Data Mining:
Concepts and Techniques

(3rd ed.)

— Chapter 9 —
Classification: Advanced Methods

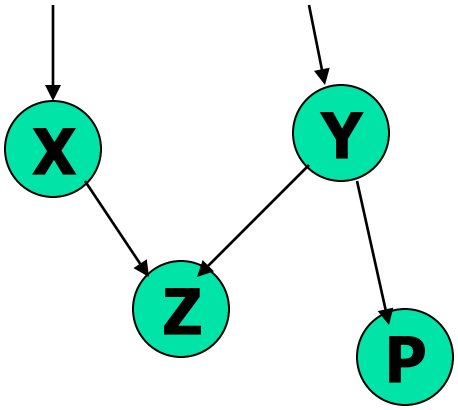
Jianjun Cheng
School of Information Science & Engineering
Lanzhou University

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

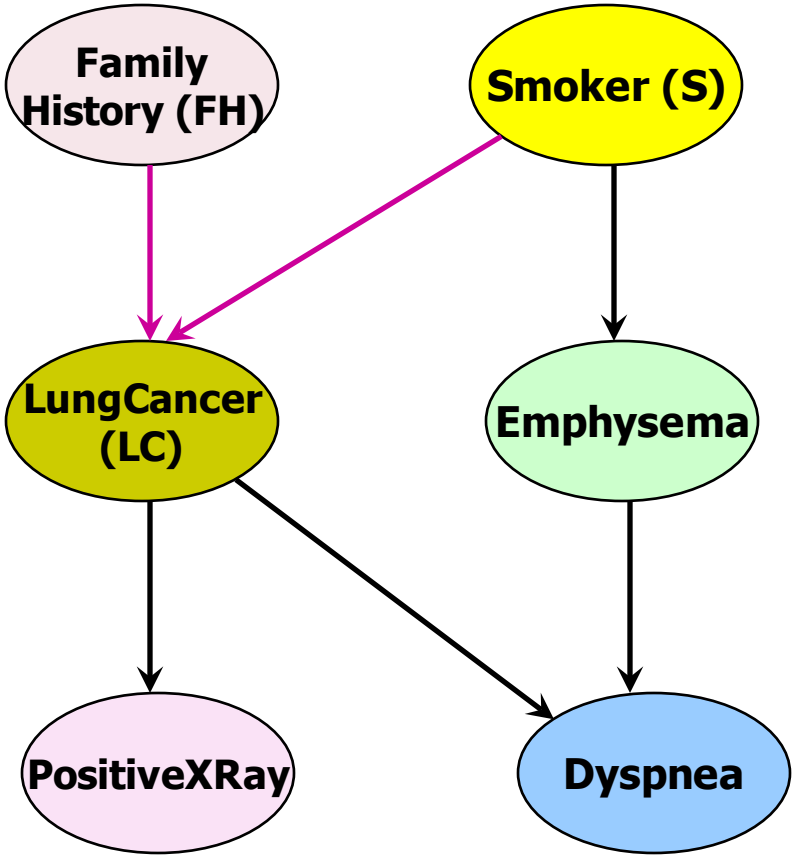
Bayesian Belief Networks

- **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



Bayesian Belief Network

CPT: Conditional Probability Table for variable LungCancer:

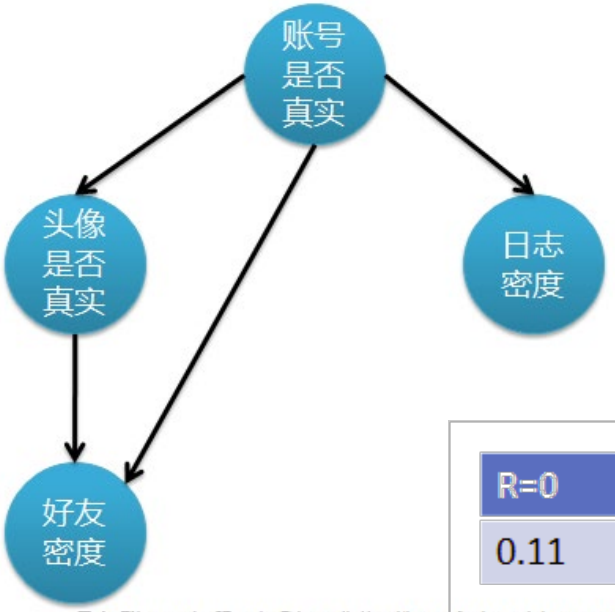
	(FH, S)	(FH, ~S)	(~FH, S)	(~FH, ~S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of $\mathbf{X(x_1,x_2,\ldots,x_n)}$, from CPT:

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

Bayesian network: an Example



R=0	R=1
0.11	0.89

	H=0	H=1
R=0	0.9	0.1
R=1	0.2	0.8

Bayesian network: an Example

$$P(R=0|H=0) = \frac{P(H=0|R=0)P(R=0)}{P(H=0)}$$

$$= \frac{P(H=0|R=0)P(R=0)}{P(H=0|R=0)P(R=0) + P(H=0|R=1)P(R=1)}$$

$$= \frac{0.9 \times 0.11}{0.9 \times 0.11 + 0.2 \times 0.89}$$

$$\approx 0.3574$$

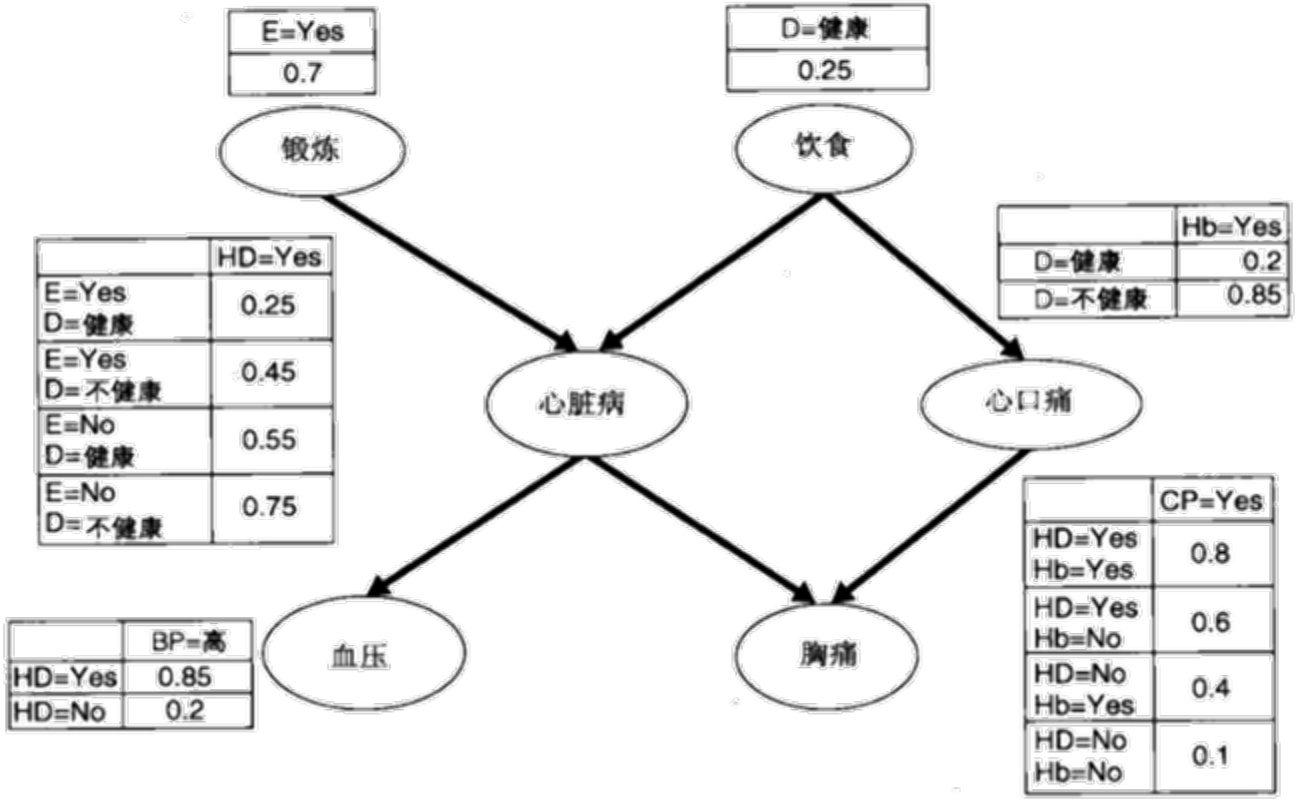
Bayesian network: an Example

- 四个随机变量：账号真实性R，头像真实性H，
  日志密度L，好友密度F。
- 其中H，L，F 可以观察到，而R无法直接观察得到
- 问题转换为通过H，L，F的观察值对R进行概率推理
- 推理过程可以如下表示：
  - 使用观察值实例化H, L和F，把随机值赋给R。
  - 计算$P(R|H,L,F) \propto P(R)P(H|R)P(L|R)P(F|R,H,L)$
    $=P(R)P(H|R)P(L|R)P(F|R,H)$

  其中相应概率值可以查条件概率表

Bayesian network: an Example

$$P(\text{心脏病}=\text{No}|\text{锻炼}=\text{No},\ \text{饮食}=\text{健康})$$
$$= 1 - P(\text{心脏病}=\text{Yes}|\text{锻炼}=\text{No},\ \text{饮食}=\text{健康})$$
$$= 1 - 0.55 = 0.45$$



判断一个人是否患有心脏疾病

- 没有先验信息
  - 计算先验概率P(HD=1)和P(HD=0)

$$P(HD=1)$$

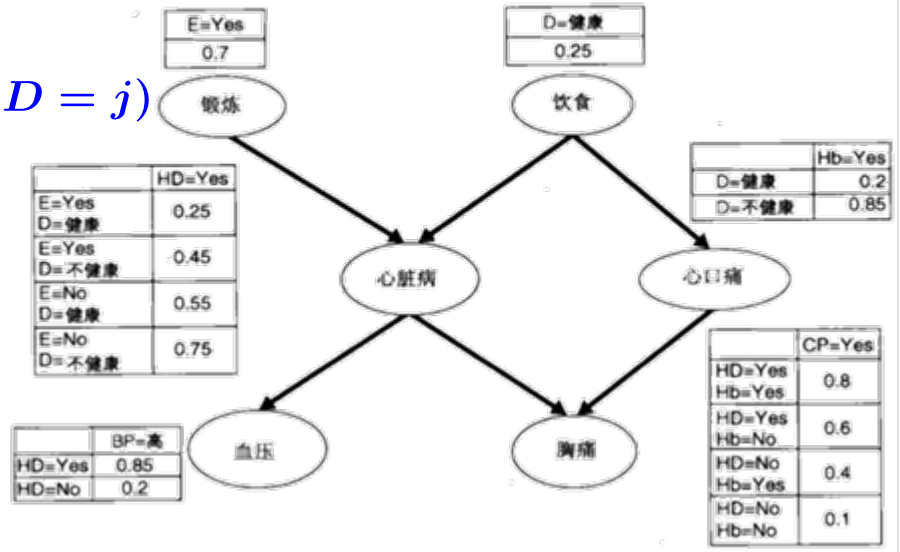
$$=\sum_{i=0}^{1}\sum_{j=0}^{1}P(HD=1|E=i,D=j)P(E=i,D=j)$$

$$P(HD=0)$$

$$=\sum_{i=0}^{1}\sum_{j=0}^{1}P(HD=0|E=i,D=j)P(E=i,D=j)$$

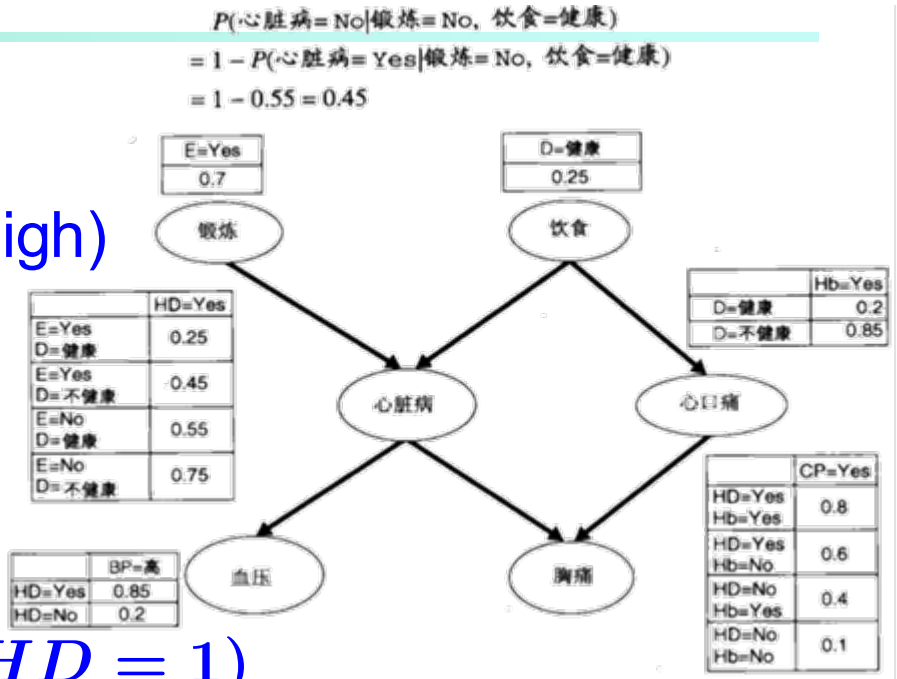
其中i=0, 1表示锻炼的两个属性值，j=0, 1分别表示不健康和健康的饮食。

$P(\text{心脏病}=\text{No}|\text{锻炼}=\text{No},\ \text{饮食}=\text{健康})$
$=1-P(\text{心脏病}=\text{Yes}|\text{锻炼}=\text{No},\ \text{饮食}=\text{健康})$
$=1-0.55=0.45$



判断一个人是否患有心脏疾病

- 高血压
  - 比较后验概率P(HD=1|BP=High)和P(HD=0|BP=High)



$$P(HD=1|BP=High)$$

$$=\frac{P(BP=High|HD=1)P(HD=1)}{P(BP=High)}$$

$$P(HD=0|BP=High)$$

$$=\frac{P(BP=High|HD=0)P(HD=0)}{P(BP=High)}$$

判断一个人是否患有心脏疾病

- 高血压，饮食健康，锻炼情况

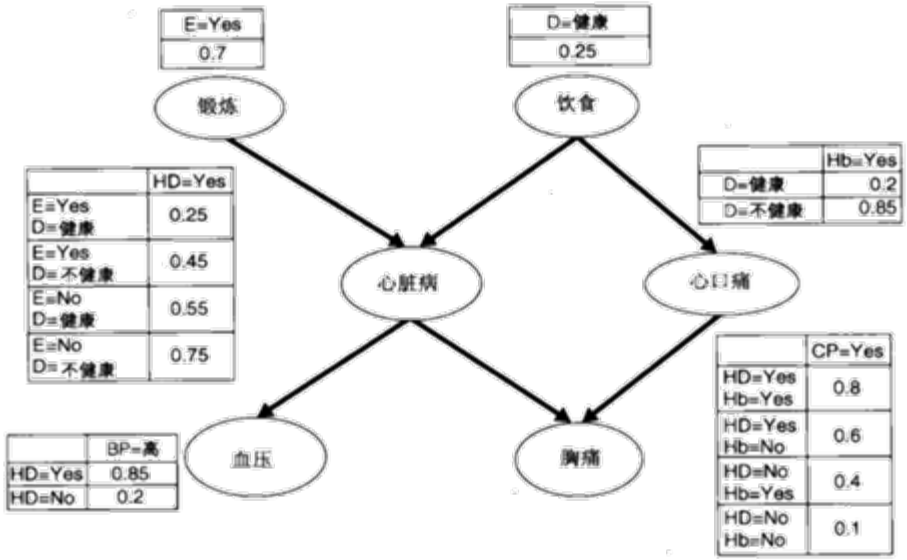
$$P(HD=1|BP=High, D=Healthy, E=Yes)$$

$$=\left[\frac{P(BP=High|HD=Yes, D=Healthy, E=Yes)}{P(BP=High|D=Healthy, E=Yes)}\right]$$

$$\cdot P(HD=Yes|D=Healthy, E=Yes)$$

$$P(\text{心脏病}=\text{No}|\text{锻炼}=\text{No},\ \text{饮食}=\text{健康})$$
$$=1-P(\text{心脏病}=\text{Yes}|\text{锻炼}=\text{No},\ \text{饮食}=\text{健康})$$
$$=1-0.55=0.45$$

- 很适合处理不完整的数据，信息有缺失时，仍可分类
- 对模型过拟合有很好的鲁棒性



$$P(HD=1|BP=High, D=Healthy, E=yes)$$

$$=\frac{P(D=Healthy, E=yes, BP=High|HD=1)P(HD=1)}{P(BP=High, D=Healthy, E=yes)}$$

$$=\left[\frac{P(D=Healthy|HD=1)P(E=Yes|HD=1, D=Healthy)}{P(BP=High, D=Healthy, E=yes)}\right]$$

$$P(BP=High|HD=1, D=Healthy, E=yes)P(HD=1)$$

$$=\left[\frac{P(BP=High|HD=1, D=Healthy, E=yes)}{P(BP=High, D=Healthy, E=yes)}\right]$$

$$P(D=Healthy, E=yes|HD=1)P(HD=1)$$

$$=\left[\frac{P(BP=High|HD=1, D=Healthy, E=yes)}{P(BP=High, D=Healthy, E=yes)}\right]$$

$$P(HD=1|D=Healthy, E=yes)P(D=Healthy, E=yes)$$

$$=\left[\frac{P(BP=High|HD=1, D=Healthy, E=yes)}{\frac{P(BP=High, D=Healthy, E=yes)}{P(D=Healthy, E=yes)}}\right] P(HD=1|D=Healthy, E=yes)$$

$$=\left[\frac{P(BP=High|HD=1, D=Healthy, E=yes)}{P(BP=High|D=Healthy, E=yes)}\right] P(HD=1|D=Heathy, E=yes)$$

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

Training Bayesian Networks: Several Scenarios

- 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.

Training Bayesian Networks: Gradient Descent

- Let D be a training set of data tuples, $X_1, X_2, \ldots, X_{|D|}$. Training the belief network means to learn the values of the CPT entries.
- Let w_{ijk} be a CPT entry for the variable $Y_i=y_{ij}$ having the parents $U_i=u_{ik}$, where $\mathbf{w_{ijk} \equiv P(Y_i=y_{ij} | U_i=u_{ik})}$, which represents the probability of value of the ith variable, Y_i, to be its jth value, y_{ij}, given the value of its parents U_i, to be the kth combination, u_{ik}
- E.g. w_{ijk}

$\mathbf{Y_i}$**=LungCancer** $\mathbf{y_{ij}}$**=yes**

$\mathbf{U_i}$**=FH, S** $\mathbf{u_{ik}}$**=yes, yes**

	(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

Training Bayesian Networks: Gradient Descent

- The w_{ijk} are viewed as weights, the set of weights is collectively referred to as **W**.
- The weights are initialized to random probability values.
- A **gradient descent** strategy performs greedy hill-climbing, and the objective is to maximize the probability

$$P_w(D) = \prod_{d=1}^{|D|} P_w(X_d)$$

Training Bayesian Networks: Gradient Descent

- Given the network topology and initialized w_{ijk} , the algorithm proceeds as follows
  - Compute the gradients: For each i, j, k, compute

$$\frac{\partial \ln P_w(D)}{\partial w_{ijk}} = \sum_{d=1}^{|D|} \frac{P(Y_i = y_{ij}, U_i = u_{ik} | X_d)}{w_{ijk}}$$

  - Take a small step in the direction of the gradient

$$w_{ijk} \leftarrow w_{ijk} + (l) \frac{\partial \ln P_w(D)}{\partial w_{ijk}}$$

  - Renormalize the weights, so that $\sum_j w_{ijk} = 1$

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

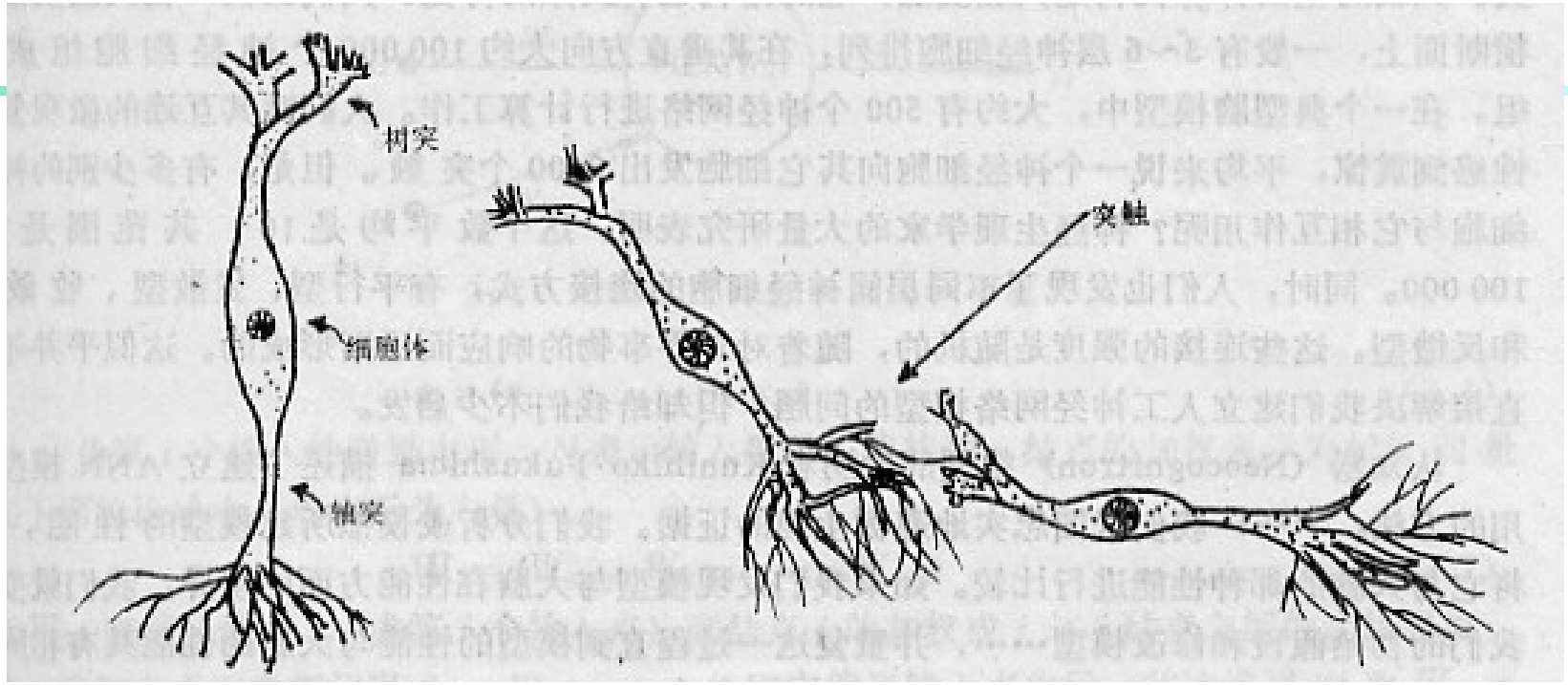
Classification by Backpropagation

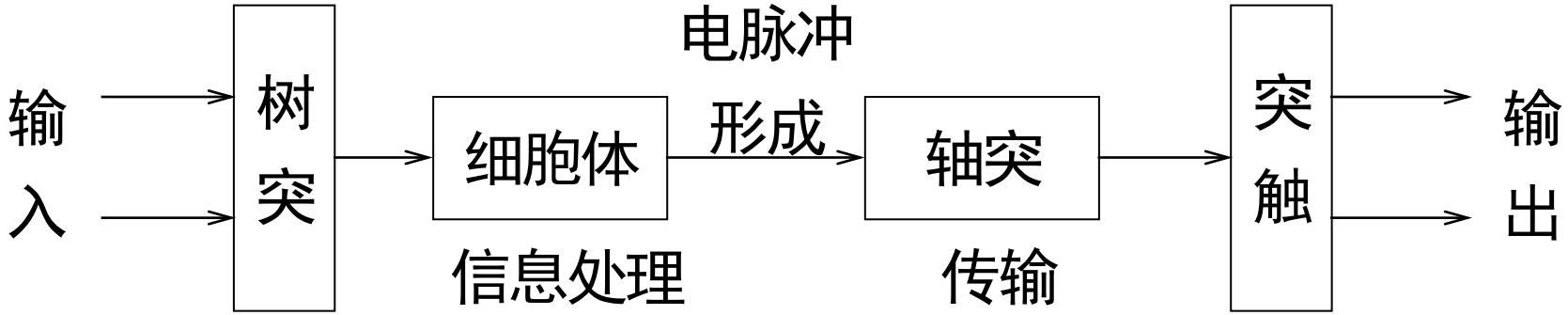
- Backpropagation: A **neural network** learning algorithm
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where each connection has a **weight** associated with it
- During the learning phase, 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

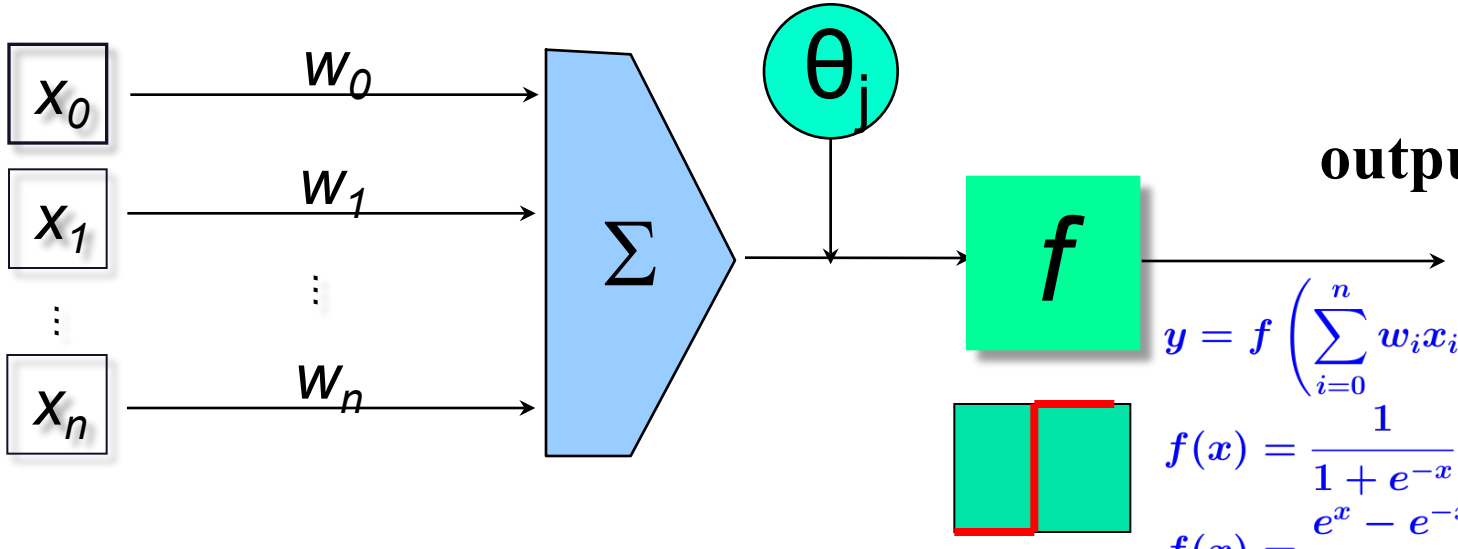
- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g., the network topology or "structure".
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of "hidden units" in the network
- Strength
  - 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

简单的神经元





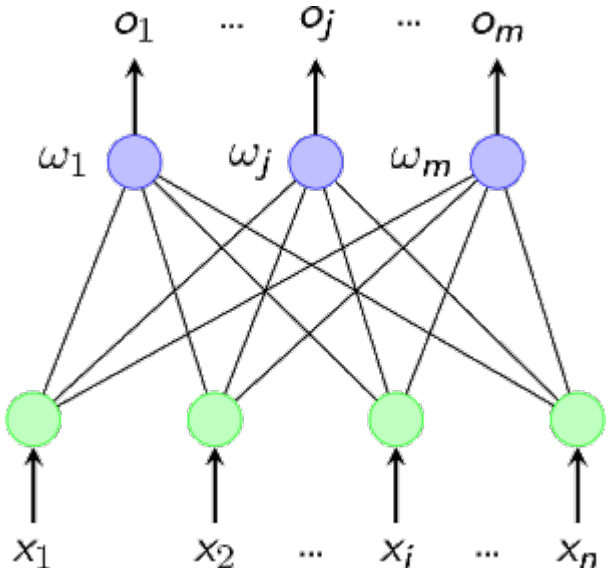
A Neuron (= a perceptron)

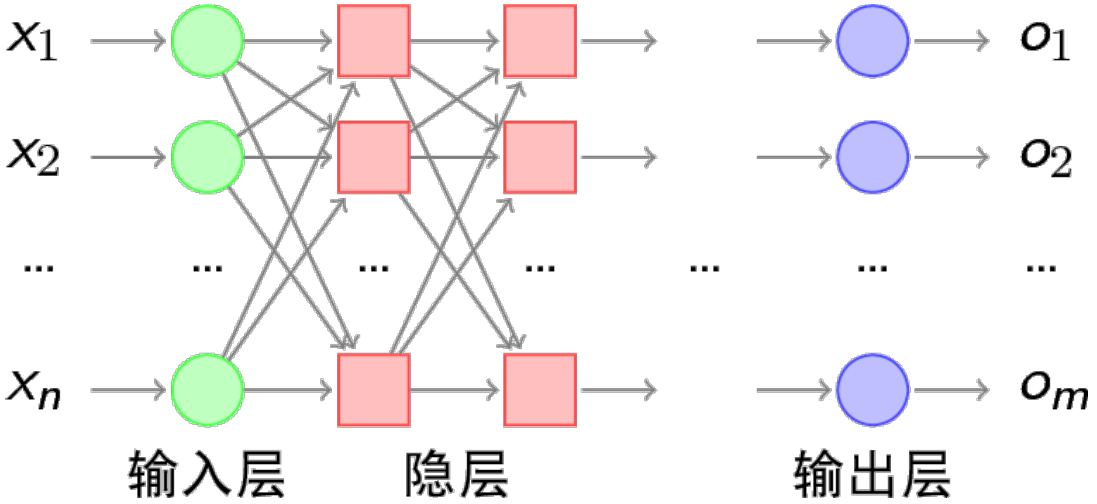


Input vector **x** | weight vector **w** | weighted sum | Activation function

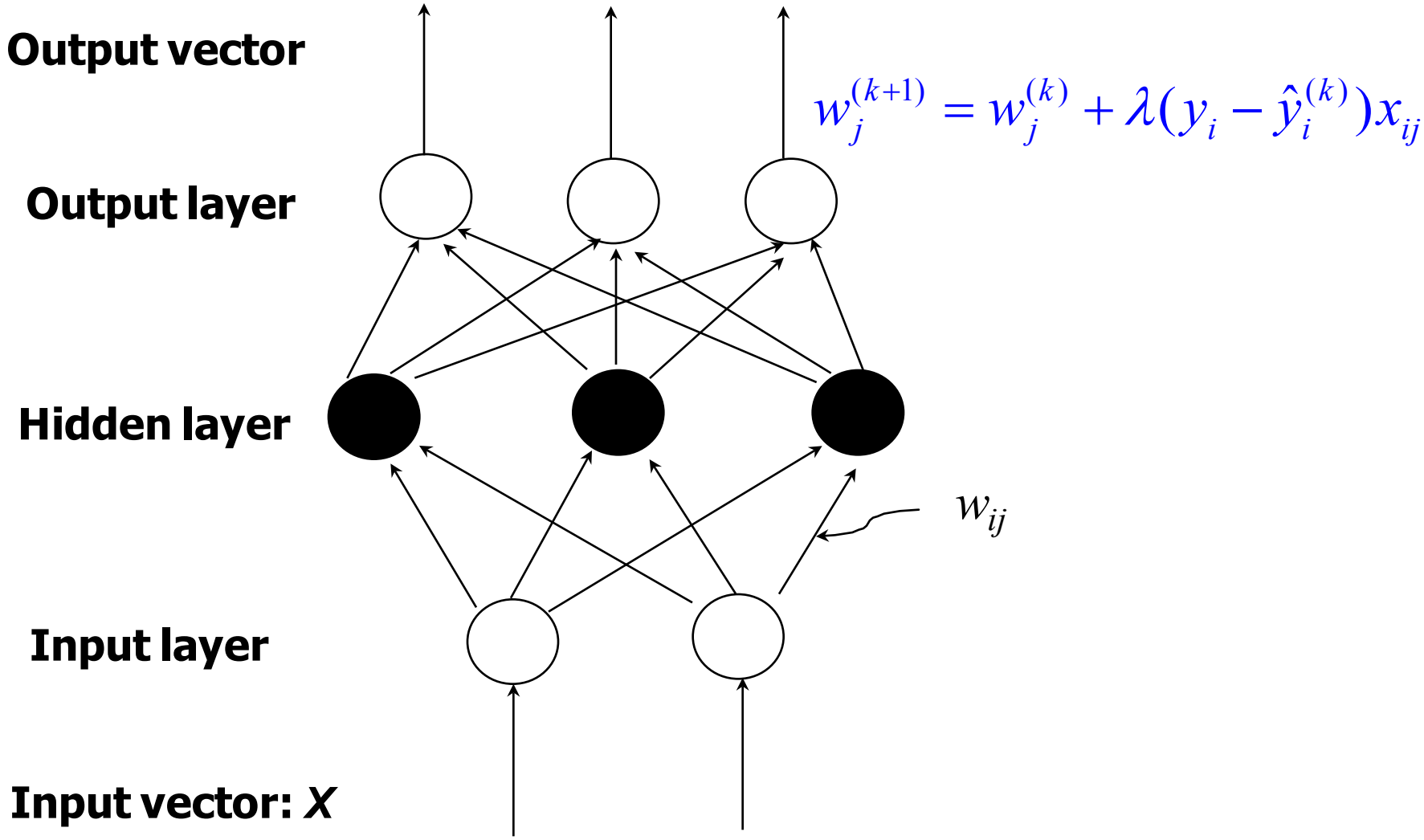
- The *n*-dimensional input vector **x** is mapped into variable y by means of the scalar product and a nonlinear function mapping

Neural Network: perceptron

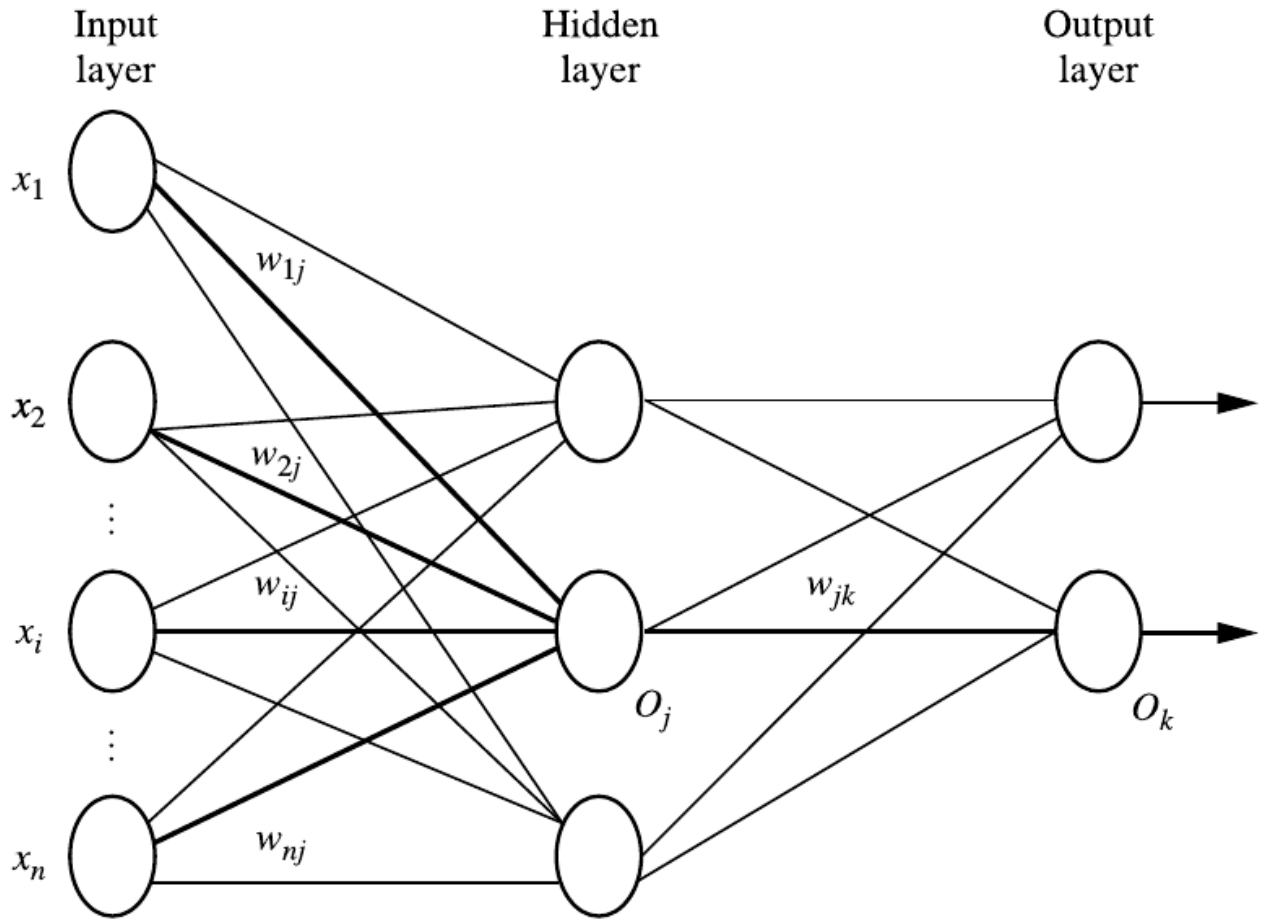




A Multi-Layer Feed-Forward Neural Network



A Multi-Layer Feed-Forward Neural Network



How A Multi-Layer Neural Network Works

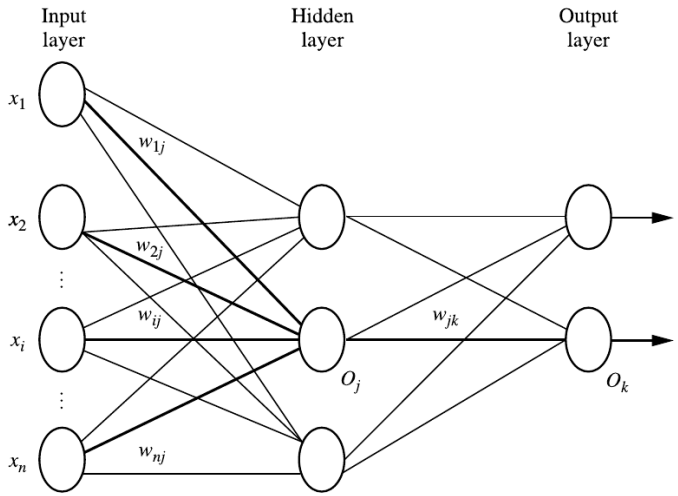
- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of hidden layers is arbitrary, usually only one
- 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.)

Algorithm: Backpropagation

Algorithm: Backpropagation. Neural network learning for classification or numeric prediction, using the backpropagation algorithm.

Input:

- D, a data set consisting of the training tuples and their associated target values;
- l, the learning rate;
- *network*, a multilayer feed-forward network.

Output: A trained neural network.

Method:

(1) Initialize all weights and biases in *network*;
(2) **while** terminating condition is not satisfied {
(3) **for** each training tuple $\boldsymbol{X}$ in D {
(4) // Propagate the inputs forward:
(5) **for** each input layer unit j {
(6) $O_j = I_j$; // output of an input unit is its actual input value
(7) **for** each hidden or output layer unit j {
(8) $I_j = \sum_i w_{ij} O_i + \theta_j$; //compute the net input of unit j with respect to the previous layer, i
(9) $O_j = \frac{1}{1+e^{-I_j}}$; } // compute the output of each unit j
(10) // Backpropagate the errors:
(11) **for** each unit j in the output layer
(12) $Err_j = O_j(1 - O_j)(T_j - O_j)$; // compute the error
(13) **for** each unit j in the hidden layers, from the last to the first hidden layer
(14) $Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$; // compute the error with respect to the next higher layer, k
(15) **for** each weight w_{ij} in *network* {
(16) $\Delta w_{ij} = (l) Err_j O_i$; // weight increment
(17) $w_{ij} = w_{ij} + \Delta w_{ij}$; } // weight update
(18) **for** each bias θ_j in *network* {
(19) $\Delta \theta_j = (l) Err_j$; // bias increment
(20) $\theta_j = \theta_j + \Delta \theta_j$; } // bias update
(21) } }

Backpropagation

- Initialize weights (to small random #s) and biases in the network
  - The weights in the network are initialized to small random numbers (e.g., ranging from −1.0 to 1.0, or −0.5 to 0.5).
  - Each unit has a bias associated with it, it is also initialized to small random numbers
- Propagate the inputs forward
  - the training tuple is fed to the input layer of the network, the inputs pass through the input units, unchanged
  - the net input and output of each unit in the hidden and output layers are computed: as a linear combination of its inputs
  - each unit in the hidden and output layers takes its net input and then applies an activation function to it

Backpropagation

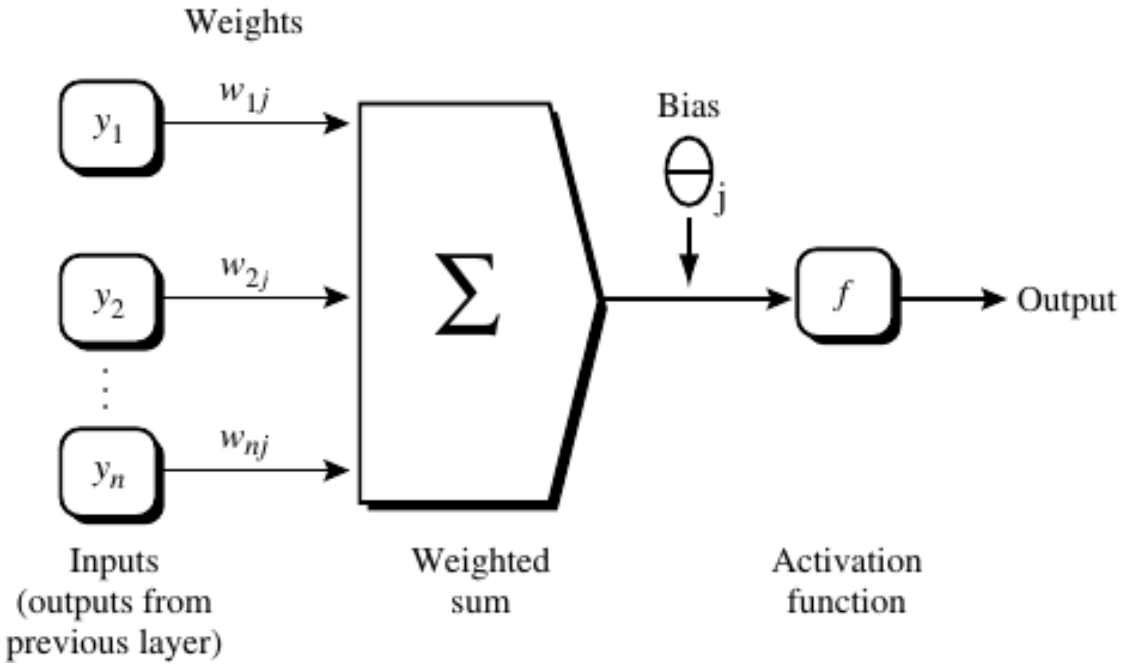
- Propagate the inputs forward
  - for an input unit j, its output O_j is its input I_j
  - for a unit j in a hidden or output layer, its net input is

$$I_j = \sum_i w_{ij} O_i + \theta_j$$

  - the output of unit j is

$$O_j = f(I_j) = \frac{1}{1+e^{-I_j}}$$

$$O_j = f(I_j) = \frac{e^{I_j} - e^{-I_j}}{e^{I_j} + e^{-I_j}}$$



Backpropagation

- Back Propagate the error
  - the error is propagated backward by updating the weights and biases to reflect the error of the network's prediction
  - for unit j in the output layer, the error Err_j is computed as

$$Err_j = O_j(1-O_j)(T_j - O_j)$$

  - for a hidden layer unit j, its error:

$$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$$

  - weights are updated as

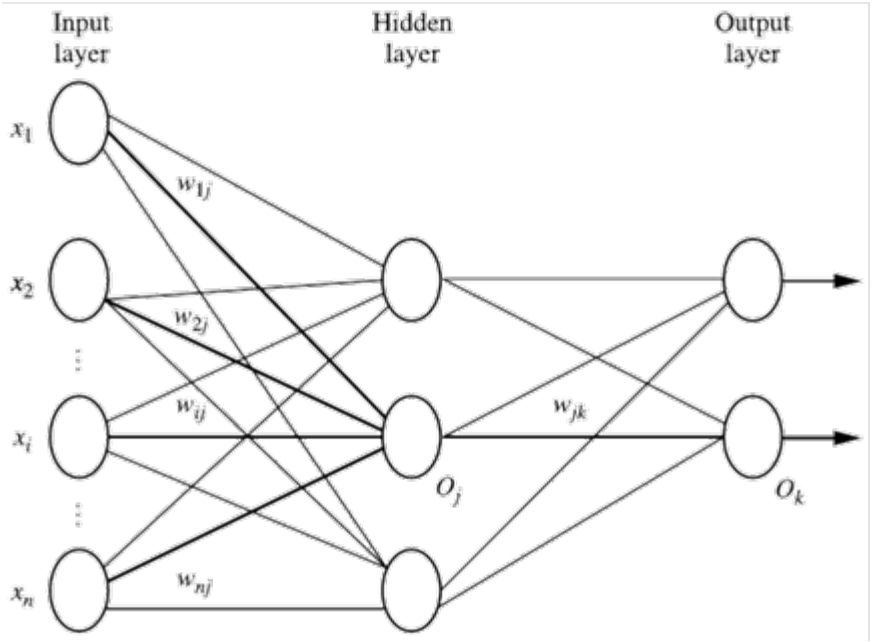
$$\Delta w_{ij} = (l) Err_j w_{jk}$$

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

  - biases are updated as

$$\Delta \theta_j = (l) Err_j$$

$$\theta_j = \theta_j + \Delta \theta_j$$

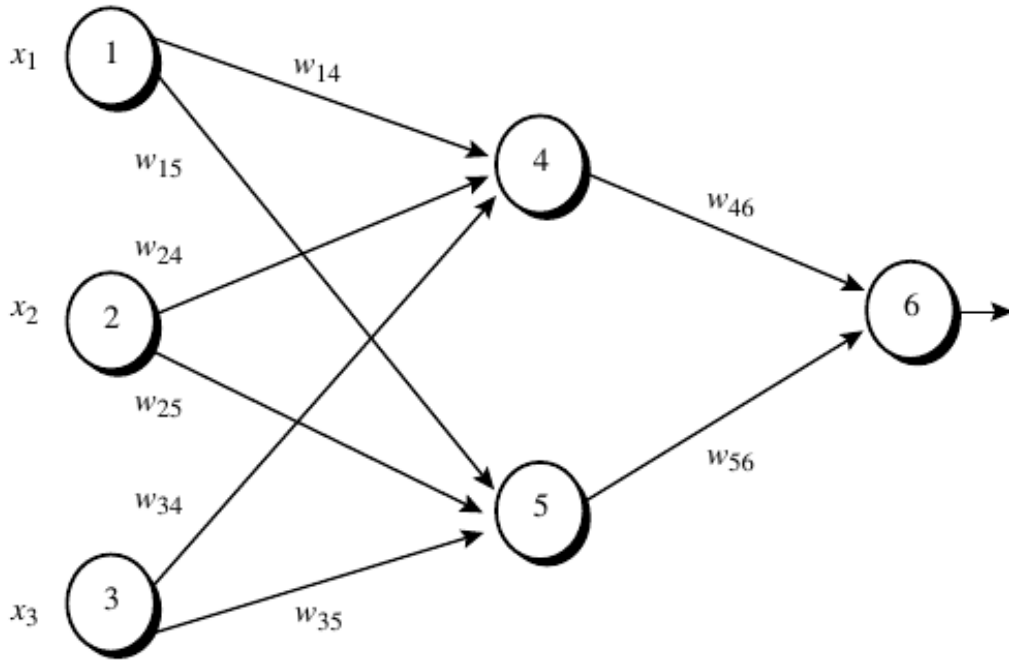


Backpropagation

- Termination condition
  - all Δw_{ij} in the previous epoch were so small as to be below some specific threshold
  - the percentage of tuples misclassified in the previous epoch is below some threshold
  - a prespecified number of epochs has expired

Example

- In the neuro network following, let the learning rate be 0.9. The initial weight and bias values of the network are given in the table, along with the first training tuple, X = (1, 0, 1), whose class label is 1.



w_{14}	0.2	w_{15}	-0.3
w_{24}	0.4	w_{25}	0.1
w_{34}	-0.5	w_{35}	0.2
w_{46}	-0.3	w_{56}	-0.2
θ_4	-0.4	θ_5	0.2
θ_6	0.1		

Example

- the net input and output calculations

Unit j	Net input, I_j	Output, O_j
4	0.2+0-0.5-0.4=-0.7	$1/(1+e^{0.7})=0.332$
5	-0.3+0+0.2+0.2=0.1	$1/(1+e^{-0.1})=0.525$
6	(-0.3)(0.332)-(0.2)(0.525) +0.1 =-0.105	$1/(1+e^{0.105})=0.474$

- calculations of the error at each node

Unit j	Err_j
6	(0.474)(1 – 0.474)(1 – 0.474) = 0.1311
5	(0.525)(1 – 0.525)(0.1311)(–0.2) = –0.0065
4	(0.332)(1 – 0.332)(0.1311)(–0.3) = –0.0087

Calculations for weight and bias updating

weight/bias	New value
w_{46}	$-0.3 + (0.9)(0.1311)(0.332) = -0.261$
w_{56}	$-0.2 + (0.9)(0.1311)(0.525) = -0.138$
w_{14}	$0.2 + (0.9)(-0.0087)(1) = 0.192$
w_{15}	$-0.3 + (0.9)(-0.0065)(1) = -0.306$
w_{24}	$0.4 + (0.9)(-0.0087)(0) = 0.4$
w_{25}	$0.1 + (0.9)(-0.0065)(0) = 0.1$
w_{34}	$-0.5 + (0.9)(-0.0087)(1) = -0.508$
w_{35}	$0.2 + (0.9)(-0.0065)(1) = 0.194$
θ_6	$0.1 + (0.9)(0.1311) = 0.218$
θ_5	$0.2 + (0.9)(-0.0065) = 0.194$
θ_4	$-0.4 + (0.9)(-0.0087) = -0.408$

Efficiency and Interpretability

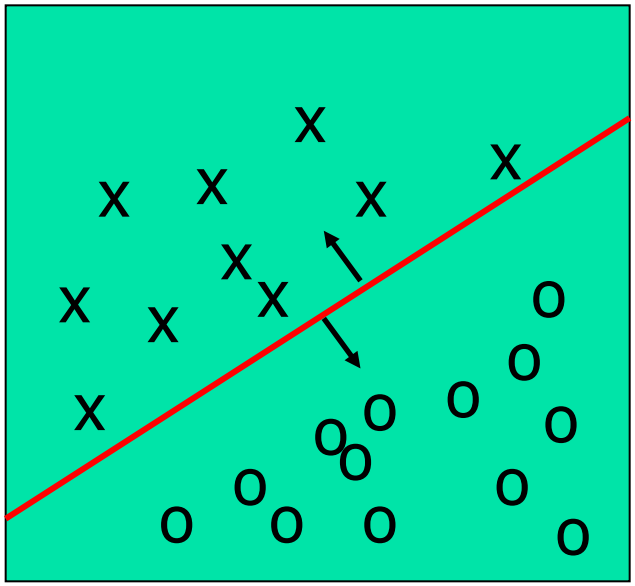
- **Efficiency** of backpropagation: Each epoch (one iteration through the training set) takes $O(|D| \times w)$, with |D| tuples and w weights, but # of epochs can be exponential to n, the number of inputs, in worst case
- For easier comprehension: **Rule extraction** by network pruning
  - Simplify 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
    - $\mathbf{x} = (x_1, x_2, x_3, \ldots)$, $y_i = +1$ or -1
    - x_1 : # of word "homepage"
    - x_2 : # of word "welcome"
- Mathematically, $x \in X = R^n$, $y \in Y = \{+1, -1\}$
  - 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



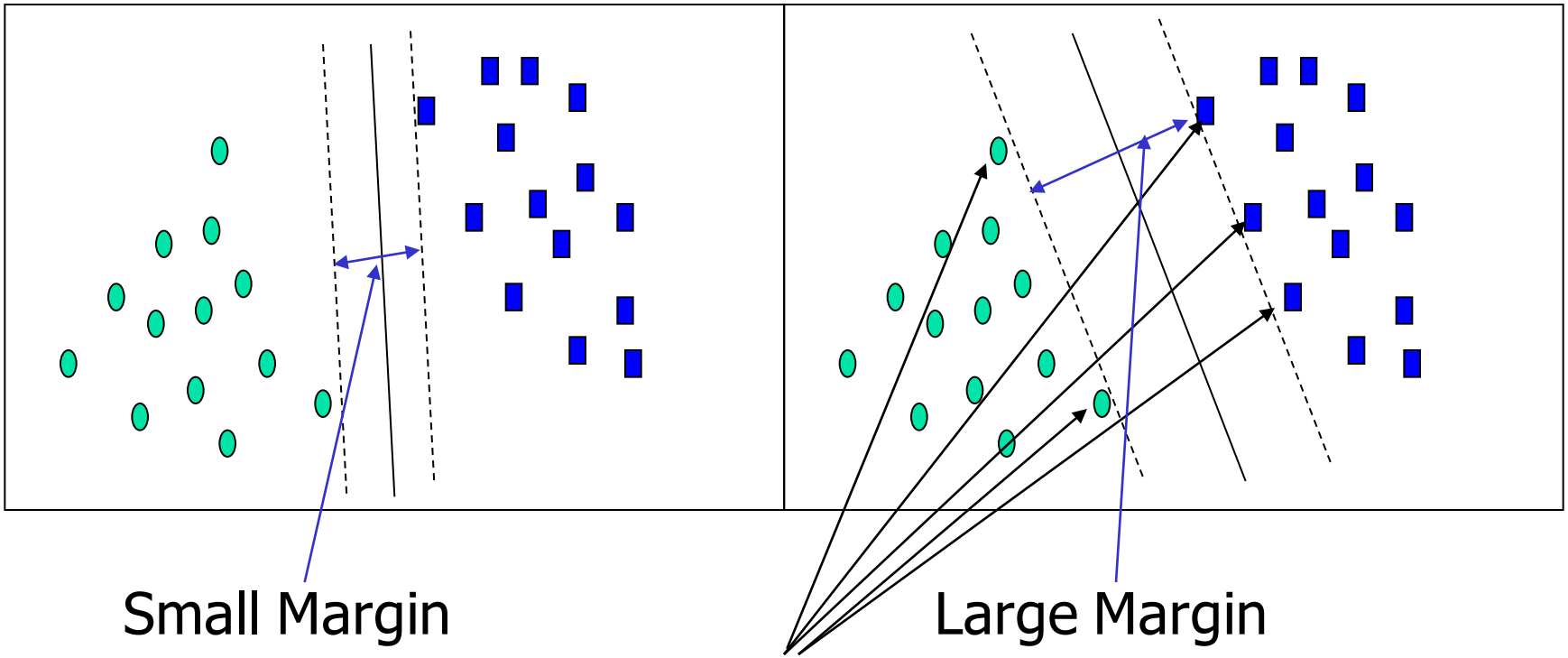
SVM—Support Vector Machines

- 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)

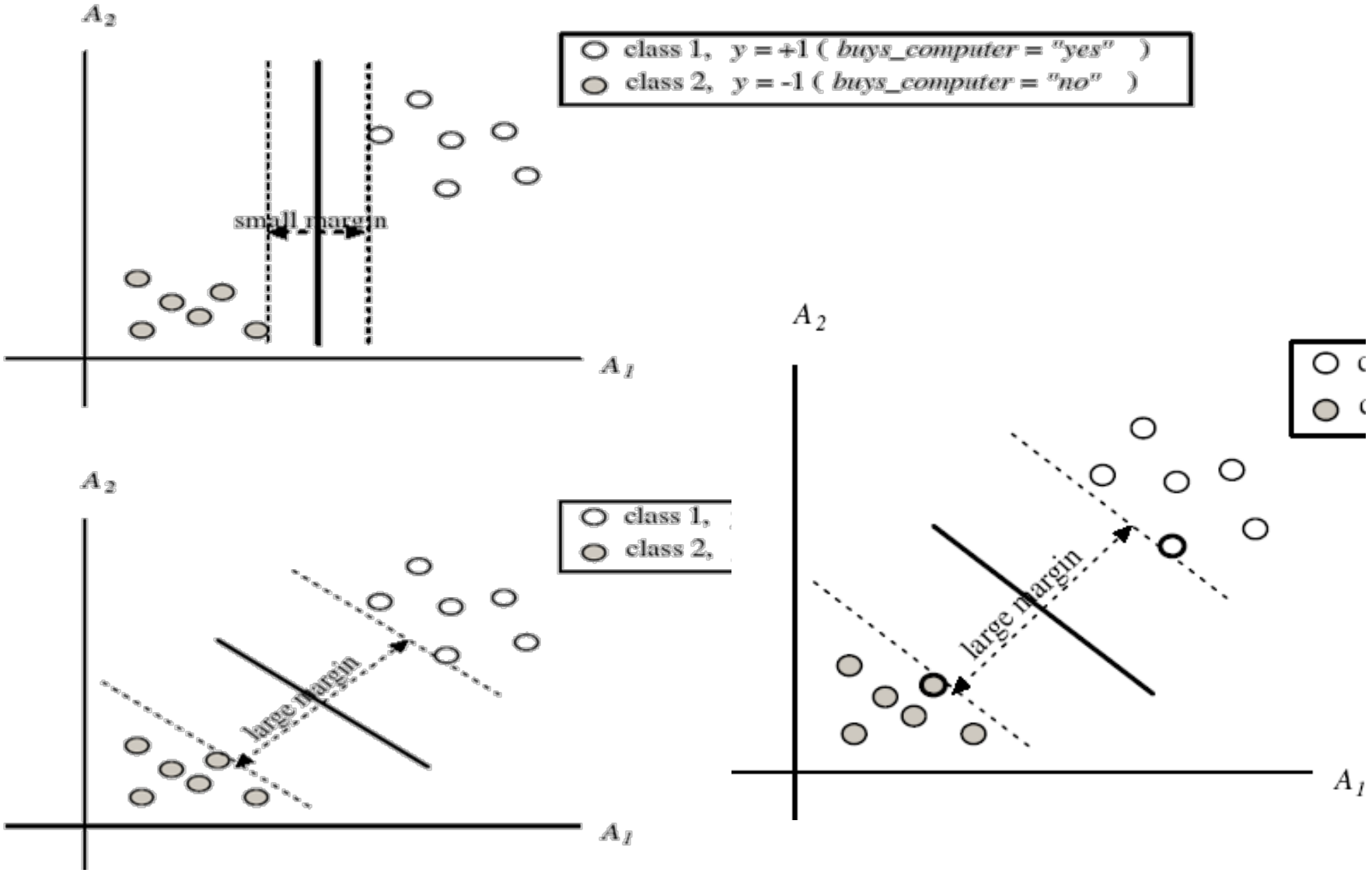
SVM—History and Applications

- 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

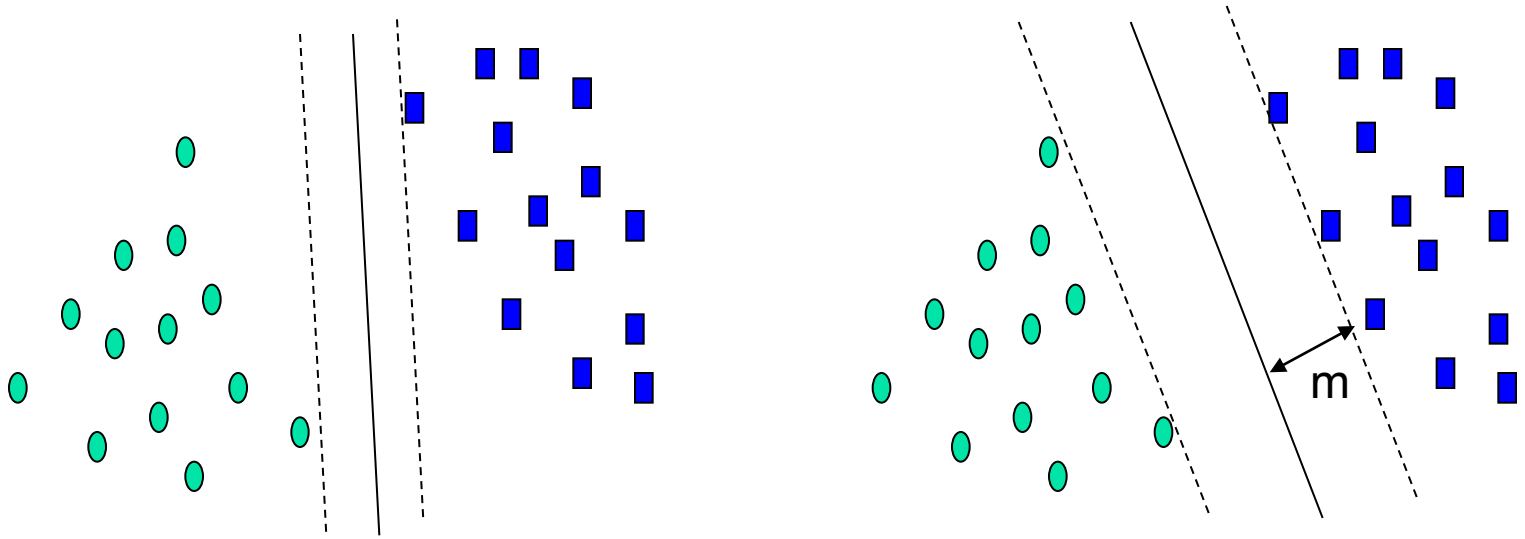
SVM—General Philosophy



SVM—Margins and Support Vectors



SVM—When Data Is Linearly Separable



Let data D be $(\mathbf{X}_1, y_1), \ldots, (\mathbf{X}_{|D|}, y_{|D|})$, 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

  $\mathbf{w} \cdot \mathbf{X} + \mathbf{b} = 0$

  where $w=\{w_1, w_2, \ldots, w_n\}$ is a weight vector and **b** a scalar (bias)
- For 2-D it can be written as

  $\mathbf{w_0 + w_1 x_1 + w_2 x_2 = 0}$
- The hyperplane defining the sides of the margin:

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

  $\mathbf{H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1}$ for $y_i = -1$
- Any training tuples that fall on hyperplanes $\mathbf{H_1}$ or $\mathbf{H_2}$ (i.e., the sides defining the margin) are **support vectors**

SVM—Linearly Separable

- This becomes a **constrained (convex) quadratic optimization** problem: Quadratic objective function and linear constraints → *Quadratic Programming (QP)* → Lagrangian multipliers

$$\min \frac{1}{2}\|w\|^2$$

$$s.t. \quad y_i(w^T x_i + b) \geq 1 \quad i = 1, 2, \cdots, n$$

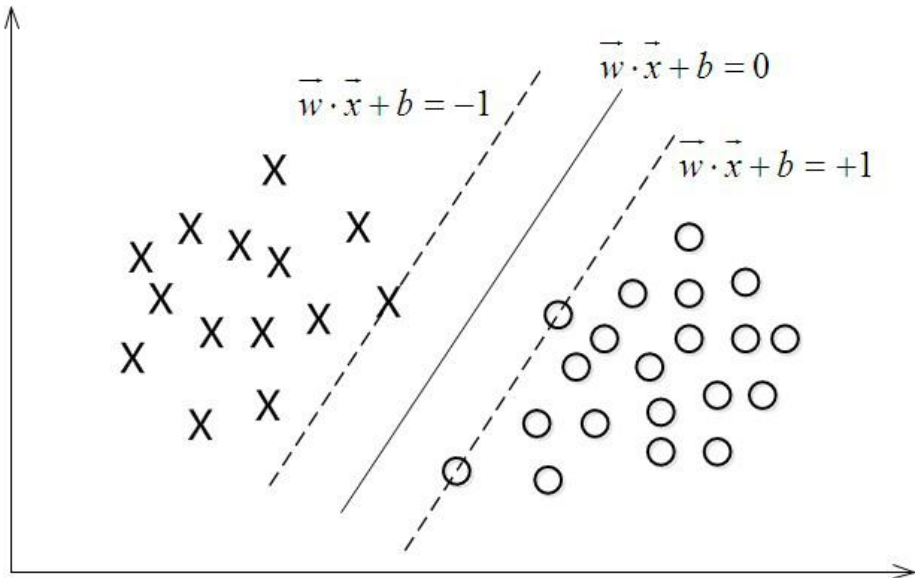
- The final model

$$w^T x + b = \left(\sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \right)^T x + b$$

$$= \sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x \rangle + b$$

SVM的理论基础

- 线性可分的情况：寻找最大边界超平面
  - **$f(x)=w^Tx+b$**
  - f(x)=0 x是超平面上的点
  - f(x)>1 x是正类的点
  - f(x)<1 x是负类的点



SVM的理论基础

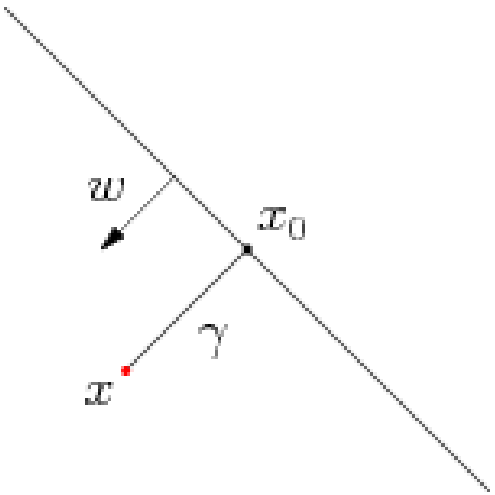
- 在平面$w^Tx+b=0$确定的情况下，$|w^Tx+b|$即为点x到该平面的距离
- 记 $\hat{\gamma} = y(w^T x + b) = y \cdot f(x)$ 为$|w^Tx+b|$，该距离为函数间隔
- 训练集T中的样本点(x_i,y_i)的函数间隔的最小值定义为超平面f(x)关于训练集T的函数间隔 $\hat{\gamma} = \min \hat{\gamma}_i \quad i = 1, 2, \cdots, n$
- w^T和b的等比例放大时，对应还是同一个超平面，但函数间隔的值也会成比例增长，趋近于无穷大
- **几何间隔**

SVM的理论基础

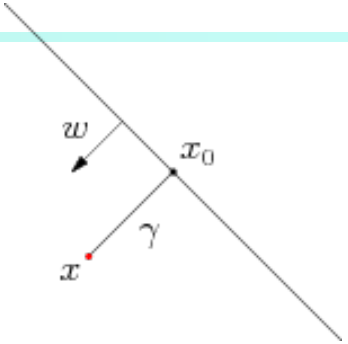
- 假定对于一个点 x，令其垂直投影到超平面上的对应点为 x_0，w是垂直于超平面的一个向量，样本x到分类间隔的距离记为 γ，根据平面几何知识，有关系 $x = x_0 + \gamma \frac{w^T}{\|w\|}$
- x_0为超平面上的点，满足超平面方程

  $f(x_0)=0$
- 代入$w^Tx+b=0$，即可得到

$$\gamma = \frac{w^T x + b}{\|w\|} = \frac{f(x)}{\|w\|}$$



SVM的理论基础



- 考虑其绝对值

$$\tilde{\gamma} = y \cdot \gamma = \frac{\tilde{\gamma}}{\|w\|}$$

该间隔定义为**几何间隔**，代表点到超平面的距离

- 对数据点进行分类，当超平面离数据点的“间隔”越大，分类的确信度也越大
- 为了使得分类的确信度尽量高，需要让所选择的超平面能够最大化这个“间隔”值
- 函数间隔不适合用于最大化这个间隔值，而几何间隔适合

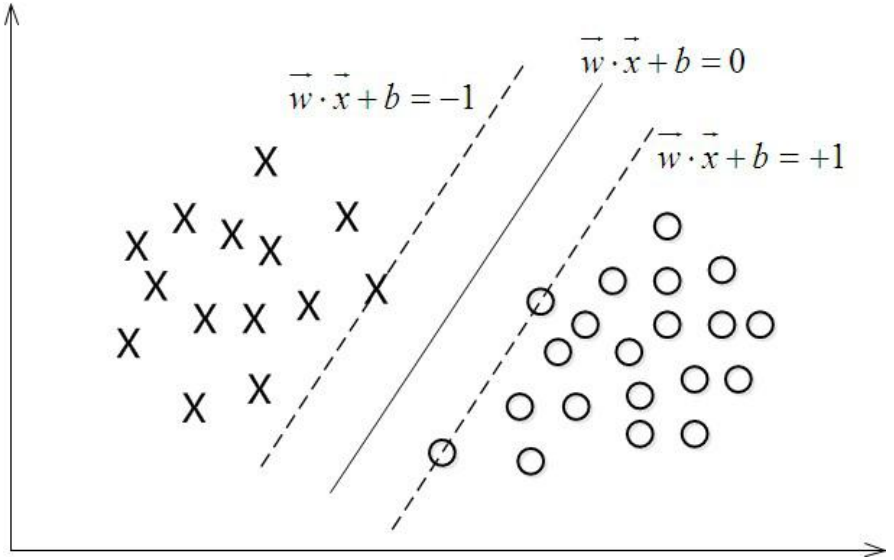
SVM的理论基础

- 最大间隔分类器的目标函数可定义为

$$\max \tilde{\gamma}$$

$$s.t. \quad y_i(w^T x_i + b) = \hat{\gamma}_i \geq \hat{\gamma} \quad i = 1, 2, \cdots, n$$

- 右图中，实线表示的找到的最优超平面，位于虚线表示的平面上的点即为**支持向量**
- 两个虚线表示的超平面之间的距离为2



- 对于支持向量： $y_i(w^T x_i+b)=1$
- 对于非支持向量： $y_i(w^T x_i+b)>1$

SVM的理论基础

- 最大间隔分类器的目标函数可定义为

$$\max \tilde{\gamma}$$

$$s.t. \quad y_i(w^T x_i + b) = \hat{\gamma}_i \geq \hat{\gamma} \quad i = 1, 2, \cdots, n$$

- 可令目标函数约束条件中的 $\hat{\gamma} = 1$
- 则目标函数为

$$\max \frac{1}{\|w\|}$$

$$s.t. \quad y_i(w^T x_i + b) \geq 1 \quad i = 1, 2, \cdots, n$$

- 为了优化方便，将其转换为等价的目标函数

$$\min \frac{1}{2}\|w\|^2$$

$$s.t. \quad y_i(w^T x_i + b) \geq 1 \quad i = 1, 2, \cdots, n$$

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

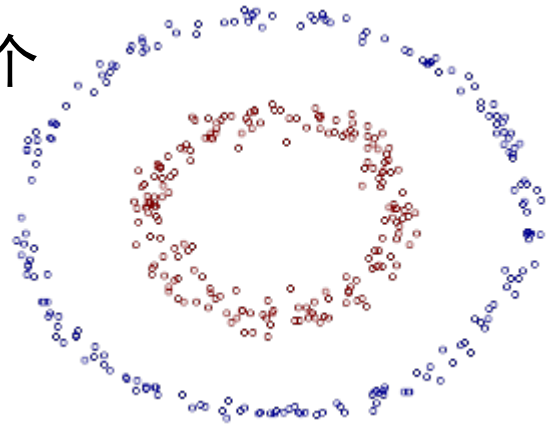
- Transform the original input data into a higher dimensional space
- Nonlinear transformation of original input data into a higher dimensional space.
- Example: Consider the following example. A 3-D input vector $\mathbf{X=(x_1, x_2, x_3)}$ is mapped into a 6-D space, Z, using the mappings $\mathbf{\varphi_1(X)=x_1}$, $\mathbf{\varphi_2(X)=x_2}$, $\mathbf{\varphi_3(X)=x_3}$, $\mathbf{\varphi_4(X)=(x_1)^2}$, $\mathbf{\varphi_5(X)=x_1x_2}$, and $\mathbf{\varphi_6(X)=x_1x_3}$. A decision hyperplane in the new space is $\mathbf{d(Z)=WZ + b}$, where **W** and **Z** are vectors.

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

- Search for a linear separating hyperplane in the new space

SVM—Linearly Inseparable

- 右图包含两类数据，分布形状分别为两个圆圈——非线性可分



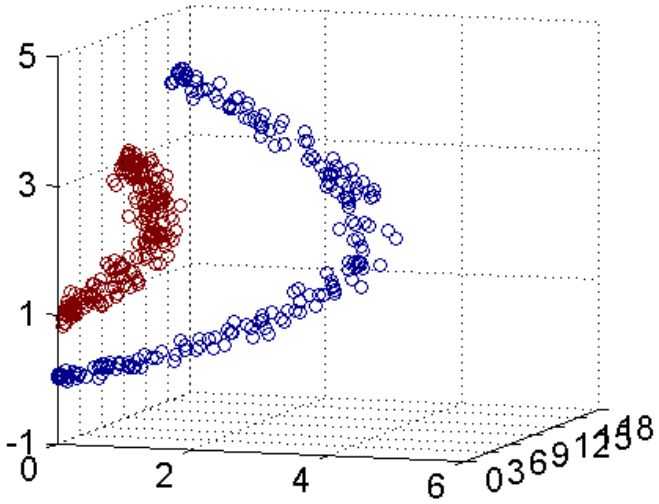
- 用x_1和x_2表示二维平面的坐标

$$a_1x_1 + a_2x_1^2 + a_3x_2 + a_4x_2^2 + a_5x_1x_2 + a_6 = 0$$

- 可构造一个5-d空间，坐标分别为

$$z_1 = x_1 \quad z_2 = x_1^2 \quad z_3 = x_2$$
$$z_4 = x_2^2 \quad z_5 = x_1x_2$$

- 右图为三维的情况



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., $K(\mathbf{X}_i, \mathbf{X}_j) = \Phi(\mathbf{X}_i)\ \Phi(\mathbf{X}_j)$
- 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:

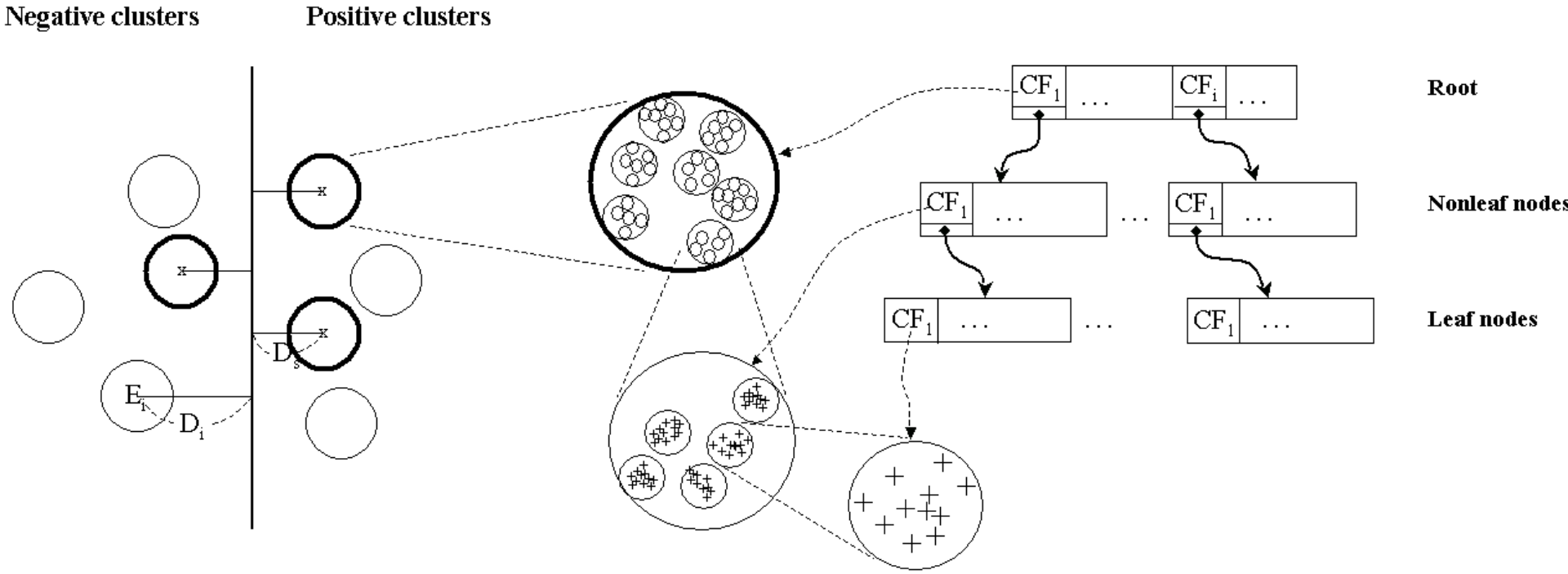
$$K(X_i, X_j) = tanh(kX_i \cdot X_j - \delta)$$

- 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
- H. Yu, J. Yang, and J. Han, "Classifying Large Data Sets Using SVM with Hierarchical Clusters", 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

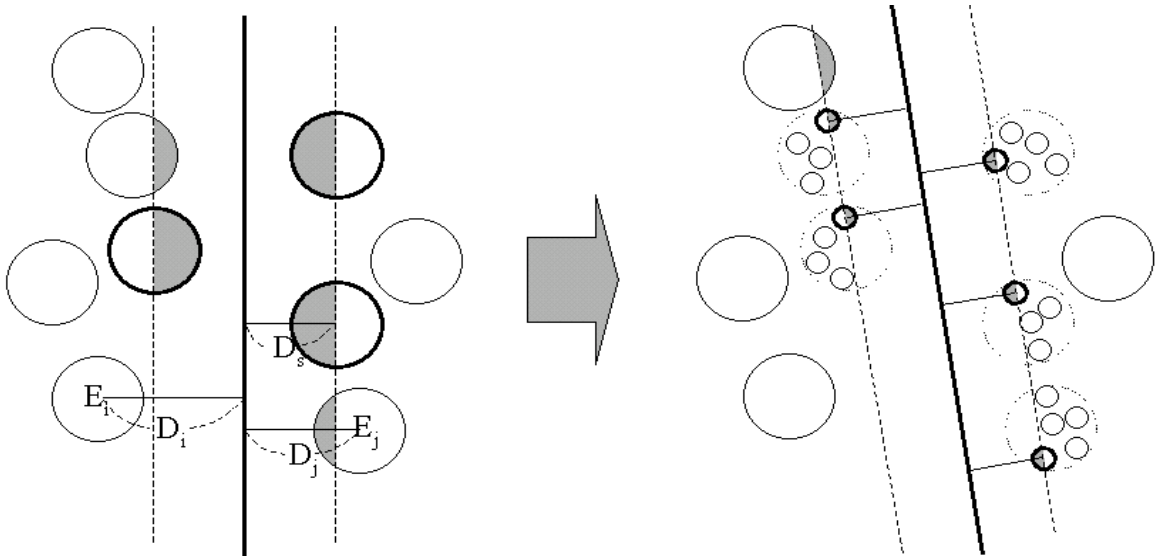
CF-Tree: Hierarchical Micro-cluster



- 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

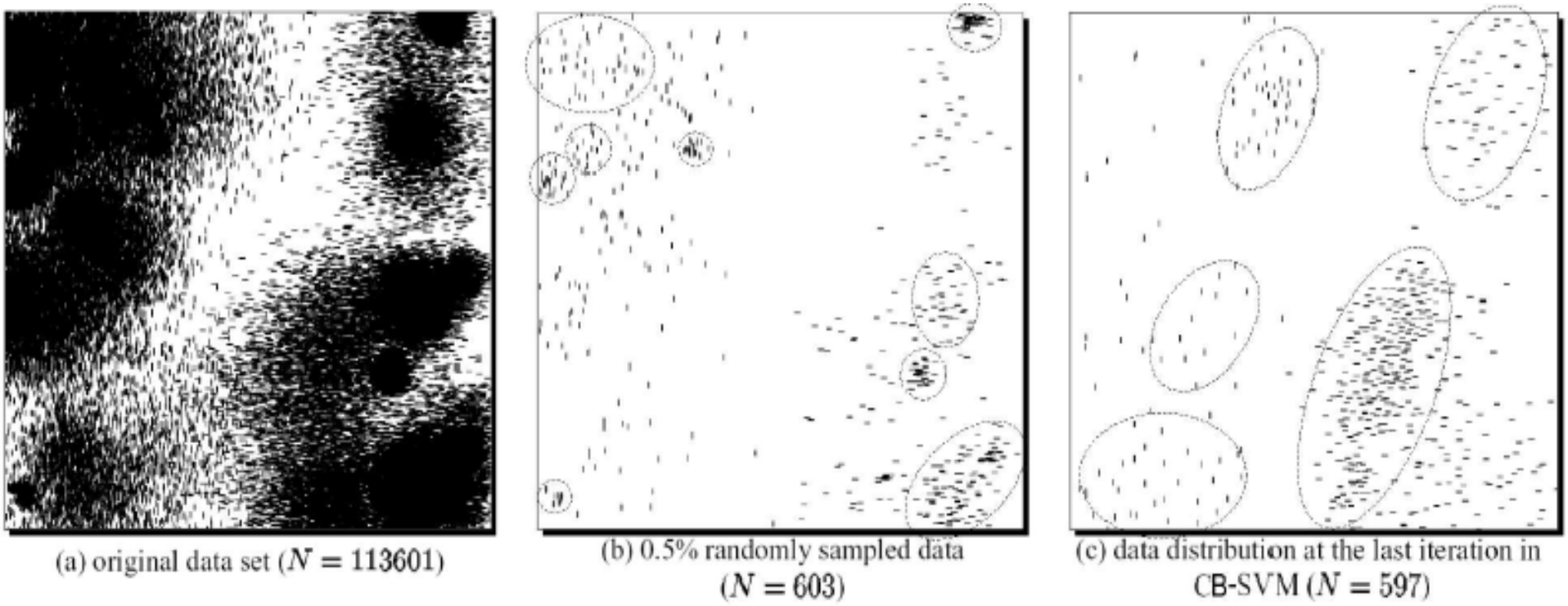
- CF tree is a suitable base structure for selective declustering
- De-cluster only the cluster E_i such that
  - $D_i - R_i < D_s$, where D_i is the distance from the boundary to the center point of E_i and R_i is the radius of E_i
  - Decluster only the cluster whose subclusters have possibilities 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



(a) original data set ($N = 113601$)

(b) 0.5% randomly sampled data ($N = 603$)

(c) data distribution at the last iteration in CB-SVM ($N = 597$)

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

    $P_1 \wedge P_2 \ldots \wedge P_l \rightarrow$ "A_{class}=C " (conf, 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

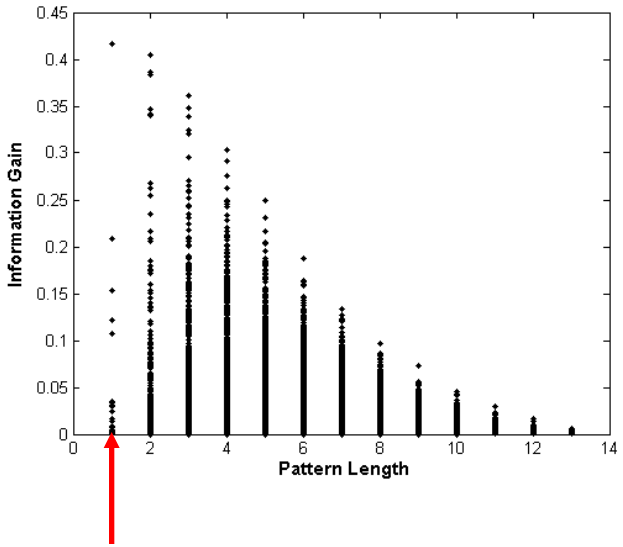
- **CBA** (Classification Based on Associations: Liu, Hsu & Ma, KDD'98)
  - Mine possible association rules in the form of
    - Cond-set (a set of attribute-value pairs) → class label
  - Build classifier: Organize rules according to decreasing precedence based on confidence and then support
- **CMAR** (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

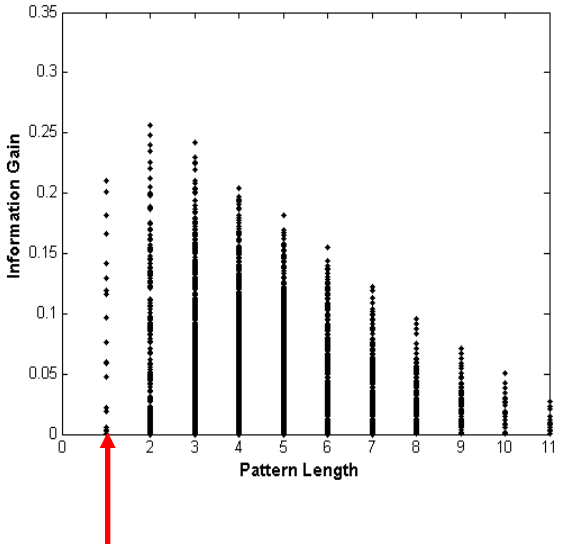
- H. Cheng, X. Yan, J. Han, and C.-W. Hsu, "Discriminative Frequent Pattern Analysis for Effective Classification", ICDE'07
- Accuracy issue
  - Increase the discriminative power
  - Increase the expressive power of the feature space
- Scalability issue
  - It is computationally infeasible to generate 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

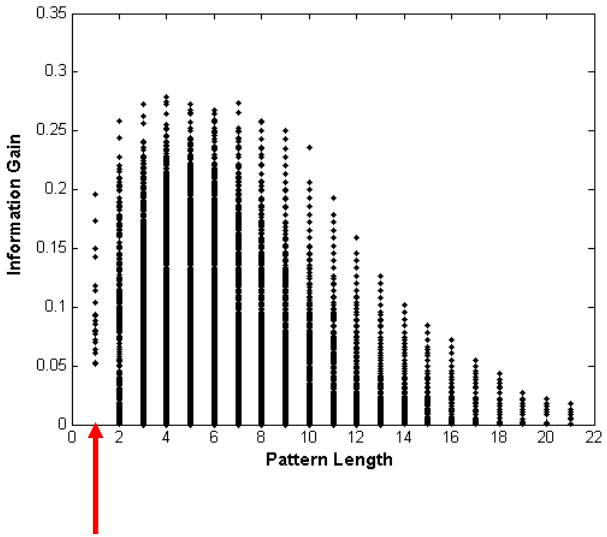
The discriminative power of some frequent patterns is higher than that of single features.



(a) Austral



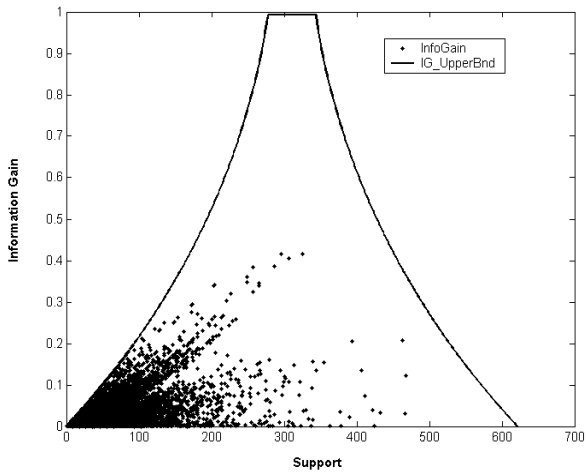
(b) Cleve



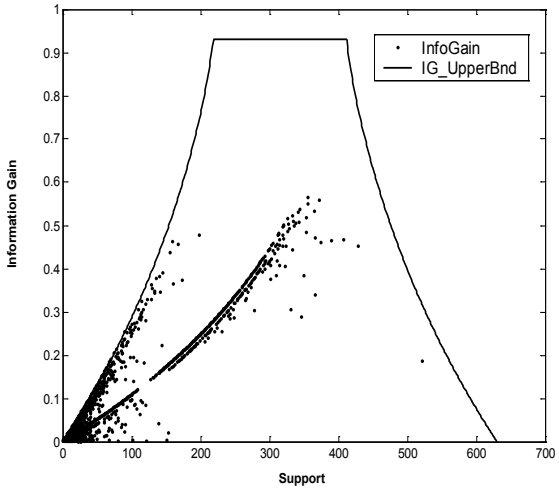
(c) Sonar

Fig. 1. Information Gain vs. Pattern Length

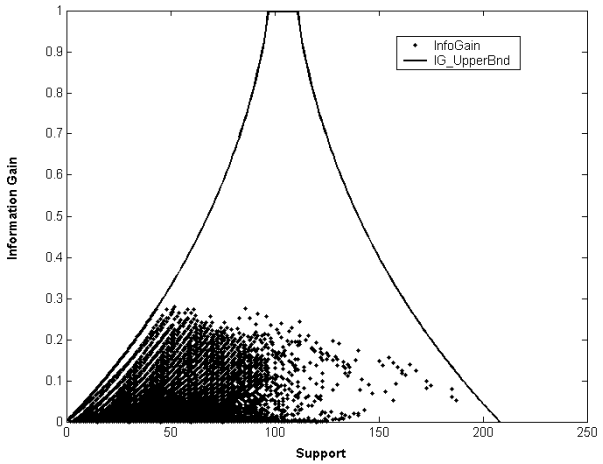
Empirical Results



(a) Austral



(b) Breast



(c) Sonar

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

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

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

Scalability Tests

Table 3. Accuracy & Time on Chess Data

min_sup	#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

min_sup	#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

$$\sup(child) \leq \sup(parent)$$

$$\sup(b) \leq \sup(a)$$

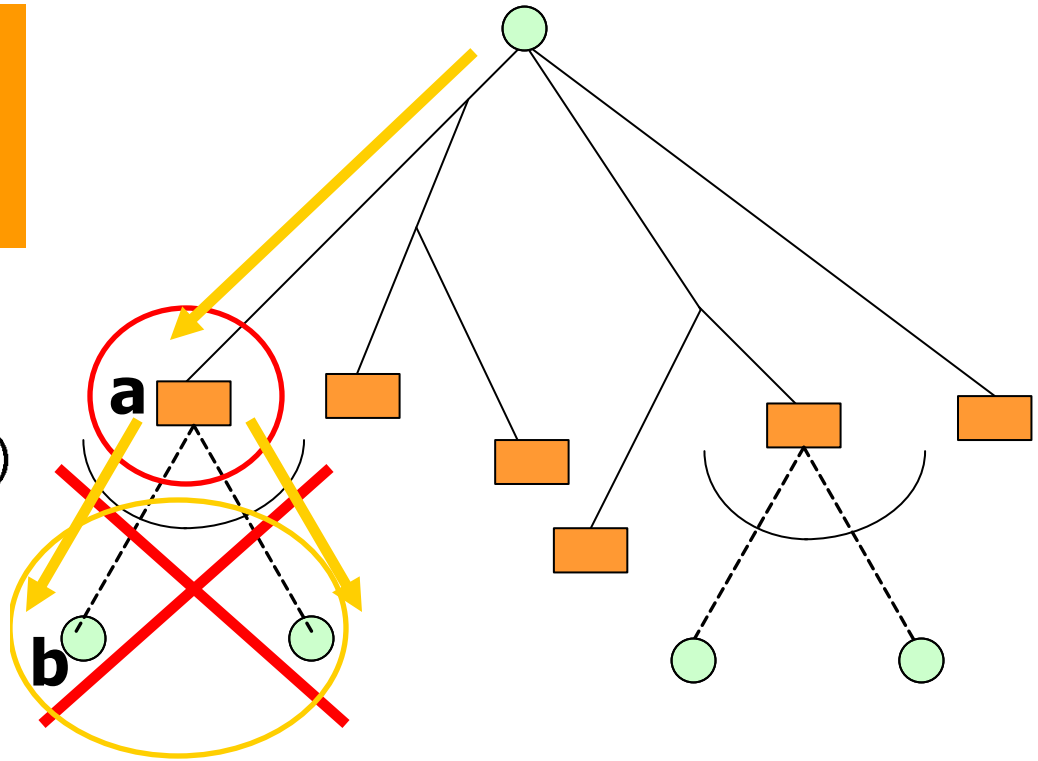
maximize $IG(C|b)$

subject to

$min\_sup \leq sup(b) \leq sup(a)$

$0 \leq sup_{+}(b) \leq sup_{+}(a)$

$0 \leq sup_{-}(b) \leq sup_{-}(a)$



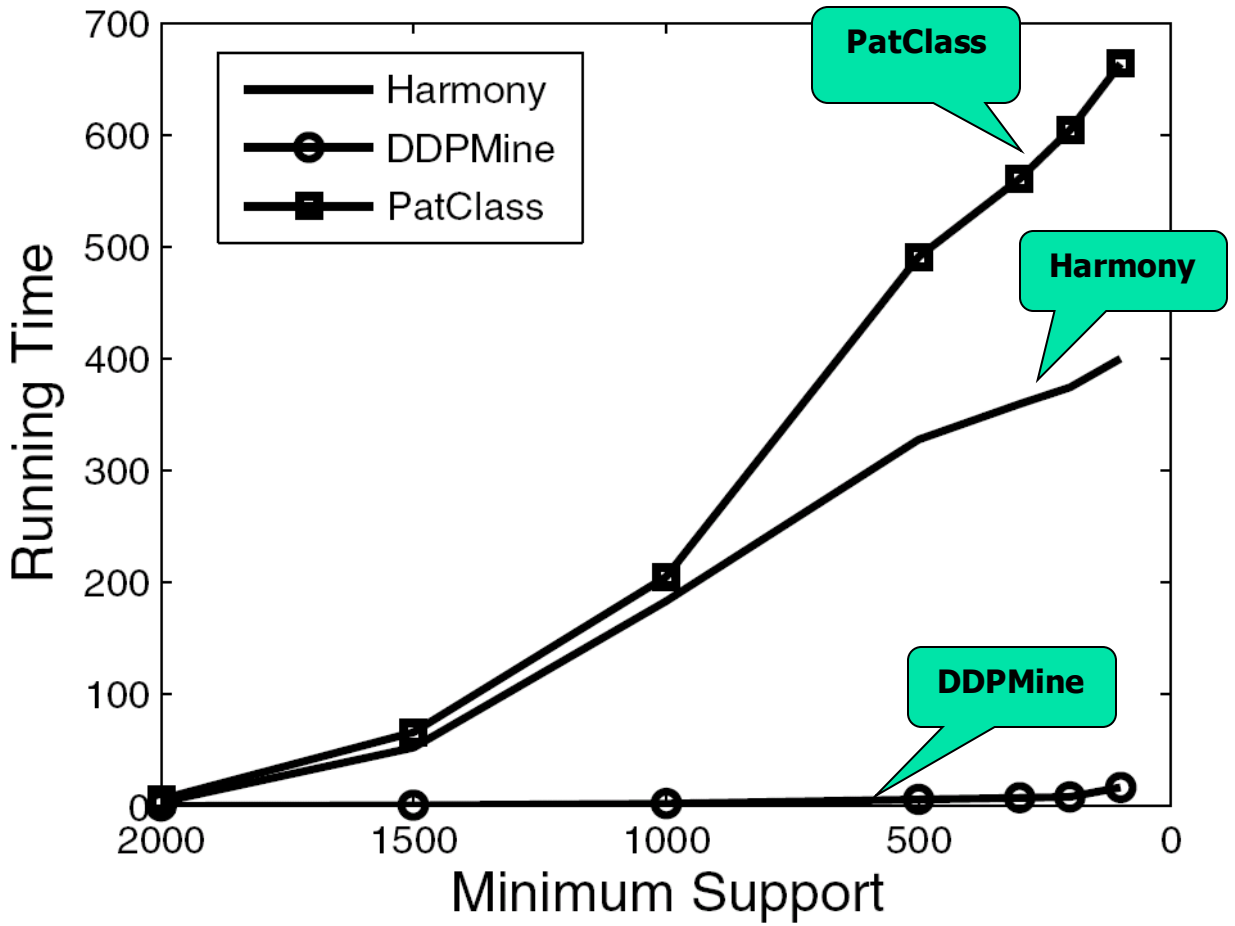
a: constant, a parent node

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

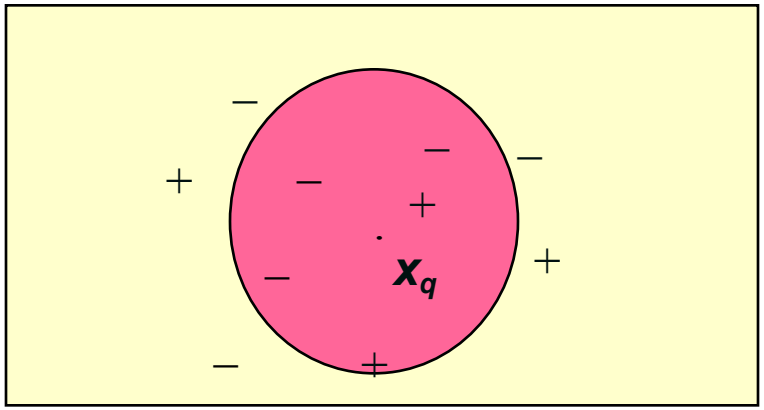
- Lazy vs. eager learning
  - **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - **Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before 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, $dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n}(x_{1i} - x_{2i})^2}$
- 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**



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 $w \equiv \frac{1}{dist(x_q, x_i)^2}$
    - 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

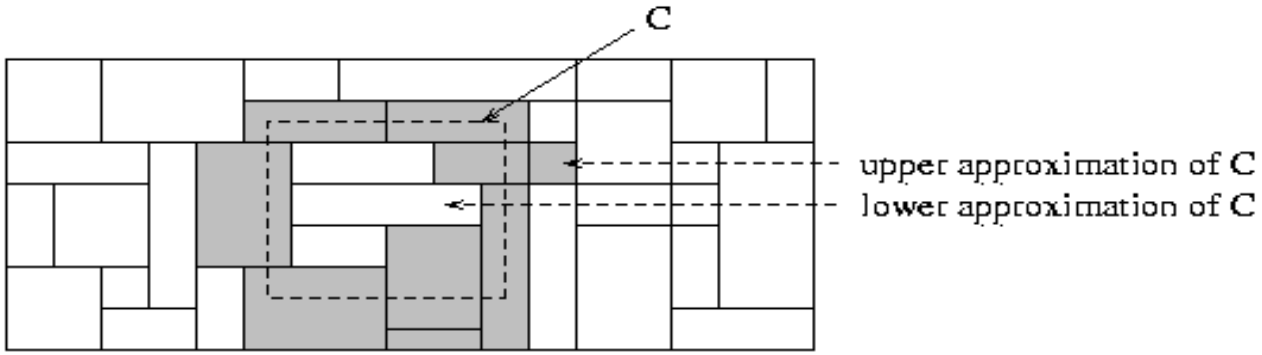
- Bayesian Belief Networks
- Classification by Backpropagation
- Support Vector Machines
- Classification by Using Frequent Patterns
- Lazy Learners (or Learning from Your Neighbors)
- Other Classification Methods
- Additional Topics Regarding Classification
- Summary

Genetic Algorithms (GA)

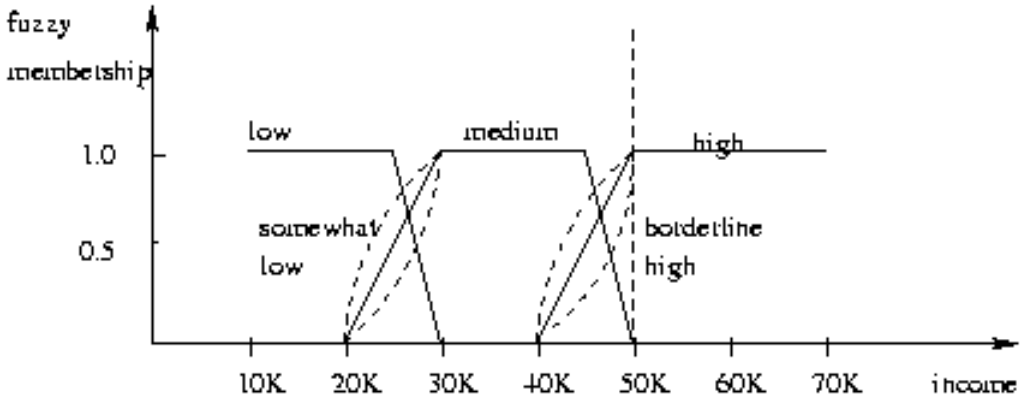
- Genetic Algorithm: based on an analogy to biological evolution
- An initial **population** is created consisting of randomly generated rules
  - Each rule is represented by a string of bits
  - E.g., if A_1 and $\neg A_2$ then C_2 can be encoded as 100
  - If an attribute has k>2 values, k bits can 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 **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 **X**, each classifier votes. **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

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

- 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 **X**, the 7-trained classifiers output: 0001010
  - Hamming distance: # of different bits between two codewords
  - H(**X**, C_1)=5, by checking # of bits between [1111111] & [0001010]
  - H(**X**, C_2)=3, H(**X**, C_3)=3, H(**X**, C_4)=1, thus C_4 as the label for **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

Semi-Supervised Classification

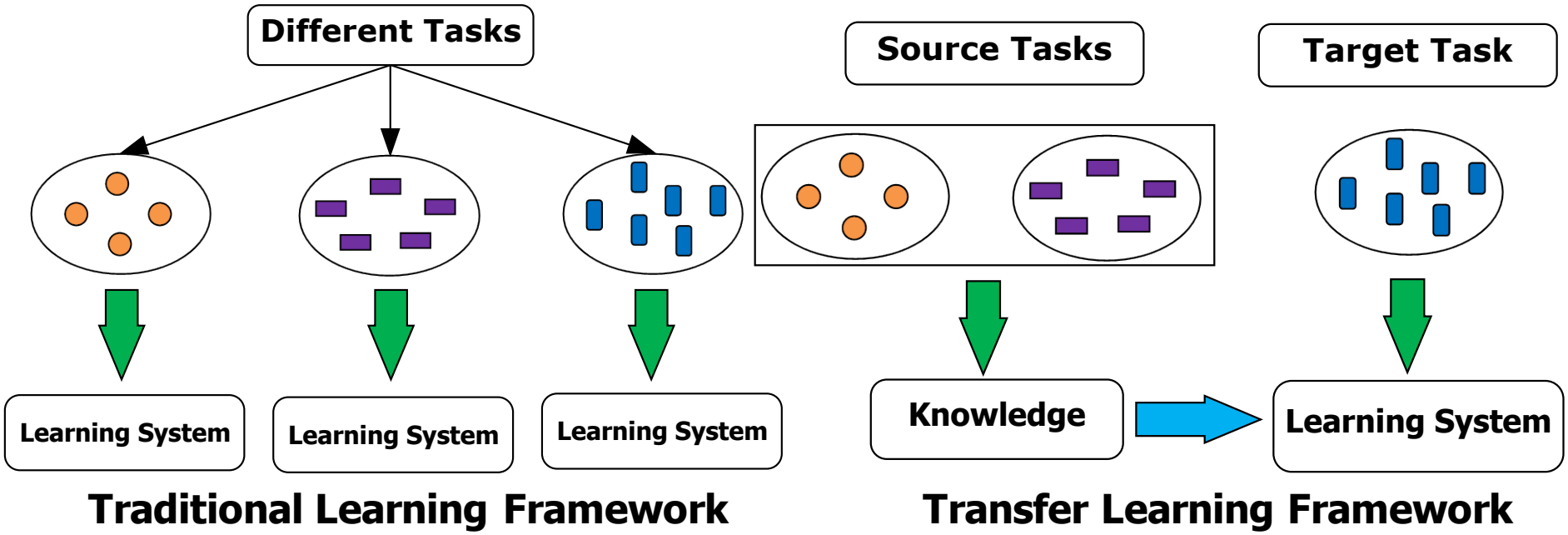
- Semi-supervised: Uses labeled and unlabeled data to build a classifier
- Self-training:
  - Build a classifier using the labeled data
  - Use it to label the unlabeled data, and those with the most confident label prediction are added to the set of labeled data
  - Repeat the above process
  - Adv: 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
  - L: a small subset of D is labeled, U: a pool of unlabeled data in D
  - Use a query function to carefully select one or more tuples from U and request labels from an oracle (a human annotator)
  - The newly labeled samples are added to 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 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 tupels 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

Surplus 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

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

- Multiple linear regression: involves more than one predictor variable
  - Training data is of the form $(\mathbf{X_1}, y_1)$, $(\mathbf{X_2}, y_2)$,…, $(\mathbf{X_{|D|}}, y_{|D|})$
  - 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

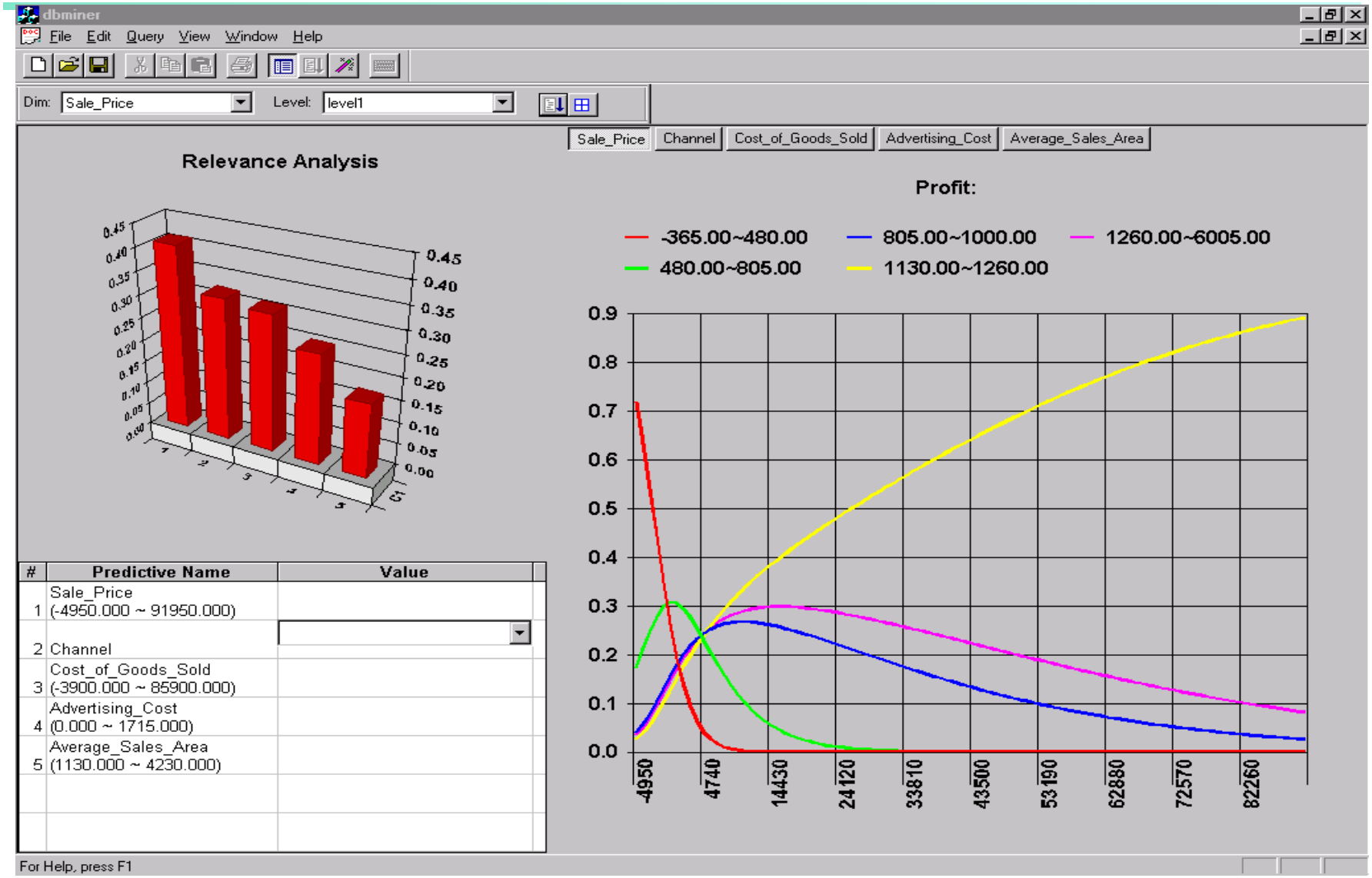
Regression Trees and Model Trees

- 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

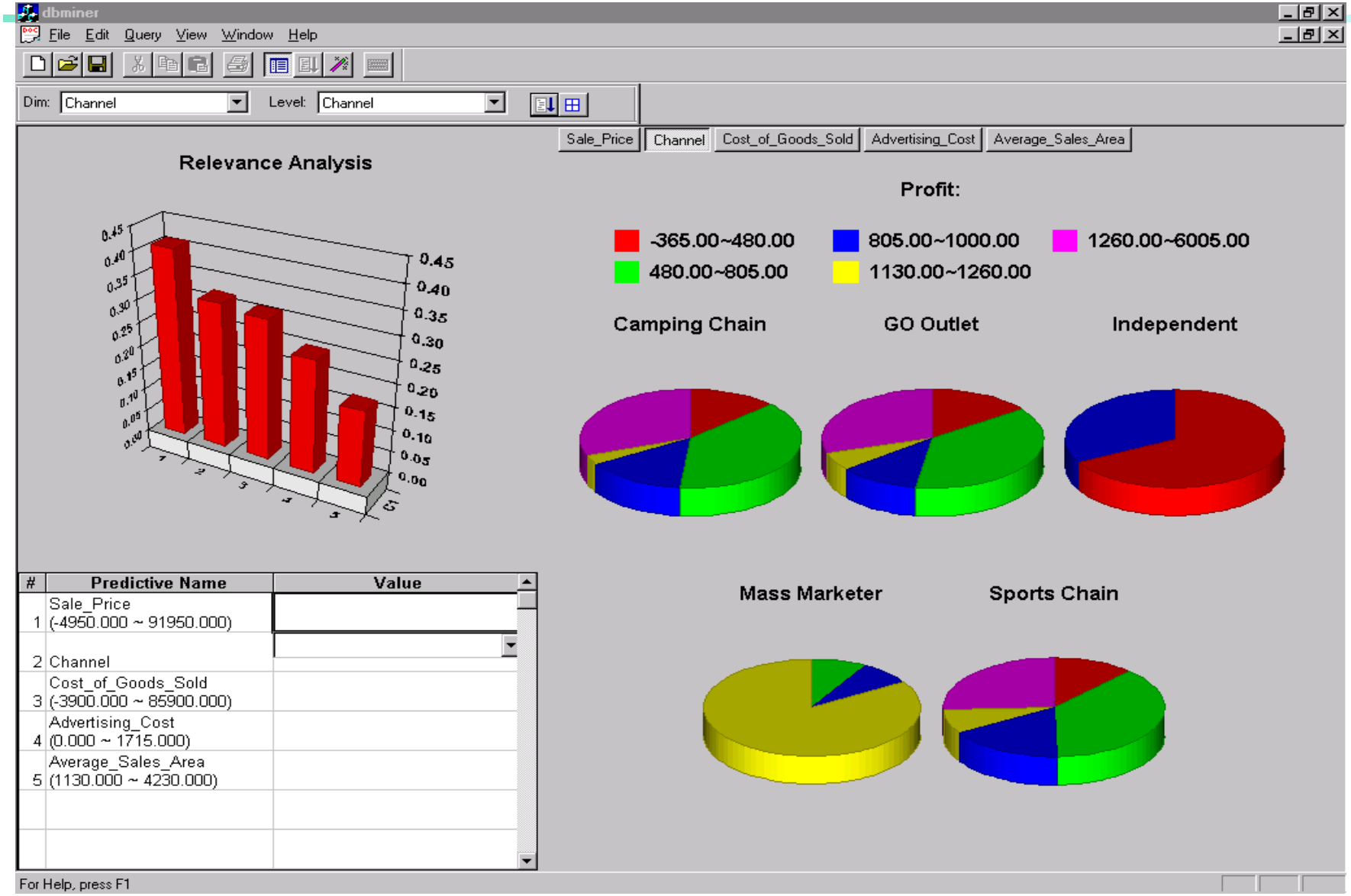
Predictive Modeling in Multidimensional Databases

- 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
  - Contains one nice chapter of SVM introduction

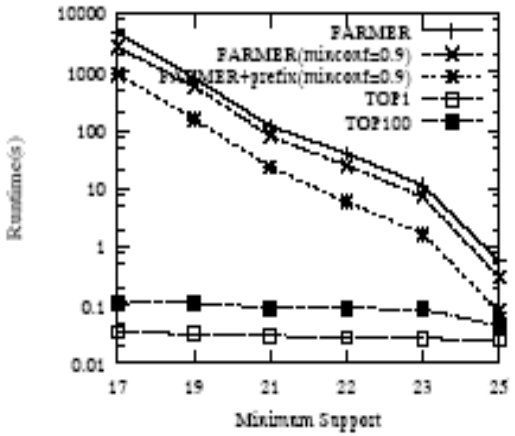
Notes about SVM—Introductory Literature

- "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

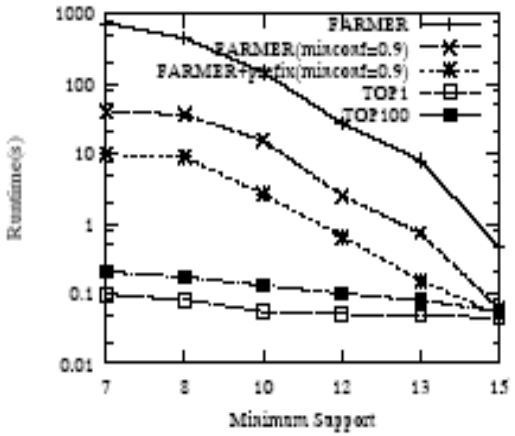
Associative Classification Can Achieve High Accuracy and Efficiency (Cong et al. SIGMOD05)

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%

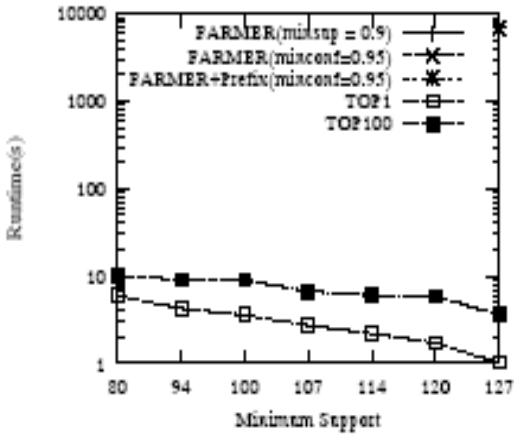
Table 2: Classification Results



(a) ALL-AML leukemia



(b) Lung Cancer

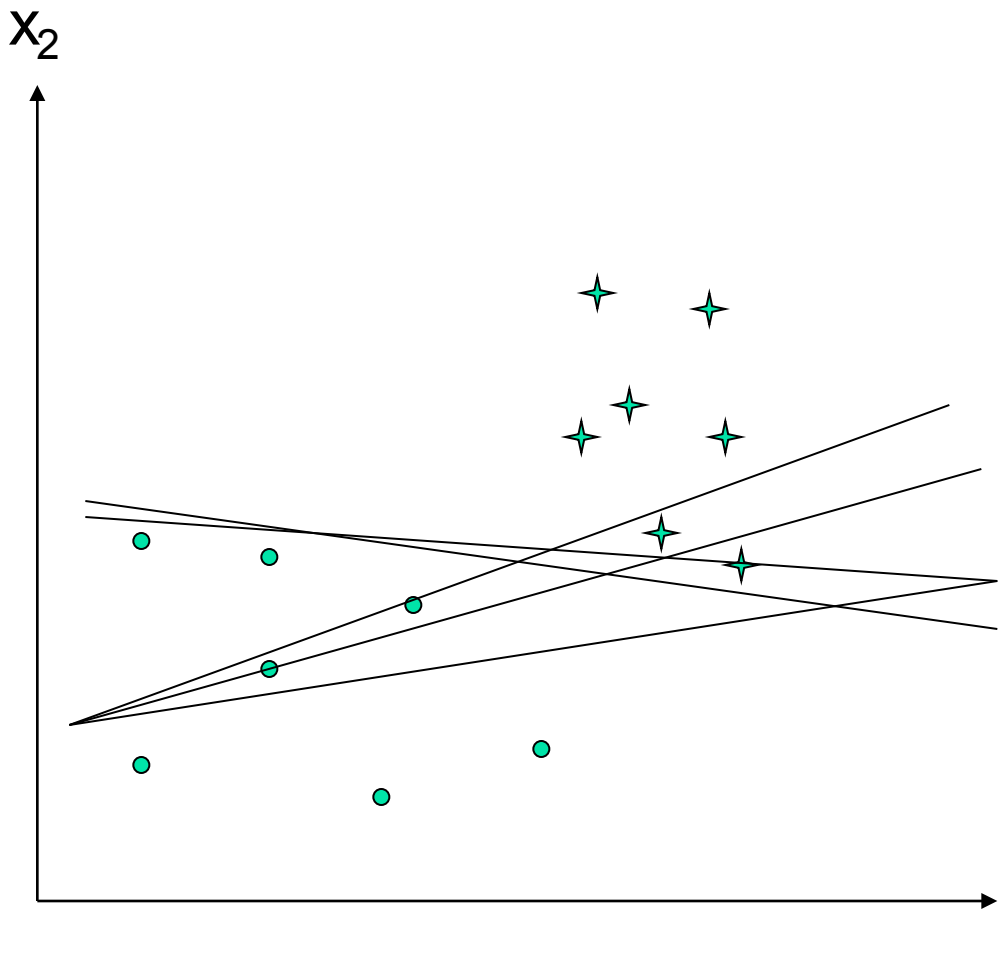


(c) Ovarian Cancer

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 $conf(R_1) \geq conf(R_2)$, then prune R_2
  - Prunes rules for which the rule antecedent and class are not positively correlated, based on a χ^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 χ^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

Perceptron & Winnow



- Vector: x, w
- Scalar: x, y, w

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

Output: classification function f(x)

$f(x_i) > 0$ for $y_i = +1$

$f(x_i) < 0$ for $y_i = -1$

f(x) => $wx + b = 0$

or $w_1x_1+w_2x_2+b = 0$

- Perceptron: update W additively
- Winnow: update W multiplicatively