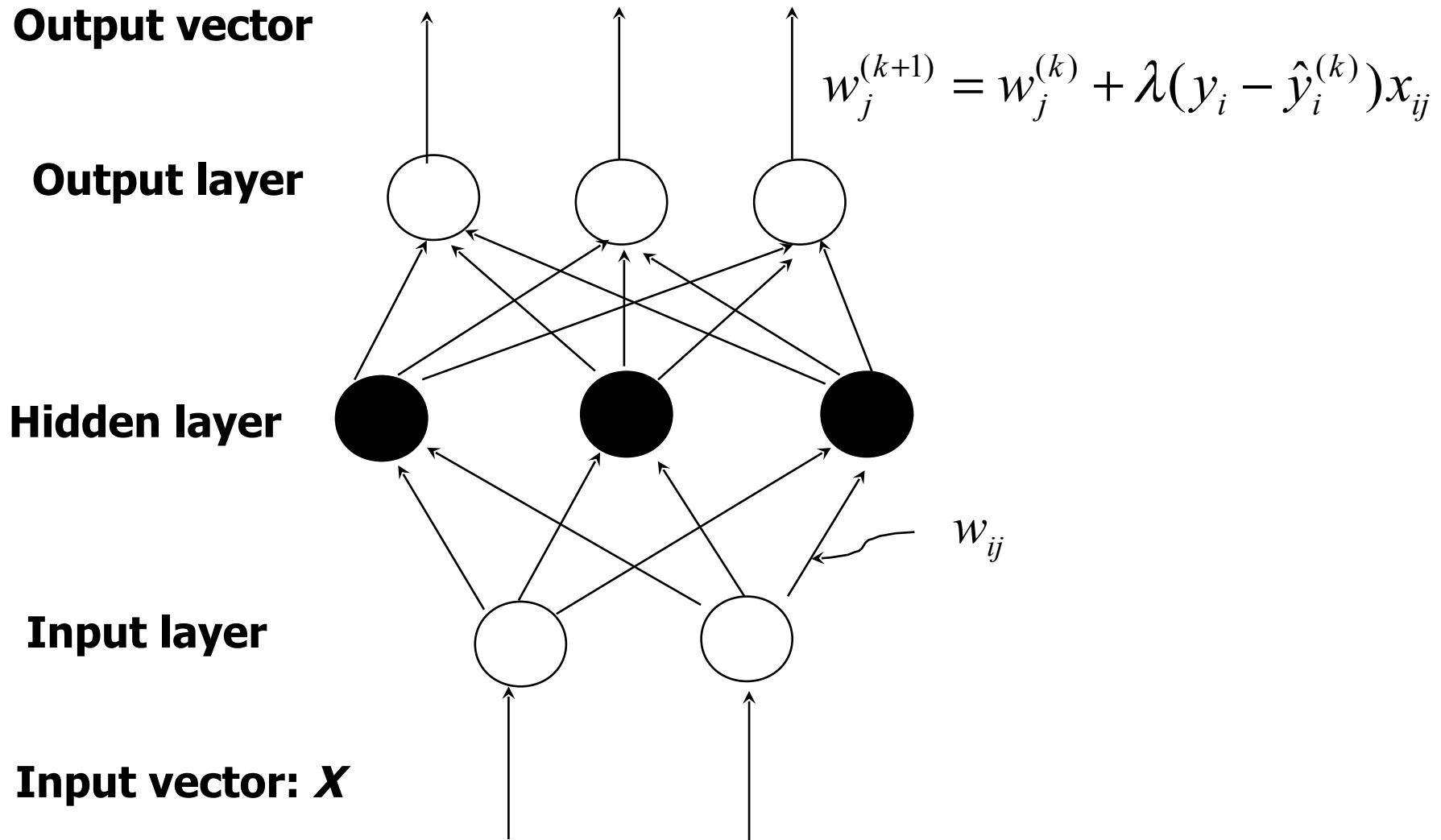


# Neural Network as a Classifier

---

- 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 an array of real-world data, e.g., hand-written letters
  - Algorithms are inherently parallel
  - Techniques have recently been developed for the extraction of rules from trained neural networks

# A Multi-Layer Feed-Forward Neural Network



# 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**: 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

---

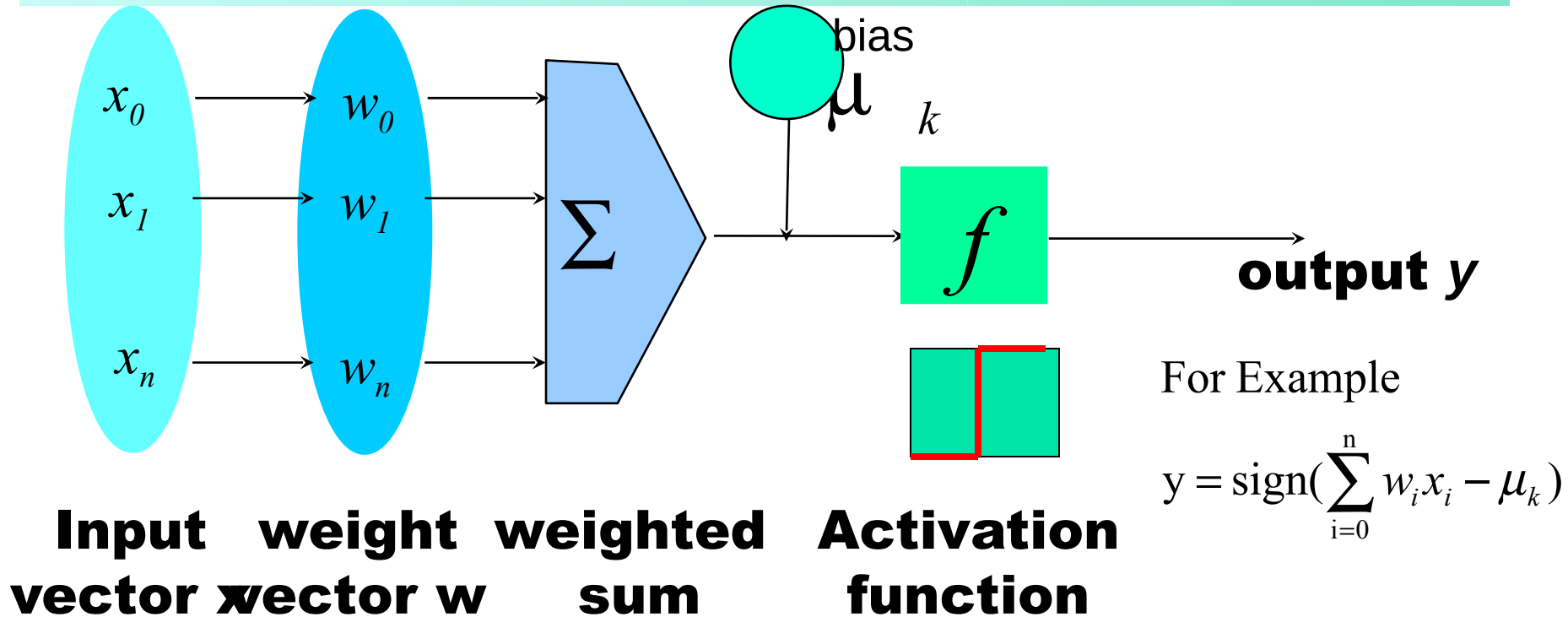
- Decide the **network topology**: Specify # of units in the *input layer*, # of *hidden layers* (if  $> 1$ ), # of units in *each hidden layer*, and # of units in the *output layer*
- Normalize the input values for each attribute measured in the training tuples to [0.0—1.0]
- One **input** unit per 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 numbers, associated with biases
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

# Neuron: A Hidden/Output Layer Unit



- An  $n$ -dimensional input vector  $\mathbf{x}$  is mapped into variable  $y$  by means of the scalar product and a nonlinear function mapping
- The inputs to unit are outputs from the previous layer. They are multiplied by their corresponding weights to form a weighted sum, which is added to the bias associated with unit. Then a nonlinear activation function is applied to it.




# Efficiency and Interpretability

---

- **Efficiency** of backpropagation: Each epoch (one iteration through the training set) takes  $O(|D| * w)$ , with  $|D|$  tuples and  $w$  weights, but # of epochs can be exponential to  $n$ , the number of inputs, in worst case
- For easier comprehension: **Rule extraction** by network pruning
  - Simplify the network structure by removing weighted links that have the least effect on the trained network
  - Then perform link, unit, or activation value clustering
  - The set of input and activation values are studied to derive rules describing the relationship between the input and hidden unit layers
- **Sensitivity analysis**: assess the impact that a given input variable has on a network output. The knowledge gained from this analysis can be represented in rules

# Chapter 9. Classification: Advanced Methods

---

- Bayesian Belief Networks
- Classification by Backpropagation
- Support Vector Machines 
- Classification by Using Frequent Patterns
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods
- Additional Topics Regarding Classification
- Summary

# Classification: A Mathematical Mapping

- **Classification:** predicts categorical class labels

- E.g., Personal homepage classification

- $x_i = (x_1, x_2, x_3, \dots)$ ,  $y_i = +1$  or  $-1$

- $x_1$  : # of word “homepage”

- $x_2$  : # of word “welcome”

- Mathematically,  $x \in X = \mathfrak{R}^n$ ,  $y \in Y = \{+1, -1\}$

- We want to derive a function  $f: X \rightarrow Y$

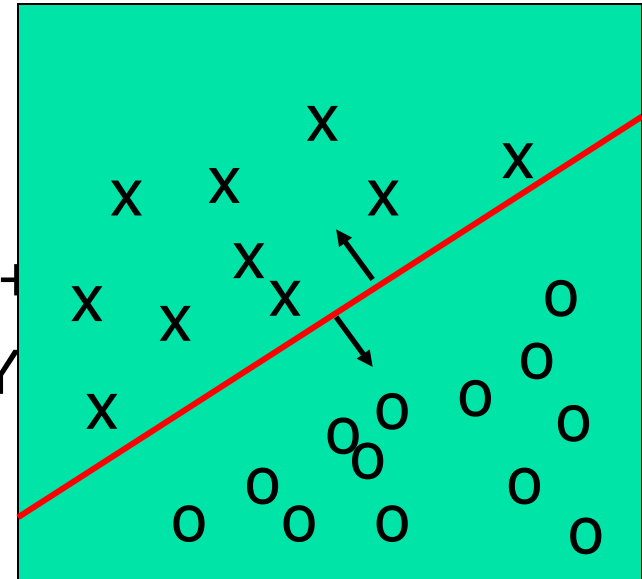
- Linear Classification

- Binary Classification problem

- Data above the red line belongs to class ‘x’

- Data below red line belongs to class ‘o’

- Examples: SVM, Perceptron, Probabilistic Classifiers

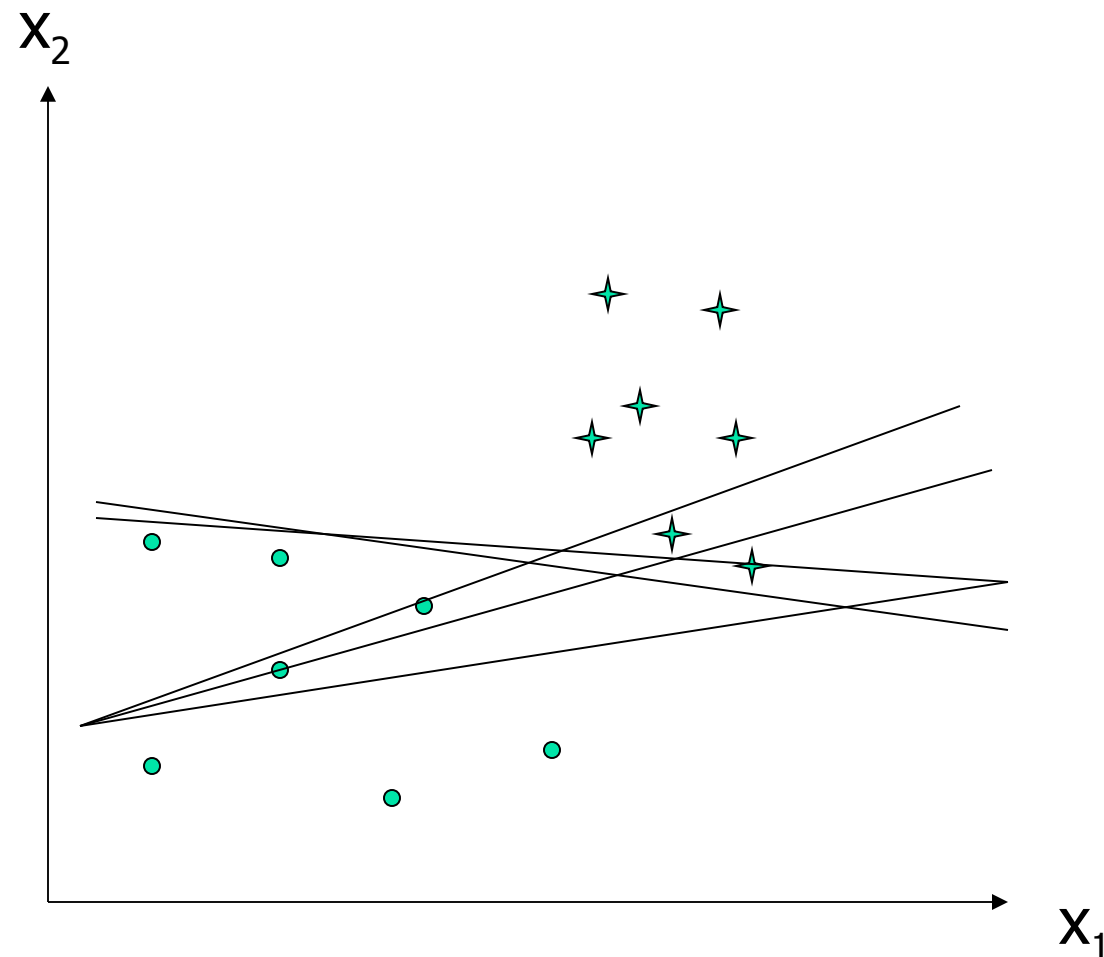


# Discriminative Classifiers

---

- Advantages
  - Prediction accuracy is generally high
    - As compared to Bayesian methods – in general
  - Robust, works when training examples contain errors
  - Fast evaluation of the learned target function
    - Bayesian networks are normally slow
- Criticism
  - Long training time
  - Difficult to understand the learned function (weights)
    - Bayesian networks can be used easily for pattern discovery
  - Not easy to incorporate domain knowledge
    - Easy in the form of priors on the data or distributions

# Perceptron & Winnow



- Vector:  $x, w$

- Scalar:  $x, y, w$

Input:  $\{(x_1, y_1), \dots\}$

Output: classification function  $f(x)$

$$f(x_i) > 0 \text{ for } y_i = +1$$
$$f(x_i) < 0 \text{ for } y_i = -1$$
$$f(x) \Rightarrow wx + b = 0$$
$$\text{or } w_1x_1 + w_2x_2 + b = 0$$

- Perceptron: update  $W$  additively

- Winnow: update  $W$  multiplicatively

# SVM—Support Vector Machines

---

- A relatively 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)

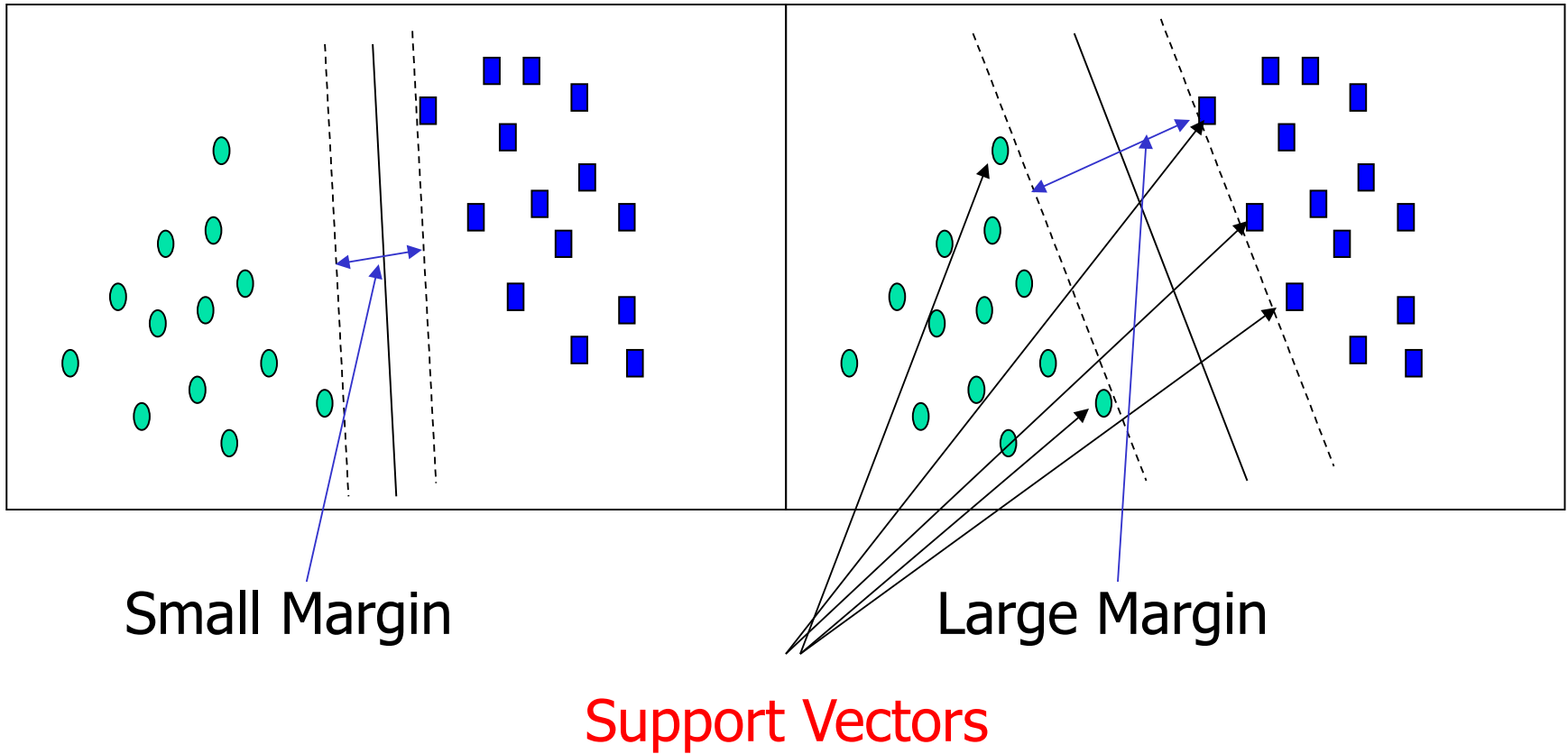


# SVM—History and Applications

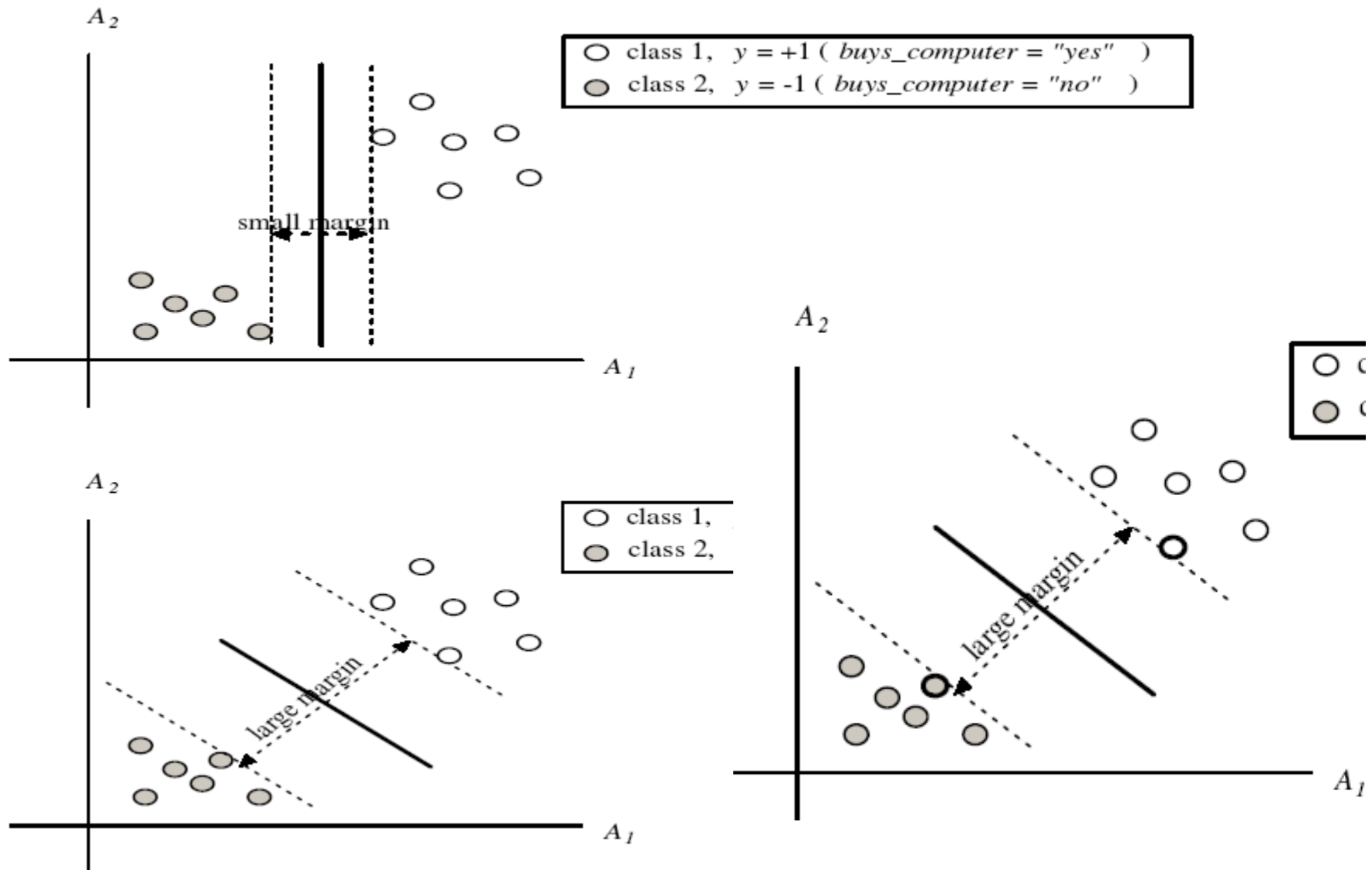
---

- 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 for: classification and numeric prediction
- Applications:
  - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

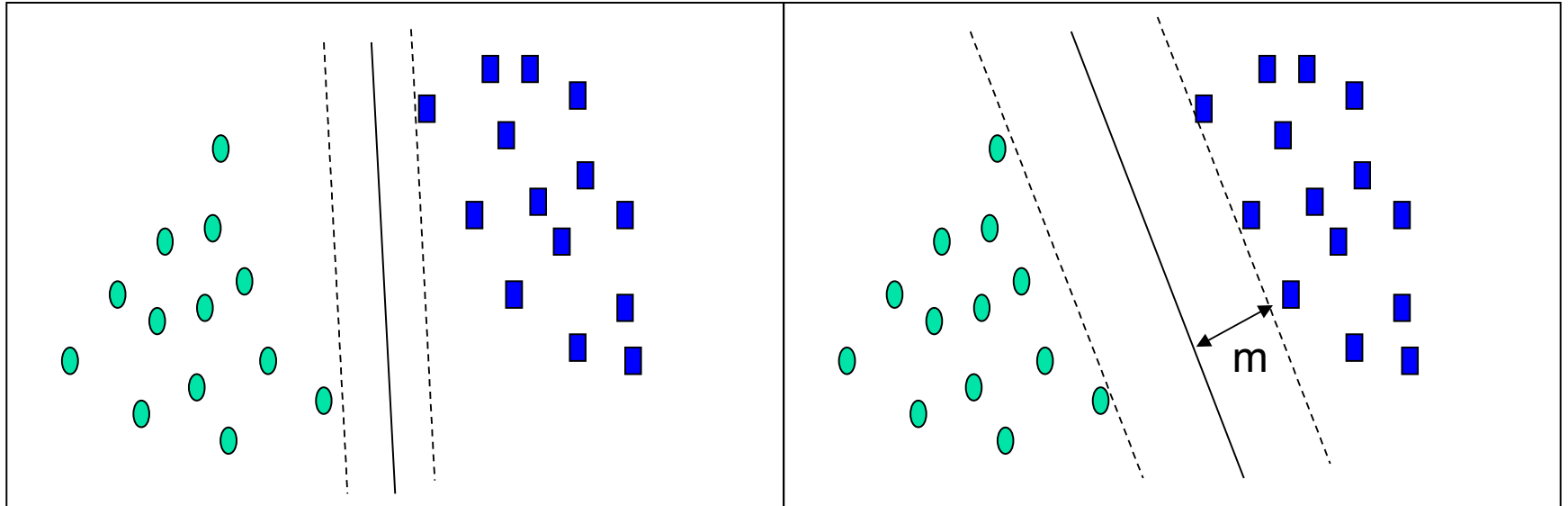
# SVM—General Philosophy



# SVM—Margins and Support Vectors



# SVM—When Data Is Linearly Separable



Let data  $D$  be  $(\mathbf{X}_1, y_1), \dots, (\mathbf{X}_{|D|}, y_{|D|})$ , where  $\mathbf{X}_i$  is the set of training tuples associated with the class labels  $y_i$

There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data)

*SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH)*

# SVM—Linearly Separable

- A separating hyperplane can be written as

$$\mathbf{W} \bullet \mathbf{X} + b = 0$$

where  $\mathbf{W} = \{w_1, w_2, \dots, w_n\}$  is a weight vector and  $b$  a scalar (bias)

- For 2-D it can be written as

$$w_0 + w_1 x_1 + w_2 x_2 = 0$$

- The hyperplane defining the sides of the margin:

$$H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 1 \quad \text{for } y_i = +1, \text{ and}$$

$$H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1 \quad \text{for } y_i = -1$$

- Any training tuples that fall on hyperplanes  $H_1$  or  $H_2$  (i.e., the sides defining the margin) are **support vectors**
- This becomes a **constrained (convex) quadratic optimization** problem: Quadratic objective function and linear constraints  $\rightarrow$  *Quadratic Programming (QP)*  $\rightarrow$  Lagrangian multipliers

# Why Is SVM Effective on High Dimensional Data?

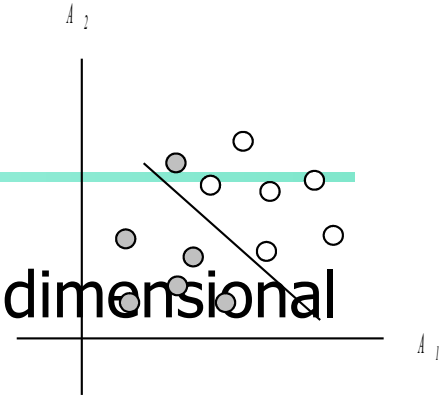
---

- The **complexity** of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The **support vectors** are the essential or critical training examples — they lie closest to the decision boundary (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high



# SVM—Linearly Inseparable

- Transform the original input data into a higher dimensional space



**Example 6.8** Nonlinear transformation of original input data into a higher dimensional space. Consider the following example. A 3D input vector  $\mathbf{X} = (x_1, x_2, x_3)$  is mapped into a 6D space  $Z$  using the mappings  $\phi_1(\mathbf{X}) = x_1, \phi_2(\mathbf{X}) = x_2, \phi_3(\mathbf{X}) = x_3, \phi_4(\mathbf{X}) = (x_1)^2, \phi_5(\mathbf{X}) = x_1x_2$ , and  $\phi_6(\mathbf{X}) = x_1x_3$ . A decision hyperplane in the new space is  $d(\mathbf{Z}) = \mathbf{WZ} + b$ , where  $\mathbf{W}$  and  $\mathbf{Z}$  are vectors. This is linear. We solve for  $\mathbf{W}$  and  $b$  and then substitute back so that we see that the linear decision hyperplane in the new ( $\mathbf{Z}$ ) space corresponds to a nonlinear second order polynomial in the original 3-D input space,

$$\begin{aligned} d(\mathbf{Z}) &= w_1x_1 + w_2x_2 + w_3x_3 + w_4(x_1)^2 + w_5x_1x_2 + w_6x_1x_3 + b \\ &= w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5 + w_6z_6 + b \end{aligned} \quad \blacksquare$$

- Search for a linear separating hyperplane in the new space

# SVM: Different Kernel functions

---

- Instead of computing the dot product on the transformed data, it is math. equivalent to applying a kernel function  $K(\mathbf{X}_i, \mathbf{X}_j)$  to the original data, i.e.,  $K(\mathbf{X}_i, \mathbf{X}_j) = \Phi(\mathbf{X}_i) \cdot \Phi(\mathbf{X}_j)$
- Typical Kernel Functions

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

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

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

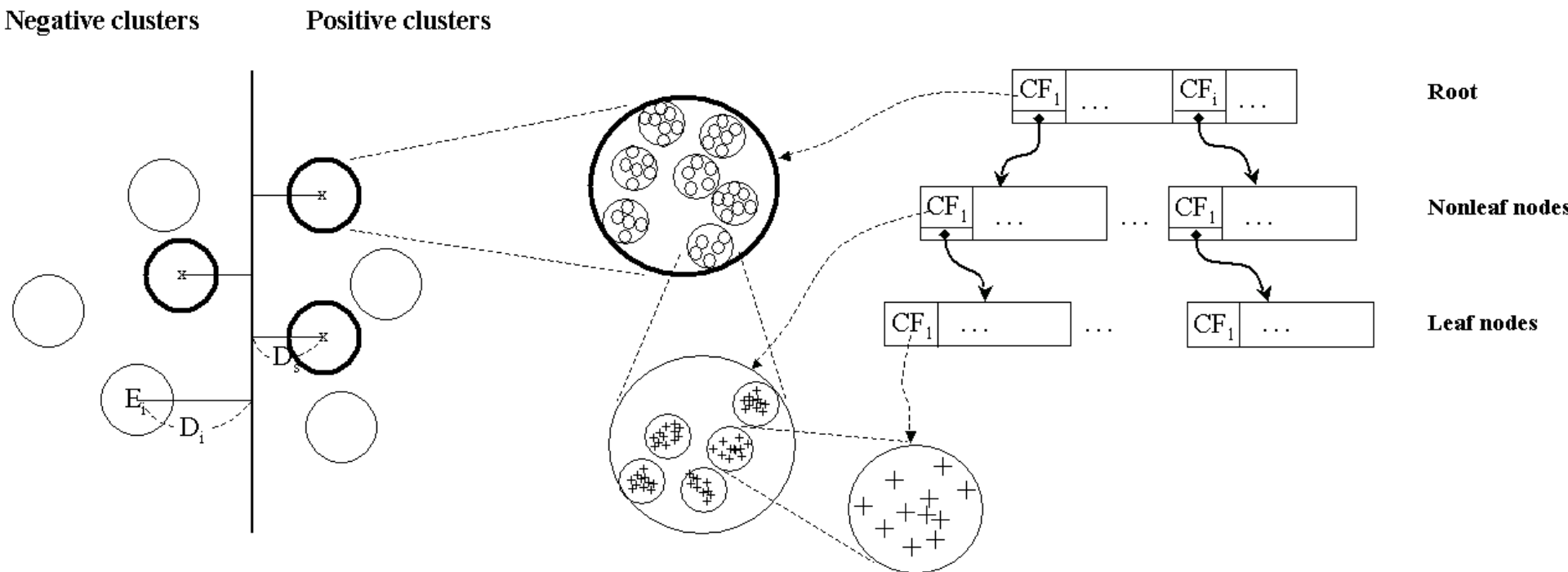
- SVM can also be used for classifying multiple ( $> 2$ ) classes and for regression analysis (with additional parameters)

# 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 H. Yu, J. Yang, J. 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

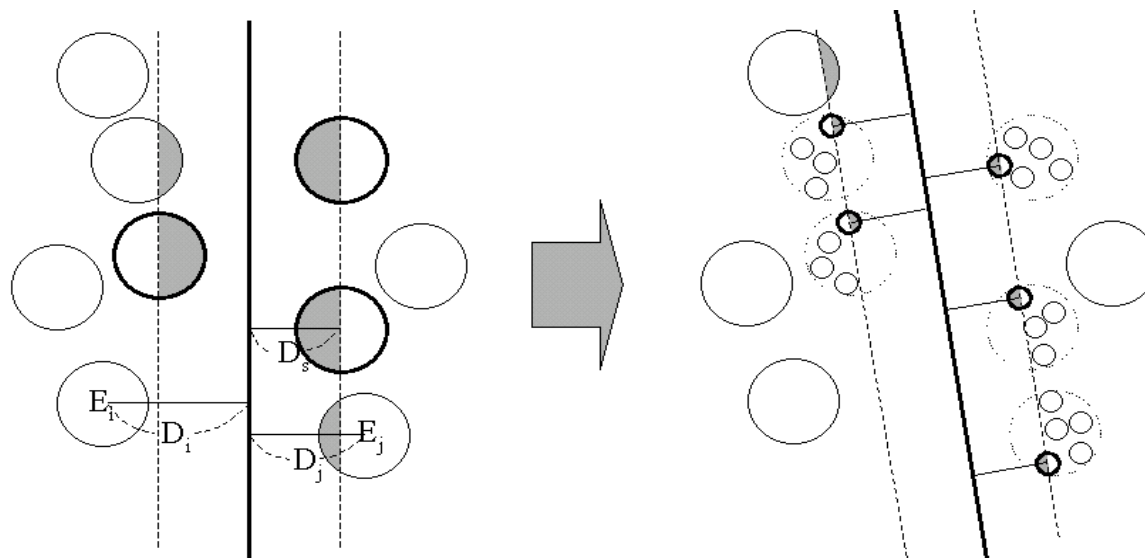
# CF-Tree: Hierarchical Micro-cluster



- Read the data set once, construct a statistical summary of the data (i.e., hierarchical clusters) given a limited amount of memory
- Micro-clustering: Hierarchical indexing structure
  - provide finer samples closer to the boundary and coarser samples farther from the boundary

# Selective Declustering: Ensure High Accuracy

- 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



# 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



# Accuracy and Scalability on Synthetic Dataset

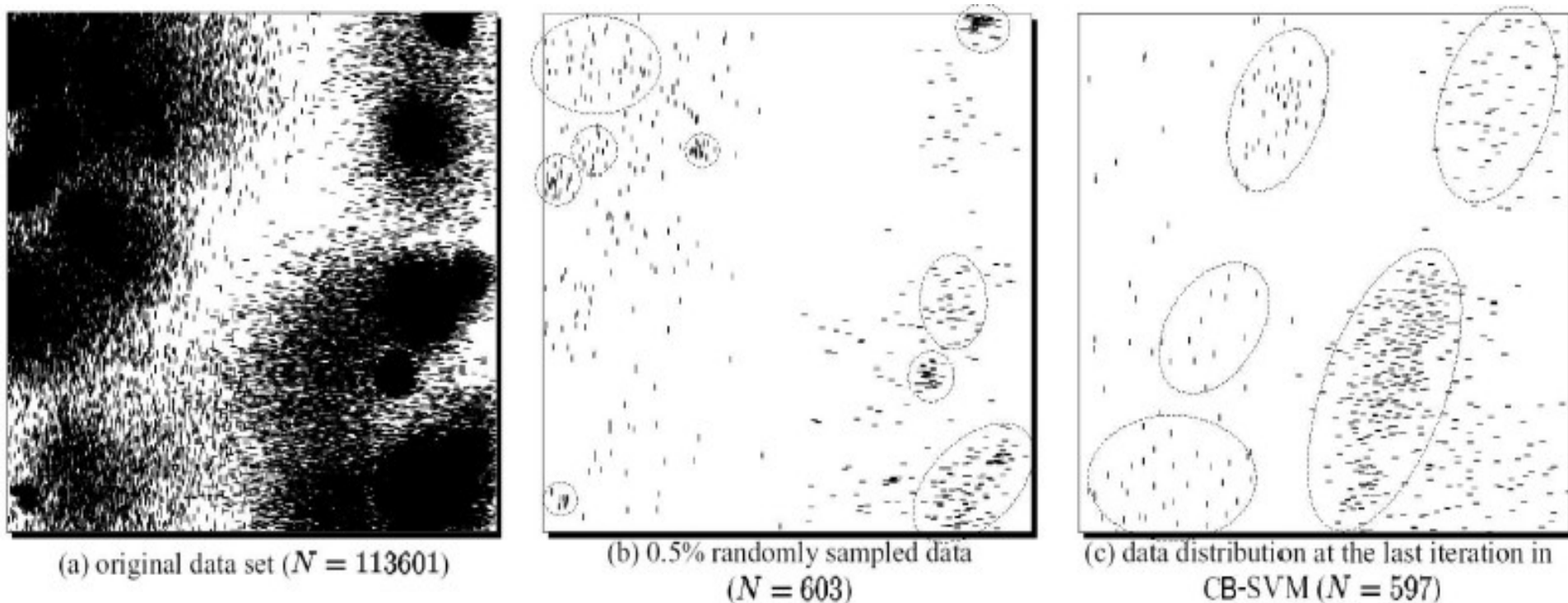


Figure 6: Synthetic data set in a two-dimensional space. '+': positive data; '-': negative data

- Experiments on large synthetic data sets shows better accuracy than random sampling approaches and far more scalable than the original SVM algorithm

# SVM vs. Neural Network

---

## ■ SVM

- Deterministic algorithm
- Nice generalization properties
- Hard to learn – learned in batch mode using quadratic programming techniques
- Using kernels can learn very complex functions

## ■ Neural Network

- Nondeterministic algorithm
- Generalizes well but doesn't have strong mathematical foundation
- Can easily be learned in incremental fashion
- To learn complex functions—use multilayer perceptron (nontrivial)

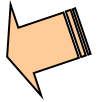
# SVM Related Links

---

- SVM Website: <http://www.kernel-machines.org/>
- Representative implementations
  - **LIBSVM**: an efficient implementation of SVM, multi-class classifications, nu-SVM, one-class SVM, including also various interfaces with java, python, etc.
  - **SVM-light**: simpler but performance is not better than LIBSVM, support only binary classification and only in C
  - **SVM-torch**: another recent implementation also written in C

# Chapter 9. Classification: Advanced Methods

---

- Bayesian Belief Networks
- Classification by Backpropagation
- Support Vector Machines
- Classification by Using Frequent Patterns 
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods
- Additional Topics Regarding Classification
- Summary

# Associative Classification

---

- Associative classification: Major steps

- Mine data to find strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels
- Association rules are generated in the form of

$$P_1 \wedge p_2 \dots \wedge p_l \rightarrow "A_{\text{class}} = C" (\text{conf}, \text{sup})$$

- Organize the rules to form a rule-based classifier

- Why effective?

- It explores highly confident associations among multiple attributes and may overcome some constraints introduced by decision-tree induction, which considers only one attribute at a time
- Associative classification has been found to be often more accurate than some traditional classification methods, such as C4.5

# Typical Associative Classification Methods

---

- **CBA** (Classification Based on Associations: Liu, Hsu & Ma, KDD'98)
  - Mine possible association rules in the form of
    - Cond-set (a set of attribute-value pairs) → class label
  - Build classifier: Organize rules according to decreasing precedence based on confidence and then support
- **CMAR** (Classification based on Multiple Association Rules: Li, Han, Pei, ICDM'01)
  - Classification: Statistical analysis on multiple rules
- **CPAR** (Classification based on Predictive Association Rules: Yin & Han, SDM'03)
  - Generation of predictive rules (FOIL-like analysis) but allow covered rules to retain with reduced weight
  - Prediction using best k rules
  - High efficiency, accuracy similar to CMAR

# Frequent Pattern-Based Classification

---

- H. Cheng, X. Yan, J. Han, and C.-W. Hsu, "Discriminative Frequent Pattern Analysis for Effective Classification", ICDE'07
- Accuracy issue
  - Increase the discriminative power
  - Increase the expressive power of the feature space
- Scalability issue
  - It is computationally infeasible to generate all feature combinations and filter them with an information gain threshold
  - Efficient method (DDPMine: FPtree pruning): H. Cheng, X. Yan, J. Han, and P. S. Yu, "Direct Discriminative Pattern Mining for Effective Classification", ICDE'08

# Frequent Pattern vs. Single Feature

The discriminative power of some frequent patterns is higher than that of single features.

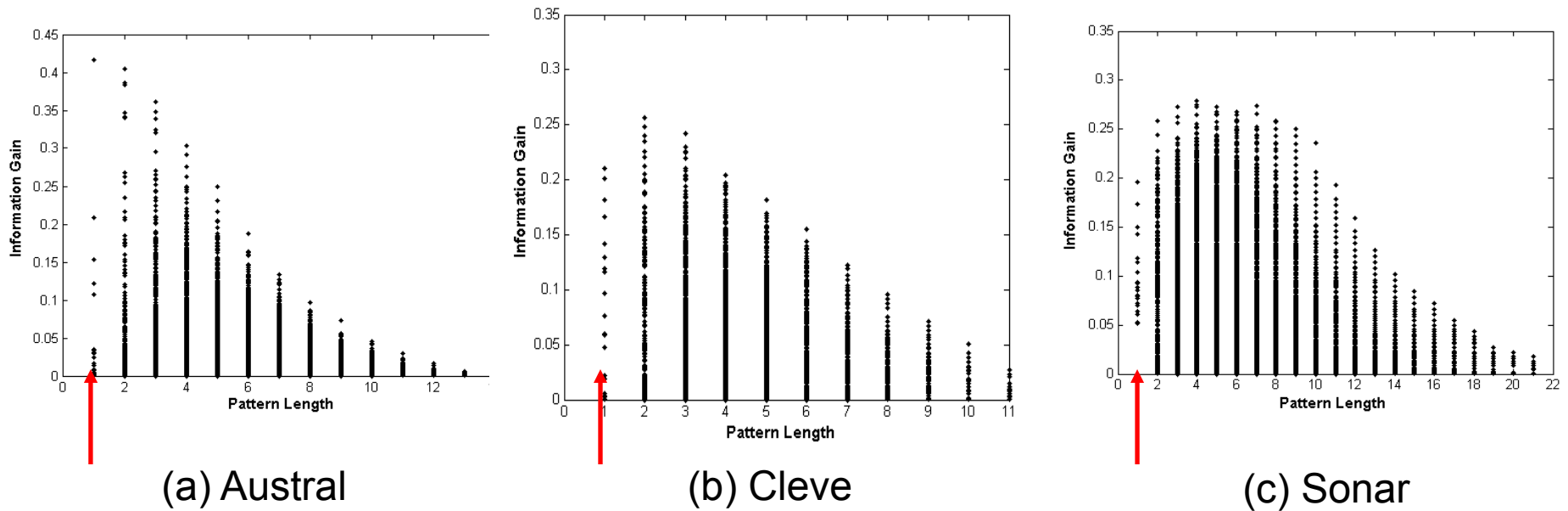


Fig. 1. Information Gain vs. Pattern Length



# Empirical Results

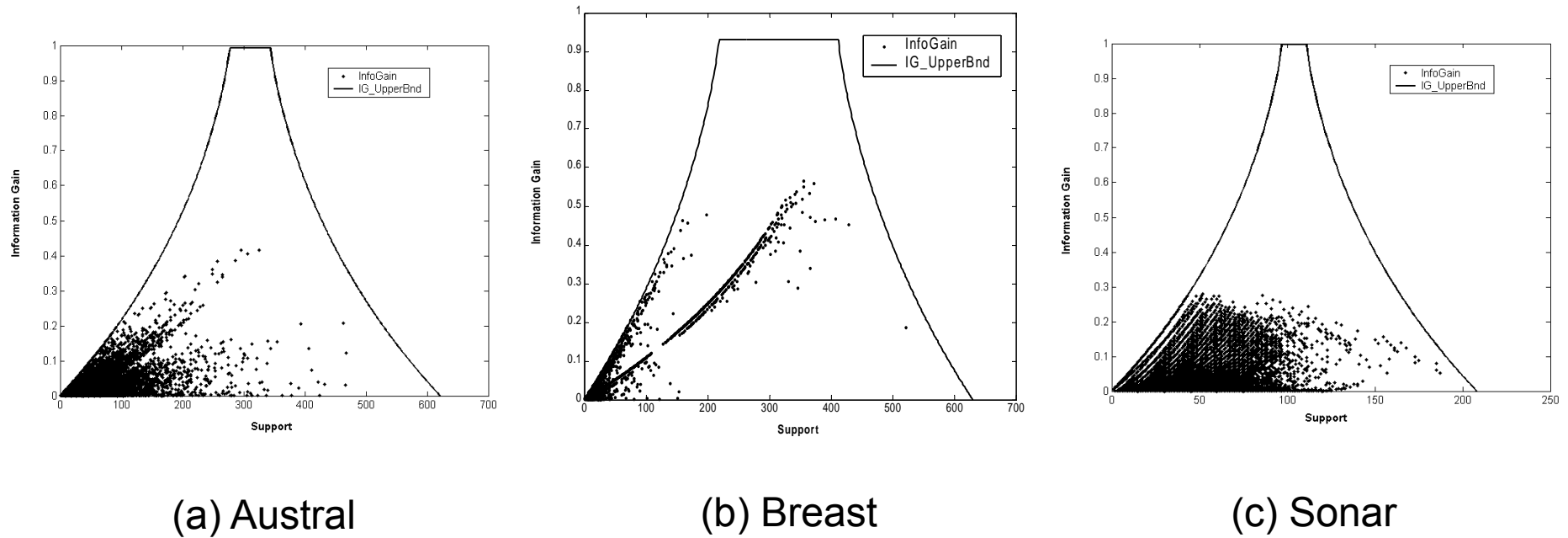


Fig. 2. Information Gain vs. Pattern Frequency

# Feature Selection

---

- Given a set of frequent patterns, both non-discriminative and redundant patterns exist, which can cause overfitting
- We want to single out the discriminative patterns and remove redundant ones
- The notion of **Maximal Marginal Relevance (MMR)** is borrowed
  - A document has high marginal relevance if it is both relevant to the query and contains minimal marginal similarity to previously selected documents

# Experimental Results

**Table 1. Accuracy by SVM on Frequent Combined Features vs. Single Features**

Data	Single Feature			Freq. Pattern	
	<i>Item_All</i>	<i>Item_FS</i>	<i>Item_RBF</i>	<i>Pat_All</i>	<i>Pat_FS</i>
anneal	<b>99.78</b>	<b>99.78</b>	99.11	99.33	99.67
austral	85.01	85.50	85.01	81.79	<b>91.14</b>
auto	83.25	84.21	78.80	74.97	<b>90.79</b>
breast	97.46	97.46	96.98	96.83	<b>97.78</b>
cleve	84.81	84.81	85.80	78.55	<b>95.04</b>
diabetes	74.41	74.41	74.55	77.73	<b>78.31</b>
glass	75.19	75.19	74.78	79.91	<b>81.32</b>
heart	84.81	84.81	84.07	82.22	<b>88.15</b>
hepatic	84.50	89.04	85.83	81.29	<b>96.83</b>
horse	83.70	84.79	82.36	82.35	<b>92.39</b>
iono	93.15	94.30	92.61	89.17	<b>95.44</b>
iris	94.00	<b>96.00</b>	94.00	95.33	<b>96.00</b>
labor	89.99	91.67	91.67	94.99	<b>95.00</b>
lymph	81.00	81.62	84.29	83.67	<b>96.67</b>
pima	74.56	74.56	76.15	76.43	<b>77.16</b>
sonar	82.71	86.55	82.71	84.60	<b>90.86</b>
vehicle	70.43	72.93	72.14	73.33	<b>76.34</b>
wine	98.33	99.44	98.33	98.30	<b>100</b>
zoo	97.09	97.09	95.09	94.18	<b>99.00</b>

**Table 2. Accuracy by C4.5 on Frequent Combined Features vs. Single Features**

Dataset	Single Features		Frequent Patterns	
	<i>Item_All</i>	<i>Item_FS</i>	<i>Pat_All</i>	<i>Pat_FS</i>
anneal	98.33	98.33	97.22	<b>98.44</b>
austral	84.53	84.53	84.21	<b>88.24</b>
auto	71.70	77.63	71.14	<b>78.77</b>
breast	95.56	95.56	95.40	<b>96.35</b>
cleve	80.87	80.87	80.84	<b>91.42</b>
diabetes	<b>77.02</b>	<b>77.02</b>	76.00	76.58
glass	75.24	75.24	76.62	<b>79.89</b>
heart	81.85	81.85	80.00	<b>86.30</b>
hepatic	78.79	85.21	80.71	<b>93.04</b>
horse	83.71	83.71	84.50	<b>87.77</b>
iono	92.30	92.30	92.89	<b>94.87</b>
iris	<b>94.00</b>	<b>94.00</b>	93.33	93.33
labor	86.67	86.67	<b>95.00</b>	91.67
lymph	76.95	77.62	74.90	<b>83.67</b>
pima	75.86	75.86	76.28	<b>76.72</b>
sonar	80.83	81.19	<b>83.67</b>	<b>83.67</b>
vehicle	70.70	71.49	<b>74.24</b>	73.06
wine	95.52	93.82	96.63	<b>99.44</b>
zoo	91.18	91.18	95.09	<b>97.09</b>

# Scalability Tests

**Table 3. Accuracy & Time on Chess Data**

<i>min_sup</i>	#Patterns	Time (s)	SVM (%)	C4.5 (%)
1	N/A	N/A	N/A	N/A
2000	68,967	44.703	92.52	97.59
2200	28,358	19.938	91.68	97.84
2500	6,837	2.906	91.68	97.62
2800	1,031	0.469	91.84	97.37
3000	136	0.063	91.90	97.06

**Table 4. Accuracy & Time on Waveform Data**

<i>min_sup</i>	#Patterns	Time (s)	SVM (%)	C4.5 (%)
1	9,468,109	N/A	N/A	N/A
80	26,576	176.485	92.40	88.35
100	15,316	90.406	92.19	87.29
150	5,408	23.610	91.53	88.80
200	2,481	8.234	91.22	87.32

# DDPMine: Branch-and-Bound Search

$$\text{sup}(\text{child}) \leq \text{sup}(\text{parent})$$

$$\text{sup}(b) \leq \text{sup}(a)$$

maximize  $IG(C|b)$

subject to

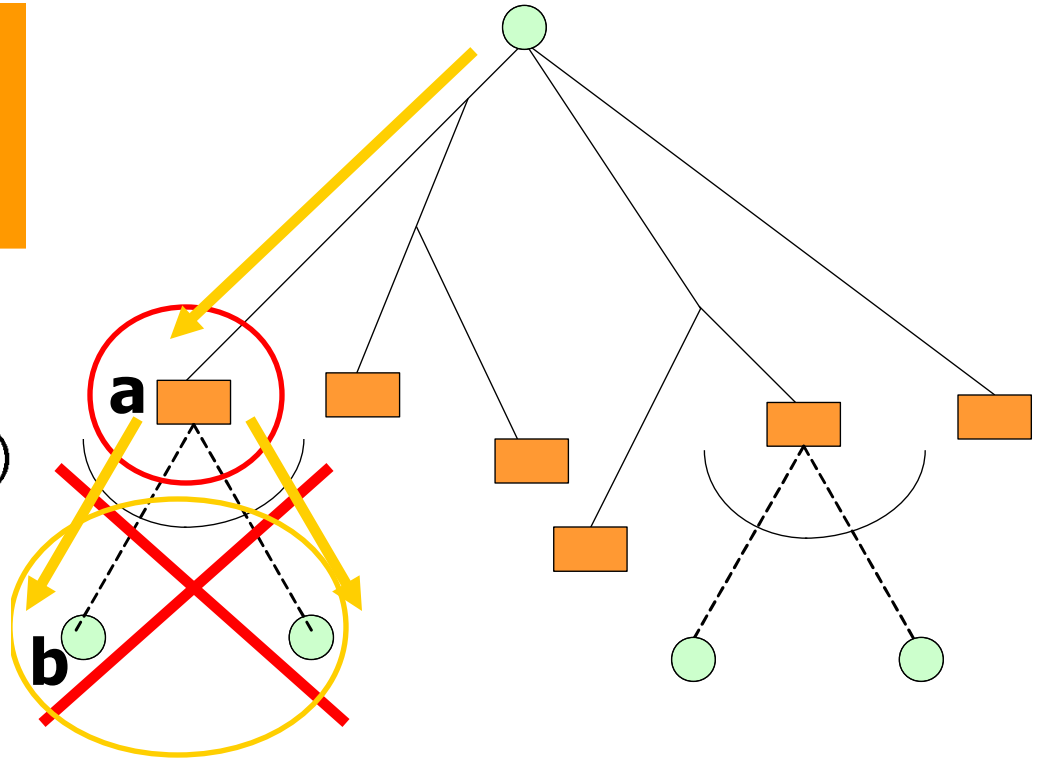
$$\text{min\_sup} \leq \text{sup}(b) \leq \text{sup}(a)$$

$$0 \leq \text{sup}_+(b) \leq \text{sup}_+(a)$$

$$0 \leq \text{sup}_-(b) \leq \text{sup}_-(a)$$

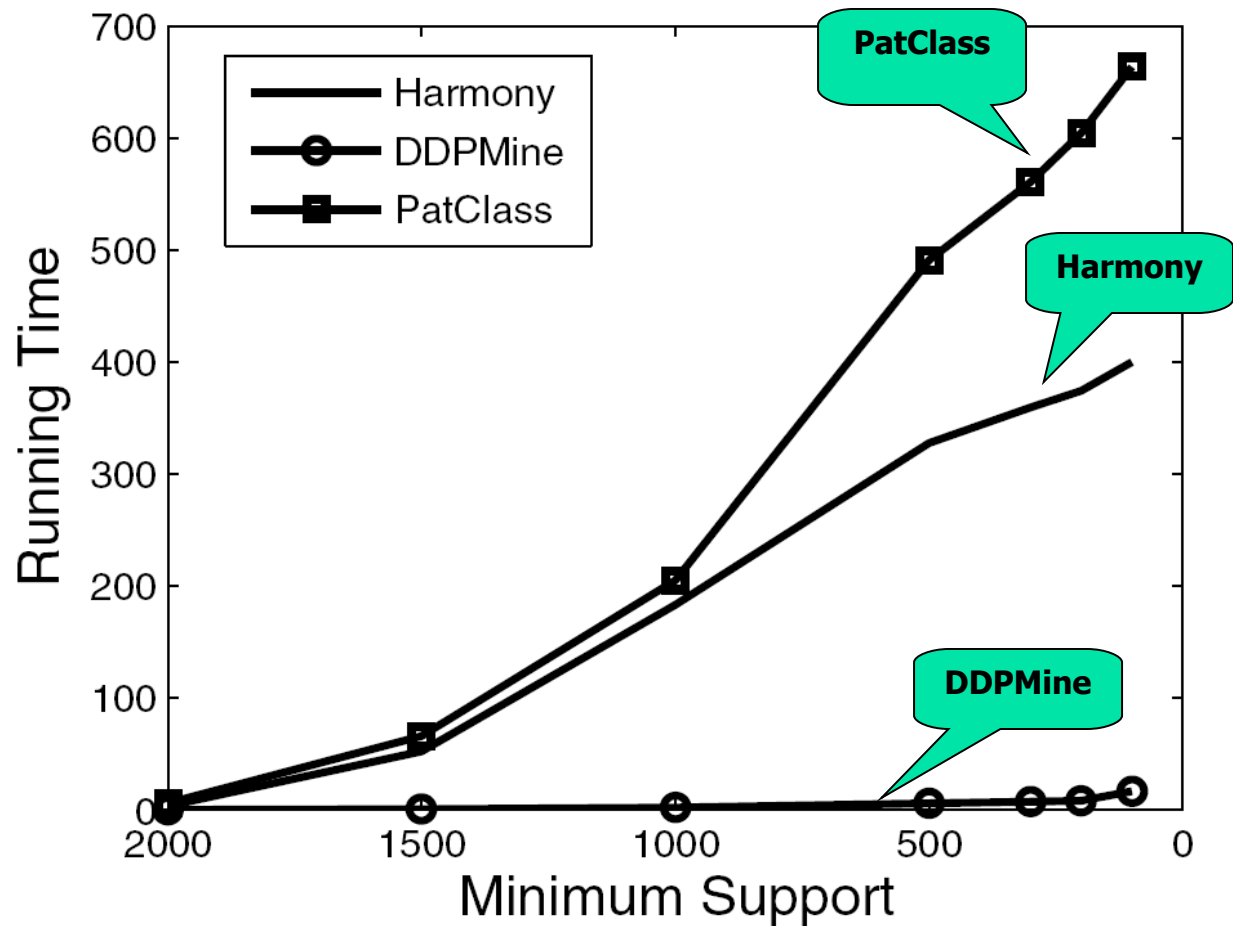
**a:** constant, a parent node

**b:** variable, a descendent



**Association between information gain and frequency**

# DDPMine Efficiency: Runtime



PatClass: ICDE'07  
Pattern  
Classification Alg.

# Chapter 9. Classification: Advanced Methods

---

- Bayesian Belief Networks
- Classification by Backpropagation
- Support Vector Machines
- Classification by Using Frequent Patterns
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods
- Additional Topics Regarding Classification
- Summary



# Lazy vs. Eager Learning

---

- Lazy vs. eager learning
  - **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - **Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form an implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space



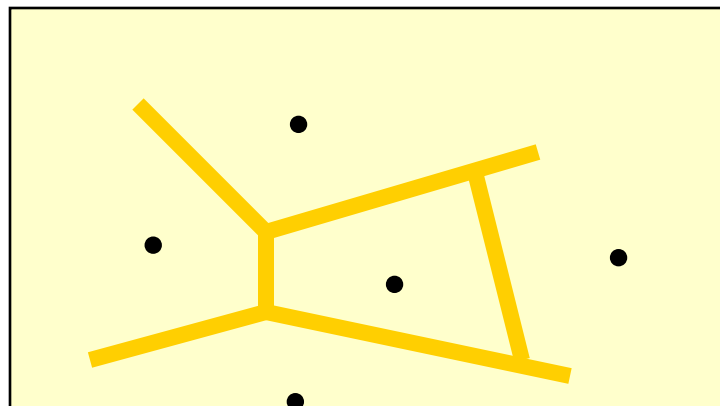
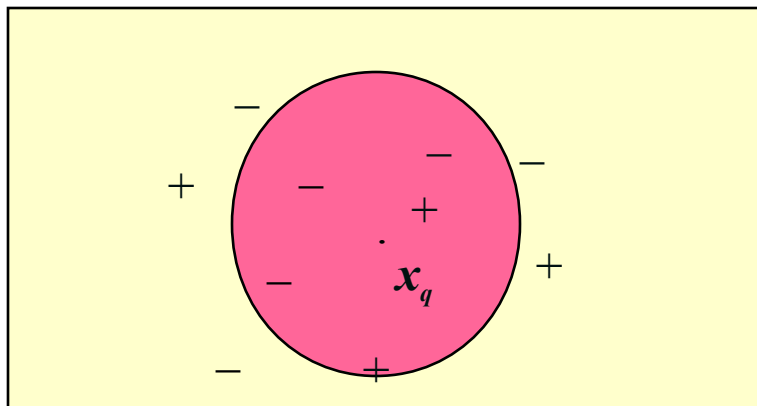
# Lazy Learner: Instance-Based Methods

---

- Instance-based learning:
  - Store training examples and delay the processing (“lazy evaluation”) until a new instance must be classified
- Typical approaches
  - $k$ -nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference

# The $k$ -Nearest Neighbor Algorithm

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



# Discussion on the *k*-NN Algorithm

---

- *k*-NN for real-valued prediction for a given unknown tuple
  - Returns the mean values of the *k* nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the *k* neighbors according to their distance to the query  $x_q$ 
$$w \equiv \frac{1}{d(x_q, x_i)^2}$$
    - Give greater weight to closer neighbors
- Robust to noisy data by averaging *k*-nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes
  - To overcome it, axes stretch or elimination of the least relevant attributes


# Case-Based Reasoning (CBR)

---

- **CBR:** Uses a database of problem solutions to solve new problems
- Store symbolic description (tuples or cases)—not points in a Euclidean space
- Applications: Customer-service (product-related diagnosis), legal ruling
- Methodology
  - Instances represented by rich symbolic descriptions (e.g., function graphs)
  - Search for similar cases, multiple retrieved cases may be combined
  - Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- Challenges
  - Find a good similarity metric
  - Indexing based on syntactic similarity measure, and when failure, backtracking, and adapting to additional cases

# Chapter 9. Classification: Advanced Methods

---

- Bayesian Belief Networks
- Classification by Backpropagation
- Support Vector Machines
- Classification by Using Frequent Patterns
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods 
- Additional Topics Regarding Classification
- Summary

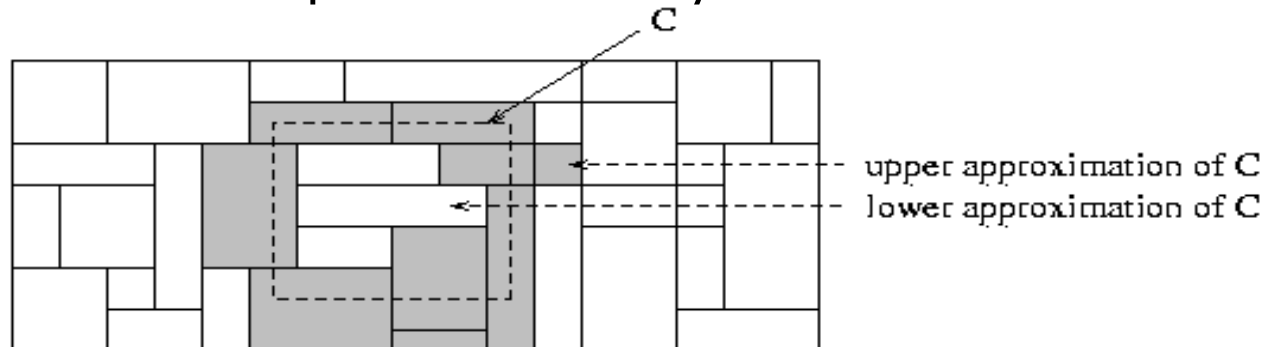
# Genetic Algorithms (GA)

---

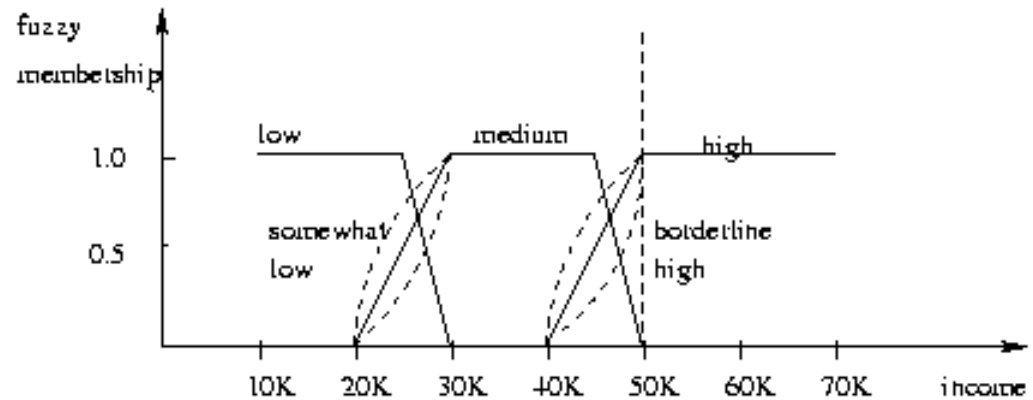
- Genetic Algorithm: based on an analogy to biological evolution
- An initial **population** is created consisting of randomly generated rules
  - Each rule is represented by a string of bits
  - E.g., if  $A_1$  and  $\neg A_2$  then  $C_2$  can be encoded as 100
  - If an attribute has  $k > 2$  values,  $k$  bits can be used
- Based on the notion of survival of the **fittest**, a new population is formed to consist of the fittest rules and their offspring
- The *fitness of a rule* is represented by its classification accuracy on a set of training examples
- Offspring are generated by *crossover* and *mutation*
- The process continues until a population  $P$  evolves *when each rule in  $P$  satisfies a prespecified threshold*
- Slow but easily parallelizable

# Rough Set Approach

- Rough sets are used to **approximately or “roughly” define equivalent classes**
- A rough set for a given class C is approximated by two sets: a **lower approximation** (certain to be in C) and an **upper approximation** (cannot be described as not belonging to C)
- Finding the minimal subsets (**reducts**) of attributes for feature reduction is NP-hard but a **discernibility matrix** (which stores the differences between attribute values for each pair of data tuples) is used to reduce the computation intensity



# Fuzzy Set Approaches



- Fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership (such as in a *fuzzy membership graph*)
- Attribute values are converted to fuzzy values. Ex.:
  - Income,  $x$ , is assigned a **fuzzy membership value** to each of the discrete categories {low, medium, high}, e.g. \$49K belongs to “medium income” with fuzzy value 0.15 but belongs to “high income” with fuzzy value 0.96
  - Fuzzy membership values do not have to sum to 1.
- Each applicable rule contributes a vote for membership in the categories
- Typically, the truth values for each predicted category are summed, and these sums are combined



# Chapter 9. Classification: Advanced Methods

---

- Bayesian Belief Networks
- Classification by Backpropagation
- Support Vector Machines
- Classification by Using Frequent Patterns
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods
- Additional Topics Regarding Classification
- Summary



# Multiclass Classification

---

- Classification involving more than two classes (i.e.,  $> 2$  Classes)
- Method 1. **One-vs.-all** (OVA): Learn a classifier one at a time
  - Given  $m$  classes, train  $m$  classifiers: one for each class
  - Classifier  $j$ : treat tuples in class  $j$  as *positive* & all others as *negative*
  - To classify a tuple  $\mathbf{X}$ , the set of classifiers vote as an ensemble
- Method 2. **All-vs.-all** (AVA): Learn a classifier for each pair of classes
  - Given  $m$  classes, construct  $m(m-1)/2$  binary classifiers
  - A classifier is trained using tuples of the two classes
  - To classify a tuple  $\mathbf{X}$ , each classifier votes.  $\mathbf{X}$  is assigned to the class with maximal vote
- Comparison
  - All-vs.-all tends to be superior to one-vs.-all
  - Problem: Binary classifier is sensitive to errors, and errors affect vote count

# Error-Correcting Codes for Multiclass Classification

- Originally designed to correct errors during data transmission for communication tasks by exploring data redundancy
- Example
  - A 7-bit codeword associated with classes 1-4
  - Given a unknown tuple  $\mathbf{X}$ , the 7-trained classifiers output: 0001010
  - Hamming distance: # of different bits between two codewords
  - $H(\mathbf{X}, C_1) = 5$ , by checking # of bits between [1111111] & [0001010]
  - $H(\mathbf{X}, C_2) = 3$ ,  $H(\mathbf{X}, C_3) = 3$ ,  $H(\mathbf{X}, C_4) = 1$ , thus  $C_4$  as the label for  $\mathbf{X}$
- Error-correcting codes can correct up to  $(h-1)/h$  1-bit error, where  $h$  is the minimum Hamming distance between any two codewords
- If we use 1-bit per class, it is equiv. to one-vs.-all approach, the code are insufficient to self-correct
- When selecting error-correcting codes, there should be good row-wise and col.-wise separation between the codewords

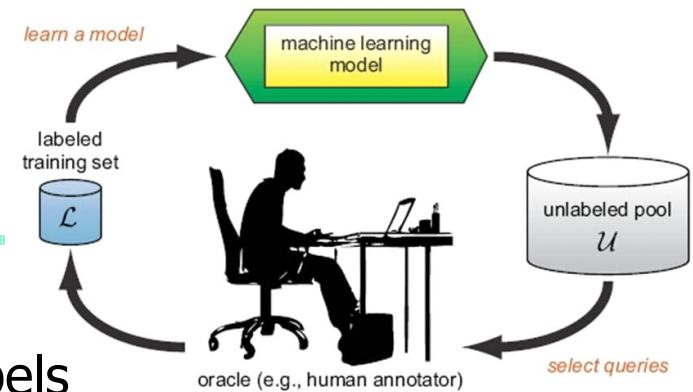
Class	Error-Corr. Codeword						
$C_1$	1	1	1	1	1	1	1
$C_2$	0	0	0	0	1	1	1
$C_3$	0	0	1	1	0	0	1
$C_4$	0	1	0	1	0	1	0

# Semi-Supervised Classification

---

- Semi-supervised: Uses labeled and unlabeled data to build a classifier
- Self-training:
  - Build a classifier using the labeled data
  - Use it to label the unlabeled data, and those with the most confident label prediction are added to the set of labeled data
  - Repeat the above process
  - Adv: easy to understand; disadv: may reinforce errors
- Co-training: Use two or more classifiers to teach each other
  - Each learner uses a mutually independent set of features of each tuple to train a good classifier, say  $f_1$
  - Then  $f_1$  and  $f_2$  are used to predict the class label for unlabeled data  $X$
  - Teach each other: The tuple having the most confident prediction from  $f_1$  is added to the set of labeled data for  $f_2$ , & vice versa
- Other methods, e.g., joint probability distribution of features and labels

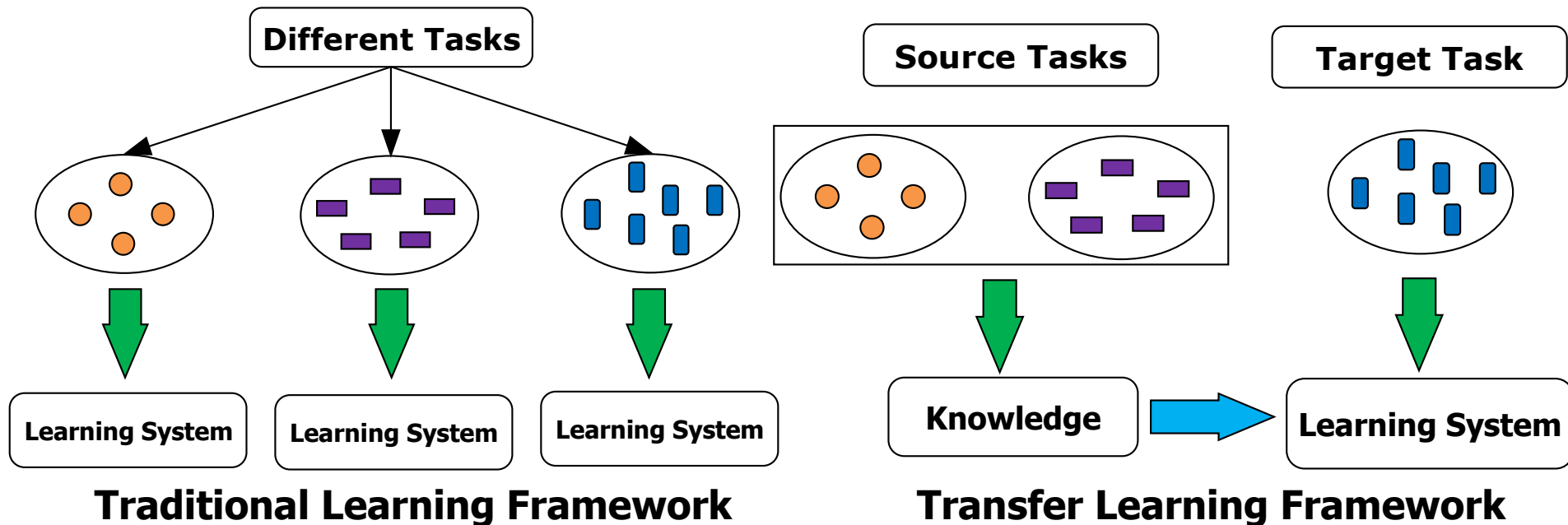
# Active Learning



- Class labels are expensive to obtain
- Active learner: query human (oracle) for labels
- Pool-based approach: Uses a pool of unlabeled data
  - $\mathcal{L}$ : a small subset of  $\mathcal{D}$  is labeled,  $\mathcal{U}$ : a pool of unlabeled data in  $\mathcal{D}$
  - Use a query function to carefully select one or more tuples from  $\mathcal{U}$  and request labels from an oracle (a human annotator)
  - The newly labeled samples are added to  $\mathcal{L}$ , and learn a model
  - Goal: Achieve high accuracy using as few labeled data as possible
- Evaluated using *learning curves*: Accuracy as a function of the number of instances queried (# of tuples to be queried should be small)
- Research issue: How to choose the data tuples to be queried?
  - Uncertainty sampling: choose the least certain ones
  - Reduce *version space*, the subset of hypotheses consistent w. the training data
  - Reduce expected entropy over  $\mathcal{U}$ : Find the greatest reduction in the total number of incorrect predictions

# Transfer Learning: Conceptual Framework

- Transfer learning: Extract knowledge from one or more source tasks and apply the knowledge to a target task
- Traditional learning: Build a new classifier for each new task
- Transfer learning: Build new classifier by applying existing knowledge learned from source tasks




# Transfer Learning: Methods and Applications

---

- Applications: Especially useful when data is outdated or distribution changes, e.g., Web document classification, e-mail spam filtering
- *Instance-based transfer learning*: Reweight some of the data from source tasks and use it to learn the target task
- TrAdaBoost (Transfer AdaBoost)
  - Assume source and target data each described by the same set of attributes (features) & class labels, but rather diff. distributions
  - Require only labeling a small amount of target data
  - Use source data in training: When a source tuple is misclassified, reduce the weight of such tuples so that they will have less effect on the subsequent classifier
- Research issues
  - Negative transfer: When it performs worse than no transfer at all
  - Heterogeneous transfer learning: Transfer knowledge from different feature space or multiple source domains
  - Large-scale transfer learning

# Chapter 9. Classification: Advanced Methods

---

- Bayesian Belief Networks
- Classification by Backpropagation
- Support Vector Machines
- Classification by Using Frequent Patterns
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods
- Additional Topics Regarding Classification
- Summary 



# Summary

---

- Effective and advanced classification methods
  - Bayesian belief network (probabilistic networks)
  - Backpropagation (Neural networks)
  - Support Vector Machine (SVM)
  - Pattern-based classification
  - Other classification methods: lazy learners (KNN, case-based reasoning), genetic algorithms, rough set and fuzzy set approaches
- Additional Topics on Classification
  - Multiclass classification
  - Semi-supervised classification
  - Active learning
  - Transfer learning

# References (1)

---

- C. M. Bishop, Neural Networks for Pattern Recognition. Oxford University Press, 1995
- C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2): 121-168, 1998
- 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
- N. Cristianini and J. Shawe-Taylor, Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2000
- A. J. Dobson. An Introduction to Generalized Linear Models. Chapman & Hall, 1990
- G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. KDD'99

# References (2)

---

- R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification, 2ed. John Wiley, 2001
- T. Hastie, R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer-Verlag, 2001
- S. Haykin, Neural Networks and Learning Machines, Prentice Hall, 2008
- D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning, 1995.
- V. Kecman, Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic, MIT Press, 2001
- W. Li, J. Han, and J. Pei, CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules, ICDM'01
- T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Machine Learning, 2000

# References (3)

---

- B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining, p. 80-86, KDD'98.
- T. M. Mitchell. Machine Learning. McGraw Hill, 1997.
- D.E. Rumelhart, and J.L. McClelland, editors, Parallel Distributed Processing, MIT Press, 1986.
- P. Tan, M. Steinbach, and V. Kumar. Introduction to Data Mining. Addison Wesley, 2005.
- S. M. Weiss and N. Indurkha. Predictive Data Mining. Morgan Kaufmann, 1997.
- I. H. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques, 2ed. Morgan Kaufmann, 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.





# OLDER SLIDES:

---

# What Is Prediction?

---

- (Numerical) prediction is similar to classification
  - construct a model
  - use model to predict continuous or ordered value for a given input
- Prediction is different from classification
  - Classification refers to predict categorical class label
  - Prediction models continuous-valued functions
- Major method for prediction: regression
  - model the relationship between one or more *independent* or **predictor** variables and a *dependent* or **response** variable
- Regression analysis
  - Linear and multiple regression
  - Non-linear regression
  - Other regression methods: generalized linear model, Poisson regression, log-linear models, regression trees



# Linear Regression

- Linear regression: involves a response variable  $y$  and a single predictor variable  $x$

$$y = w_0 + w_1 x$$

where  $w_0$  (y-intercept) and  $w_1$  (slope) are regression coefficients

- Method of least squares: estimates the best-fitting straight line

$$w_1 = \frac{\sum_{i=1}^{|D|} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|} (x_i - \bar{x})^2} \quad w_0 = \bar{y} - w_1 \bar{x}$$

- Multiple linear regression: involves more than one predictor variable
  - Training data is of the form  $(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \dots, (\mathbf{X}_{|D|}, y_{|D|})$
  - Ex. For 2-D data, we may have:  $y = w_0 + w_1 x_1 + w_2 x_2$
  - Solvable by extension of least square method or using SAS, S-Plus
  - Many nonlinear functions can be transformed into the above



# Nonlinear Regression

- Some nonlinear models can be modeled by a polynomial function
- A polynomial regression model can be transformed into linear regression model. For example,

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

convertible to linear with new variables:  $x_2 = x^2$ ,  $x_3 = x^3$

$$y = w_0 + w_1 x + w_2 x_2 + w_3 x_3$$

- Other functions, such as power function, can also be transformed to linear model
- Some models are intractable nonlinear (e.g., sum of exponential terms)
  - possible to obtain least square estimates through extensive calculation on more complex formulae

# Other Regression-Based Models

---

- Generalized linear model:
  - Foundation on which linear regression can be applied to modeling categorical response variables
  - Variance of  $y$  is a function of the mean value of  $y$ , not a constant
  - Logistic regression: models the prob. of some event occurring as a linear function of a set of predictor variables
  - Poisson regression: models the data that exhibit a Poisson distribution
- Log-linear models: (for categorical data)
  - Approximate discrete multidimensional prob. distributions
  - Also useful for data compression and smoothing
- Regression trees and model trees
  - Trees to predict continuous values rather than class labels

# Regression Trees and Model Trees

---

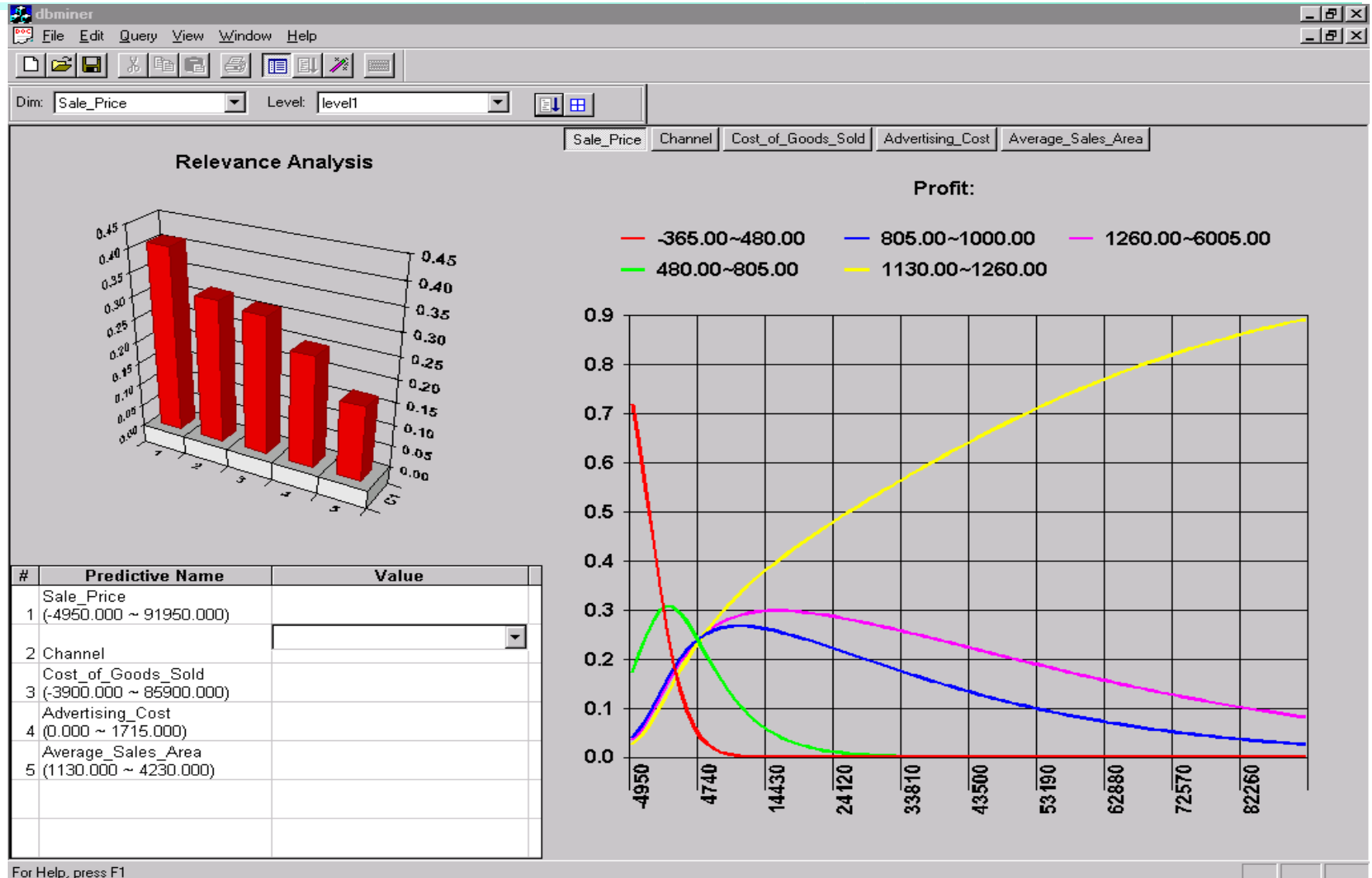
- Regression tree: proposed in CART system (Breiman et al. 1984)
  - CART: Classification And Regression Trees
  - Each leaf stores a *continuous-valued prediction*
  - It is the *average value of the predicted attribute* for the training tuples that reach the leaf
- Model tree: proposed by Quinlan (1992)
  - Each leaf holds a regression model—a multivariate linear equation for the predicted attribute
  - A more general case than regression tree
- Regression and model trees tend to be more accurate than linear regression when the data are not represented well by a simple linear model

# Predictive Modeling in Multidimensional Databases

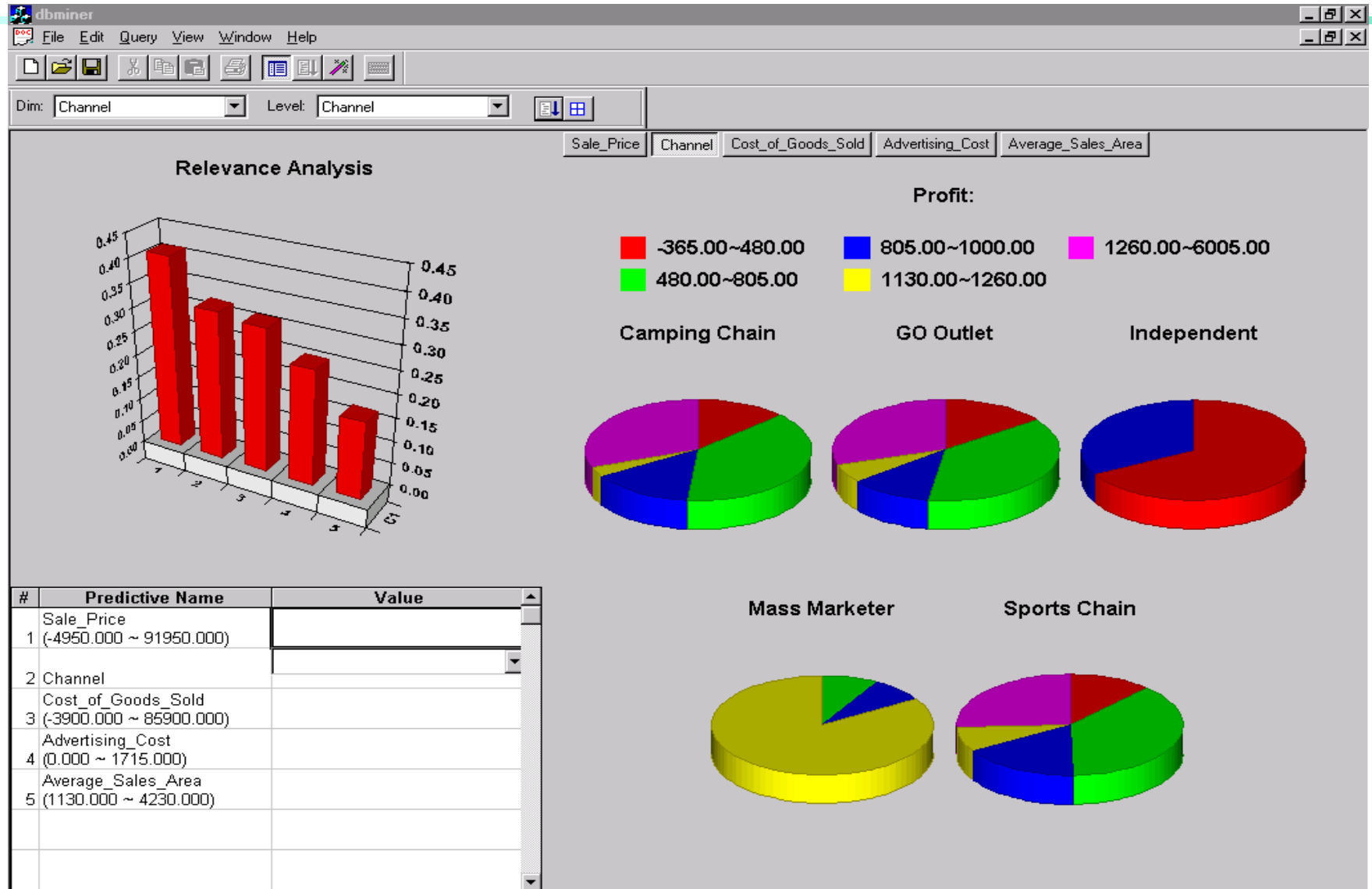
---

- Predictive modeling: Predict data values or construct generalized linear models based on the database data
- One can only predict value ranges or category distributions
- Method outline:
  - Minimal generalization
  - Attribute relevance analysis
  - Generalized linear model construction
  - Prediction
- Determine the major factors which influence the prediction
  - Data relevance analysis: uncertainty measurement, entropy analysis, expert judgement, etc.
- Multi-level prediction: drill-down and roll-up analysis

# Prediction: Numerical Data



# Prediction: Categorical Data



# SVM—Introductory Literature

---

- “Statistical Learning Theory” by Vapnik: extremely hard to understand, containing many errors too.
- C. J. C. Burges.  
**A Tutorial on Support Vector Machines for Pattern Recognition.**  
*Knowledge Discovery and Data Mining*, 2(2), 1998.
  - 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

# Notes about SVM— Introductory Literature

---

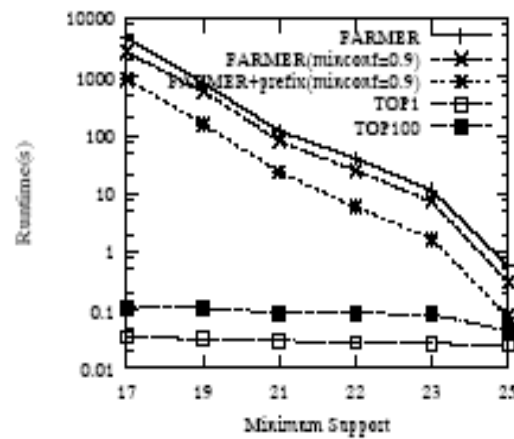
- “Statistical Learning Theory” by **Vapnik**: difficult to understand, containing many errors.
- C. J. C. **Burges**.  
**A Tutorial on Support Vector Machines for Pattern Recognition.**  
*Knowledge Discovery and Data Mining*, 2(2), 1998.
  - Easier than Vapnik’s book, but still not introductory level; the examples are not so intuitive
- The book An Introduction to Support Vector Machines by **Cristianini and Shawe-Taylor**
  - Not introductory level, but the explanation about Mercer’s Theorem is better than above literatures
- Neural Networks and Learning Machines by **Haykin**
  - Contains a nice chapter on SVM introduction



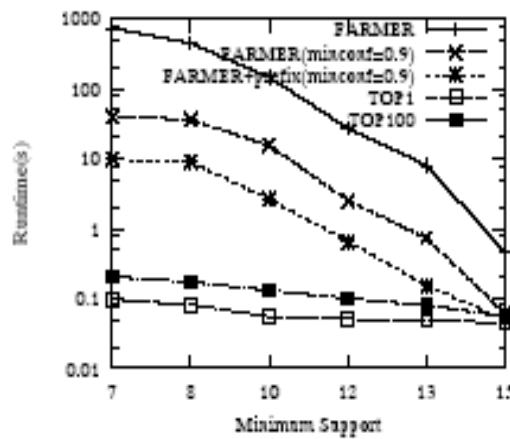
# Associative Classification Can Achieve High Accuracy and Efficiency (Cong et al. SIGMOD05)

Dataset	RCBT	CBA	IRG Classifier	C4.5 family			SVM
				single tree	bagging	boosting	
AML/ALL (ALL)	91.18%	91.18%	64.71%	91.18%	91.18%	91.18%	97.06%
Lung Cancer(LC)	97.99%	81.88%	89.93%	81.88%	96.64%	81.88%	96.64%
Ovarian Cancer(OC)	97.67%	93.02%	-	97.67%	97.67%	97.67%	97.67%
Prostate Cancer(PC)	97.06%	82.35%	88.24%	26.47%	26.47%	26.47%	79.41%
Average Accuracy	95.98%	87.11%	80.96%	74.3%	77.99%	74.3%	92.70%

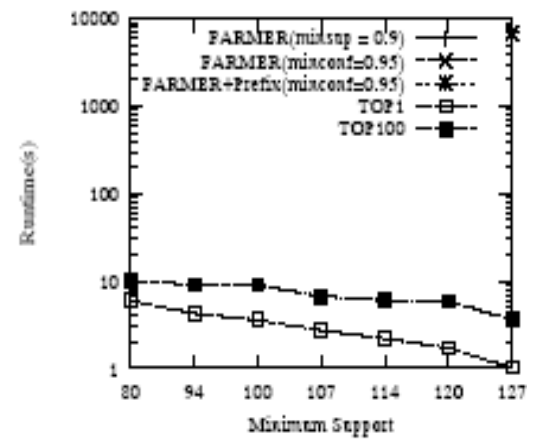
Table 2: Classification Results



(a) ALL-AML leukemia



(b) Lung Cancer



(c) Ovarian Cancer

# A Closer Look at CMAR

- **CMAR** (Classification based on Multiple Association Rules: Li, Han, Pei, ICDM'01)
- Efficiency: Uses an enhanced FP-tree that maintains the distribution of class labels among tuples satisfying each frequent itemset
- Rule pruning whenever a rule is inserted into the tree
  - Given two rules,  $R_1$  and  $R_2$ , if the antecedent of  $R_1$  is more general than that of  $R_2$  and  $\text{conf}(R_1) \geq \text{conf}(R_2)$ , then prune  $R_2$
  - Prunes rules for which the rule antecedent and class are not positively correlated, based on a  $\chi^2$  test of statistical significance
- Classification based on generated/pruned rules
  - If only *one rule* satisfies tuple  $X$ , assign the class label of the rule
  - If a *rule set*  $S$  satisfies  $X$ , CMAR
    - divides  $S$  into groups according to class labels
    - uses a weighted  $\chi^2$  measure to find the strongest group of rules, based on the statistical correlation of rules within a group
    - assigns  $X$  the class label of the strongest group