

A photograph of four ginger and white kittens sitting on a dark, textured tree branch. The kittens are looking in various directions, some up and some forward. The background is a lush green forest, and there are clusters of bright pink flowers in the foreground on both sides.

# Chapter 9. Classification: Advanced Methods

Meng Jiang

CS412 Summer 2017:

Introduction to Data Mining

# Classification: Advanced Methods

- **Bayesian Belief Networks**
- Neural Networks
- Support Vector Machines
- Rule/Pattern-based Classification
- Lazy Learners and K-Nearest Neighbors
- Other Classification Methods: Genetic Algorithms, Fuzzy Sets and Rough Sets
- Additional Topics: Semi-Supervised Methods, Active Learning, etc.

# Bayesian Belief Networks

- **Bayesian belief network (or Bayesian network, probabilistic network):**

- allows *class conditional independencies* between *subsets* of variables

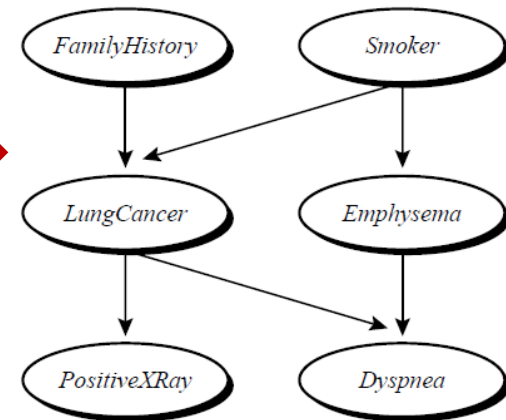
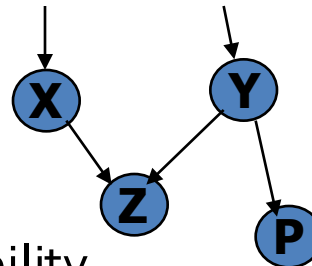
	<i>FH, S</i>	<i>FH, ~S</i>	<i>~FH, S</i>	<i>~FH, ~S</i>
<i>LC</i>	0.8	0.5	0.7	0.1
<i>~LC</i>	0.2	0.5	0.3	0.9

- Two components:

- A *directed acyclic graph* (called a structure)
- A set of *conditional probability tables* (CPTs)

- A (*directed acyclic*) graphical model of *causal influence* 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

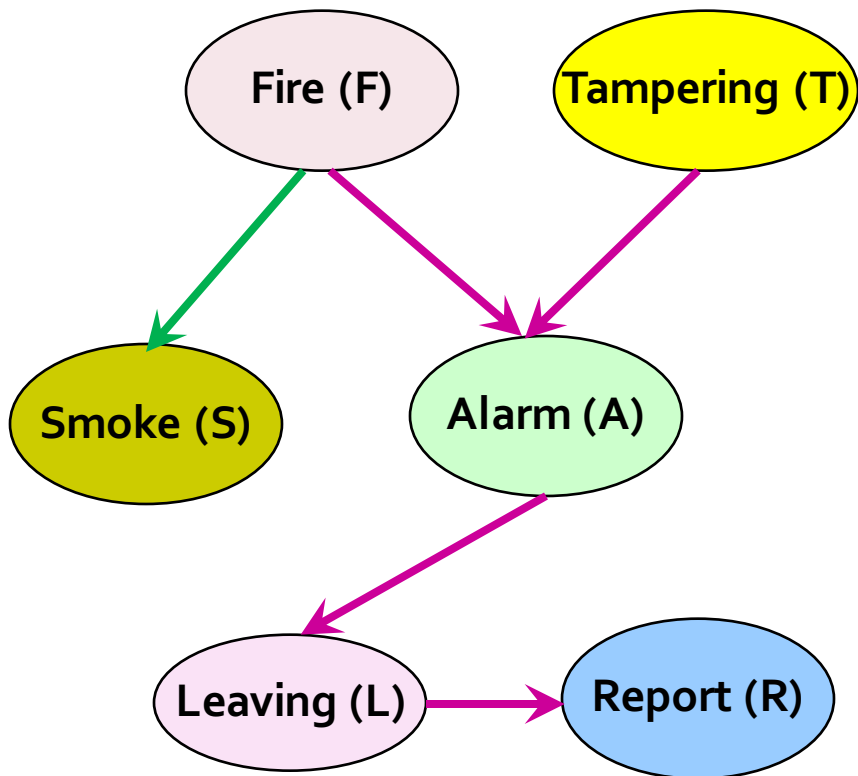
No dependency between Z and P

Has no loops/cycles



# A Bayesian Network and Some of Its CPTs

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



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

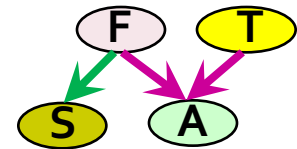
$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{Parents}(x_i))$$

## CPT: Conditional Probability Tables

Fire	Smoke	$\Theta_{s f}$
True	True	.90
False	True	.01

Fire	Tampering	Alarm	$\Theta_{a f,t}$
True	True	True	.5
True	False	True	.99
False	True	True	.85
False	False	True	.0001

# How Are Bayesian Networks Constructed?



- **Subjective construction:** Identification of (direct) causal structure
  - People are quite good at identifying direct causes from a given set of variables & whether the set contains all relevant direct causes
  - Markovian assumption: Each variable becomes independent of its non-effects once its direct causes are known
    - E.g.,  $S \leftarrow F \rightarrow A \leftarrow T$ , path  $S \rightarrow A$  is blocked once we know  $F \rightarrow A$
  - HMM (Hidden Markov Model): often used to model dynamic systems whose states are not observable, yet their outputs are
- **Synthesis from other specifications**
  - E.g., from a formal system design: block diagrams & info flow
- **Learning from data** (e.g., from medical records or student admission record)
  - Learn parameters give its structure or learn both structure and parms
  - Maximum likelihood principle: favors Bayesian networks that maximize the probability of observing the given data set

# 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.

# Classification: Advanced Methods

- Bayesian Belief Networks
- **Neural Networks**
- Support Vector Machines
- Rule/Pattern-based Classification
- Lazy Learners and K-Nearest Neighbors
- Other Classification Methods: Genetic Algorithms, Fuzzy Sets and Rough Sets
- Additional Topics: Semi-Supervised Methods, Active Learning, etc.

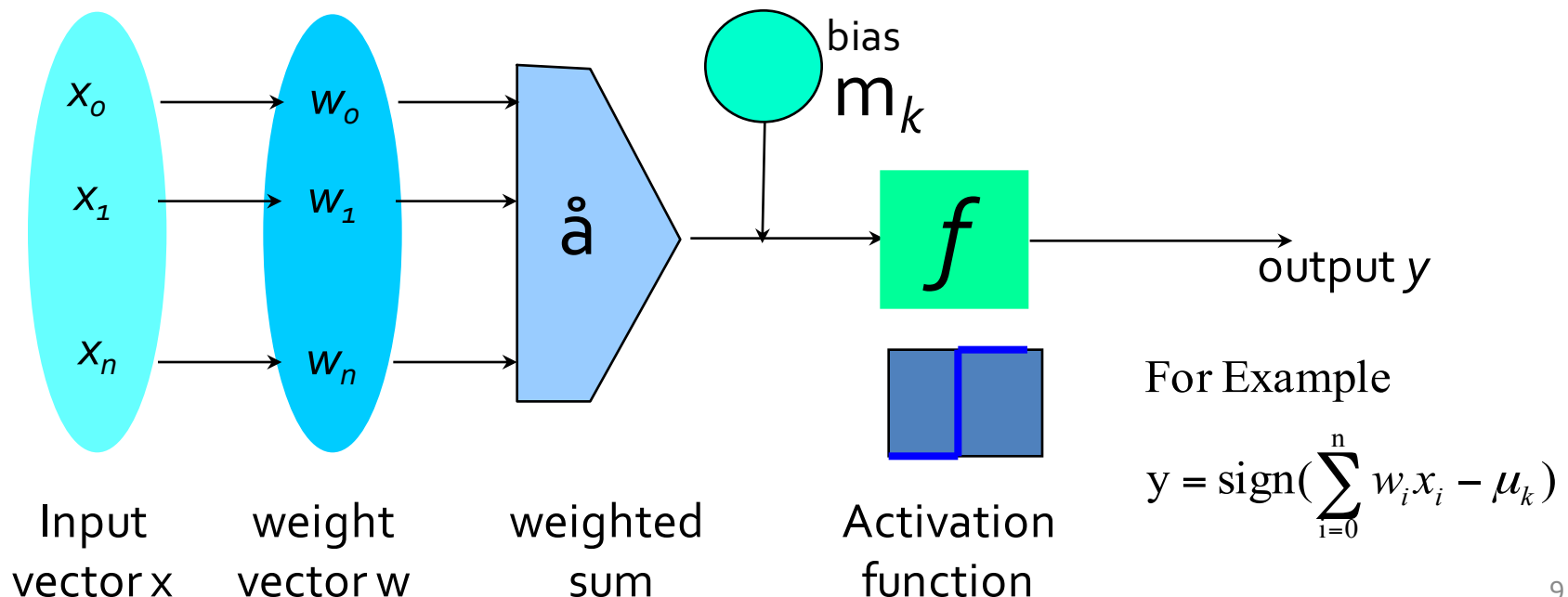
# Neural Network for Classification

- 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
- Backpropagation: A **neural network** learning algorithm



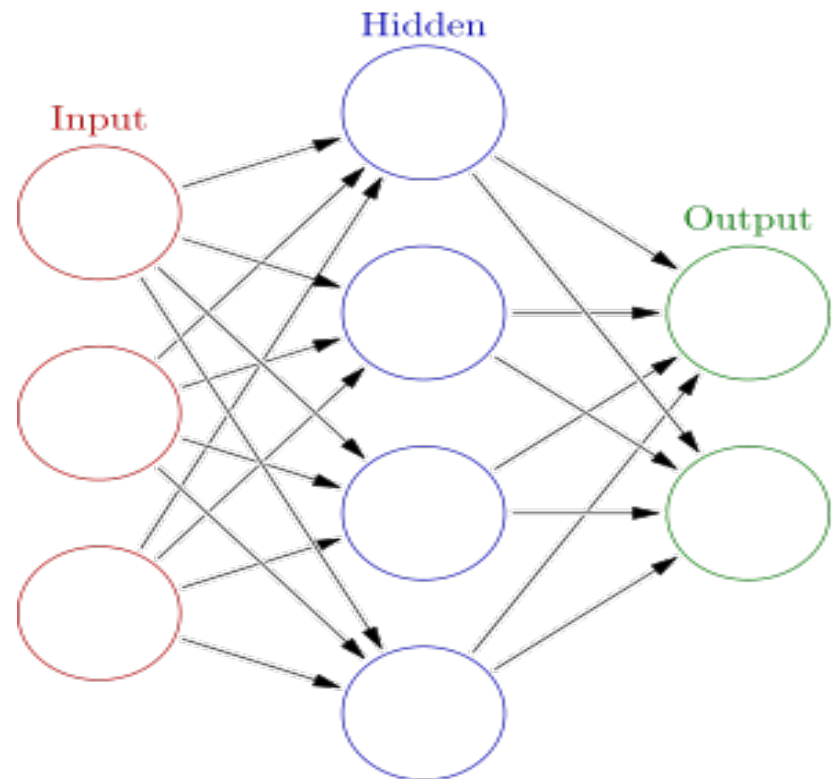
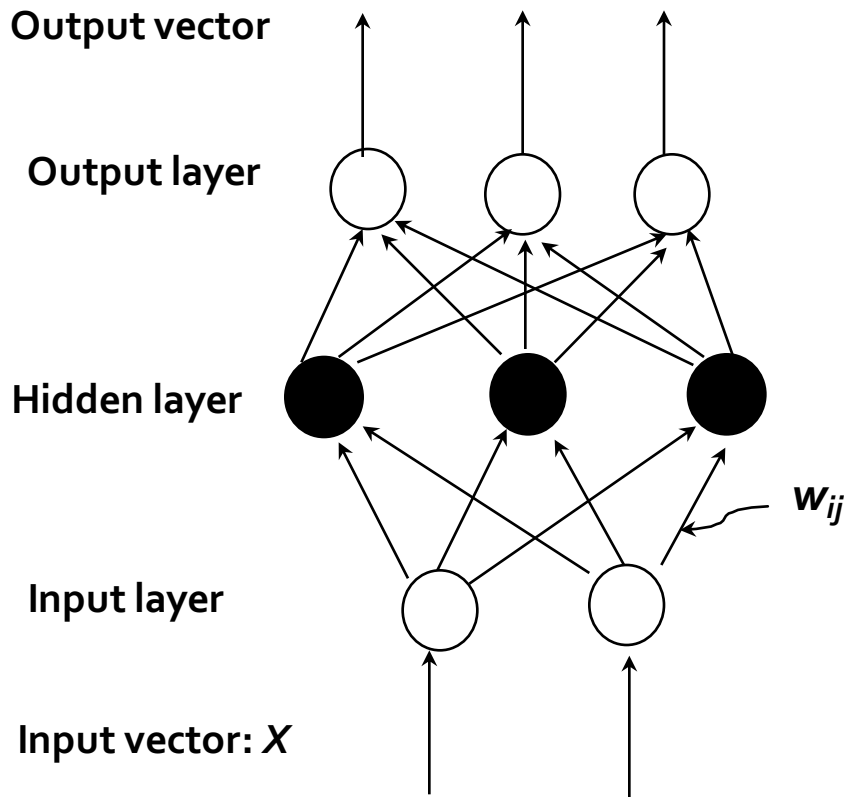
# Neuron: A Hidden/Output Layer Unit

- 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.



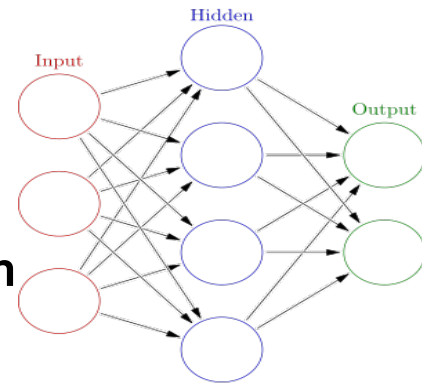
# A Multi-Layer Feed-Forward Neural Network

$$w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})x_{ij}$$



# 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

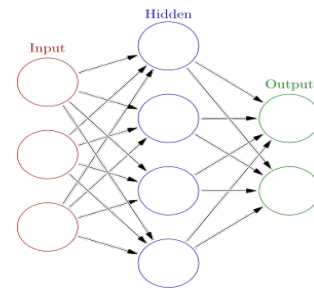


# Defining a Network Topology

- 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*

# Back Propagation

- **Back propagation:** Reset weights on the "front" neural units and this is sometimes done in combination with training where the correct result is known
- 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.)



# From Neural Networks to Deep Learning

- Train networks with many layers (vs. shallow nets with just a couple of layers)
- Multiple layers work to build an improved feature space
  - First layer learns 1<sup>st</sup> order features (e.g., edges, ...)
  - 2<sup>nd</sup> layer learns higher order features (combinations of first layer features, combinations of edges, etc.)
  - In current models, layers often learn in an unsupervised mode and discover general features of the input space—serving multiple tasks related to the unsupervised instances (image recognition, etc.)
  - Then final layer features are fed into supervised layer(s)
    - And entire network is often subsequently tuned using supervised training of the entire net, using the initial weightings learned in the unsupervised phase
  - Could also do fully supervised versions (back-propagation)

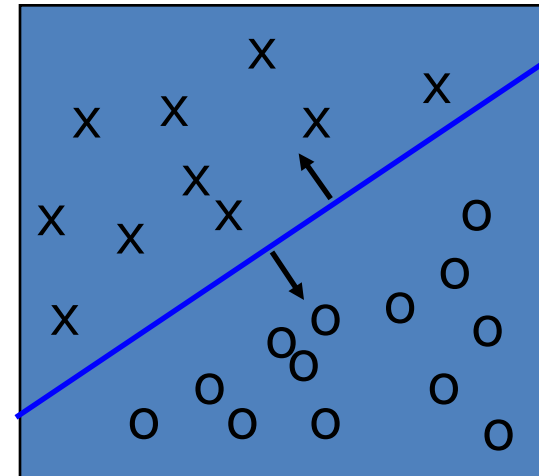


# Classification: Advanced Methods

- Bayesian Belief Networks
- Neural Networks
- **Support Vector Machines**
- Rule/Pattern-based Classification
- Lazy Learners and K-Nearest Neighbors
- Other Classification Methods: Genetic Algorithms, Fuzzy Sets and Rough Sets
- Additional Topics: Semi-Supervised Methods, Active Learning, etc.

# 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
  - 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

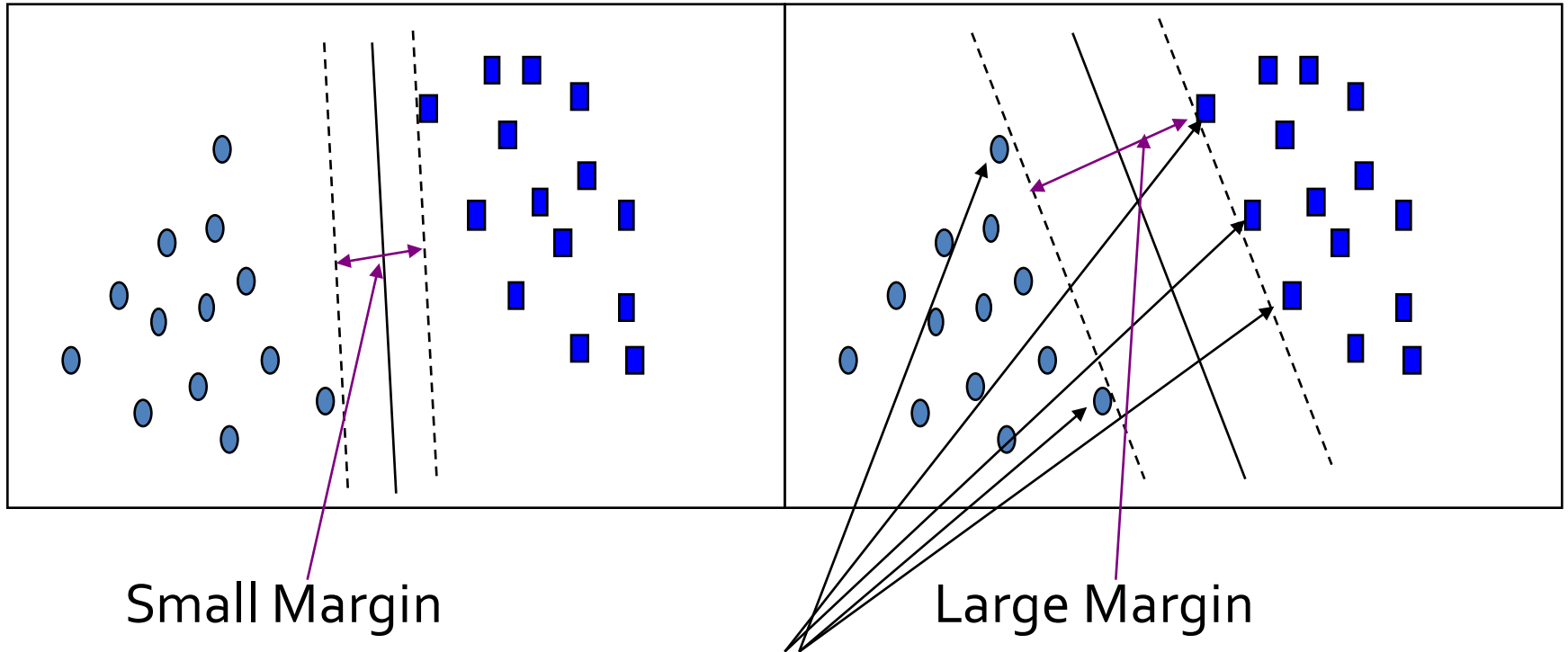
# SVM: Support Vector Machines

- 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

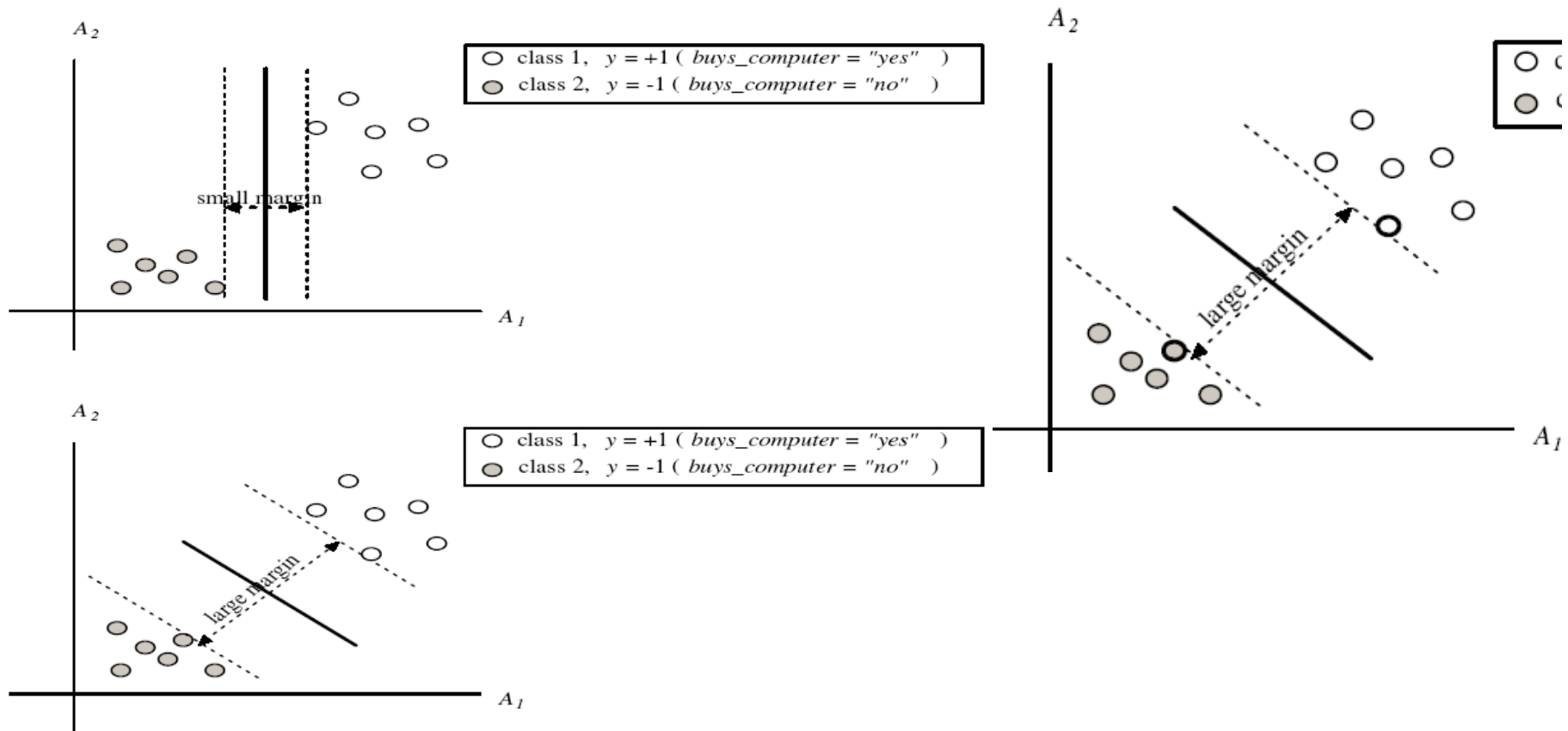
# SVM: General Philosophy



**Support Vectors**



# SVM: Margins and Support Vectors

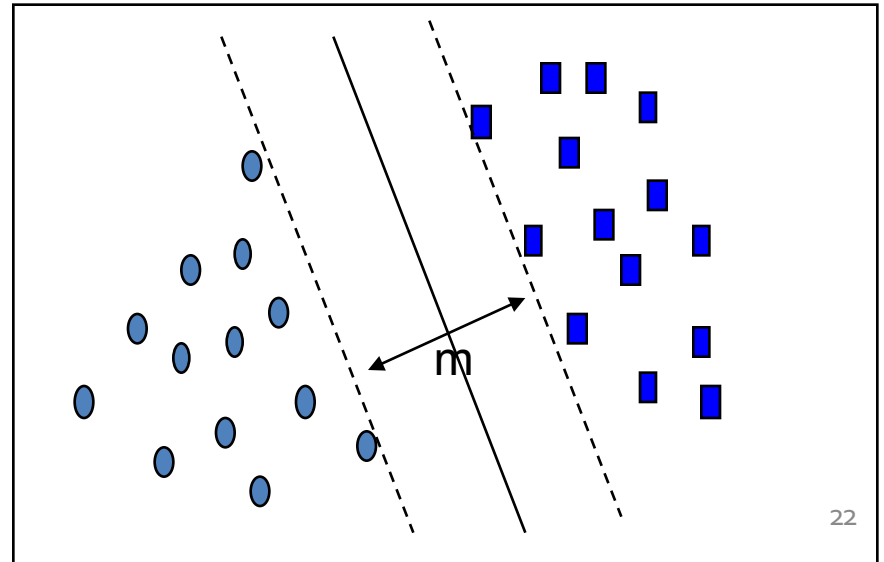
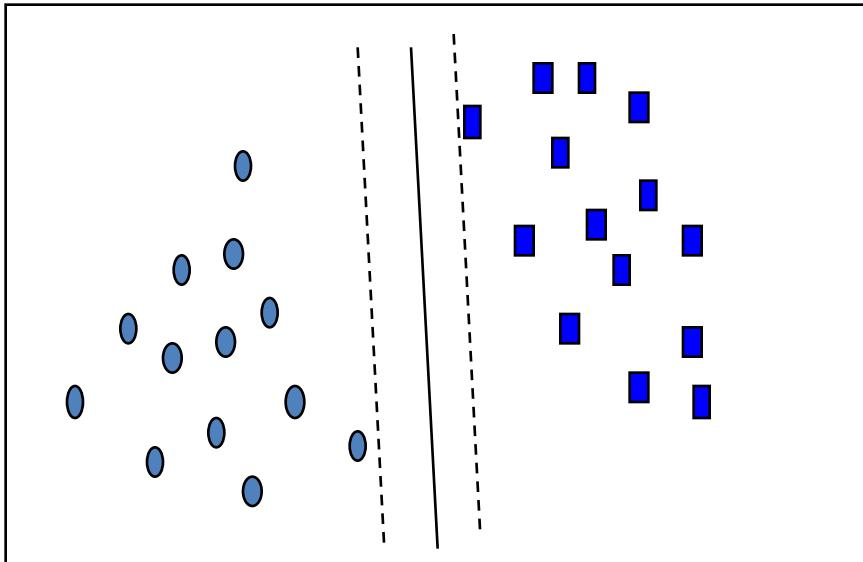


# SVM: When Data Is Linearly Separable

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

# Kernel Functions for Nonlinear Classification

- Instead of computing the dot product on the transformed data, it is mathematically 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  $h$  :  $K(\mathbf{X}_i, \mathbf{X}_j) = (\mathbf{X}_i \cdot \mathbf{X}_j + 1)^h$

Gaussian radial basis function kernel :  $K(\mathbf{X}_i, \mathbf{X}_j) = e^{-\|\mathbf{X}_i - \mathbf{X}_j\|^2 / 2\sigma^2}$

Sigmoid kernel :  $K(\mathbf{X}_i, \mathbf{X}_j) = \tanh(\kappa \mathbf{X}_i \cdot \mathbf{X}_j - \delta)$

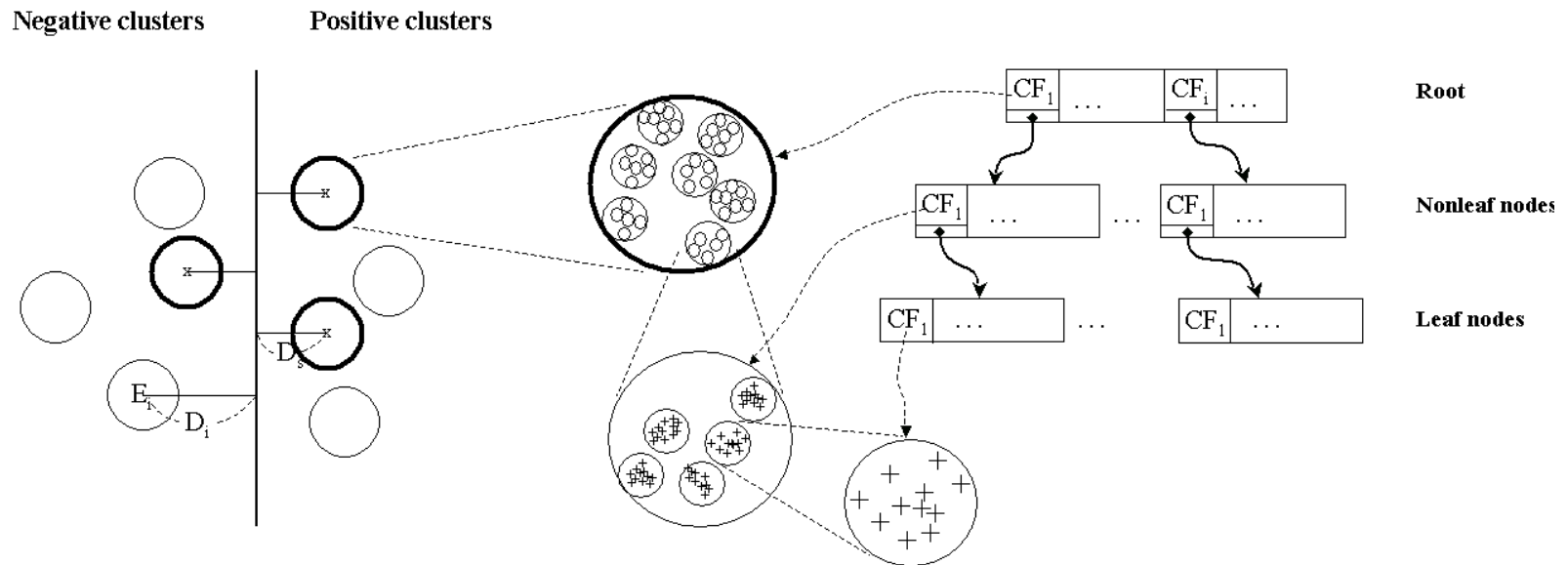
- SVM can also be used for classifying multiple ( $> 2$ ) classes and for regression analysis (with additional parameters)

# Scaling SVM by Hierarchical Micro-Clustering

- SVM is not scalable to # of data objects in terms of training time and memory usage
- CB-SVM (Clustering-Based SVM): H. Yu, J. Yang, and J. Han, "[Classifying Large Data Sets Using SVM with Hierarchical Clusters](#)", KDD'03
- Clustering-Based SVM: Algorithm outline
  - Construct two CF-trees (hierarchical clusters)
  - 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
  - Repeat until nothing is accumulated
- At deriving support vectors, de-cluster micro-clusters near "candidate vector" to ensure high classification accuracy



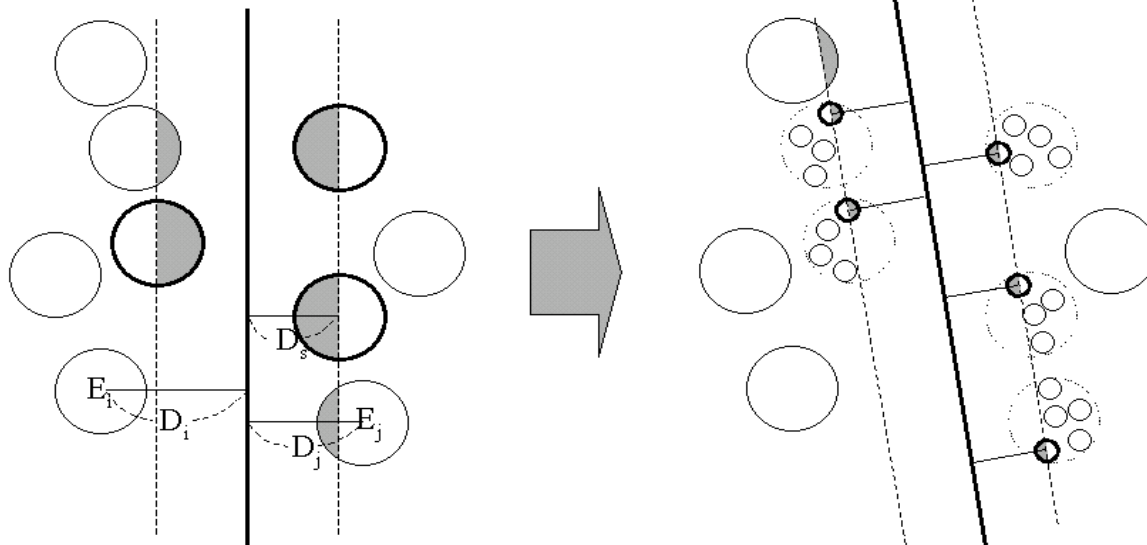
# CF-Tree: Hierarchical Micro-cluster



- One scan of the data set: Construct two CF-trees (i.e., statistical summary of the data) from positive and negative data sets independently
- 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



# Accuracy and Scalability on Synthetic Dataset

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

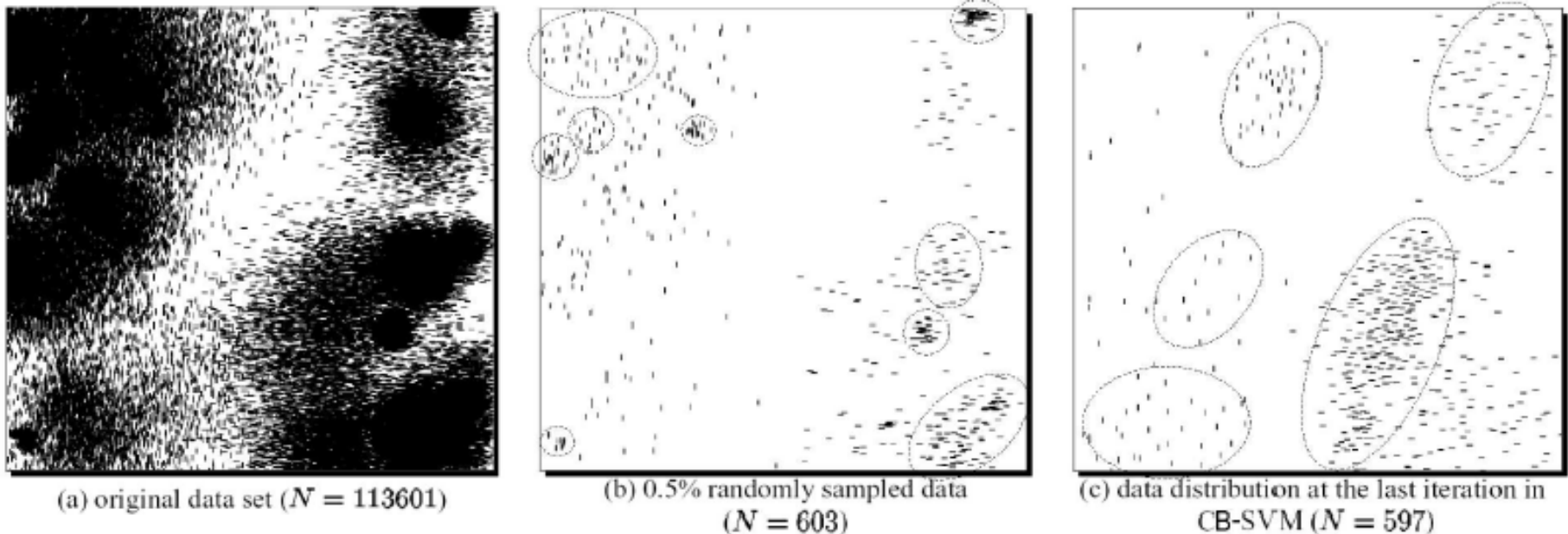


Figure 6: Synthetic data set in a two-dimensional space. '+' : positive data; '-' : negative data

# 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

# Classification: Advanced Methods

- Bayesian Belief Networks
- Neural Networks
- Support Vector Machines
- **Rule/Pattern-based Classification**
- Lazy Learners and K-Nearest Neighbors
- Other Classification Methods: Genetic Algorithms, Fuzzy Sets and Rough Sets
- Additional Topics: Semi-Supervised Methods, Active Learning, etc.

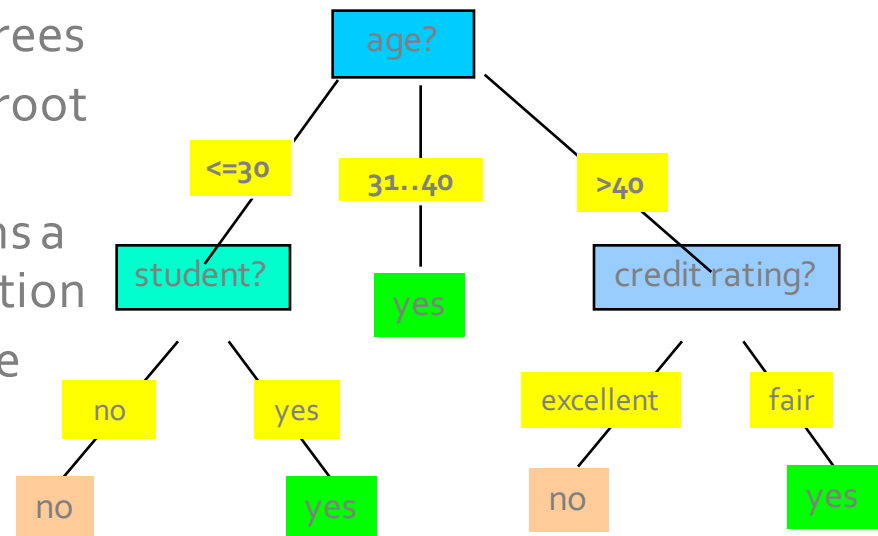
# Using IF-THEN Rules for Classification

- Represent the knowledge in the form of **IF-THEN** rules  
     $R_1$ : IF *age* = youth AND *student* = yes THEN *buys\_computer* = yes
  - Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: *coverage* and *accuracy*
  - $n_{\text{covers}}$  = # of tuples covered by  $R_1$
  - $n_{\text{correct}}$  = # of tuples correctly classified by  $R_1$
  - $\text{coverage}(R_1) = n_{\text{covers}} / |D|$  /\*  $D$ : training data set \*/
  - $\text{accuracy}(R_1) = n_{\text{correct}} / n_{\text{covers}}$
- If more than one rule are triggered, need **conflict resolution**
  - **Size ordering**: assign the highest priority to the triggering rules that has the “toughest” requirement (i.e., with the *most attribute tests*)
  - **Class-based ordering**: decreasing order of *prevalence* or *misclassification cost per class*
  - **Rule-based ordering (decision list)**: rules are organized into one long priority list, according to some measure of rule quality or by experts



# Rule Extraction from a Decision Tree

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



IF *age* = young AND *student* = no

IF *age* = young AND *student* = yes

IF *age* = mid-age

IF *age* = old AND *credit\_rating* = excellent

IF *age* = old AND *credit\_rating* = fair

THEN *buys\_computer* = no

THEN *buys\_computer* = yes

THEN *buys\_computer* = yes

THEN *buys\_computer* = no

THEN *buys\_computer* = yes

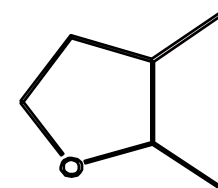
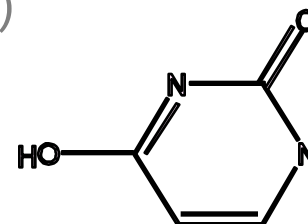
# Rule Induction: Sequential Covering Method

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

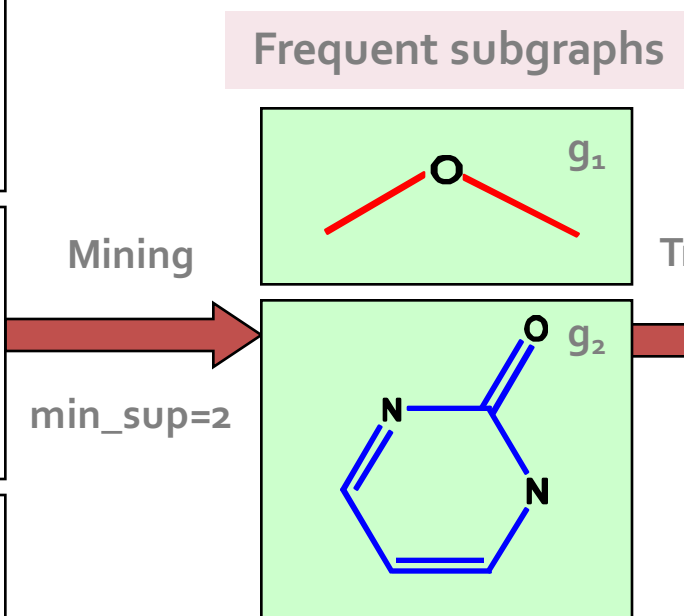
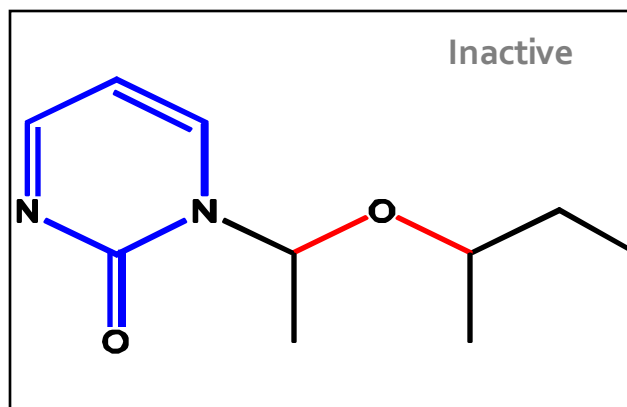
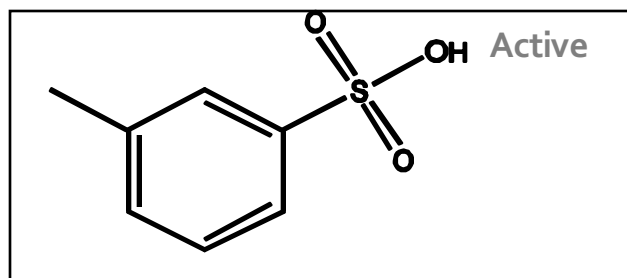
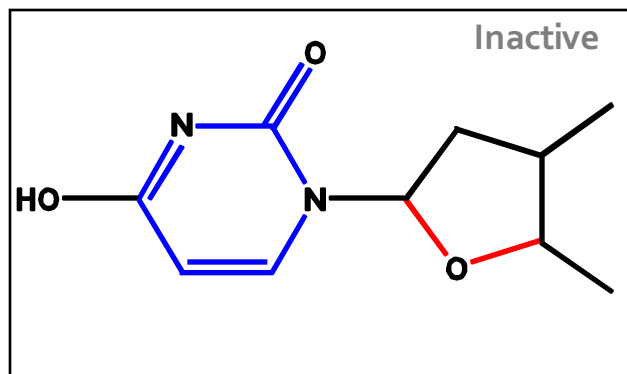
# Pattern-Based Classification, Why?



- **Pattern-based classification:** An integration of both themes
- **Why pattern-based classification?**
  - **Feature construction**
    - Higher order; compact; discriminative
    - E.g., single word → phrase (Apple pie, Apple i-pad)
  - **Complex data modeling**
    - Graphs (no predefined feature vectors)
    - Sequences
    - Semi-structured/unstructured Data



# Pattern-Based Classification on Graphs



Use frequent patterns as features for classification

Transform

$g_1$	$g_2$	Class
1	1	0
0	0	1
1	1	0

# Associative or Pattern-Based Classification

- **Data:** Transactions, microarray data, ... → **Patterns or association rules**
- **Classification Methods** (Some interesting work):
  - CBA [Liu, Hsu & Ma, KDD'98]: Use high-conf., high-support *class association rules* to build classifiers
  - Emerging patterns [Dong & Li, KDD'99]: Patterns whose support changes significantly between the two classes
  - CMAR [Li, Han & Pei, ICDM'01]: Multiple rules in prediction
  - CPAR [Yin & Han, SDM'03]: Beam search on multiple prediction rules
  - RCBT [Cong et al., SIGMOD'05]: Build classifier based on mining top-k covering rule groups with row enumeration (for high-dimensional data)
  - Lazy classifier [Veloso, Meira & Zaki, ICDM'06]: For a test  $t$ , project training data  $D$  on  $t$ , mine rules from  $D_t$ , predict on the best rule
  - Discriminative pattern-based classification [Cheng et al., ICDE'07]

# CBA: Classification Based on Associations

- CBA [Liu, Hsu and Ma, KDD'98]
- Method
  - Mine high-confidence, high-support class association rules
  - LHS: conjunctions of attribute-value pairs; RHS: class labels
$$p_1 \wedge p_2 \dots \wedge p_l \rightarrow "A_{\text{class-label}} = C" \text{ (confidence, support)}$$
  - Rank rules in descending order of confidence and support
  - Classification: Apply the first rule that matches a test case; o.w. apply the default rule
  - Effectiveness: Often found more accurate than some traditional classification methods, such as C4.5
  - Why? — Exploring high confident associations among multiple attributes may overcome some constraints introduced by some classifiers that consider only one attribute at a time

# CMAR: Classification Based on Multiple Association Rules

- 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 label are not positively correlated, based on the  $\chi^2$  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  $X$ 
    - Divide  $S$  into groups according to class labels
    - Use a weighted  $\chi^2$  measure to find the strongest group of rules, based on the statistical correlation of rules within a group
    - Assign  $X$  the class label of the strongest group
- CMAR improves model construction efficiency and classification accuracy

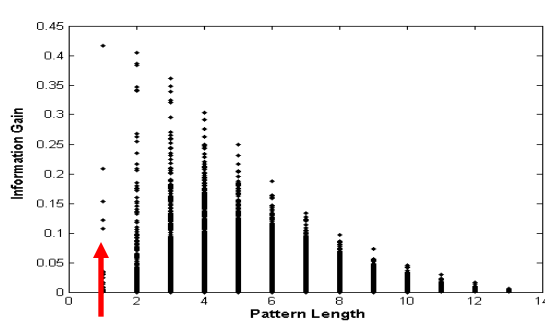
# Discriminative Pattern-Based Classification

- Discriminative patterns as features for classification [Cheng et al., ICDE'07]
- **Principle:** Mining discriminative frequent patterns as high-quality features and then apply any classifier
- **Framework (PatClass)**
  - Feature construction by *frequent itemset mining*
  - Feature selection (e.g., using Maximal Marginal Relevance (MMR))
    - Select discriminative features (i.e., that are relevant but minimally similar to the previously selected ones)
    - Remove redundant or closely correlated features
  - Model learning
    - Apply a general classifier, such as SVM or C4.5, to build a classification model

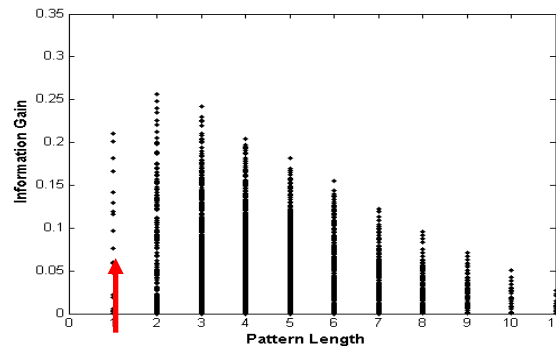


# On the Power of Discriminative Patterns

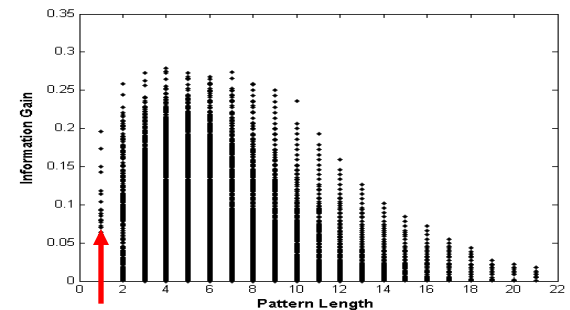
- K-itemsets are often more informative than single features (1-itemsets) in classification
- Computation on real datasets shows: The discriminative power of k-itemsets (for  $k > 1$  but often  $\leq 10$ ) is higher than that of single features



(a) Austral



(b) Cleve

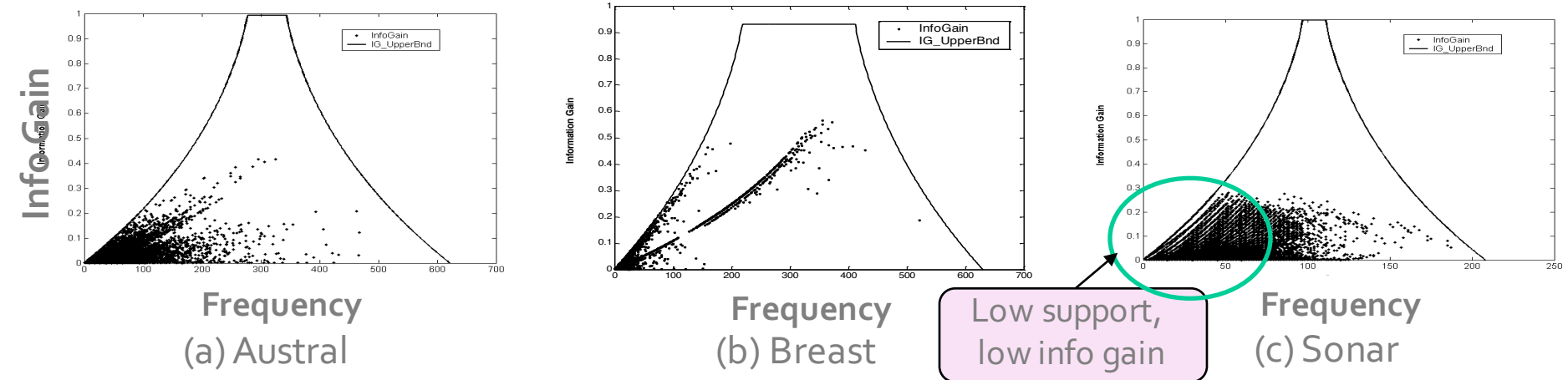


(c) Sonar

Information Gain vs. Pattern Length

# Information Gain vs. Pattern Frequency

- Computation on real datasets shows: Pattern frequency (but not too frequent) is strongly tied with the discriminative power (information gain)
- Information gain upper bound monotonically increases with pattern frequency



Information Gain Formula:  $IG(C | X) = H(C) - H(C | X)$  Conditional entropy of study focus

Entropy of given data  $H(C) = -\sum_{i=1}^m p_i \log_2(p_i)$   $H(C | X) = \sum_j P(X = x_j) H(Y | X = x_j)$

# Discriminative Pattern-Based Classification: 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>

# Discriminative Pattern-Based Classification: Scalability Tests

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

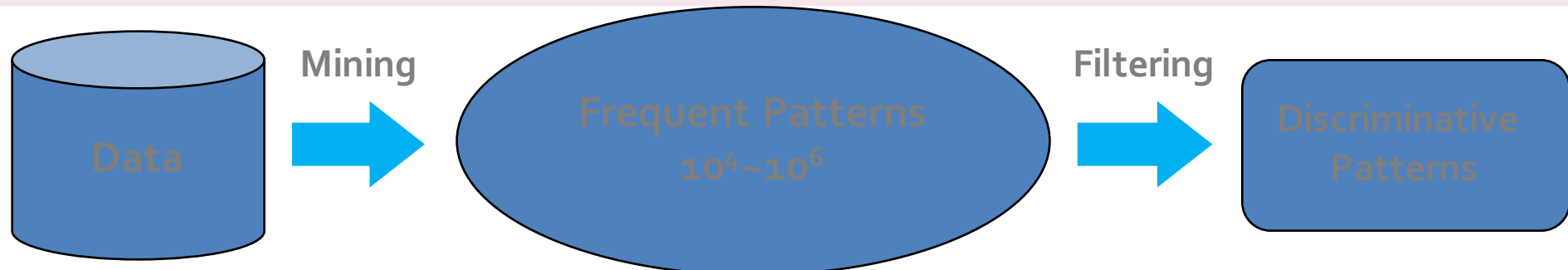
<i>min_sup</i>	#Patterns	Time (s)	SVM (%)	C4.5 (%)
1	N/A	N/A	N/A	N/A
2000	68,967	44.703	92.52	97.59
2200	28,358	19.938	91.68	97.84
2500	6,837	2.906	91.68	97.62
2800	1,031	0.469	91.84	97.37
3000	136	0.063	91.90	97.06

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

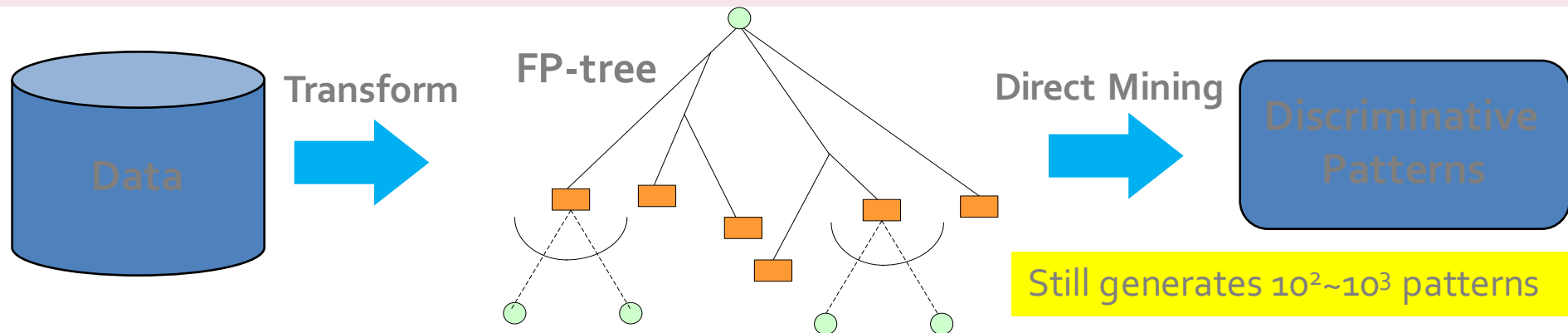
<i>min_sup</i>	#Patterns	Time (s)	SVM (%)	C4.5 (%)
1	9,468,109	N/A	N/A	N/A
80	26,576	176.485	92.40	88.35
100	15,316	90.406	92.19	87.29
150	5,408	23.610	91.53	88.80
200	2,481	8.234	91.22	87.32

# Mining Concise Set of Discriminative Patterns

Frequent pattern mining, then getting discriminative patterns: Expensive, large model



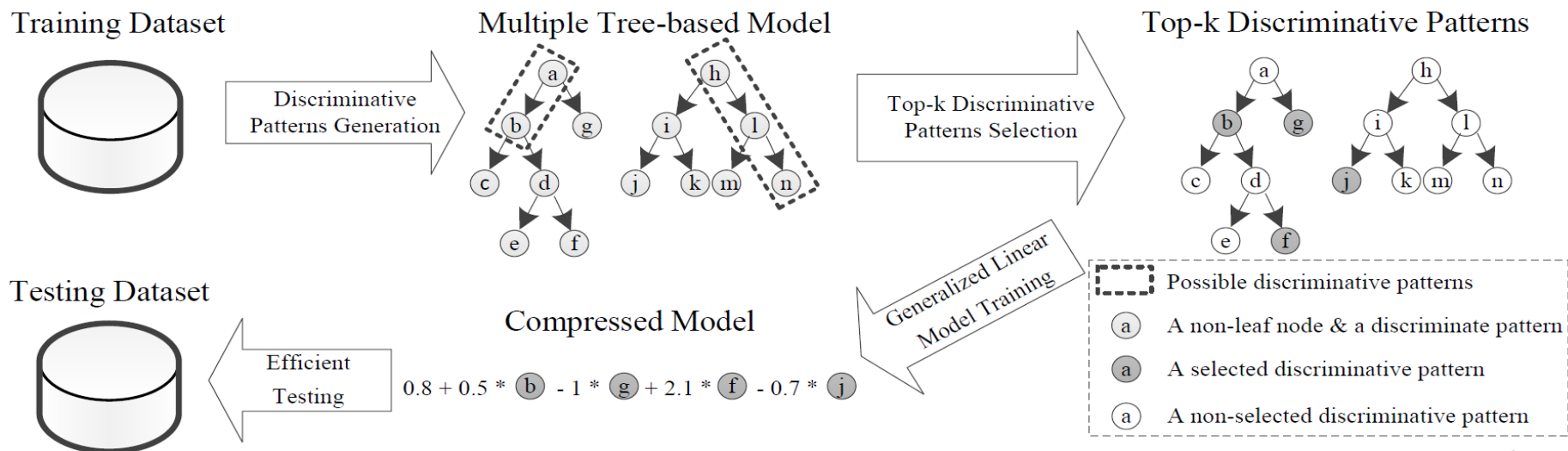
DDPMine [Cheng et al., ICDE'08]: Direct mining of discriminative patterns: Efficient



DPClass [Shang et al, SDM'16]: A better solution — Efficient, effective, and generating a very limited number of (such as only 20 or so) patterns

# DPClass: Discriminative Pattern-based Classification

- Input: A feature table for training data
- Adopt every prefix path in an (extremely) random forest as a candidate pattern
  - The split points of continuous variables are automatically chosen by random forest → No discretization!
- Run top-k (e.g., top-20) pattern selection based on training data
- Train a generalized linear model (e.g., logistic regression) based on “bag-of-patterns” representations of training data



# Explanatory Discriminative Patterns: Generation

- Example: For each patient, we have several uniformly sampled features as follows
  - Age (A): Positive Integers no more than 60
  - Gender (G): Male or Female.
  - Lab Test 1 (LT1): Categorical values from (A, B, O, AB)
  - Lab Test 2 (LT2): Continuous values in  $[0..1]$
- The positive label of the hypo-disease will be given when at least one of the following rules holds
  - (age > 18) and (gender = Male) and (LT1 = AB) and (LT2  $\geq$  0.6)
  - (age > 18) and (gender = Female) and (LT1 = O) and (LT2  $\geq$  0.5)
  - (age  $\leq$  18) and (LT2  $\geq$  0.9)
- Training:  $10^5$  random patients + 0.1% noise
  - Flip the binary labels with 0.1% probability
- Testing:  $5 \times 10^4$  random patients in test

# Explanatory Discriminative Patterns: Evaluation

- Accuracy:
  - **DPClass 99.99% (perfect)**
  - DDPMine 95.64% (reasonable)
- Top-3 Discriminative Patterns:
  - **DPClass (perfect):**
    - (age > 18) and (gender = Female) and (LT<sub>1</sub> = O) and (LT<sub>2</sub> ≥ 0.496)
    - (age ≤ 18) and (LT<sub>2</sub> ≥ 0.900)
    - (age > 18) and (gender = Male) and (LT<sub>1</sub> = AB) and (LT<sub>2</sub> ≥ 0.601)
  - DDPMine (poor):
    - (LT<sub>2</sub> > 0.8)
    - (gender = Male) and (LT<sub>1</sub> = AB) and (LT<sub>2</sub> ≥ 0.6) and (LT<sub>2</sub> < 0.8)
    - (gender = Female) and (LT<sub>1</sub> = O) and (LT<sub>2</sub> ≥ 0.6) and (LT<sub>2</sub> < 0.8)



# A Comparison on Classification Accuracy

- DPClass: Discriminative & frequent at the same time, then select top-k
  - Only top-20 patterns are used in DPClass
- Two methods on pattern selection
  - Forward vs. LASSO
  - In comparison with DDPMine and Random Forest, DPClass maintains high accuracy
- An extension of DPClass has been applied to health study
  - Cheng et al, "Mining Discriminative Patterns to Predict Health Status for Cardiopulmonary Patients", *ACM-BCB'16*

	Dataset	DPClass (Forward)	DPClass (LASSO)	DDPMine	Random Forest
low-dimensional data	adult	<b>85.66%</b>	84.33%	83.42%	85.45%
	hypo	<b>99.58%</b>	99.28%	92.69%	97.22%
	sick	98.35%	<b>98.87%</b>	93.82%	94.03%
	crx	<b>89.35%</b>	87.96%	87.96%	<b>89.35%</b>
	sonar	<b>85.29%</b>	83.82%	73.53%	83.82%
	chess	92.25%	92.05%	90.04%	<b>94.22%</b>
high-dimensional data	namao	97.17%	96.94%	96.83%	<b>97.86%</b>
	musk	95.92%	95.71%	93.29%	<b>96.60%</b>
	madelon	74.50%	<b>76.00%</b>	59.84%	56.50%

# Classification: Advanced Methods

- Bayesian Belief Networks
- Neural Networks
- Support Vector Machines
- Rule/Pattern-based Classification
- **Lazy Learners and K-Nearest Neighbors**
- Other Classification Methods: Genetic Algorithms, Fuzzy Sets and Rough Sets
- Additional Topics: Semi-Supervised Methods, Active Learning, etc.

# Lazy vs. Eager Learning

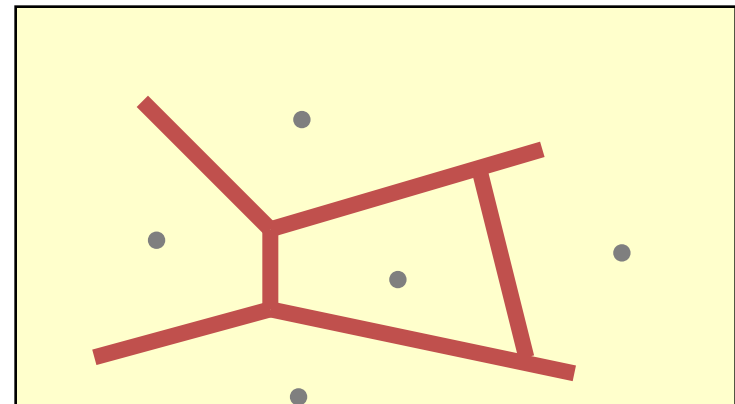
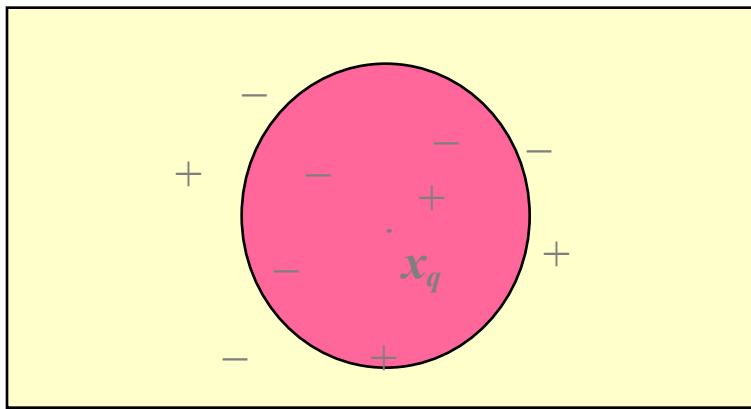
- Lazy vs. eager learning
  - **Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - **Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before 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  $x_q$ 
    - 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

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

# 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

# Classification: Advanced Methods

- Bayesian Belief Networks
- Neural Networks
- Support Vector Machines
- Rule/Pattern-based Classification
- Lazy Learners and K-Nearest Neighbors
- **Other Classification Methods: Genetic Algorithms, Fuzzy Sets and Rough Sets**
- Additional Topics: Semi-Supervised Methods, Active Learning, etc.

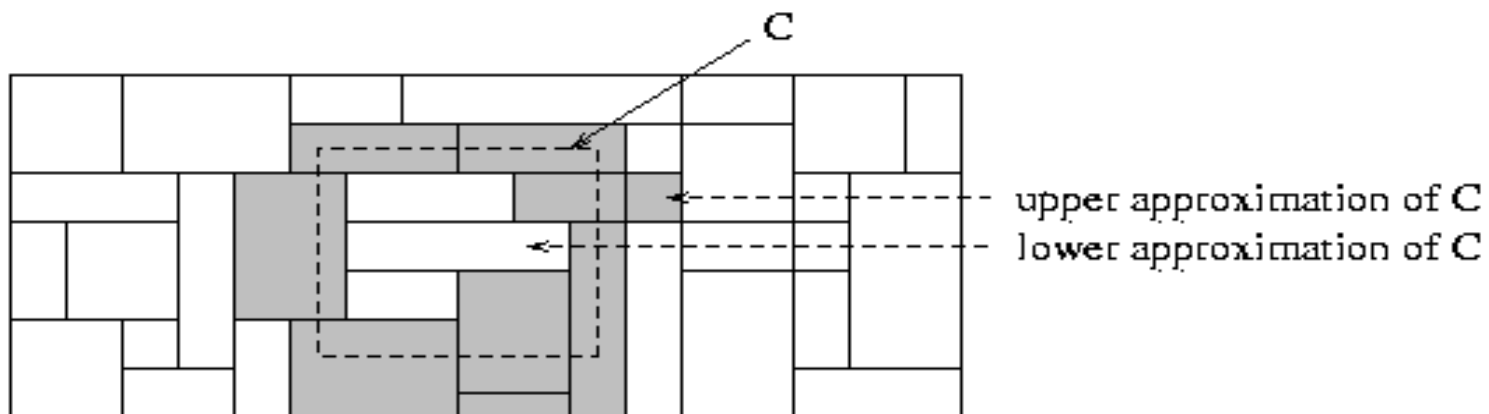


# Genetic Algorithms (GA)

- 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 pre-specified threshold*
- Slow but easily parallelizable

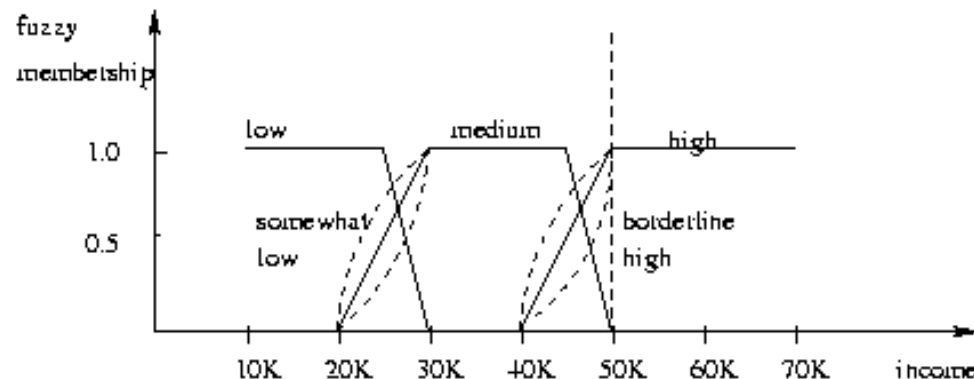
# Rough Set Approach

- 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



# Classification: Advanced Methods

- Bayesian Belief Networks
- Neural Networks
- Support Vector Machines
- Rule/Pattern-based Classification
- Lazy Learners and K-Nearest Neighbors
- Other Classification Methods: Genetic Algorithms, Fuzzy Sets and Rough Sets
- **Additional Topics: Semi-Supervised Methods, Active Learning, etc.**

# Multiclass Classification

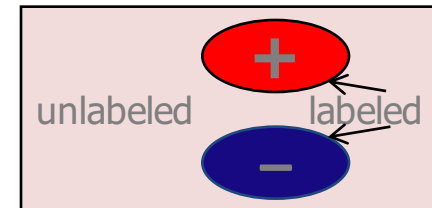
- Classification involving more than two classes (i.e.,  $> 2$  Classes)
- Method 1. **One-vs.-all** (OVA): Learn a classifier one at a time
  - Given  $m$  classes, train  $m$  classifiers: one for each class
  - Classifier  $j$ : treat tuples in class  $j$  as *positive* & all others as *negative*
  - 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

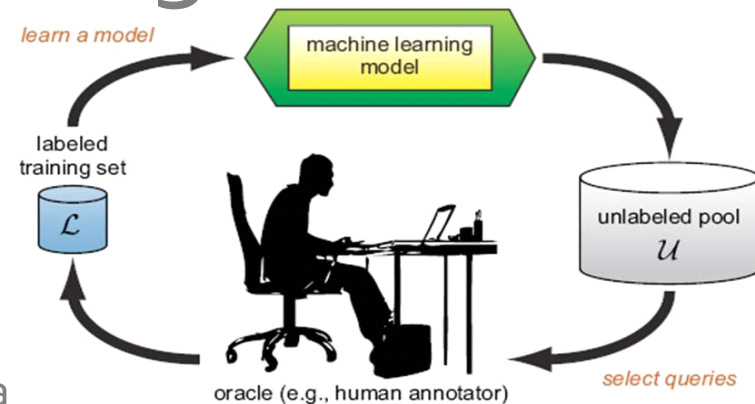
- Originally designed to correct errors during data transmission for communication tasks by exploring data redundancy
  - Example
    - A 7-bit codeword associated with classes 1-4
- | Class | Error-Corr. Codeword |   |   |   |   |   |   |
|-------|----------------------|---|---|---|---|---|---|
| $C_1$ | 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 |
- 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  $(h-1)/2$  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

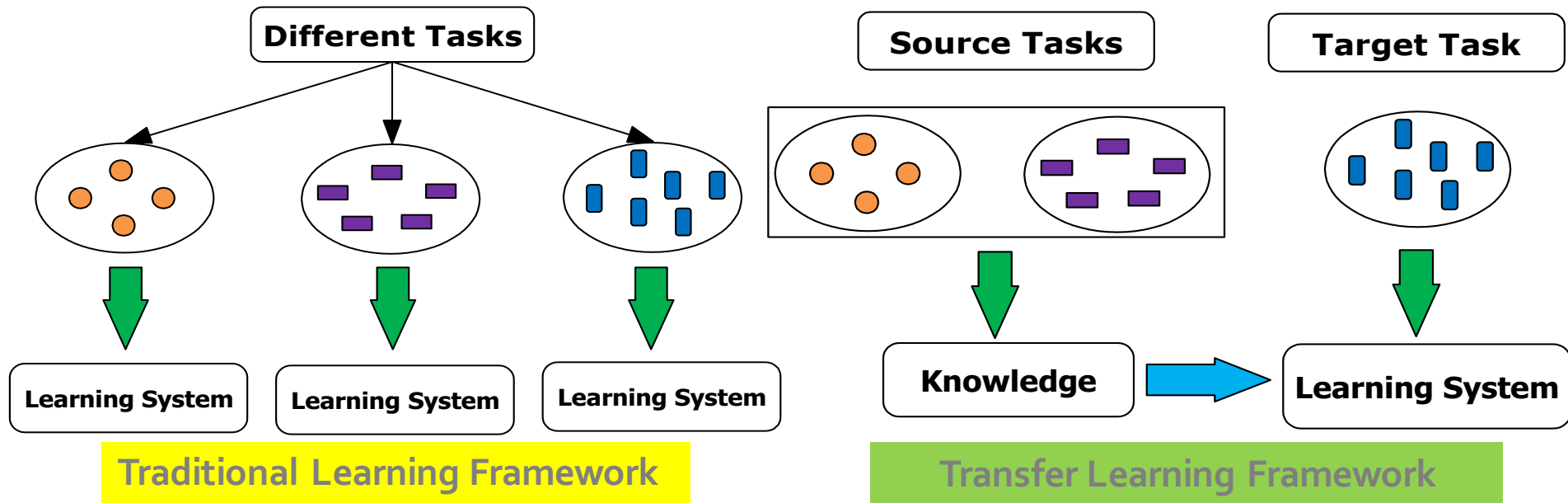


- Class labels are expensive to obtain
- Active learner: query human (oracle) for labels
- Pool-based approach: Uses a pool of unlabeled data
  - $\mathcal{L}$ : a small subset of  $D$  is labeled,  $\mathcal{U}$ : a pool of unlabeled data in  $D$
  - Use a query function to carefully select one or more 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

# 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

- C. M. Bishop, Neural Networks for Pattern Recognition. Oxford University Press, 1995
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth International Group, 1984
- C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery, 2(2): 121-168, 1998
- N. Cristianini and J. Shawe-Taylor, Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2000
- H. Yu, J. Yang, and J. Han. Classifying large data sets using SVM with hierarchical clusters. KDD'03
- A. J. Dobson. An Introduction to Generalized Linear Models. Chapman & Hall, 1990
- 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
- H. Cheng, X. Yan, J. Han & C.-W. Hsu, Discriminative Frequent Pattern Analysis for Effective Classification, ICDE'07
- W. Cohen. Fast effective rule induction. ICML'95

# References (cont.)

- H. Cheng, X. Yan, J. Han & P. S. Yu, Direct Discriminative Pattern Mining for Effective Classification, ICDE'08
- G. Cong, K. Tan, A. Tung & X. Xu. Mining Top-k Covering Rule Groups for Gene Expression Data, SIGMOD'05
- M. Deshpande, M. Kuramochi, N. Wale & G. Karypis. Frequent Substructure-based Approaches for Classifying Chemical Compounds, TKDE'05
- G. Dong & J. Li. Efficient Mining of Emerging Patterns: Discovering Trends and Differences, KDD'99
- W. Fan, K. Zhang, H. Cheng, J. Gao, X. Yan, J. Han, P. S. Yu & O. Verscheure. Direct Mining of Discriminative and Essential Graphical and Itemset Features via Model-based Search Tree, KDD'08
- W. Li, J. Han & J. Pei. CMAR: Accurate and Efficient Classification based on Multiple Class-association Rules, ICDM'01
- B. Liu, W. Hsu & Y. Ma. Integrating Classification and Association Rule Mining, KDD'98
- J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. ECML'93
- Jingbo Shang, Wenzhu Tong, Jian Peng, and Jiawei Han, "[DPClass: An Effective but Concise Discriminative Patterns-Based Classification Framework](#)", SDM'16
- J. Wang and G. Karypis. HARMONY: Efficiently Mining the Best Rules for Classification, SDM'05
- X. Yin & J. Han. CPAR: Classification Based on Predictive Association Rules, SDM'03