UNIVERSITÀ DI PISA

Dottorato di Ricerca in Ingegneria dell'Informazione

Data Mining and Soft Computing

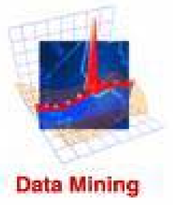
Francisco Herrera

Research Group on Soft Computing and Information Intelligent Systems (SCI^{2}S)
Dept. of Computer Science and A.I.
University of Granada, Spain

Email: herrera@decsai.ugr.es
http://sci2s.ugr.es

http://decsai.ugr.es/~herrera

DECSAI
Universidad de Granada



Summary

1. Introduction to Data Mining and Knowledge Discovery
2. Data Preparation
3. Introduction to Prediction, Classification, Clustering and Association
4. Data Mining - From the Top 10 Algorithms to the New Challenges
5. Introduction to Soft Computing. Focusing our attention in Fuzzy Logic and Evolutionary Computation
6. Soft Computing Techniques in Data Mining: Fuzzy Data Mining and Knowledge Extraction based on Evolutionary Learning
7. Genetic Fuzzy Systems: State of the Art and New Trends
8. Some Advanced Topics I: Classification with Imbalanced Data Sets
9. Some Advanced Topics II: Subgroup Discovery
10. Some advanced Topics III: Data Complexity
11. Final talk: How must I Do my Experimental Study? Design of Experiments in Data Mining/Computational Intelligence. Using Non-parametric Tests. Some Cases of Study.

Slides used for preparing this talk:

Top 10 Algorithms in Data Mining Research
prepared for ICDM 2006

10 Challenging Problems in Data Mining Research
prepared for ICDM 2005

CS490D:
Introduction to Data Mining
Prof. Chris Clifton

Association Analysis: Basic Concepts and Algorithms

Lecture Notes for Chapter 6

Introduction to Data Mining

by Tan, Steinbach, Kumar

DATA MINING
Introductory and Advanced Topics
Margaret H. Dunham

Outline

- Top 10 Algorithms in Data Mining Research
  - Introduction
  - Classification
  - Statistical Learning
  - Bagging and Boosting
  - Clustering
  - Association Analysis
  - Link Mining – Text Mining
  - Top 10 Algorithms
- 10 Challenging Problems in Data Mining Research
- Concluding Remarks

Outline

- Top 10 Algorithms in Data Mining Research
  - Introduction
  - Classification
  - Statistical Learning
  - Bagging and Boosting
  - Clustering
  - Association Analysis
  - Link Mining – Text Mining
  - Top 10 Algorithms
- 10 Challenging Problems in Data Mining Research
- Concluding Remarks

From the Top 10 Algorithms to the New Challenges in Data Mining

"Discussion Panels at ICDM 2005 and 2006"

ICDM

IEEE International Conference on Data Mining

Sponsored by the IEEE Computer Society

http://www.cs.uvm.edu/~icdm/

From the Top 10 Algorithms to the New Challenges in Data Mining

“Discussion Panels at ICDM 2005 and 2006”

Top 10 Algorithms in Data Mining Research
prepared for ICDM 2006

10 Challenging Problems in Data Mining Research
prepared for ICDM 2005

Top 10 Algoritms in Data Mining Research

prepared for ICDM 2006

http://www.cs.uvm.edu/~icdm/algorithms/index.shtml

Coordinators

Xindong Wu

University of Vermont

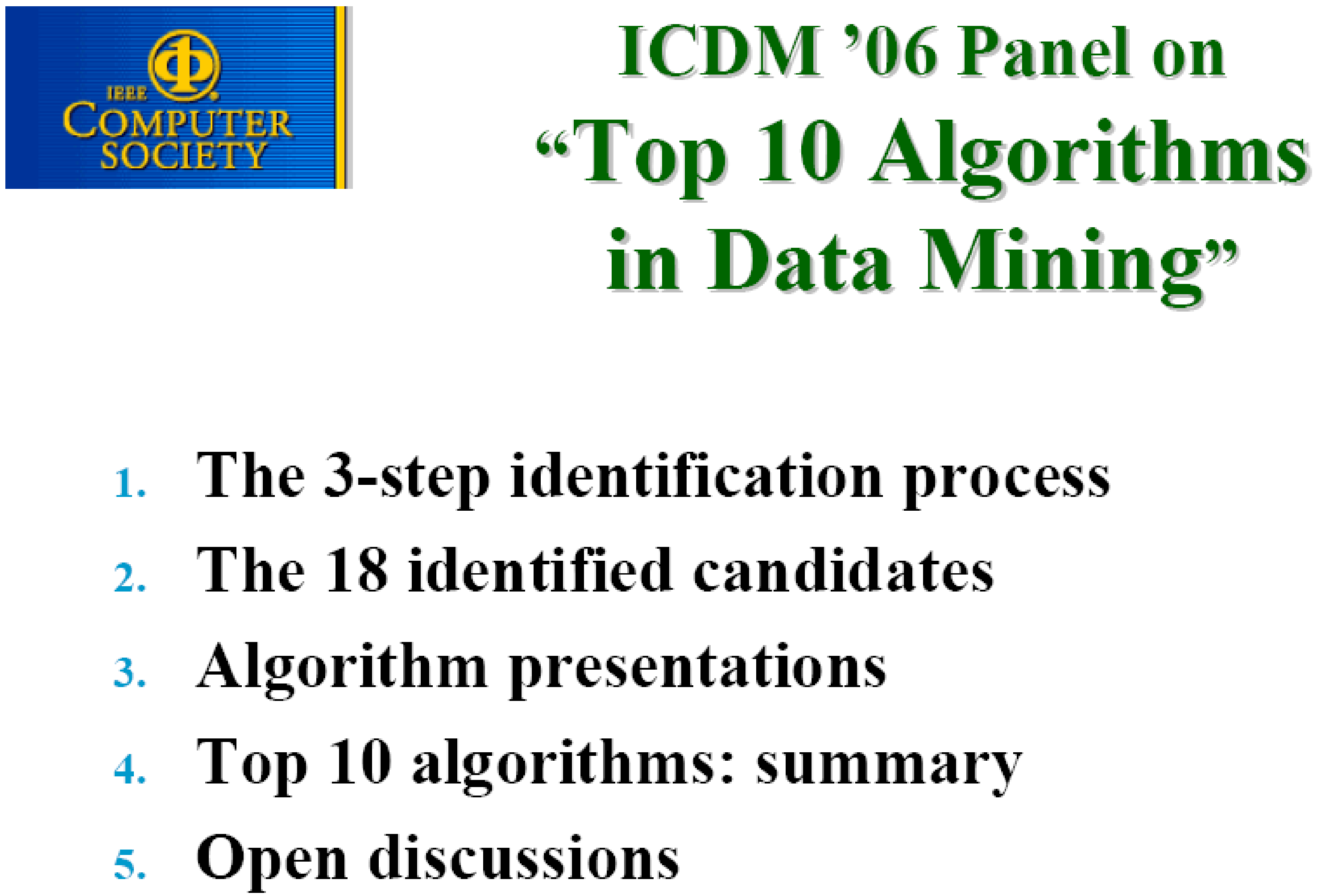
http://www.cs.uvm.edu/~xwu/home.html

Vipin Kumar

University of Minessota

http://www-users.cs.umn.edu/~kumar/

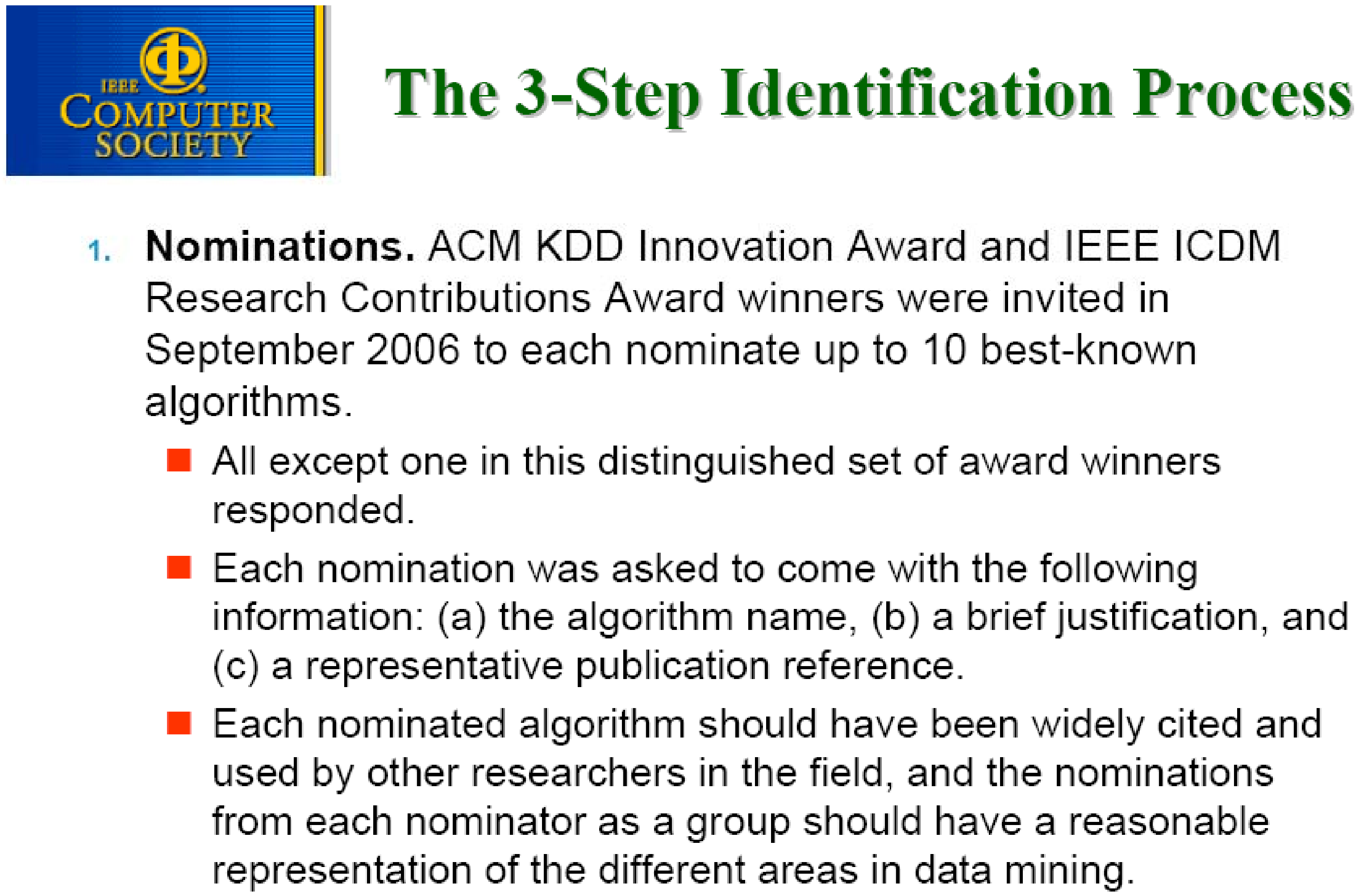
ICDM’06 Panel on “Top 10 Algorithms in Data Mining”



ICDM ’06 Panel on “Top 10 Algorithms in Data Mining”

1. The 3-step identification process
2. The 18 identified candidates
3. Algorithm presentations
4. Top 10 algorithms: summary
5. Open discussions

ICDM'06 Panel on "Top 10 Algorithms in Data Mining"



The 3-Step Identification Process

1. **Nominations.** ACM KDD Innovation Award and IEEE ICDM Research Contributions Award winners were invited in September 2006 to each nominate up to 10 best-known algorithms.
   - All except one in this distinguished set of award winners responded.
   - Each nomination was asked to come with the following information: (a) the algorithm name, (b) a brief justification, and (c) a representative publication reference.
   - Each nominated algorithm should have been widely cited and used by other researchers in the field, and the nominations from each nominator as a group should have a reasonable representation of the different areas in data mining.

ICDM'06 Panel on "Top 10 Algorithms in Data Mining"

The 3-Step Identification Process (2)

2. **Verification.** Each nomination was verified for its citations on Google Scholar in late October 2006, and those nominations that did not have at least 50 citations were removed.
   - 18 nominations survived and were then organized in 10 topics.
3. **Voting** by the wider community.
   - (a) Program Committee members of KDD-06, ICDM '06, and SDM '06 and (b) ACM KDD Innovation Award and IEEE ICDM Research Contributions Award winners were invited to each vote for up to 10 well-known algorithms.
   - The top 10 algorithms are ranked by their number of votes, and when there is a tie, the alphabetic order is used.

ICDM'06 Panel on "Top 10 Algorithms in Data Mining"

Agenda

1. The 3-step identification process
2. **The 18 identified candidates**
3. Algorithm presentations
4. Top 10 algorithms: summary
5. Open discussions

Outline

ICDM’06 Panel on “Top 10 Algorithms in Data Mining”

The 18 Identified Candidates

Classification

- #1. C4.5: Quinlan, J. R. 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc.
- #2. CART: L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and Regression Trees. Wadsworth, Belmont, CA, 1984.
- #3. K Nearest Neighbours (kNN): Hastie, T. and Tibshirani, R. 1996. Discriminant Adaptive Nearest Neighbor Classification. IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI). 18, 6 (Jun. 1996), 607-616.
- #4. Naive Bayes Hand, D.J., Yu, K., 2001. Idiot's Bayes: Not So Stupid After All? Internat. Statist. Rev. 69, 385-398.

Classification Using Decision Trees

- ***Partitioning based:*** Divide search space into rectangular regions.
- Tuple placed into class based on the region within which it falls.
- DT approaches differ in how the tree is built: ***DT Induction***
- Internal nodes associated with attribute and arcs with values for that attribute.
- Algorithms: ID3, C4.5, CART

Decision Tree

Given:

- $D = \{t_1, \ldots, t_n\}$ where $t_i=<t_{i1}, \ldots, t_{ih}>$
- Database schema contains $\{A_1, A_2, \ldots, A_h\}$
- Classes $C=\{C_1, \ldots., C_m\}$

Decision or Classification Tree is a tree associated with D such that

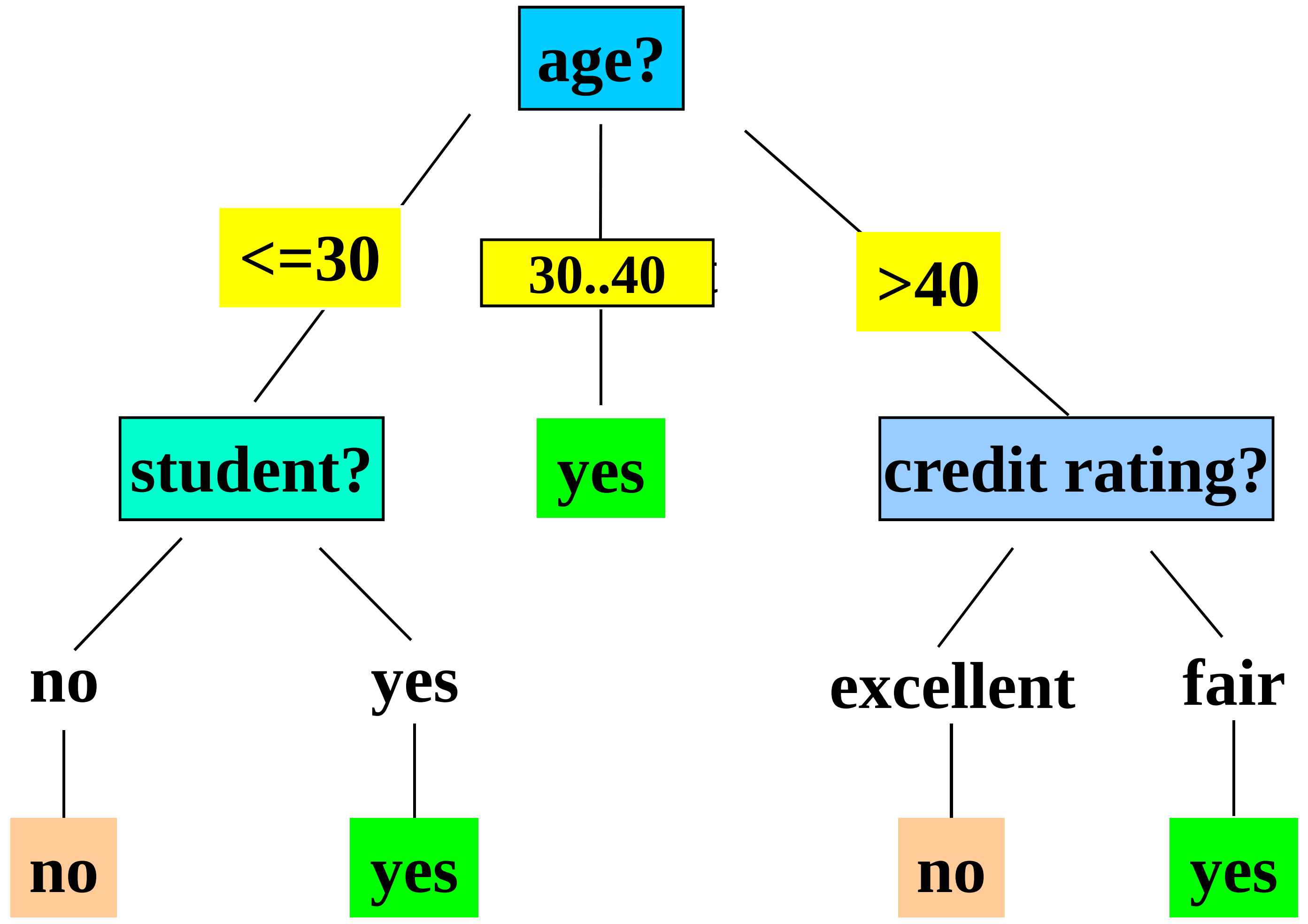
- Each internal node is labeled with attribute, A_i
- Each arc is labeled with predicate which can be applied to attribute at parent
- Each leaf node is labeled with a class, C_j

Training Dataset

This follows an example from Quinlan's ID3

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Output: A Decision Tree for *"buys_computer"*



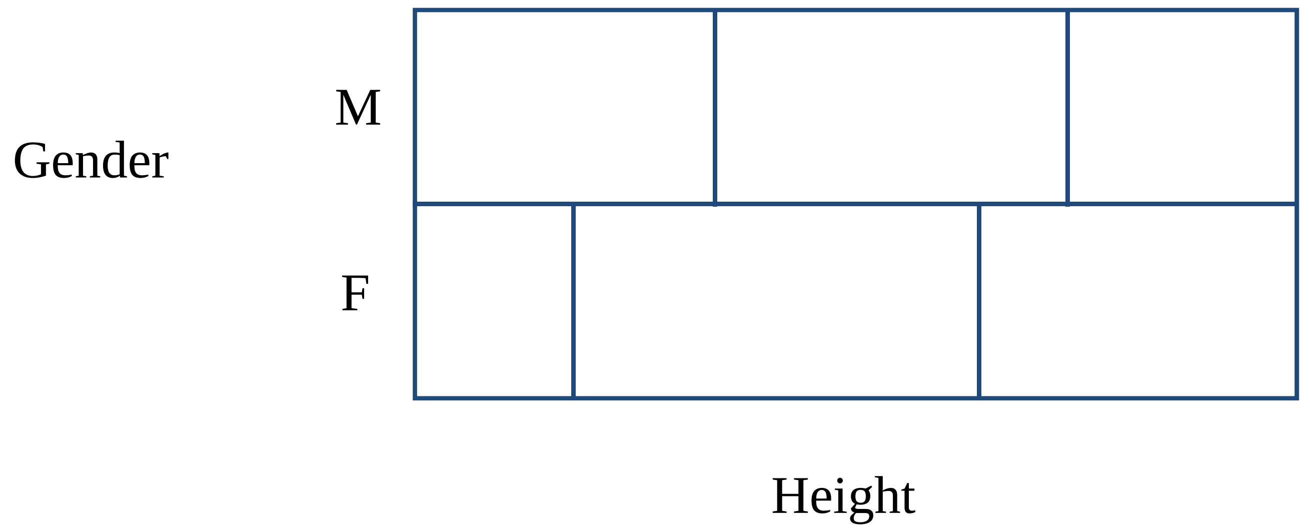
Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

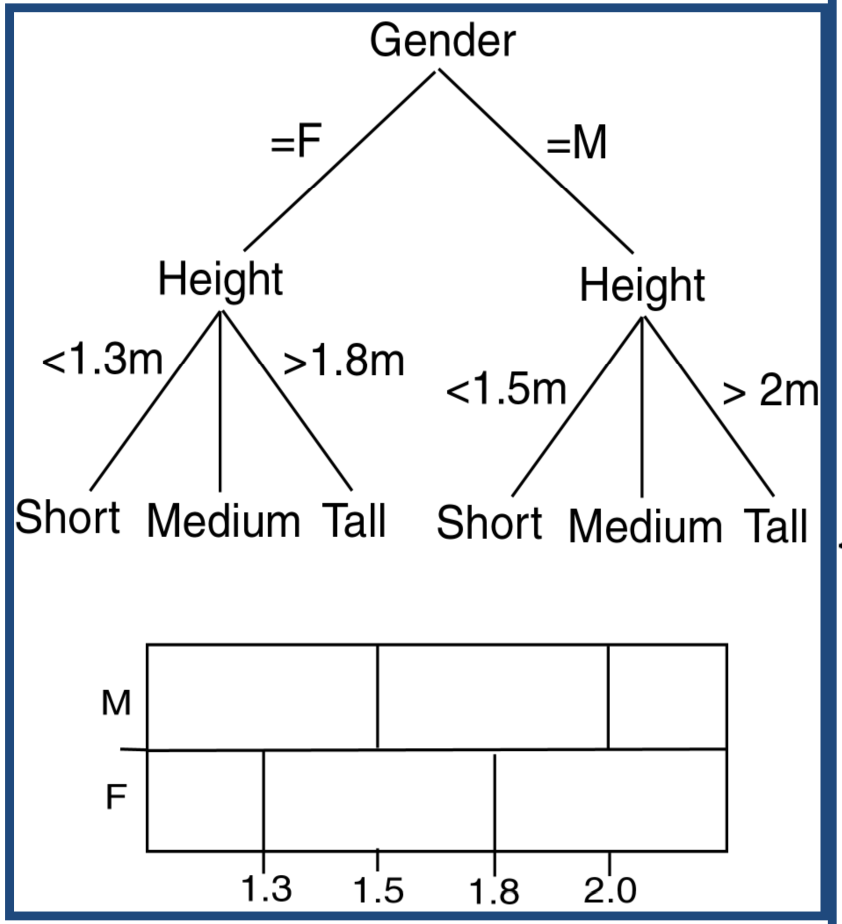
DT Induction

```
Input:
   $D$      //Training data
Output:
   $T$      //Decision Tree
DTBuild Algorithm:
         //Simplistic algorithm to illustrate naive approach to building DT
   $T = \emptyset$;
   Determine best splitting criterion;
   $T$ = Create root node node and label with splitting attribute;
   $T$ = Add arc to root node for each split predicate and label;
   for each arc do
      $D$ = Database created by applying splitting predicate to $D$;
      if stopping point reached for this path then
         $T'$ = Create leaf node and label with appropriate class;
      else
         $T' = DTBuild(D)$;
      $T$ = Add $T'$ to arc;
```

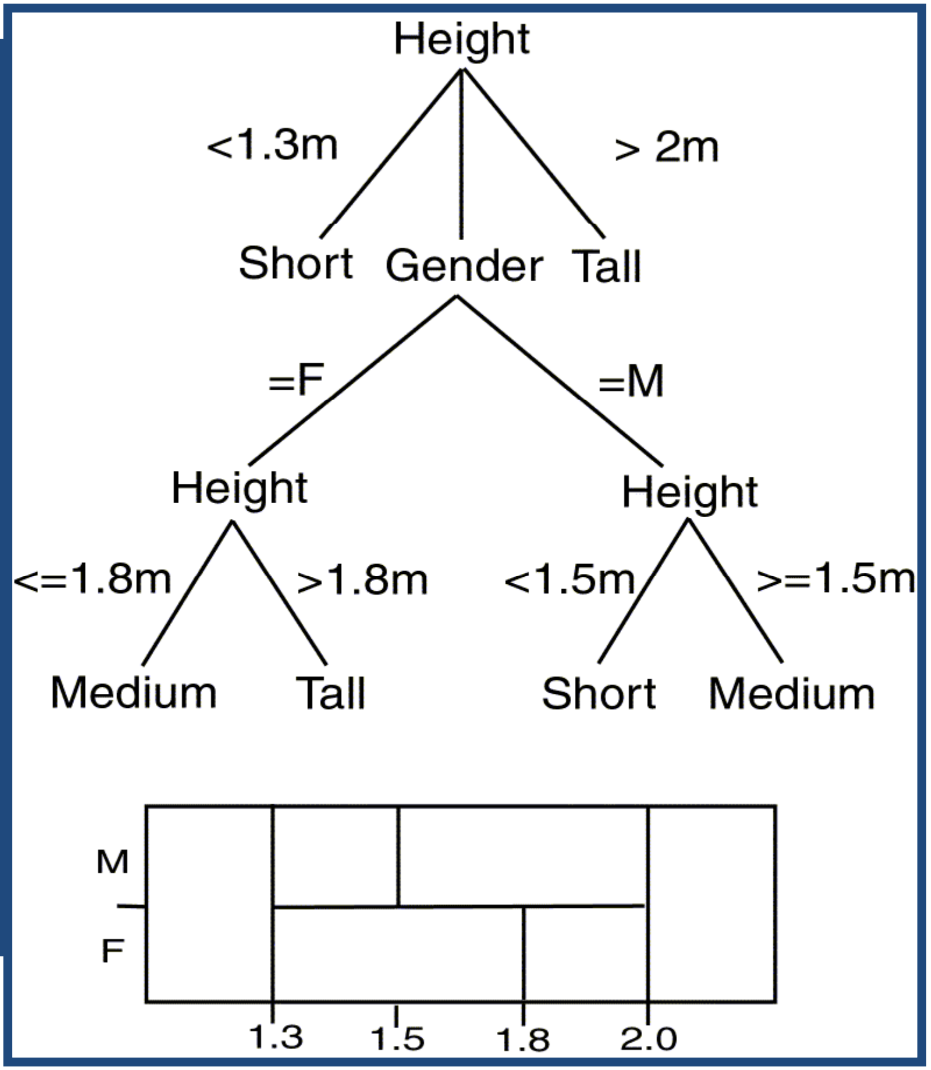
DT Splits Area



Comparing DTs



Balanced



Deep

DT Issues

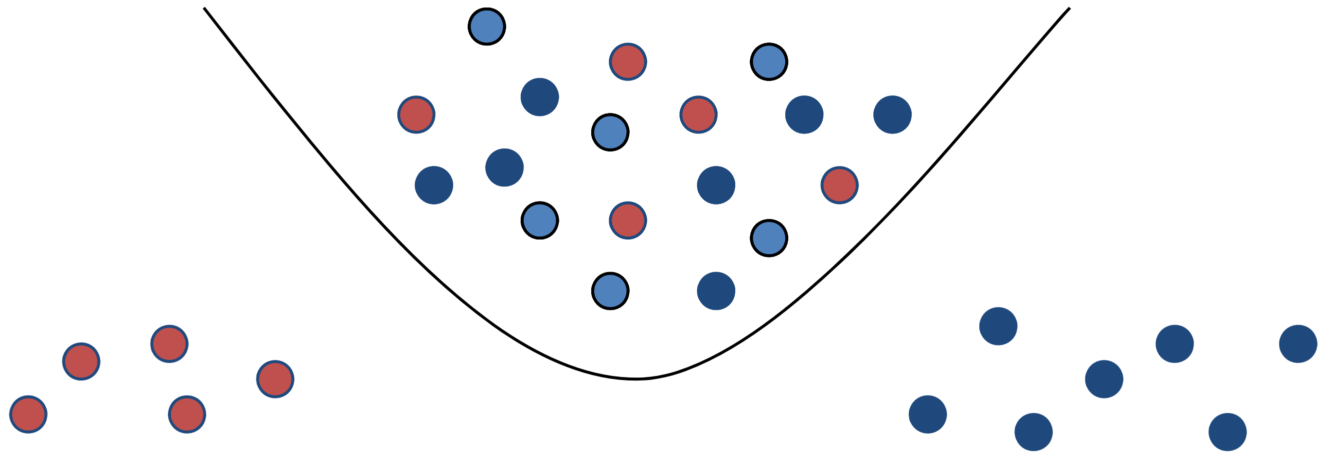
- Choosing Splitting Attributes
- Ordering of Splitting Attributes
- Splits
- Tree Structure
- Stopping Criteria
- Training Data
- Pruning

Decision Tree Induction is often based on Information Theory

So

$\longrightarrow$

Information



Information/Entropy

- Given probabilitites p_1, p_2, .., p_s whose sum is 1, ***Entropy*** is defined as:

$$\boxed{H(p_1, p_2, ..., p_s) = \sum_{i=1}^{s} (p_i log(1/p_i))}$$

- Entropy measures the amount of randomness or surprise or uncertainty.
- Goal in classification
  - no surprise
  - entropy = 0

Attribute Selection Measure: Information Gain (ID3/C4.5)

- **Select the attribute with the highest information gain**
- **S contains s_i tuples of class C_i for i = {1, …, m}**
- **information measures info required to classify any arbitrary tuple**

$$I(s_1, s_2, \ldots, s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log_2 \frac{s_i}{s}$$

- **entropy of attribute A with values $\{a_1, a_2, \ldots, a_v\}$**

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + \ldots + s_{mj}}{s} I(s_{1j}, \ldots, s_{mj})$$

- **information gained by branching on attribute A**

$$Gain(A) = I(s_1, s_2, \ldots, s_m) - E(A)$$

Attribute Selection by Information Gain Computation

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"
- I(p, n) = I(9, 5) =0.940
- Compute the entropy for *age*:

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
30…40	4	0	0
>40	3	2	0.971

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

$\frac{5}{14} I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

$$Gain(age) = I(p,n) - E(age) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

Other Attribute Selection Measures

- Gini index (CART, IBM IntelligentMiner)
  - All attributes are assumed continuous-valued
  - Assume there exist several possible split values for each attribute
  - May need other tools, such as clustering, to get the possible split values
  - Can be modified for categorical attributes

Gini Index (IBM IntelligentMiner)

- If a data set *T* contains examples from *n* classes, gini index, *gini*(*T*) is defined as

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2$$

where p_j is the relative frequency of class *j* in *T.*

- If a data set *T* is split into two subsets T_1 and T_2 with sizes N_1 and N_2 respectively, the *gini* index of the split data contains examples from *n* classes, the *gini* index *gini*(*T*) is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute provides the smallest $gini_{split}(T)$ is chosen to split the node (*need to enumerate all possible splitting points for each attribute*).

Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
- 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 node holds the class prediction
- Rules are easier for humans to understand
- Example

IF *age* = "<=30" AND *student* = "*no*" THEN *buys_computer* = "*no*"

IF *age* = "<=30" AND *student* = "*yes*" THEN *buys_computer* = "*yes*"

IF *age* = "31…40" THEN *buys_computer* = "*yes*"

IF *age* = ">40" AND *credit_rating* = "*excellent*" THEN *buys_computer* = "*yes*"

IF *age* = "<=30" AND *credit_rating* = "*fair*" THEN *buys_computer* = "*no*"

Avoid Overfitting in Classification

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the “best pruned tree”

Approaches to Determine the Final Tree Size

- Separate training (2/3) and testing (1/3) sets
- Use cross validation, e.g., 10-fold cross validation
- Use all the data for training
  - but apply a statistical test (e.g., chi-square) to estimate whether expanding or pruning a node may improve the entire distribution
- Use minimum description length (MDL) principle
  - halting growth of the tree when the encoding is minimized

Enhancements to basic decision tree induction

- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

Decision Tree vs. Rules

- Tree has implied order in which splitting is performed.
- Tree created based on looking at all classes.

- Rules have no ordering of predicates.
- Only need to look at one class to generate its rules.

Scalable Decision Tree Induction Methods in Data Mining Studies

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why decision tree induction in data mining?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods

Scalable Decision Tree Induction Methods in Data Mining Studies

- SLIQ (EDBT'96 — Mehta et al.)
  - builds an index for each attribute and only class list and the current attribute list reside in memory
- SPRINT (VLDB'96 — J. Shafer et al.)
  - constructs an attribute list data structure
- PUBLIC (VLDB'98 — Rastogi & Shim)
  - integrates tree splitting and tree pruning: stop growing the tree earlier
- RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - separates the scalability aspects from the criteria that determine the quality of the tree
  - builds an AVC-list (attribute, value, class label)

Instance-Based Methods

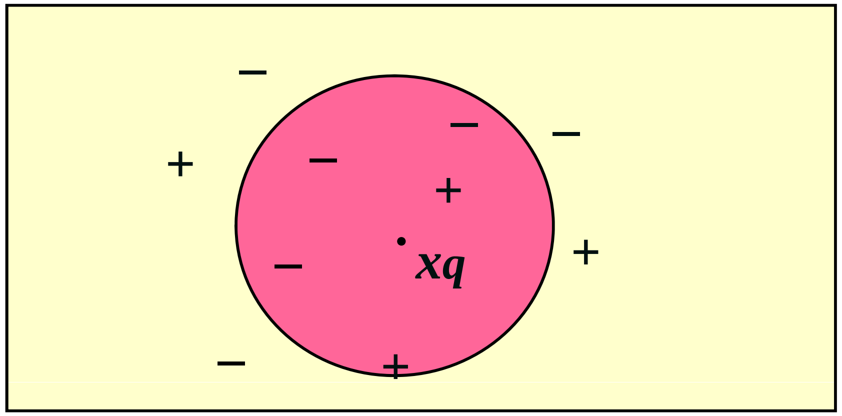
- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs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

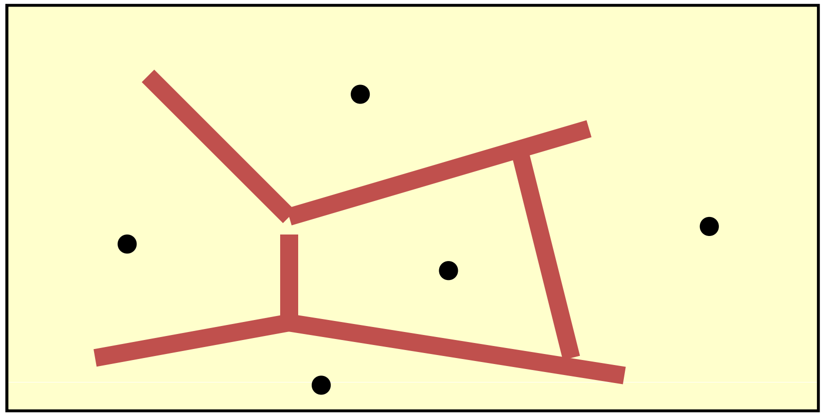
Classification Using Distance

- Place items in class to which they are "closest".
- Must determine distance between an item and a class.
- Classes represented by
  - ***Centroid:*** Central value.
  - ***Medoid:*** Representative point.
  - Individual points
- Algorithm: KNN

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.
- The target function could be discrete- or real- valued.
- For discrete-valued, the *k*-NN returns the most common value among the k training examples nearest to *xq*.
- Voronoi diagram: the decision surface induced by 1-NN for a typical set of training examples.

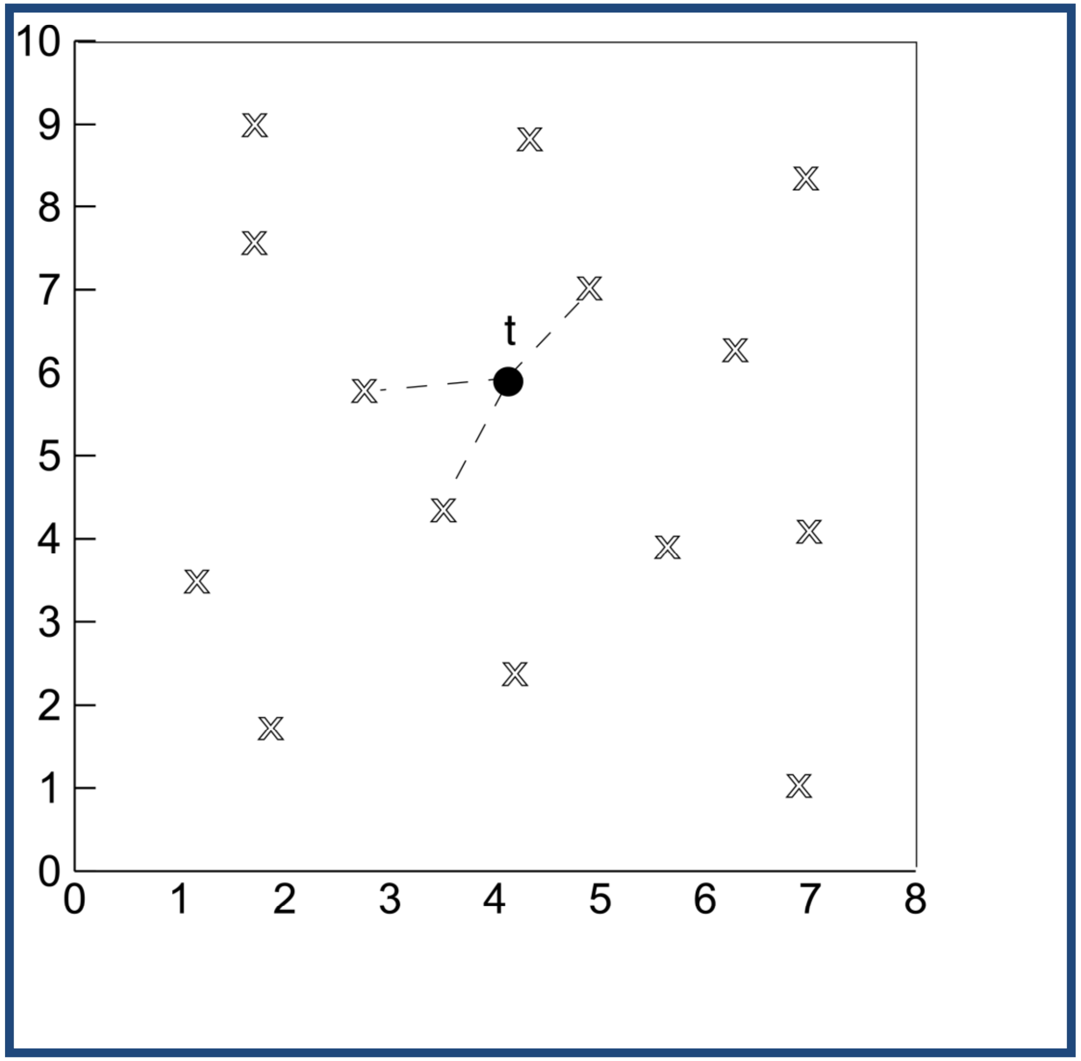




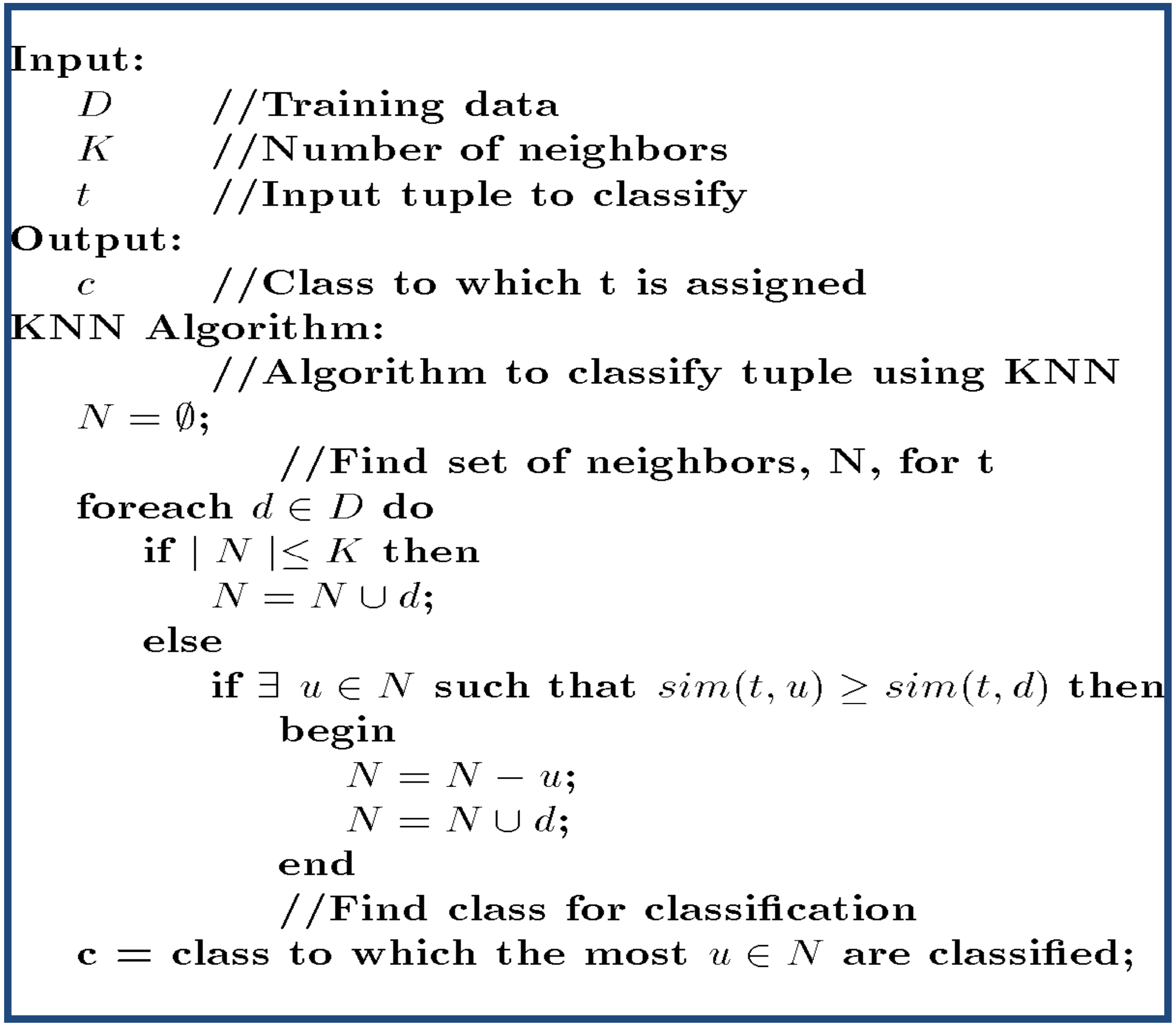
K Nearest Neighbor (KNN):

- Training set includes classes.
- Examine K items near item to be classified.
- New item placed in class with the most number of close items.
- O(q) for each tuple to be classified. (Here q is the size of the training set.)

KNN



KNN Algorithm



```
Input:
    $D$         //Training data
    $K$         //Number of neighbors
    $t$         //Input tuple to classify
Output:
    $c$         //Class to which t is assigned
KNN Algorithm:
                //Algorithm to classify tuple using KNN
    $N = \emptyset$;
                    //Find set of neighbors, N, for t
    foreach $d \in D$ do
        if $| N | \leq K$ then
            $N = N \cup d$;
        else
            if $\exists$ $u \in N$ such that $sim(t, u) \geq sim(t, d)$ then
                begin
                    $N = N - u$;
                    $N = N \cup d$;
                end
                //Find class for classification
    c $=$ class to which the most $u \in N$ are classified;
```

Bayesian Classification: Why?

- <u>Probabilistic learning</u>: Calculate explicit probabilities for hypothesis, among the most practical approaches to certain types of learning problems
- <u>Incremental</u>: Each training example can incrementally increase/decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.
- <u>Probabilistic prediction</u>: Predict multiple hypotheses, weighted by their probabilities
- <u>Standard</u>: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayesian Theorem: Basics

- Let X be a data sample whose class label is unknown
- Let H be a hypothesis that X belongs to class C
- For classification problems, determine P(H|X): the probability that the hypothesis holds given the observed data sample X
- P(H): prior probability of hypothesis H (i.e. the initial probability before we observe any data, reflects the background knowledge)
- P(X): probability that sample data is observed
- P(X|H) : probability of observing the sample X, given that the hypothesis holds

Bayes' Theorem

- Given training data *X, posteriori probability of a hypothesis H, P(H|X)* follows the Bayes theorem

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

- Informally, this can be written as

  posterior =likelihood x prior / evidence

- MAP (maximum posteriori) hypothesis

$$h_{MAP} \equiv \underset{h \in H}{\arg\max}\, P(h|D) = \underset{h \in H}{\arg\max}\, P(D|h)P(h).$$

- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent:

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

- The product of occurrence of say 2 elements x_1 and x_2, given the current class is C, is the product of the probabilities of each element taken separately, given the same class $P([y_1,y_2],C) = P(y_1,C) * P(y_2,C)$
- No dependence relation between attributes
- Greatly reduces the computation cost, only count the class distribution.
- Once the probability $P(X|C_i)$ is known, assign X to the class with maximum $P(X|C_i)*P(C_i)$

Training dataset

Class:
C1:buys_computer= 'yes'
C2:buys_computer= 'no'

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

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
30…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayesian Classifier: Example

- Compute P(X/Ci) for each class

  P(age="<30" | buys_computer="yes") = 2/9=0.222
  P(age="<30" | buys_computer="no") = 3/5 =0.6
  P(income="medium" | buys_computer="yes")= 4/9 =0.444
  P(income="medium" | buys_computer="no") = 2/5 = 0.4
  P(student="yes" | buys_computer="yes)= 6/9 =0.667
  P(student="yes" | buys_computer="no")= 1/5=0.2
  P(credit_rating="fair" | buys_computer="yes")=6/9=0.667
  P(credit_rating="fair" | buys_computer="no")=2/5=0.4

X=(age<=30 ,income =medium, student=yes,credit_rating=fair)

P(X|Ci) : P(X|buys_computer="yes")= 0.222 x 0.444 x 0.667 x 0.0.667 =0.044
P(X|buys_computer="no")= 0.6 x 0.4 x 0.2 x 0.4 =0.019

P(X|Ci)*P(Ci) : P(X|buys_computer="yes") * P(buys_computer="yes")=0.028
P(X|buys_computer="yes") * P(buys_computer="yes")=0.007

X belongs to class "buys_computer=yes"

Naïve Bayesian Classifier: Comments

- Advantages :
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence , therefore loss of accuracy
  - Practically, dependencies exist among variables
  - E.g., hospitals: patients: Profile: age, family history etc

    Symptoms: fever, cough etc., Disease: lung cancer, diabetes etc
  - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks

ICDM'06 Panel on “Top 10 Algorithms in Data Mining”

The 18 Identified Candidates

Statistical Learning

- #5. SVM: Vapnik, V. N. 1995. The Nature of Statistical Learning Theory. Springer-Verlag New York, Inc.
- #6. EM: McLachlan, G. and Peel, D. (2000). Finite Mixture Models. J. Wiley, New York.

Outline

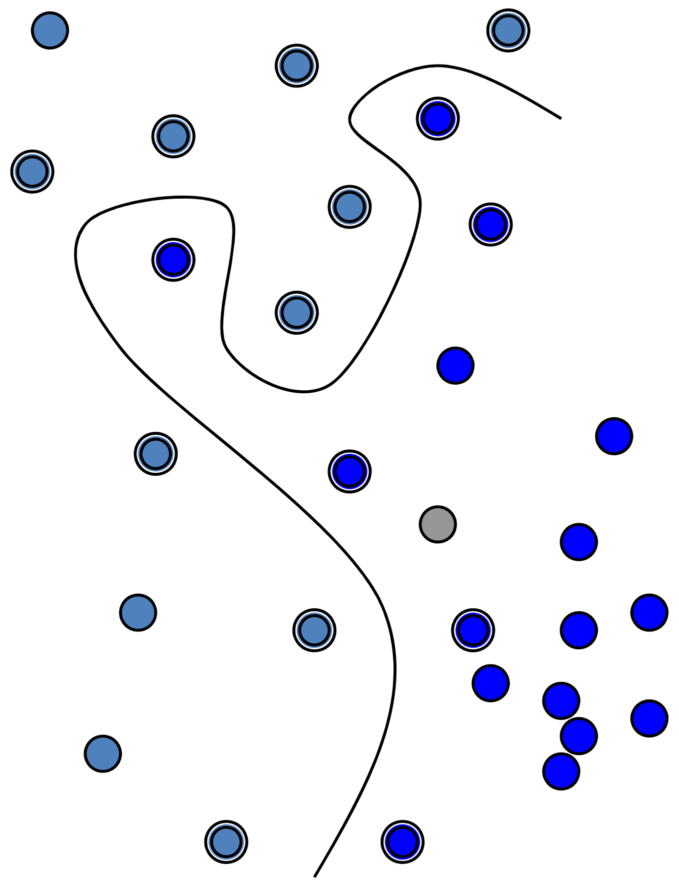
- Top 10 Algorithms in Data Mining Research
  - Introduction
  - Classification
  - Statistical Learning
  - Bagging and Boosting
  - Clustering
  - Association Analysis
  - Link Mining – Text Mining
  - Top 10 Algorithms
- 10 Challenging Problems in Data Mining Research
- Concluding Remarks

Support vector machine (SVM)

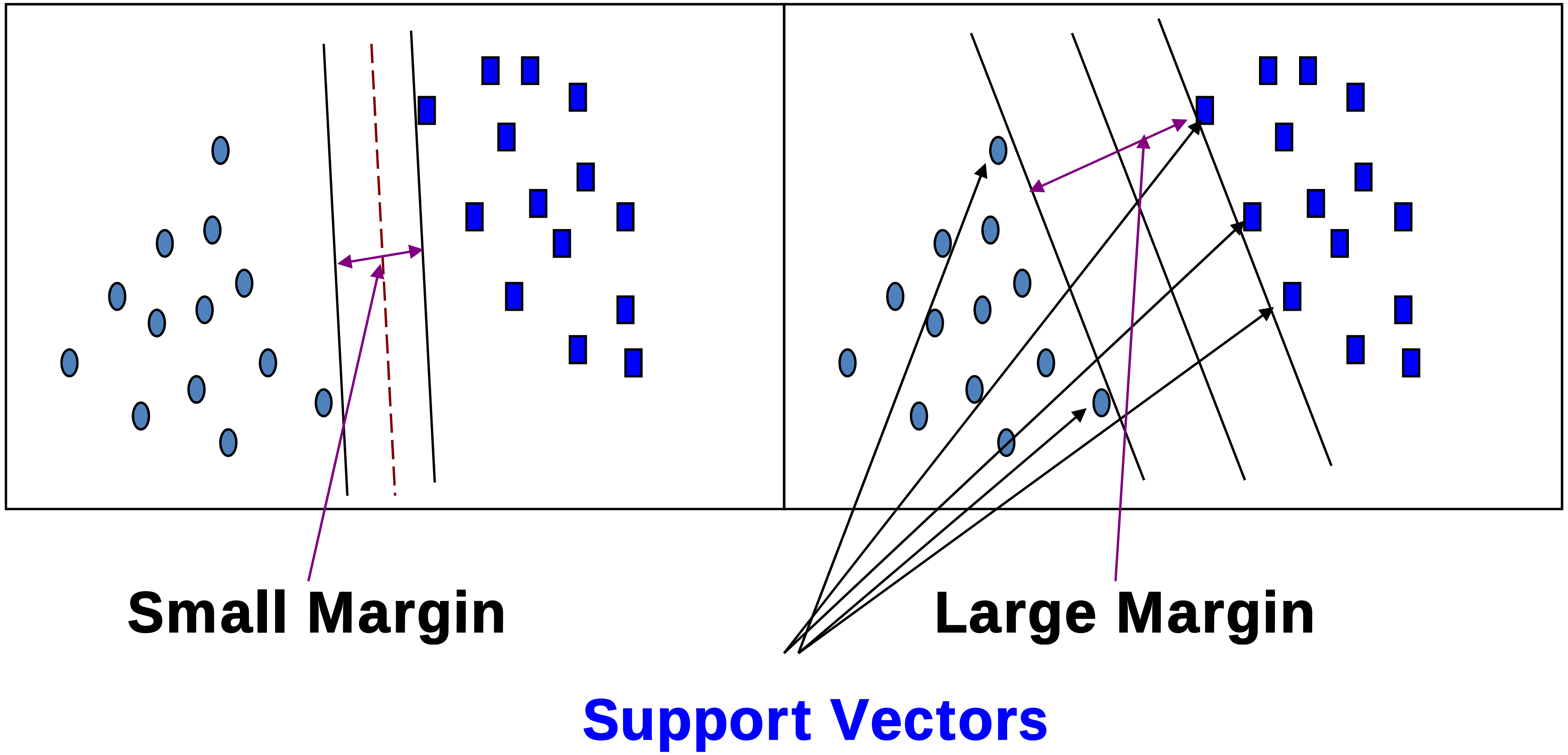
- Classification is essentially finding the best boundary between classes.
- Support vector machine finds the best boundary points called support vectors and build classifier on top of them.
- Linear and Non-linear support vector machine.

Example of general SVM

The dots with shadow around them are support vectors. Clearly they are the best data points to represent the boundary. The curve is the separating boundary.

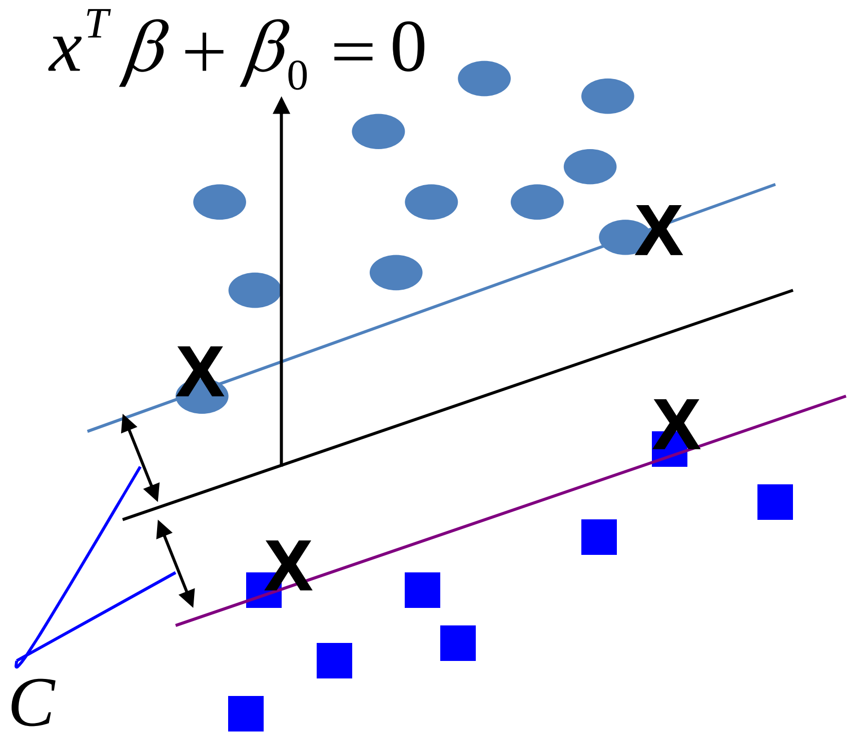


SVM – Support Vector Machines



Optimal Hyper plane, separable case.

- In this case, class 1 and class 2 are separable.
- The representing points are selected such that the margin between two classes are maximized.
- Crossed points are support vectors.



SVM – Cont.

- Linear Support Vector Machine

Given a set of points $x_i \in \Re^n$ with label $\mathrm{y_i} \in \{-1,1\}$
The SVM finds a hyperplane defined by the pair $(\mathrm{w,b})$
(where w is the normal to the plane and b is the distance from the origin)

s.t. $$y_i(x_i \cdot w + b) \geq +1 \quad i = 1,\ldots, N$$

x – feature vector, b- bias, y- class label, $||w||$ ***- margin***

Analysis of Separable case.

1. Through out our presentation, the training data consists of N pairs:(x1,y1), (x2,y2) ,…, (Xn,Yn).
2. Define a hyper plane:

$$\{x : f(x) = x^T \beta + \beta_0 = 0\}$$

where β is a unit vector. The classification rule is :

$$G(x) = sign[x^T \beta + \beta_0]$$

Analysis Cont.

3. So the problem of finding optimal hyperplane turns to: $\beta, \beta_0, \|\beta\|=1$

   Maximizing C on

   Subject to constrain:

$$y_i(x_i^T \beta + \beta_0) > C, i = 1, \ldots, N.$$

4. It's the same as :

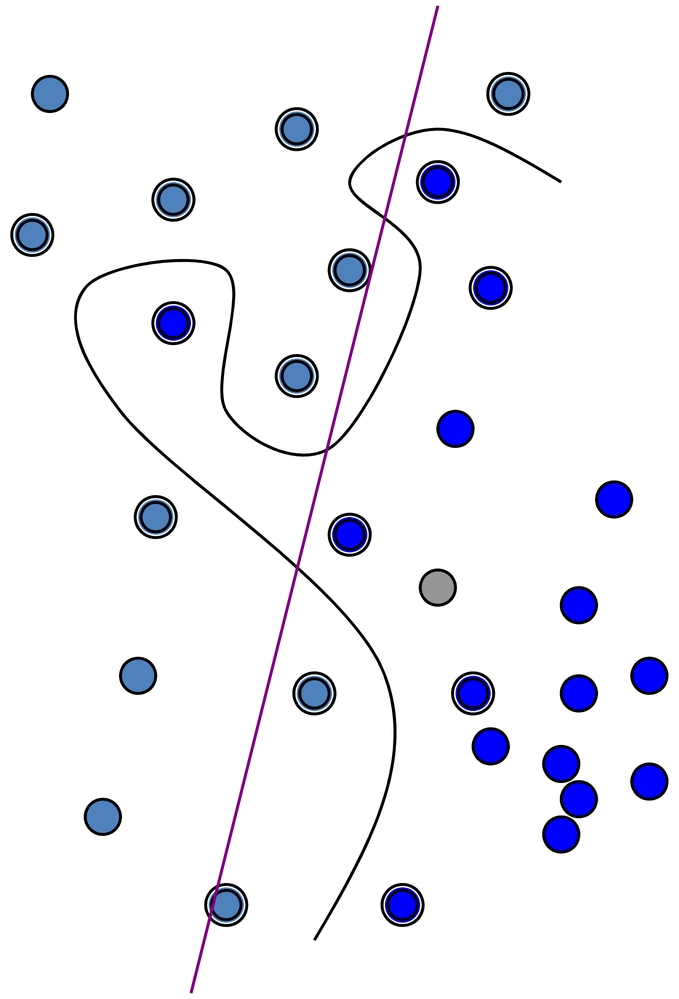
   Minimizing $\|\beta\|$ subject to

$$y_i(x_i^T \beta + \beta_0) > 1, i = 1, \ldots, N.$$

General SVM

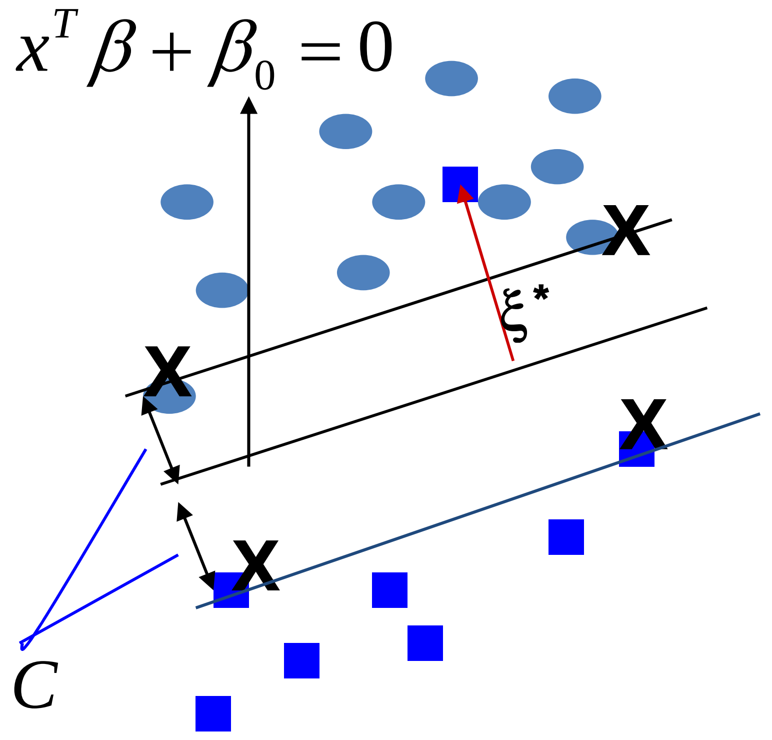
This classification problem clearly do not have a good optimal linear classifier.

Can we do better?
A non-linear boundary as shown will do fine.



Non-separable case

When the data set is non-separable as shown in the right figure, we will assign weight to each support vector which will be shown in the constraint.



Non-Linear SVM

Classification using SVM (*w,b*)

$$x_i \cdot w + b \overset{?}{>} 0$$

In non linear case we can see this as

$$\underset{\uparrow}{K}(x_i, w) + b \overset{?}{>} 0$$

Kernel – Can be thought of as doing dot product in some high dimensional space

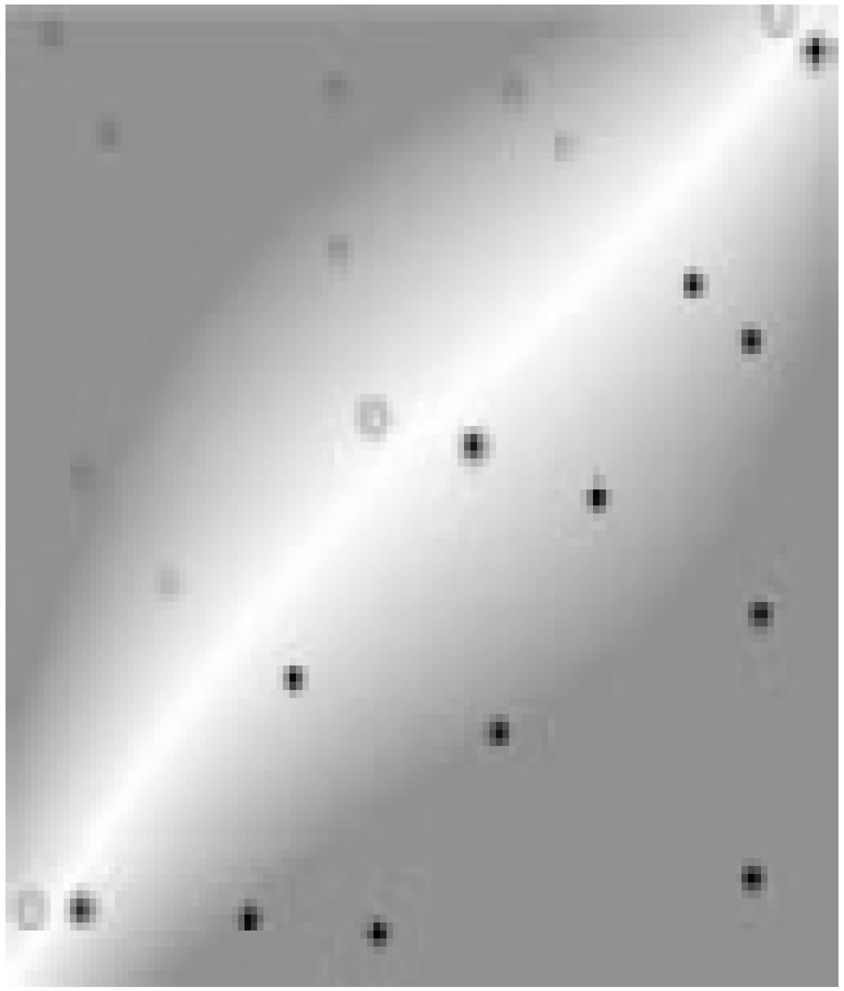
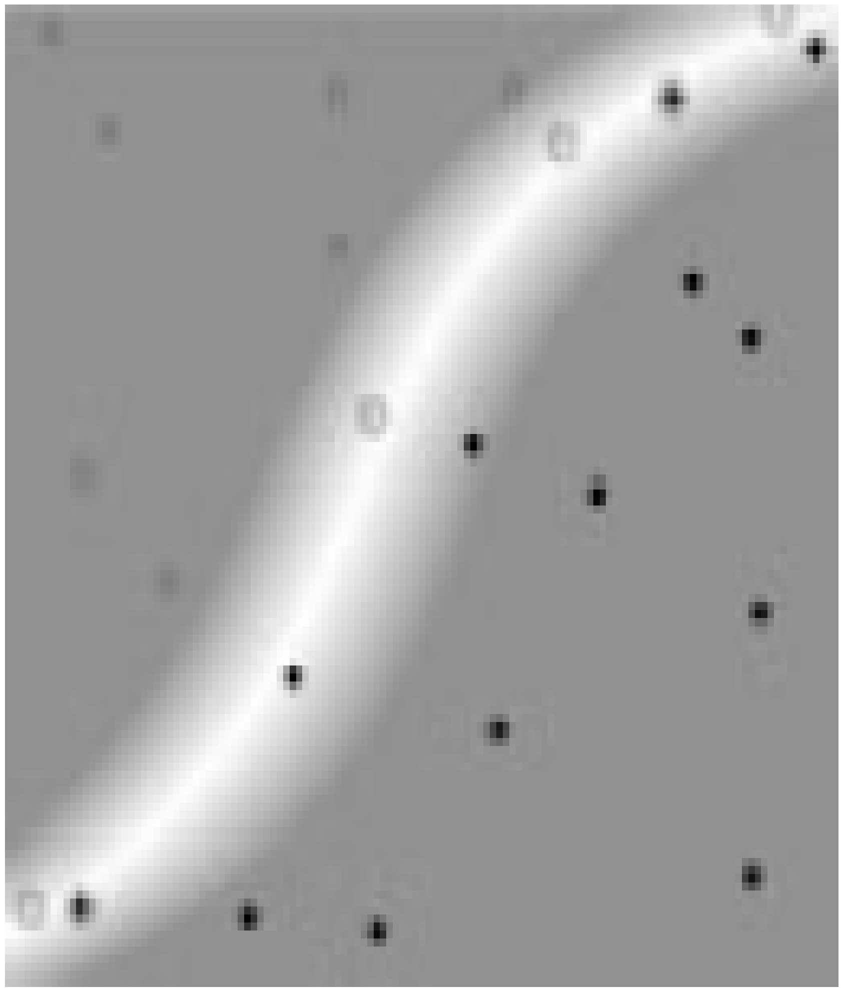
General SVM Cont.

Similar to linear case, the solution can be written as:

$$f(x) = h(x)^T \beta + \beta_0 = \sum_{i=1}^{N} \alpha_i y_i \langle h(x_i), h(x_{i'}) \rangle + \beta_0$$

But function "h" is of very high dimension sometimes infinity, does it mean SVM is impractical?

Resulting Surfaces

Reproducing Kernel.

Look at the dual problem, the solution

only depends on $\langle h(x_i), h(x_{i'}) \rangle$

Traditional functional analysis tells us we

need to only look at their kernel

representation: K(X,X')= $\langle h(x_i), h(x_{i'}) \rangle$

Which lies in a much smaller dimension

Space than “h”.

Restrictions and typical kernels.

- Kernel representation does not exist all the time, Mercer’s condition (Courant and Hilbert,1953) tells us the condition for this kind of existence.
- There are a set of kernels proven to be effective, such as polynomial kernels and radial basis kernels.

Example of polynomial kernel.

r degree polynomial:

K(x,x')=(1+<x,x'>)^d.

For a feature space with two inputs: x1,x2 and

a polynomial kernel of degree 2.

K(x,x')=(1+<x,x'>)^2

Let $h_1(x)=1, h_2(x)=\sqrt{2}x_1, h_3(x)=\sqrt{2}x_2, h_4(x)=x_1^2, h_5(x)=x_2^2$

and $h_6(x)=\sqrt{2}x_1x_2$, then K(x,x')=<h(x),h(x')>.

SVM vs. Neural Network

- SVM
  - Relatively new concept
  - Nice Generalization properties
  - Hard to learn – learned in batch mode using quadratic programming techniques
  - Using kernels can learn very complex functions
- Neural Network
  - Quiet Old
  - Generalizes well but doesn’t have strong mathematical foundation
  - Can easily be learned in incremental fashion
  - To learn complex functions – use multilayer perceptron (not that trivial)

Open problems of SVM.

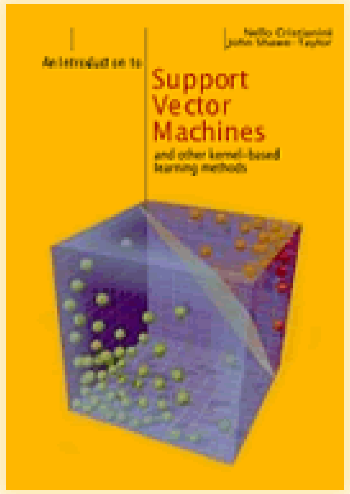
- How do we choose Kernel function for a specific set of problems. Different Kernel will have different results, although generally the results are better than using hyper planes.
- Comparisons with Bayesian risk for classification problem. Minimum Bayesian risk is proven to be the best. When can SVM achieve the risk.

Open problems of SVM

- For very large training set, support vectors might be of large size. Speed thus becomes a bottleneck.
- A optimal design for multi-class SVM classifier.

SVM Related Links

- http://svm.dcs.rhbnc.ac.uk/
- http://www.kernel-machines.org/
- C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Knowledge Discovery and Data Mining*, 2(2), 1998.
- SVM*light* – Software (in C) http://ais.gmd.de/~thorsten/svm_light
- BOOK: An Introduction to Support Vector Machines
  N. Cristianini and J. Shawe-Taylor
  Cambridge University Press, 2000



Outline

- Top 10 Algorithms in Data Mining Research
  - Introduction
  - Classification
  - Statistical Learning
  - Bagging and Boosting
  - Clustering
  - Association Analysis
  - Link Mining – Text Mining
  - Top 10 Algorithms
- 10 Challenging Problems in Data Mining Research
- Concluding Remarks

ICDM'06 Panel on "Top 10 Algorithms in Data Mining"

IEEE COMPUTER SOCIETY

The 18 Identified Candidates

Bagging and Boosting

- #13. AdaBoost: Freund, Y. and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 55, 1 (Aug. 1997), 119-139.

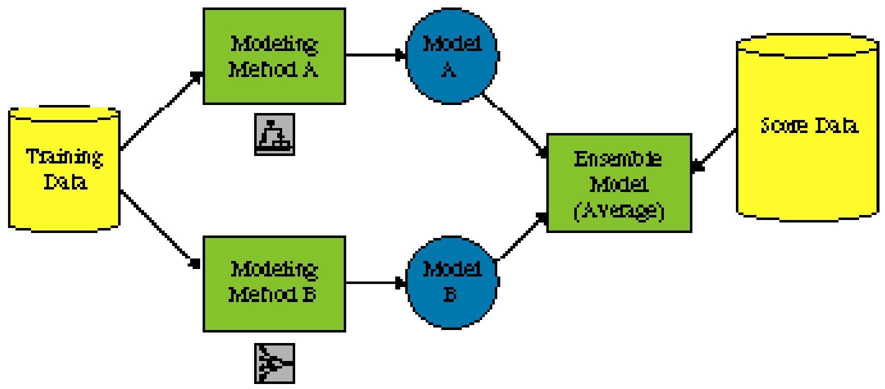
Combining classifiers

- Examples: classification trees and neural networks, several neural networks, several classification trees, etc.
- Average results from different models
- Why?
  - Better classification performance than individual classifiers
  - More resilience to noise
- Why not?
  - Time consuming
  - Overfitting

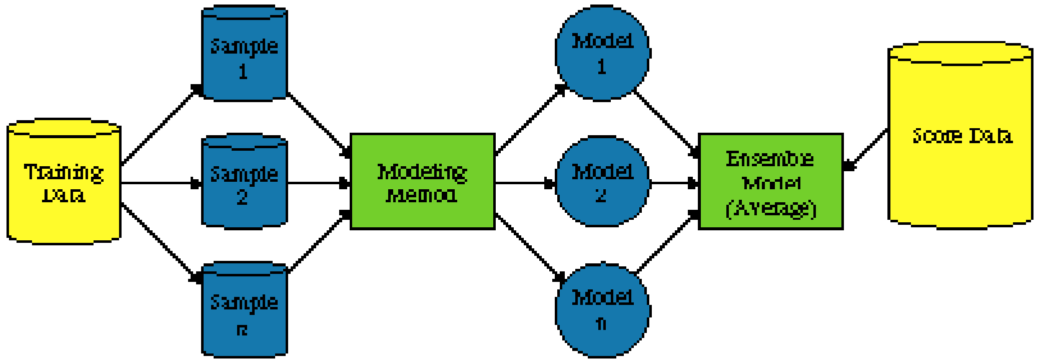
Combining classifiers

Ensemble methods classification

- *Manipulation with model* (Model= M(α))



- *Manipulation with data set*



Bagging and Boosting

- Bagging=*Manipulation with data set*

- Boosting = *Manipulation with model*

Bagging and Boosting

- General idea

$$\text{Training data} \xrightarrow{\text{Classification method (CM)}} \textbf{Classifier C}$$

$$\text{Altered Training data} \xrightarrow{\text{CM}} \textbf{Classifier C1}$$

$$\text{Altered Training data} \xrightarrow{\text{CM}} \textbf{Classifier C2}$$

……..

$$\text{Aggregation ....} \longrightarrow \textbf{Classifier C*}$$

Bagging

- Breiman, 1996
- Derived from bootstrap (Efron, 1993)

- Create classifiers using training sets that are bootstrapped (drawn with replacement)

- Average results for each case

Bagging

- Given a set S of s samples
- Generate a bootstrap sample T from S. Cases in S may not appear in T or may appear more than once.
- Repeat this sampling procedure, getting a sequence of k independent training sets
- A corresponding sequence of classifiers C1,C2,…,Ck is constructed for each of these training sets, by using the same classification algorithm
- To classify an unknown sample X,let each classifier predict or vote
- The Bagged Classifier C* counts the votes and assigns X to the class with the “most” votes

Bagging Example (Opitz, 1999)

Original	1	2	3	4	5	6	7	8
Training set 1	2	7	8	3	7	6	3	1
Training set 2	7	8	5	6	4	2	7	1
Training set 3	3	6	2	7	5	6	2	2
Training set 4	4	5	1	4	6	4	3	8

Boosting

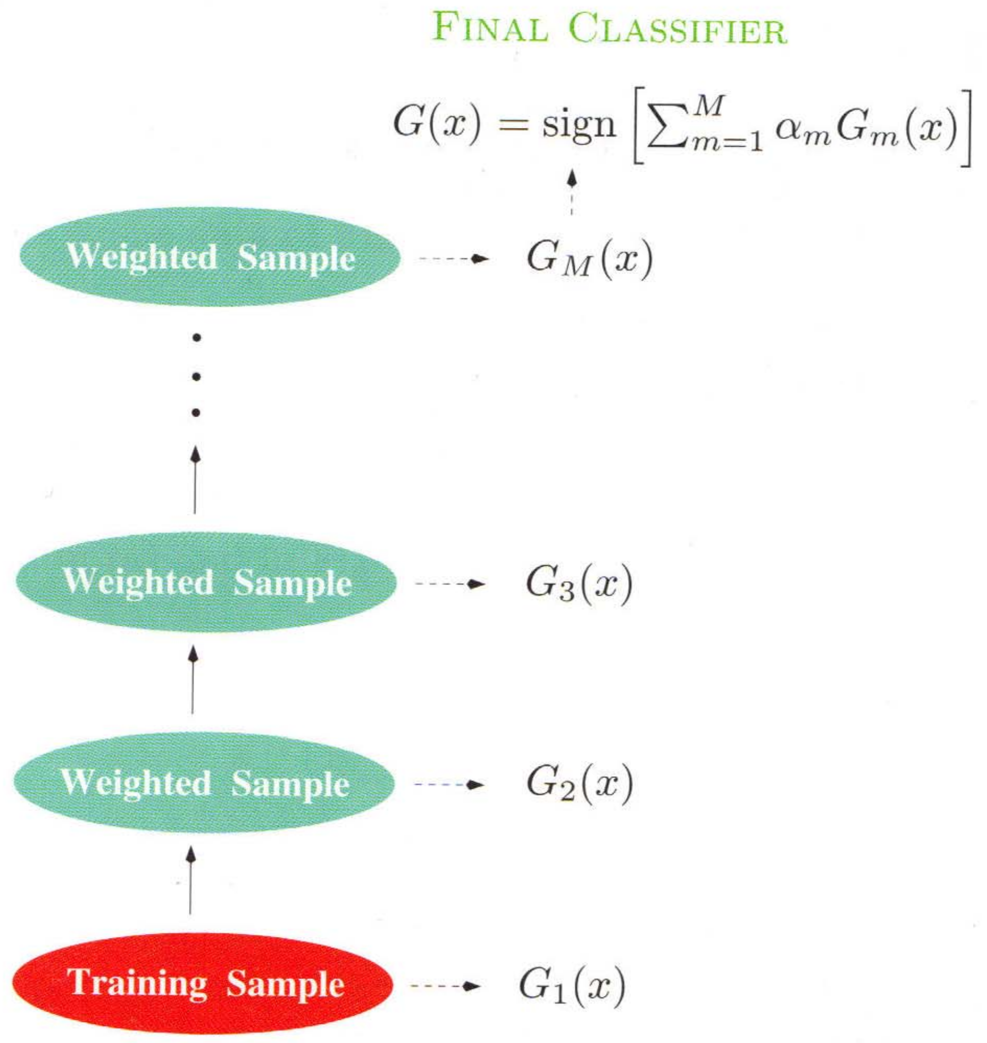
- A family of methods
- Sequential production of classifiers
- Each classifier is dependent on the previous one, and focuses on the previous one's errors
- Examples that are incorrectly predicted in previous classifiers are chosen more often or weighted more heavily

Boosting Technique — Algorithm

- Assign every example an equal weight *1/N*
- *For t = 1, 2, …, T Do*
  - Obtain a hypothesis (classifier) $h^{(t)}$ under $w^{(t)}$
  - Calculate the error of *h(t)* and re-weight the examples based on the error . Each classifier is dependent on the previous ones. Samples that are incorrectly predicted are weighted more heavily
  - Normalize $w^{(t+1)}$ to sum to 1 (weights assigned to different classifiers sum to 1)
- Output a weighted sum of all the hypothesis, with each hypothesis weighted according to its accuracy on the training set

Boosting

The idea



Ada-Boosting

- Freund and Schapire, 1996
- Two approaches
  - Select examples according to error in previous classifier (more representatives of misclassified cases are selected) – more common
  - Weigh errors of the misclassified cases higher (all cases are incorporated, but weights are different) – does not work for some algorithms

Ada-Boosting

- Define ε_k as the sum of the probabilities for the misclassified instances for current classifier C_k
- Multiply probability of selecting misclassified cases by

$$\beta_k = (1 - \varepsilon_k)/ \varepsilon_k$$

- "Renormalize" probabilities (i.e., rescale so that it sums to 1)
- Combine classifiers $C_1 \ldots C_k$ using weighted voting where C_k has weight $\log(\beta_k)$

Boosting Example (Opitz, 1999)

Original	1	2	3	4	5	6	7	8
Training set 1	2	7	8	3	7	6	3	1
Training set 2	1	4	5	4	1	5	6	4
Training set 3	7	1	5	8	1	8	1	4
Training set 4	1	1	6	1	1	3	1	5

Outline

- Top 10 Algorithms in Data Mining Research
  - Introduction
  - Classification
  - Statistical Learning
  - Bagging and Boosting
  - Clustering
  - Association Analysis
  - Link Mining – Text Mining
  - Top 10 Algorithms
- 10 Challenging Problems in Data Mining Research
- Concluding Remarks

ICDM'06 Panel on "Top 10 Algorithms in Data Mining"

The 18 Identified Candidates

Clustering

- #11. K-Means: MacQueen, J. B., Some methods for classification and analysis of multivariate observations, in Proc. 5th Berkeley Symp. Mathematical Statistics and Probability, 1967.
- #12. BIRCH Zhang, T., Ramakrishnan, R., and Livny, M. 1996. BIRCH: an efficient data clustering method for very large databases. In SIGMOD '96.

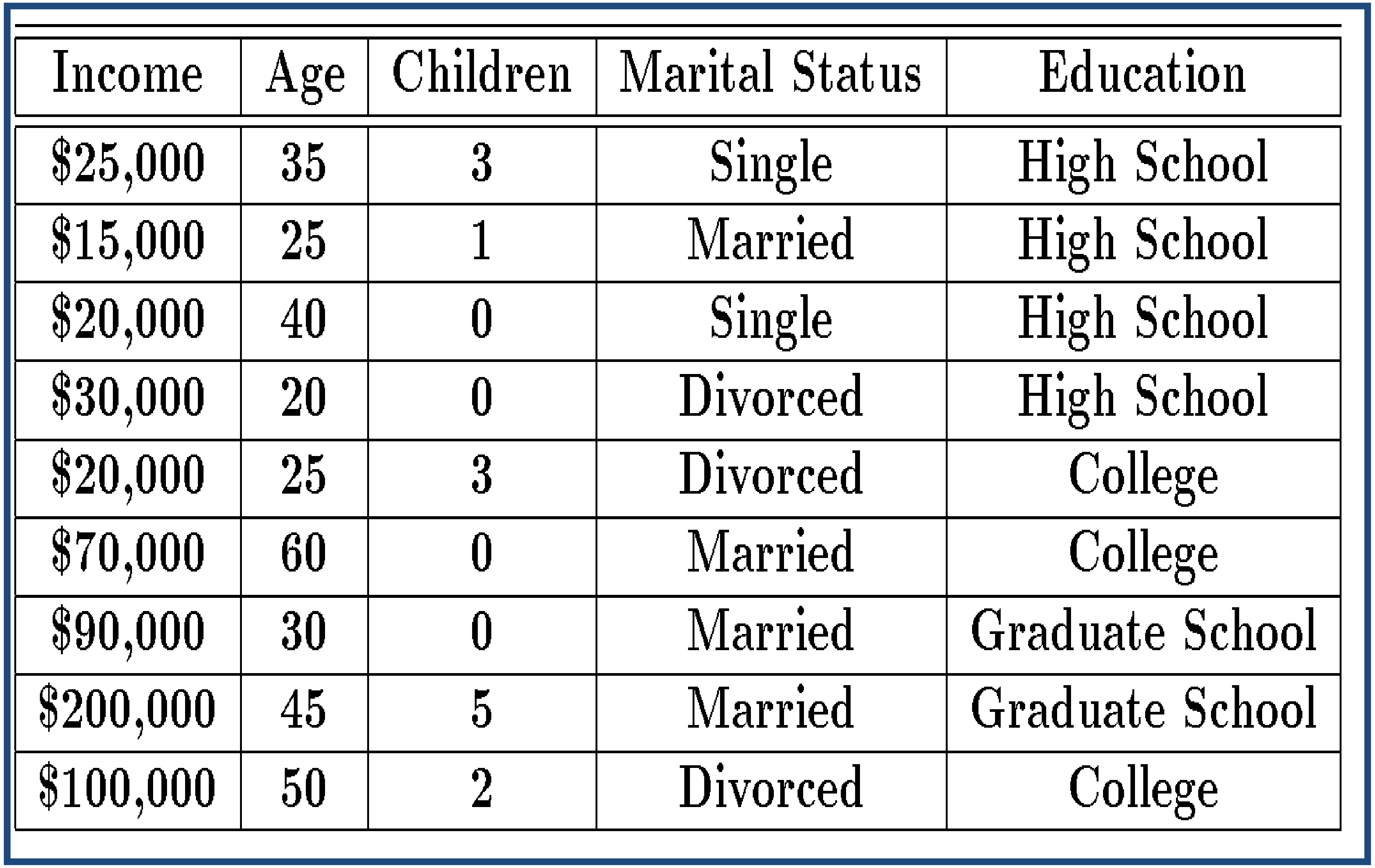
Clustering Problem

- Given a database $D=\{t_1,t_2,\ldots,t_n\}$ of tuples and an integer value k, the ***Clustering Problem*** is to define a mapping $f:D \rightarrow \{1,..,k\}$ where each t_i is assigned to one cluster K_j, $1<=j<=k$.
- A ***Cluster***, K_j, contains precisely those tuples mapped to it.
- Unlike classification problem, clusters are not known a priori.

Clustering Examples

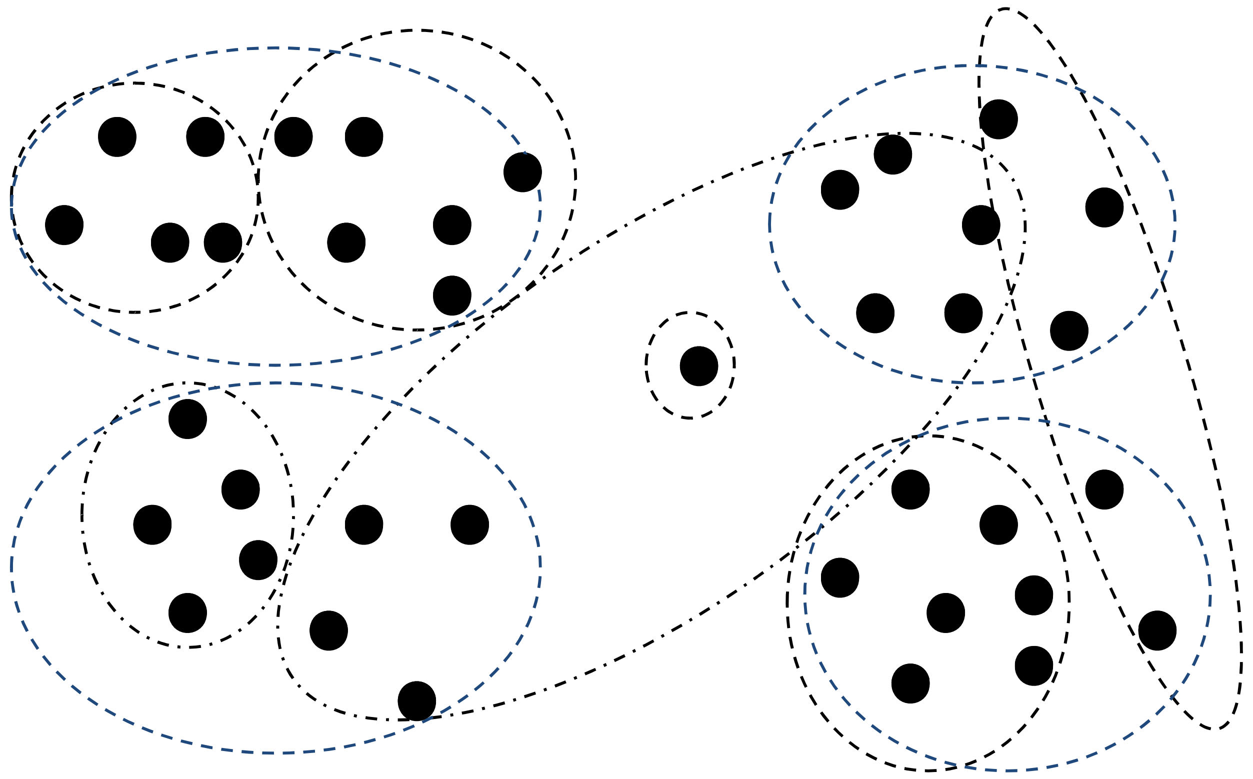
- ***Segment*** customer database based on similar buying patterns.
- Group houses in a town into neighborhoods based on similar features.
- Identify new plant species
- Identify similar Web usage patterns

Clustering Example



Income	Age	Children	Marital Status	Education
$25,000	35	3	Single	High School
$15,000	25	1	Married	High School
$20,000	40	0	Single	High School
$30,000	20	0	Divorced	High School
$20,000	25	3	Divorced	College
$70,000	60	0	Married	College
$90,000	30	0	Married	Graduate School
$200,000	45	5	Married	Graduate School
$100,000	50	2	Divorced	College

Clustering Levels



Size Based

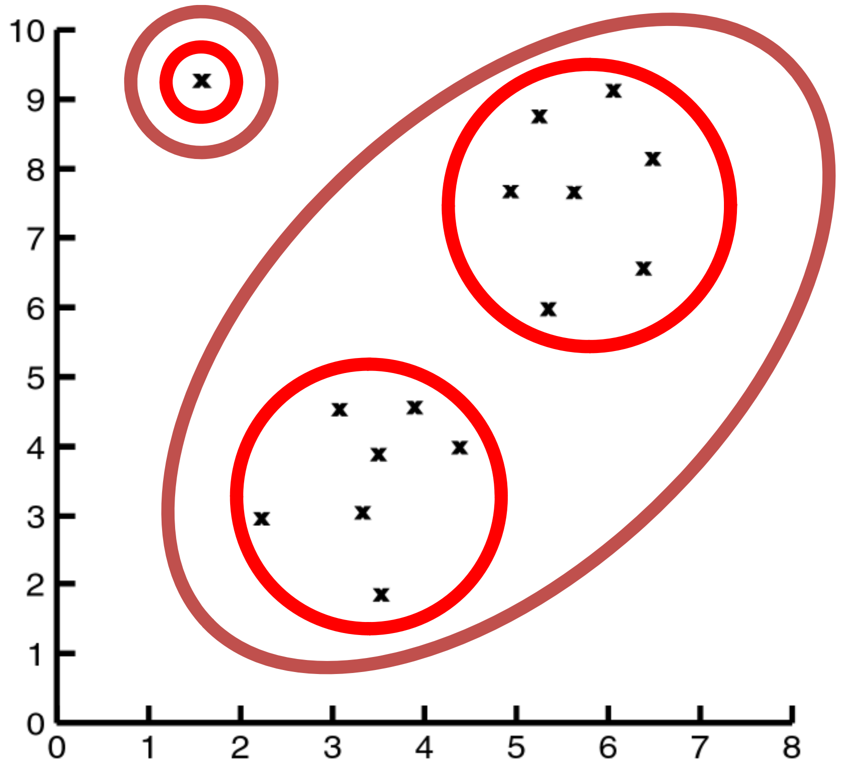
Clustering vs. Classification

- No prior knowledge
  - Number of clusters
  - Meaning of clusters
- Unsupervised learning

Clustering Issues

- Outlier handling
- Dynamic data
- Interpreting results
- Evaluating results
- Number of clusters
- Data to be used
- Scalability

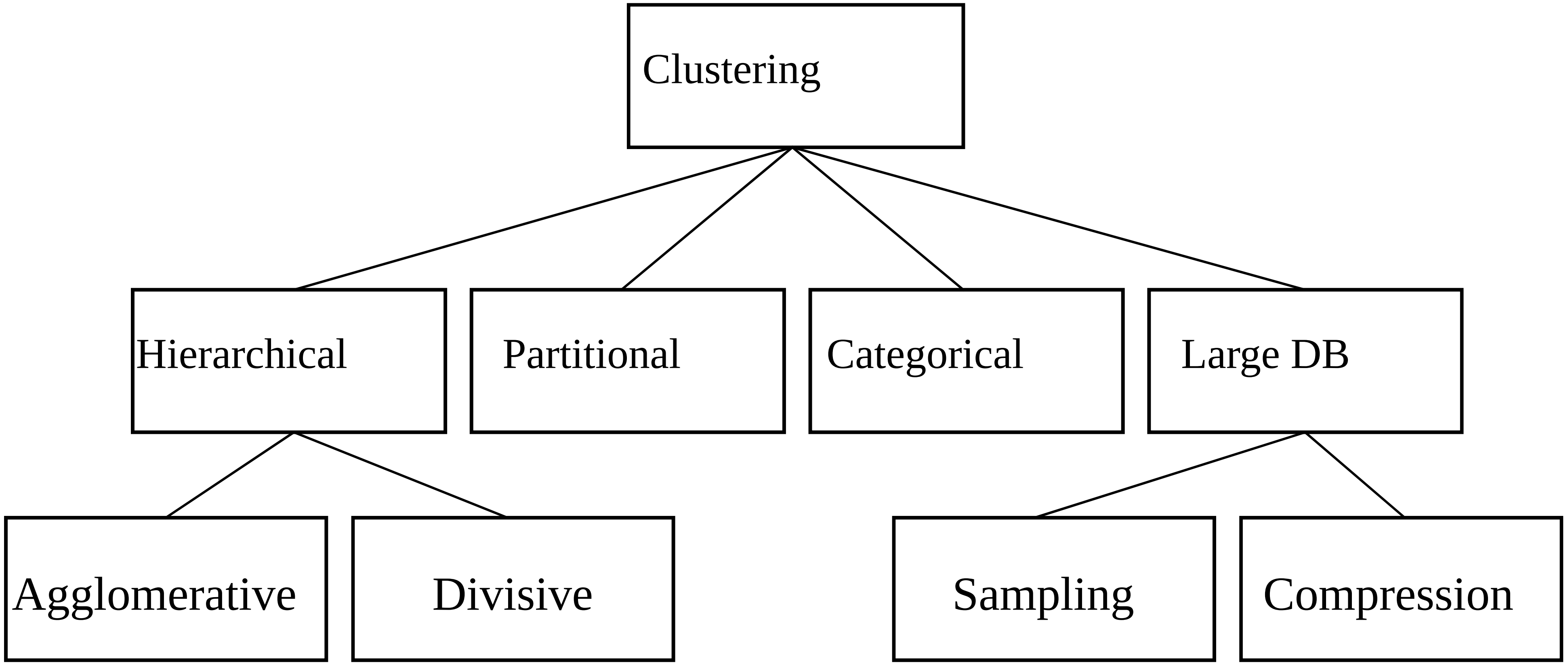
Impact of Outliers on Clustering



Types of Clustering

- ***Hierarchical*** – Nested set of clusters created.
- ***Partitional*** – One set of clusters created.
- ***Incremental*** – Each element handled one at a time.
- ***Simultaneous*** – All elements handled together.
- ***Overlapping/Non-overlapping***

Clustering Approaches



Cluster Parameters

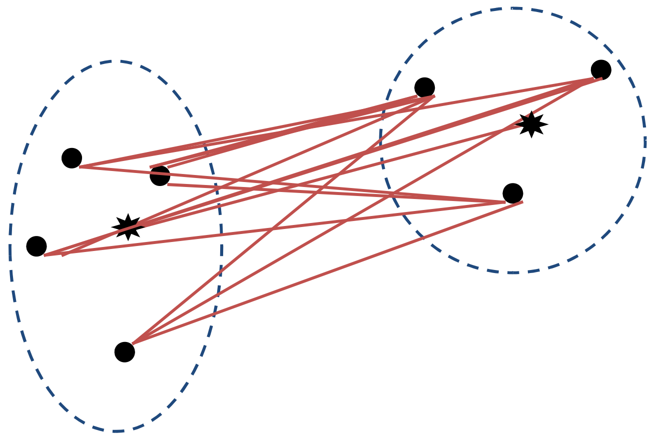
$$centroid = C_m = \frac{\sum_{i=1}^{N}(t_{mi})}{N}$$

$$radius = R_m = \sqrt{\frac{\sum_{i=1}^{N}(t_{mi} - C_m)^2}{N}}$$

$$diameter = D_m = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{N}(t_{mi} - t_{mj})^2}{(N)(N-1)}}$$

Distance Between Clusters

- ***Single Link***: smallest distance between points
- ***Complete Link:*** largest distance between points
- ***Average Link:*** average distance between points
- ***Centroid:*** distance between centroids



Hierarchical Clustering

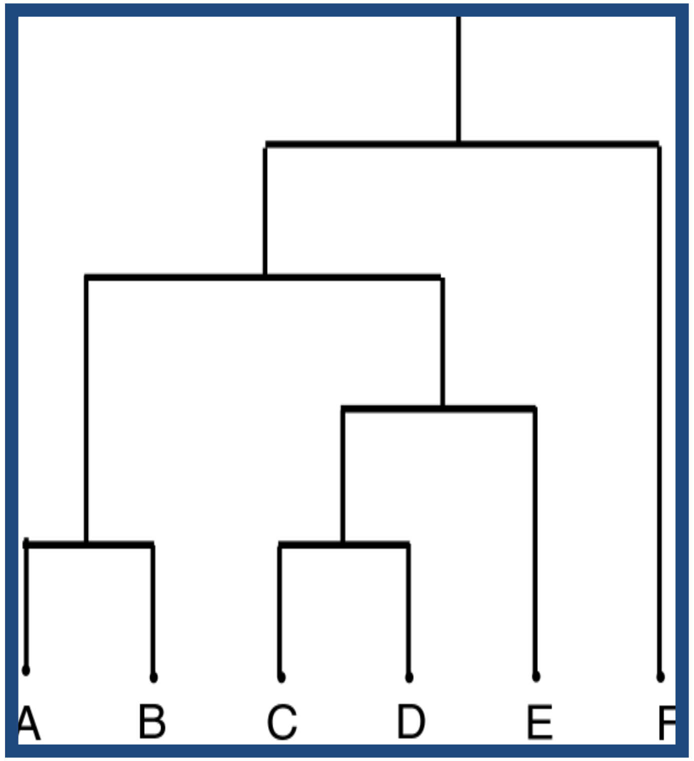
- Clusters are created in levels actually creating sets of clusters at each level.
- ***Agglomerative***
  - Initially each item in its own cluster
  - Iteratively clusters are merged together
  - Bottom Up
- ***Divisive***
  - Initially all items in one cluster
  - Large clusters are successively divided
  - Top Down

Hierarchical Algorithms

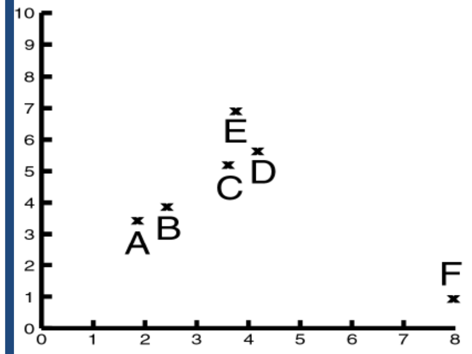
- Single Link
- MST Single Link
- Complete Link
- Average Link

Dendrogram

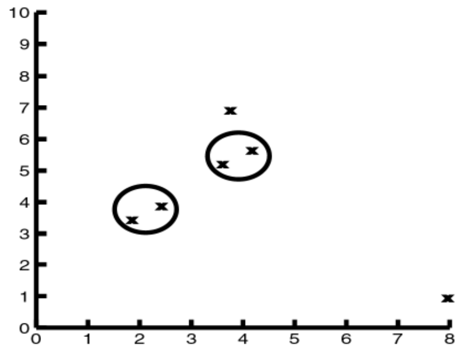
- ***Dendrogram:*** a tree data structure which illustrates hierarchical clustering techniques.
- Each level shows clusters for that level.
  - Leaf – individual clusters
  - Root – one cluster
- A cluster at level i is the union of its children clusters at level i+1.



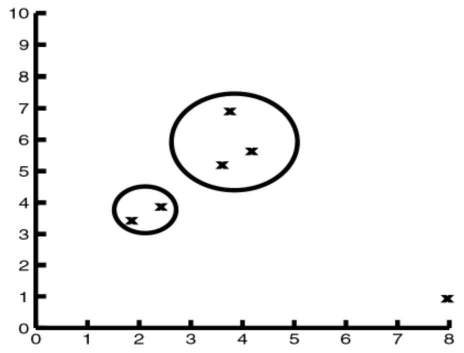
Levels of Clustering



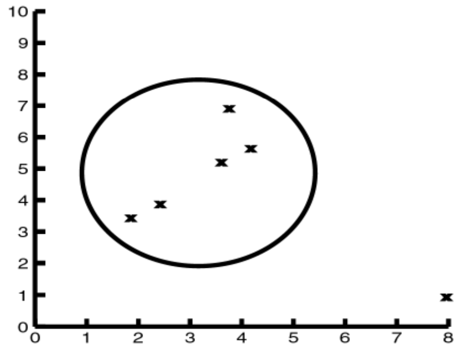
a) Six Clusters



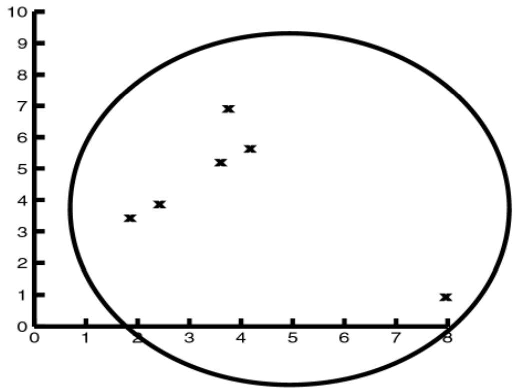
b) Four Clusters



c) Three Clusters



d) Two Clusters



e) One Cluster

Partitional Clustering

- Nonhierarchical
- Creates clusters in one step as opposed to several steps.
- Since only one set of clusters is output, the user normally has to input the desired number of clusters, k.
- Usually deals with static sets.

Partitional Algorithms

- MST
- Squared Error
- K-Means
- Nearest Neighbor
- PAM
- BEA
- GA

K-Means

- Initial set of clusters randomly chosen.
- Iteratively, items are moved among sets of clusters until the desired set is reached.
- High degree of similarity among elements in a cluster is obtained.
- Given a cluster $K_i=\{t_{i1},t_{i2},\ldots,t_{im}\}$, the ***cluster mean*** is $m_i = (1/m)(t_{i1} + \ldots + t_{im})$

K-Means Example

- Given: {2,4,10,12,3,20,30,11,25}, k=2
- Randomly assign means: m_1=3,m_2=4
- K_1={2,3}, K_2={4,10,12,20,30,11,25}, m_1=2.5,m_2=16
- K_1={2,3,4},K_2={10,12,20,30,11,25}, m_1=3,m_2=18
- K_1={2,3,4,10},K_2={12,20,30,11,25}, m_1=4.75,m_2=19.6
- K_1={2,3,4,10,11,12},K_2={20,30,25}, m_1=7,m_2=25
- Stop as the clusters with these means are the same.

K-Means Algorithm

```
Input:
    $D = \{t_1, t_2, ..., t_n\}$    // Set of elements
    $A$    // Adjacency matrix showing distance between elements.
    $k$    // Number of desired clusters.
Output:
    $K$    // Set of clusters.
K-Means Algorithm:
    assign initial values for means $m_1, m_2, ..., m_k$ ;
    repeat
        assign each item $t_i$ to the cluster which has the closest mean ;
        calculate new mean for each cluster;
    until convergence criteria is met;
```

Clustering Large Databases

- Most clustering algorithms assume a large data structure which is memory resident.
- Clustering may be performed first on a sample of the database then applied to the entire database.
- Algorithms
  - BIRCH
  - DBSCAN
  - CURE

Desired Features for Large Databases

- One scan (or less) of DB
- Online
- Suspendable, stoppable, resumable
- Incremental
- Work with limited main memory
- Different techniques to scan (e.g. sampling)
- Process each tuple once

BIRCH

- Balanced Iterative Reducing and Clustering using Hierarchies
- Incremental, hierarchical, one scan
- Save clustering information in a tree
- Each entry in the tree contains information about one cluster
- New nodes inserted in closest entry in tree

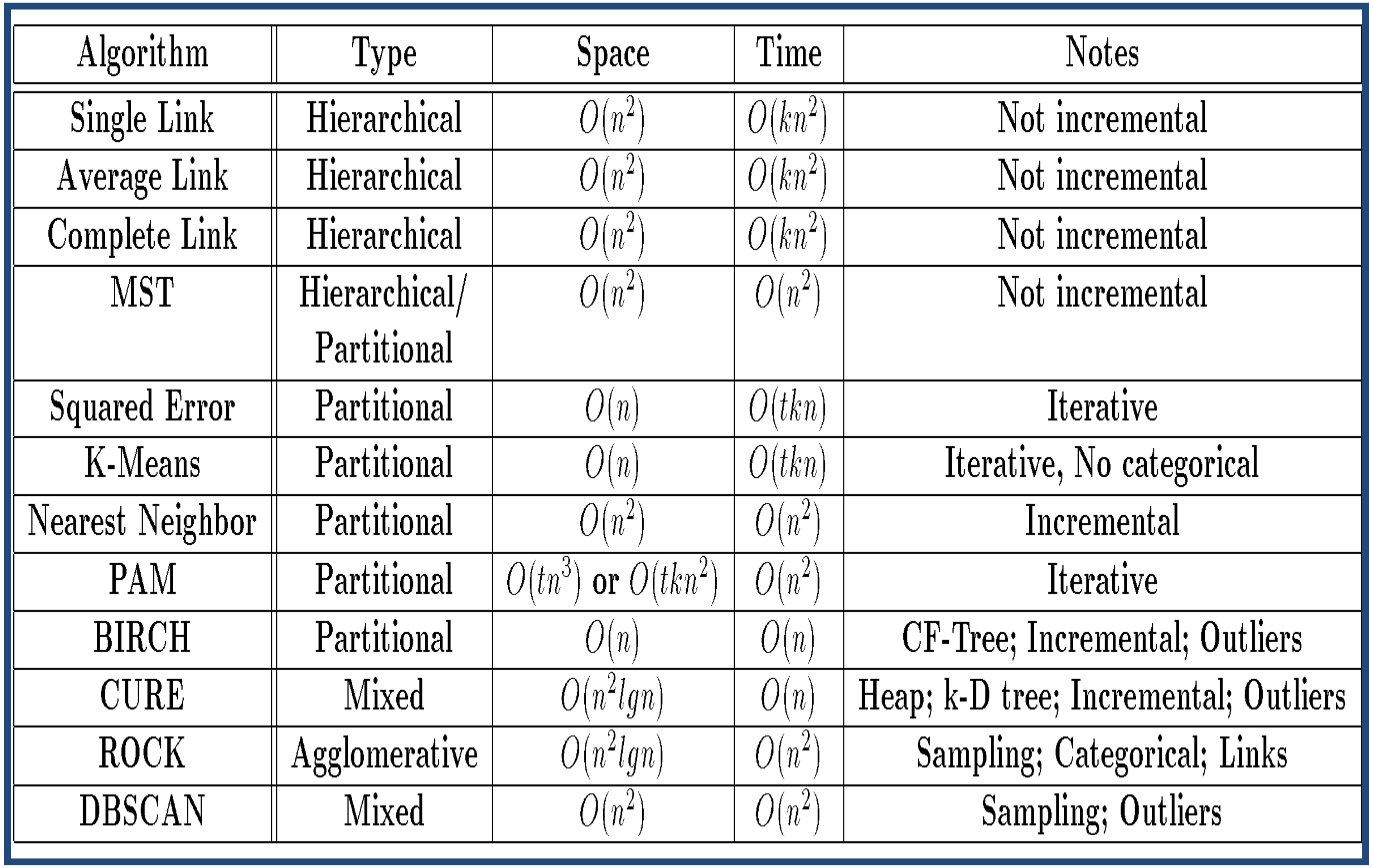
Clustering Feature

- CT Triple: $(N, LS, \overrightarrow{SS})$
  - N: Number of points in cluster
  - $\overrightarrow{LS}$: Sum of points in the cluster
  - SS: Sum of squares of points in the cluster
- CF Tree
  - Balanced search tree
  - Node has CF triple for each child
  - Leaf node represents cluster and has CF value for each subcluster in it.
  - Subcluster has maximum diameter

BIRCH Algorithm

```
Input:
    $D = \{t_1, t_2, ..., t_n\}$    // Set of elements
    $T$    // Threshold for CF tree construction.
Output:
    $K$    // Set of clusters.
BIRCH Clustering Algorithm:
    for each $t_i \in D$ do
        determine correct leaf node for $t_i$ insertion;
        if threshold condition is not violated then
            add $t_i$ to cluster and update CF triples;
        else
            if room to insert $t_i$ then
                insert $t_i$ as single cluster and update CF triples;
            else
                split leaf node and redistribute CF features;
```

Comparison of Clustering Techniques



Algorithm	Type	Space	Time	Notes
Single Link	Hierarchical	$O(n^2)$	$O(kn^2)$	Not incremental
Average Link	Hierarchical	$O(n^2)$	$O(kn^2)$	Not incremental
Complete Link	Hierarchical	$O(n^2)$	$O(kn^2)$	Not incremental
MST	Hierarchical/ Partitional	$O(n^2)$	$O(n^2)$	Not incremental
Squared Error	Partitional	$O(n)$	$O(tkn)$	Iterative
K-Means	Partitional	$O(n)$	$O(tkn)$	Iterative, No categorical
Nearest Neighbor	Partitional	$O(n^2)$	$O(n^2)$	Incremental
PAM	Partitional	$O(tn^3)$ or $O(tkn^2)$	$O(n^2)$	Iterative
BIRCH	Partitional	$O(n)$	$O(n)$	CF-Tree; Incremental; Outliers
CURE	Mixed	$O(n^2 lgn)$	$O(n)$	Heap; k-D tree; Incremental; Outliers
ROCK	Agglomerative	$O(n^2 lgn)$	$O(n^2)$	Sampling; Categorical; Links
DBSCAN	Mixed	$O(n^2)$	$O(n^2)$	Sampling; Outliers

Outline

- Top 10 Algorithms in Data Mining Research
  - Introduction
  - Classification
  - Statistical Learning
  - Bagging and Boosting
  - Clustering
  - Association Analysis
  - Link Mining – Text Mining
  - Top 10 Algorithms
- 10 Challenging Problems in Data Mining Research
- Concluding Remarks

ICDM'06 Panel on "Top 10 Algorithms in Data Mining"

IEEE COMPUTER SOCIETY

The 18 Identified Candidates

Association Analysis

- #7. Apriori: Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In VLDB '94.
- #8. FP-Tree: Han, J., Pei, J., and Yin, Y. 2000. Mining frequent patterns without candidate generation. In SIGMOD '00.

ICDM'06 Panel on "Top 10 Algorithms in Data Mining"

IEEE COMPUTER SOCIETY

The 18 Identified Candidates

Sequential Patterns

- #14. GSP: Srikant, R. and Agrawal, R. 1996. Mining Sequential Patterns: Generalizations and Performance Improvements. In Proceedings of the 5th International Conference on Extending Database Technology, 1996.
- #15. PrefixSpan: J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal and M-C. Hsu. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In ICDE '01.

Graph Mining

- #18. gSpan: Yan, X. and Han, J. 2002. gSpan: Graph-Based Substructure Pattern Mining. In ICDM '02.

Association Rule Problem

- Given a set of items $I=\{I_1,I_2,\ldots,I_m\}$ and a database of transactions $D=\{t_1,t_2, \ldots, t_n\}$ where $t_i=\{I_{i1},I_{i2}, \ldots, I_{ik}\}$ and $I_{ij} \in I$, the ***Association Rule Problem*** is to identify all association rules $X \Rightarrow Y$ with a minimum support and confidence.
- Link Analysis
- ***NOTE:*** Support of $X \Rightarrow Y$ is same as support of $X \cup Y$.

Association Rule Definitions

- ***Set of items:*** $I=\{I_1,I_2,\ldots,I_m\}$
- ***Transactions:*** $D=\{t_1,t_2,\ \ldots,\ t_n\},\ t_j \subseteq I$
- ***Itemset:*** $\{I_{i1},I_{i2},\ \ldots,\ I_{ik}\} \subseteq I$
- ***Support of an itemset:*** Percentage of transactions which contain that itemset.
- ***Large (Frequent) itemset:*** Itemset whose number of occurrences is above a threshold.

Association Rule Definitions

- ***Association Rule (AR):*** implication $X \Rightarrow Y$ where $X,Y \subseteq I$ and $X \cap Y$ = ;
- ***Support of AR (s) $X \Rightarrow Y$***: Percentage of transactions that contain $X \cup Y$
- ***Confidence of AR (α) $X \Rightarrow Y$:*** Ratio of number of transactions that contain $X \cup Y$ to the number that contain X

Example: Market Basket Data

- Items frequently purchased together:

  $\text{Bread} \Rightarrow \text{PeanutButter}$

- Uses:
  - Placement
  - Advertising
  - Sales
  - Coupons
- Objective: increase sales and reduce costs

Association Rules Example

Transaction	Items
t_1	Bread,Jelly,PeanutButter
t_2	Bread,PeanutButter
t_3	Bread,Milk,PeanutButter
t_4	Beer,Bread
t_5	Beer,Milk

I = { Beer, Bread, Jelly, Milk, PeanutButter}

Support of {Bread,PeanutButter} is 60%

Association Rules Ex (cont'd)

$X \Rightarrow Y$	s	α
Bread $\Rightarrow$ PeanutButter	60%	75%
PeanutButter $\Rightarrow$ Bread	60%	100%
Beer $\Rightarrow$ Bread	20%	50%
PeanutButter $\Rightarrow$ Jelly	20%	33.3%
Jelly $\Rightarrow$ PeanutButter	20%	100%
Jelly $\Rightarrow$ Milk	0%	0%

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

  $\Rightarrow$ Computationally prohibitive!

Mining Association Rules

TID	*Items*
1	**Bread, Milk**
2	**Bread, Diaper, Beer, Eggs**
3	**Milk, Diaper, Beer, Coke**
4	**Bread, Milk, Diaper, Beer**
5	**Bread, Milk, Diaper, Coke**

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

Observations:

- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may decouple the support and confidence requirements

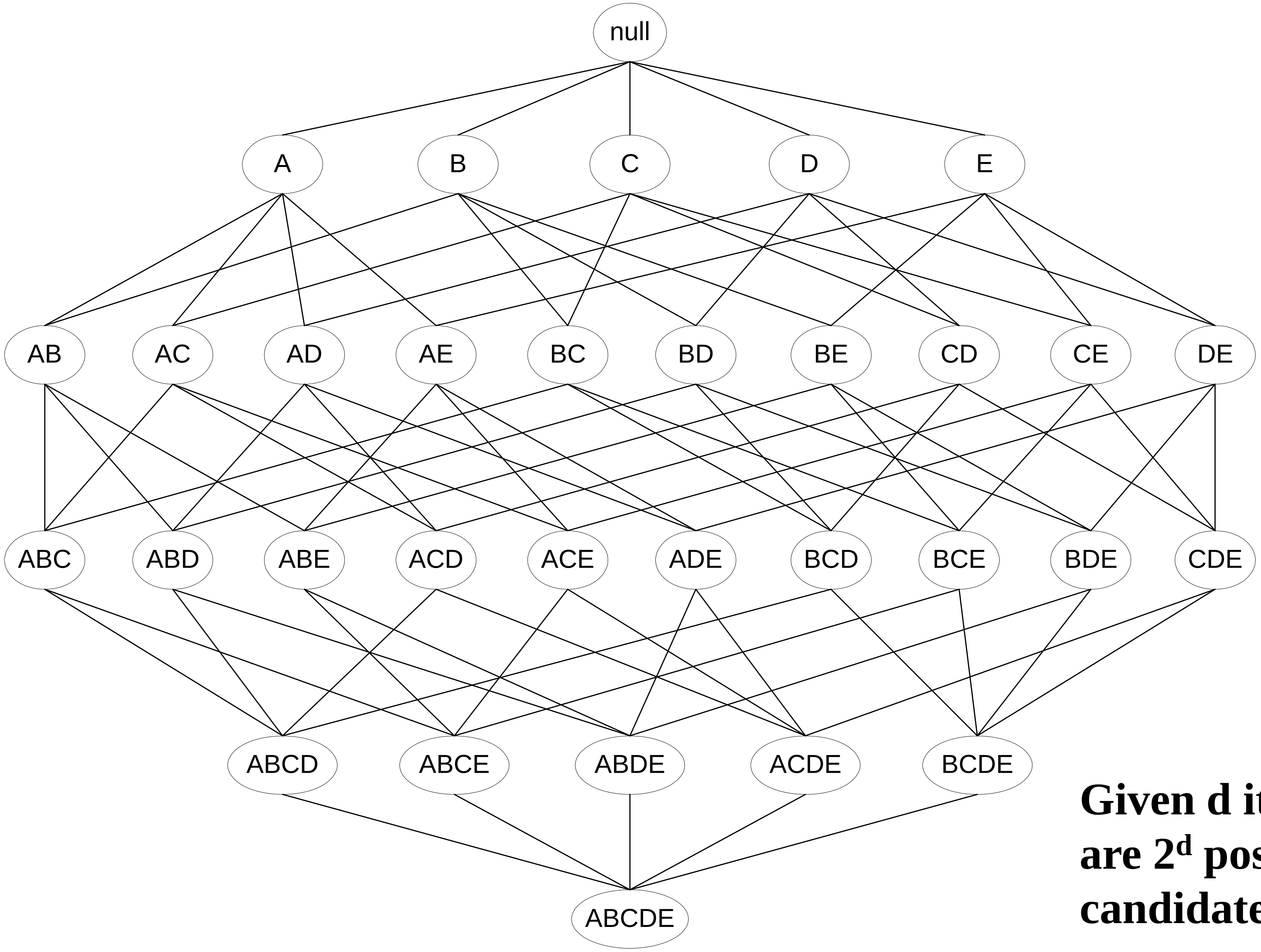
Association Rule Techniques

1. Find Large Itemsets.
2. Generate rules from frequent itemsets.

Mining Association Rules

- Two-step approach:
  1. Frequent Itemset Generation
     - Generate all itemsets whose support $\geq$ minsup
  2. Rule Generation
     - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
- Frequent itemset generation is still computationally expensive

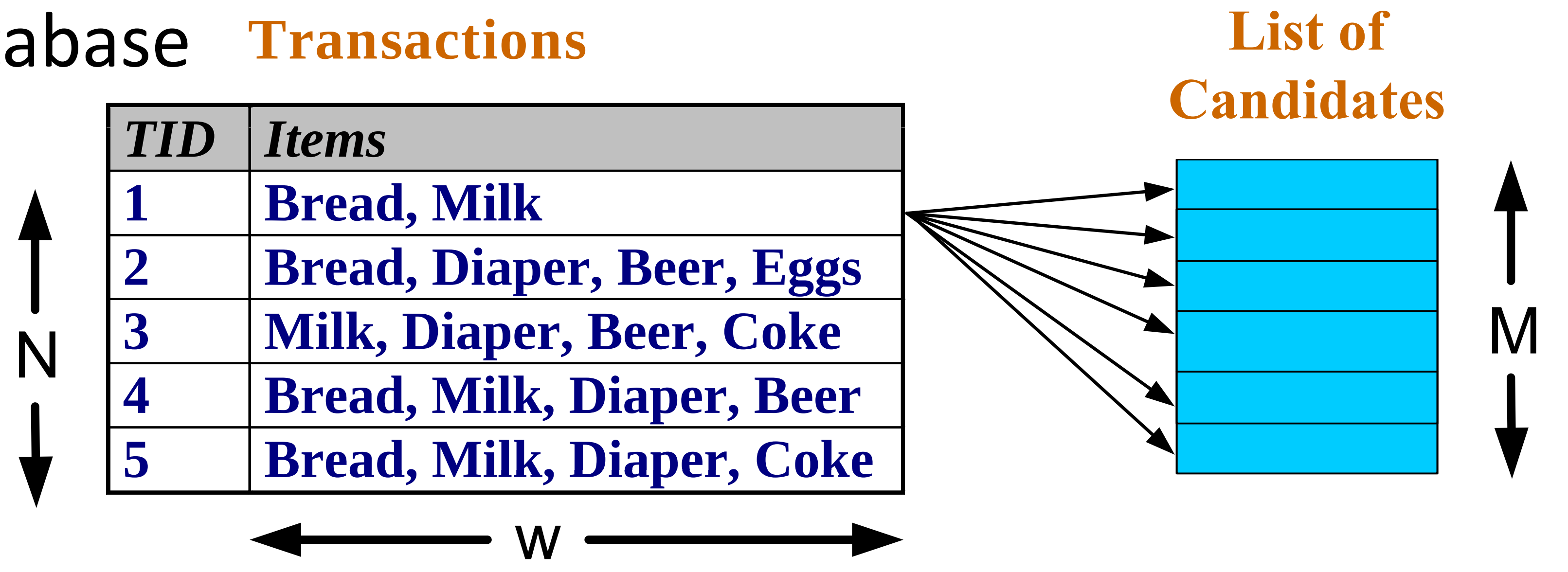
Frequent Itemset Generation



Given d items, there are 2^d possible candidate itemsets

Frequent Itemset Generation

- Brute-force approach:
  - Each itemset in the lattice is a candidate frequent itemset
  - Count the support of each candidate by scanning the database

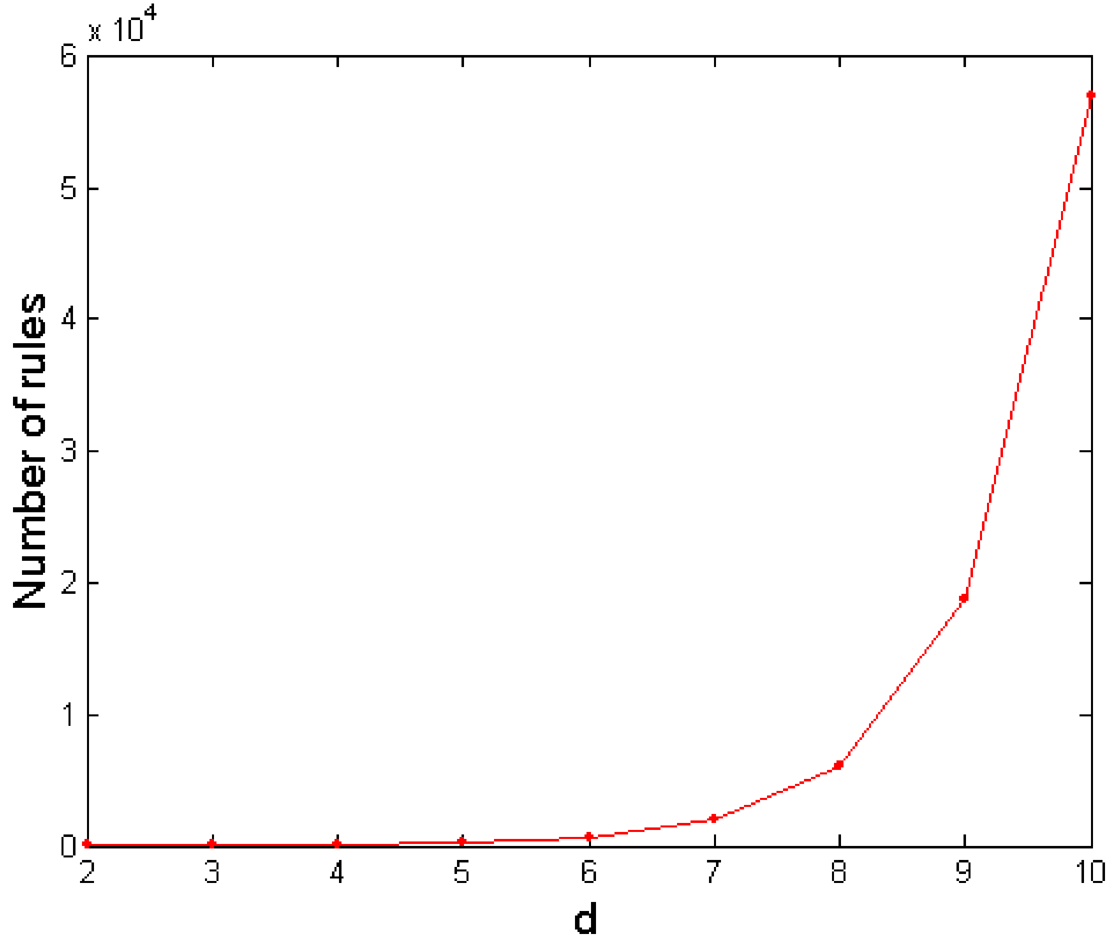


TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

  - Match each transaction against every candidate
  - Complexity ~ O(NMw) => Expensive since $M = 2^d$!!!

Computational Complexity

- Given d unique items:
  - Total number of itemsets = 2^d
  - Total number of possible association rules:



$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right]$$

$$= 3^d - 2^{d+1} + 1$$

If d=6, R = 602 rules

Frequent Itemset Generation Strategies

- Reduce the number of candidates (M)
  - Complete search: $M=2^d$
  - Use pruning techniques to reduce M
- Reduce the number of transactions (N)
  - Reduce size of N as the size of itemset increases
  - Used by DHP and vertical-based mining algorithms
- Reduce the number of comparisons (NM)
  - Use efficient data structures to store the candidates or transactions
  - No need to match every candidate against every transaction

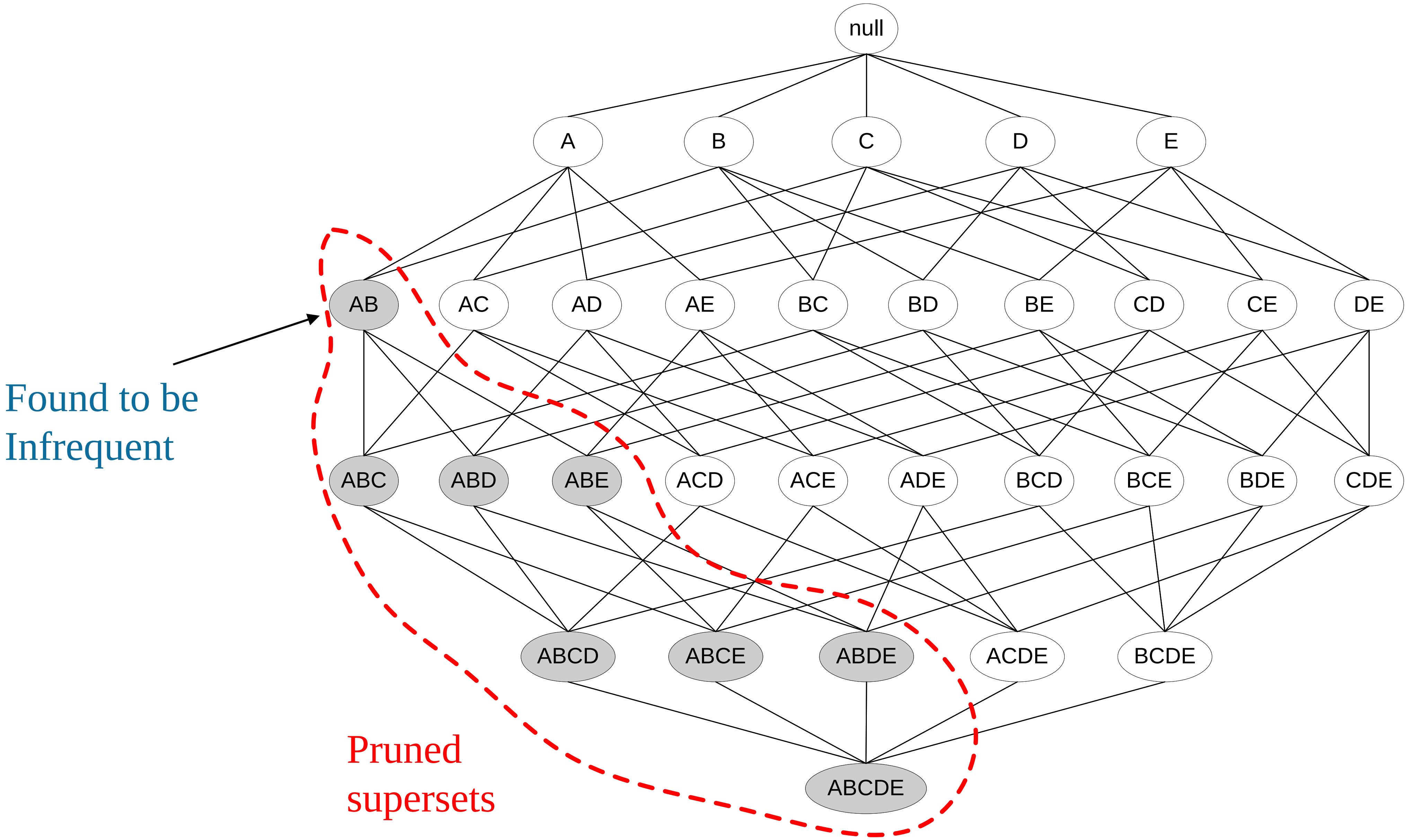
Reducing Number of Candidates

- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent
- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

  - Support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of support

Illustrating Apriori Principle



Apriori

- ***Large Itemset Property:***

Any subset of a large itemset is large.

- Contrapositive:

If an itemset is not large,

none of its supersets are large.

Apriori Ex (cont'd)

Pass	Candidates	Large Itemsets
1	{Beer},{Bread},{Jelly}, {Milk},{PeanutButter}	{Beer},{Bread}, {Milk},{PeanutButter}
2	{Beer,Bread},{Beer,Milk}, {Beer,PeanutButter},{Bread,Milk}, {Bread,PeanutButter},{Milk,PeanutButter}	{Bread,PeanutButter}

s=30% α = 50%

Apriori Algorithm

- Tables:
  Lk = Set of k-itemsets which are frequent
  Ck = Set of k-itemsets which could be frequent
- Method:

```
Init. Let k=1
    Generate L1 (frequent itemsets of length 1)
Repeat until no new frequent itemsets are identified
    a)Generate C(k+1) candidate itemsets from Lk frequent
      itemsets
    b)Count the support of each candidate by scanning the
      DB
    c)Eliminate candidates that are infrequent, leaving only
      those that are frequent
```

Illustrating Apriori Principle

L1

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Minimum Support = 3

a) No need to generate candidates involving Coke (or Eggs)

C2

Itemset
{Bread,Milk}
{Bread,Beer}
{Bread,Diaper}
{Milk, Beer}
{Milk,Diaper}
{Beer,Diaper}

b) Counting

Itemset	Count
{Bread,Milk}	**3**
{Bread,Beer}	**2**
{Bread,Diaper}	**3**
{Milk,Beer}	**2**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

c) Filter non-frequent

L2

Itemset	Count
{Bread,Milk}	**3**
{Bread,Diaper}	**3**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

a) No need to generate candidates involving {Bread, Beer}

C3

Itemset
{Bread,Milk,Diaper}

b) Counting
c) Filter

L3

Itemset	Count
{Bread,Milk,Diaper}	**3**

Apriori-Gen Example

Transaction	Items
t_1	Blouse
t_2	Shoes,Skirt,TShirt
t_3	Jeans,TShirt
t_4	Jeans,Shoes,TShirt
t_5	Jeans,Shorts
t_6	Shoes,TShirt
t_7	Jeans,Skirt
t_8	Jeans,Shoes,Shorts,TShirt
t_9	Jeans
t_{10}	Jeans,Shoes,TShirt
t_{11}	TShirt
t_{12}	Blouse,Jeans,Shoes,Skirt,TShirt
t_{13}	Jeans,Shoes,Shorts,TShirt
t_{14}	Shoes,Skirt,TShirt
t_{15}	Jeans,TShirt
t_{16}	Skirt,TShirt
t_{17}	Blouse,Jeans,Skirt
t_{18}	Jeans,Shoes,Shorts,TShirt
t_{19}	Jeans
t_{20}	Jeans,Shoes,Shorts,TShirt

Apriori-Gen Example (cont’d)

Scan	Candidates	Large Itemsets
1	{Blouse},{Jeans},{Shoes}, {Shorts},{Skirt},{TShirt}	{Jeans},{Shoes},{Shorts} {Skirt},{Tshirt}
2	{Jeans,Shoes},{Jeans,Shorts},{Jeans,Skirt}, {Jeans,TShirt},{Shoes,Shorts},{Shoes,Skirt}, {Shoes,TShirt},{Shorts,Skirt},{Shorts,TShirt}, {Skirt,TShirt}	{Jeans,Shoes},{Jeans,Shorts}, {Jeans,TShirt},{Shoes,Shorts}, {Shoes,TShirt},{Shorts,TShirt}, {Skirt,TShirt}
3	{Jeans,Shoes,Shorts},{Jeans,Shoes,TShirt}, {Jeans,Shorts,TShirt},{Jeans,Skirt,TShirt}, {Shoes,Shorts,TShirt},{Shoes,Skirt,TShirt}, {Shorts,Skirt,TShirt}	{Jeans,Shoes,Shorts}, {Jeans,Shoes,TShirt}, {Jeans,Shorts,TShirt}, {Shoes,Shorts,TShirt}
4	{Jeans,Shoes,Shorts,TShirt}	{Jeans,Shoes,Shorts,TShirt}
5	$\emptyset$	$\emptyset$

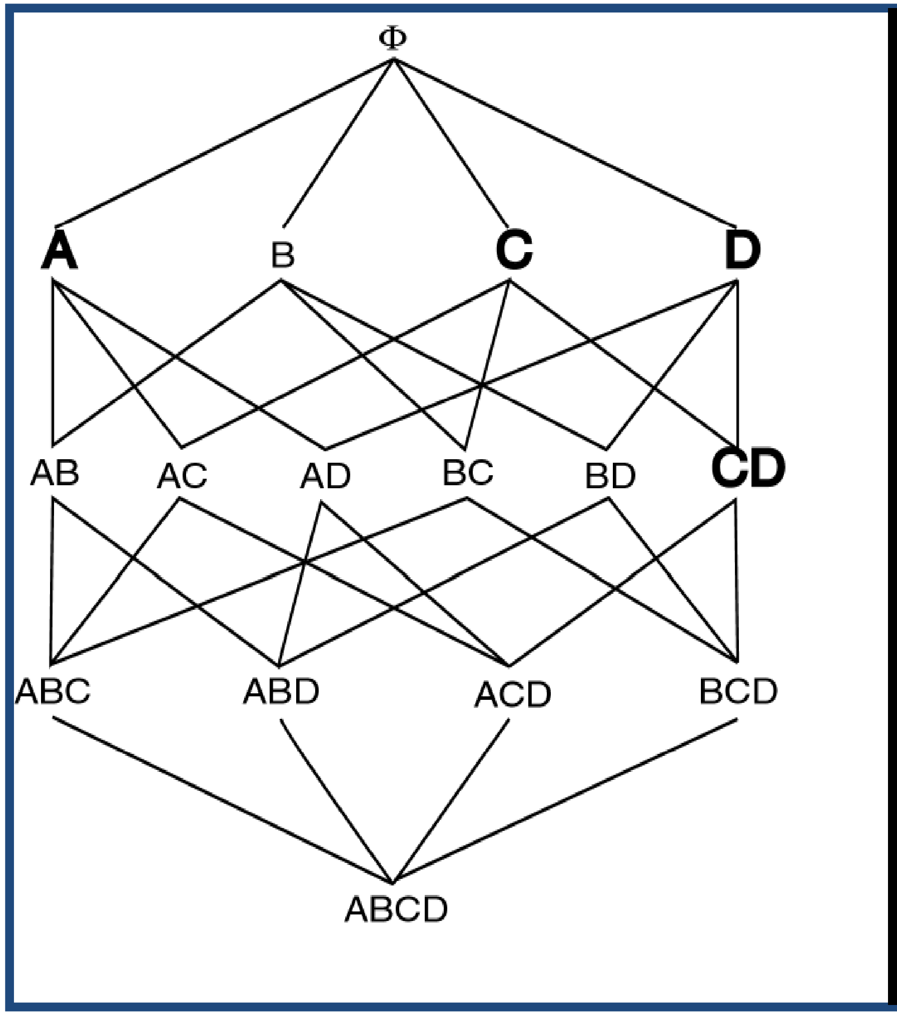
Apriori Adv/Disadv

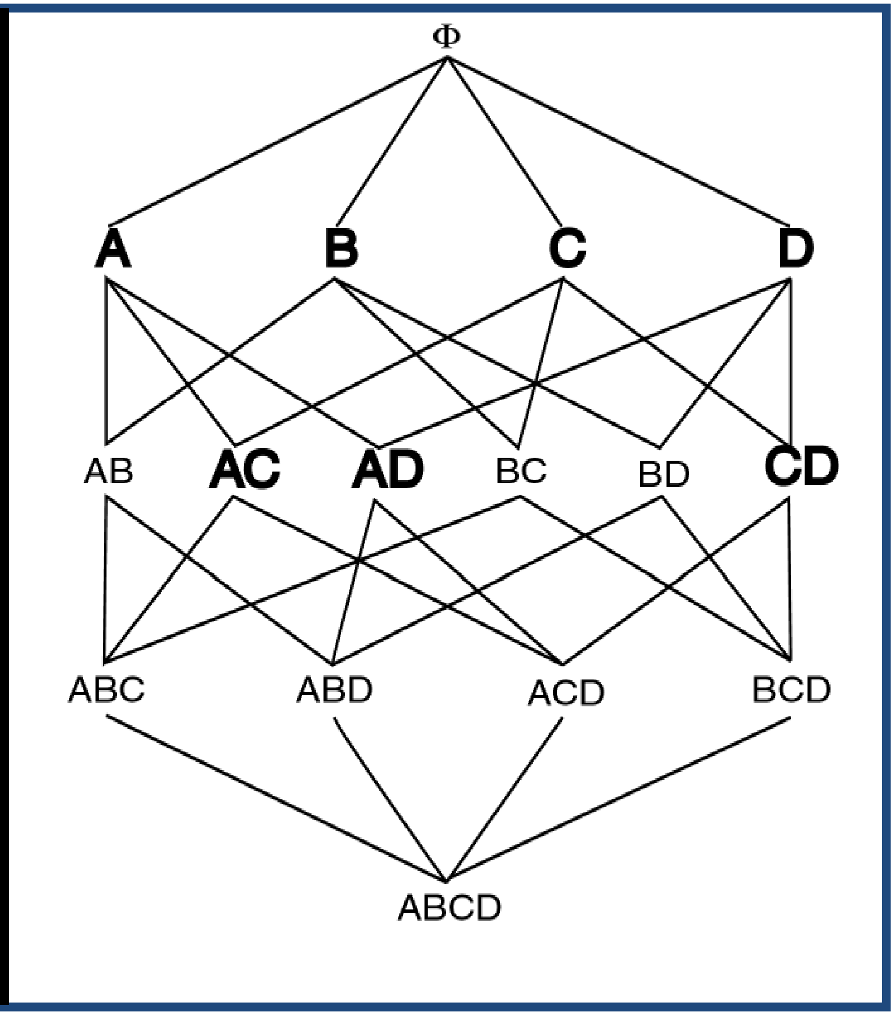
- ***Advantages:***
  - Uses large itemset property.
  - Easily parallelized
  - Easy to implement.
- ***Disadvantages:***
  - Assumes transaction database is memory resident.
  - Requires up to m database scans.

Sampling

- Large databases
- Sample the database and apply Apriori to the sample.
- ***Potentially Large Itemsets (PL):*** Large itemsets from sample
- ***Negative Border (BD^-):***
  - Generalization of Apriori-Gen applied to itemsets of varying sizes.
  - Minimal set of itemsets which are not in PL, but whose subsets are all in PL.

Negative Border Example





PL

PL $\cup BD^{-}(PL)$

Sampling Algorithm

```
1.  $D_s$ = sample of Database D;
2.  PL = Large itemsets in $D_s$ using smalls;
3.  C = PL $\cup$ $BD^-$(PL);
4.  Count C in Database using s;
5.  ML = large itemsets in $BD^-$(PL);
6.  If ML = $\emptyset$ then done
7.      else C = repeated application of $BD^-$;
8.          Count C in Database;
```

Sampling Example

- Find AR assuming s = 20%
- D_s = { t_1,t_2}
- Smalls = 10%
- PL = {{Bread}, {Jelly}, {PeanutButter}, {Bread,Jelly}, {Bread,PeanutButter}, {Jelly, PeanutButter}, {Bread,Jelly,PeanutButter}}
- BD^-(PL)={{Beer},{Milk}}
- ML = {{Beer}, {Milk}}
- Repeated application of BD^- generates all remaining itemsets

Sampling Adv/Disadv

- ***Advantages:***
  - Reduces number of database scans to one in the best case and two in worst.
  - Scales better.
- ***Disadvantages:***
  - Potentially large number of candidates in second pass

Partitioning

- Divide database into partitions $D^1, D^2, \ldots, D^p$
- Apply Apriori to each partition
- Any large itemset must be large in at least one partition.

Partitioning Algorithm

```
1. Divide D into partitions $D^1, D^2, \ldots, D^{p;}$
2. For I = 1 to p do
3.     $L^i$ = Apriori($D^i$);
4. $C = L^1 \cup \ldots \cup L^p$;
5. Count C on D to generate L;
```

Partitioning Example

Transaction	Items
t_1	Bread,Jelly,PeanutButter
t_2	Bread,PeanutButter
t_3	Bread,Milk,PeanutButter
t_4	Beer,Bread
t_5	Beer,Milk

D^1

D^2

S=10%

L^1 ={{Bread}, {Jelly}, {PeanutButter}, {Bread,Jelly}, {Bread,PeanutButter}, {Jelly, PeanutButter}, {Bread,Jelly,PeanutButter}}

L^2 ={{Bread}, {Milk}, {PeanutButter}, {Bread,Milk}, {Bread,PeanutButter}, {Milk, PeanutButter}, {Bread,Milk,PeanutButter}, {Beer}, {Beer,Bread}, {Beer,Milk}}

Partitioning Adv/Disadv

- ***Advantages:***
  - Adapts to available main memory
  - Easily parallelized
  - Maximum number of database scans is two.
- ***Disadvantages:***
  - May have many candidates during second scan.

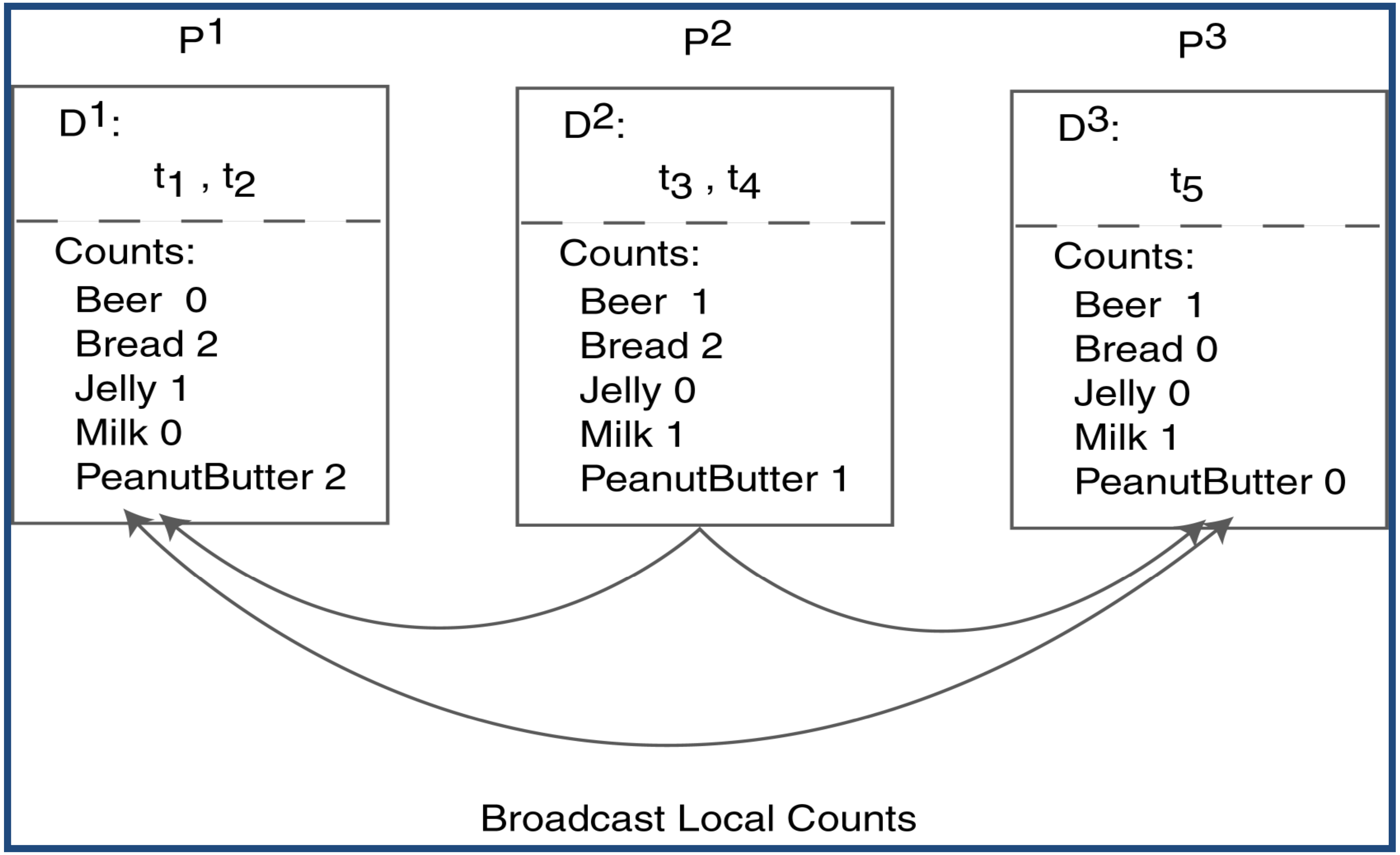
Parallelizing AR Algorithms

- Based on Apriori
- Techniques differ:
  - What is counted at each site
  - How data (transactions) are distributed
- Data Parallelism
  - Data partitioned
  - Count Distribution Algorithm
- Task Parallelism
  - Data and candidates partitioned
  - Data Distribution Algorithm

Count Distribution Algorithm(CDA)

```
1.  Place data partition at each site.
2.  In Parallel at each site do
3.      $C_1$ = Itemsets of size one in I;
4.      Count $C_{1;}$
5.      Broadcast counts to all sites;
6.      Determine global large itemsets of size 1, $L_1$;
7.      i = 1;
8.      Repeat
9.          i = i + 1;
10.         $C_i$ = Apriori-Gen($L_{i-1}$);
11.         Count $C_{i;}$
12.         Broadcast counts to all sites;
13.         Determine global large itemsets of size i, $L_i$;
14.     until no more large itemsets found;
```

CDA Example

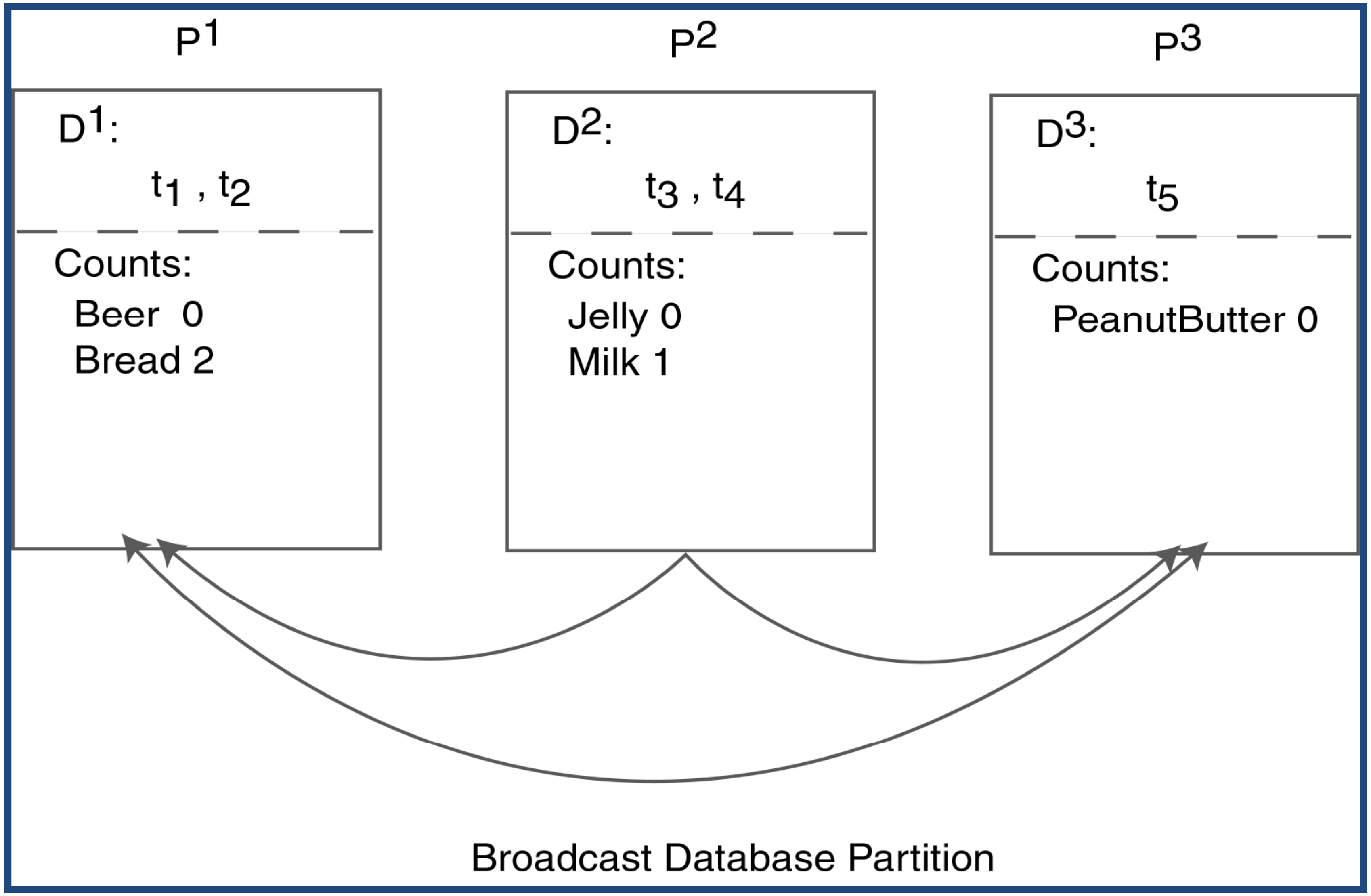


P^1
P^2
P^3
D^1:
t_1 , t_2
Counts:
Beer 0
Bread 2
Jelly 1
Milk 0
PeanutButter 2
D^2:
t_3 , t_4
Counts:
Beer 1
Bread 2
Jelly 0
Milk 1
PeanutButter 1
D^3:
t_5
Counts:
Beer 1
Bread 0
Jelly 0
Milk 1
PeanutButter 0
Broadcast Local Counts

Data Distribution Algorithm(DDA)

```
1.  Place data partition at each site.
2.  In Parallel at each site do
3.      Determine local candidates of size 1 to count;
4.      Broadcast local transactions to other sites;
5.      Count local candidates of size 1 on all data;
6.      Determine large itemsets of size 1 for local
            candidates;
7.      Broadcast large itemsets to all sites;
8.      Determine $L_1$;
9.      i = 1;
10.     Repeat
11.         i = i + 1;
12.         $C_i$ = Apriori-Gen($L_{i-1}$);
13.         Determine local candidates of size i to count;
14.         Count, broadcast, and find $L_i$;
15.     until no more large itemsets found;
```

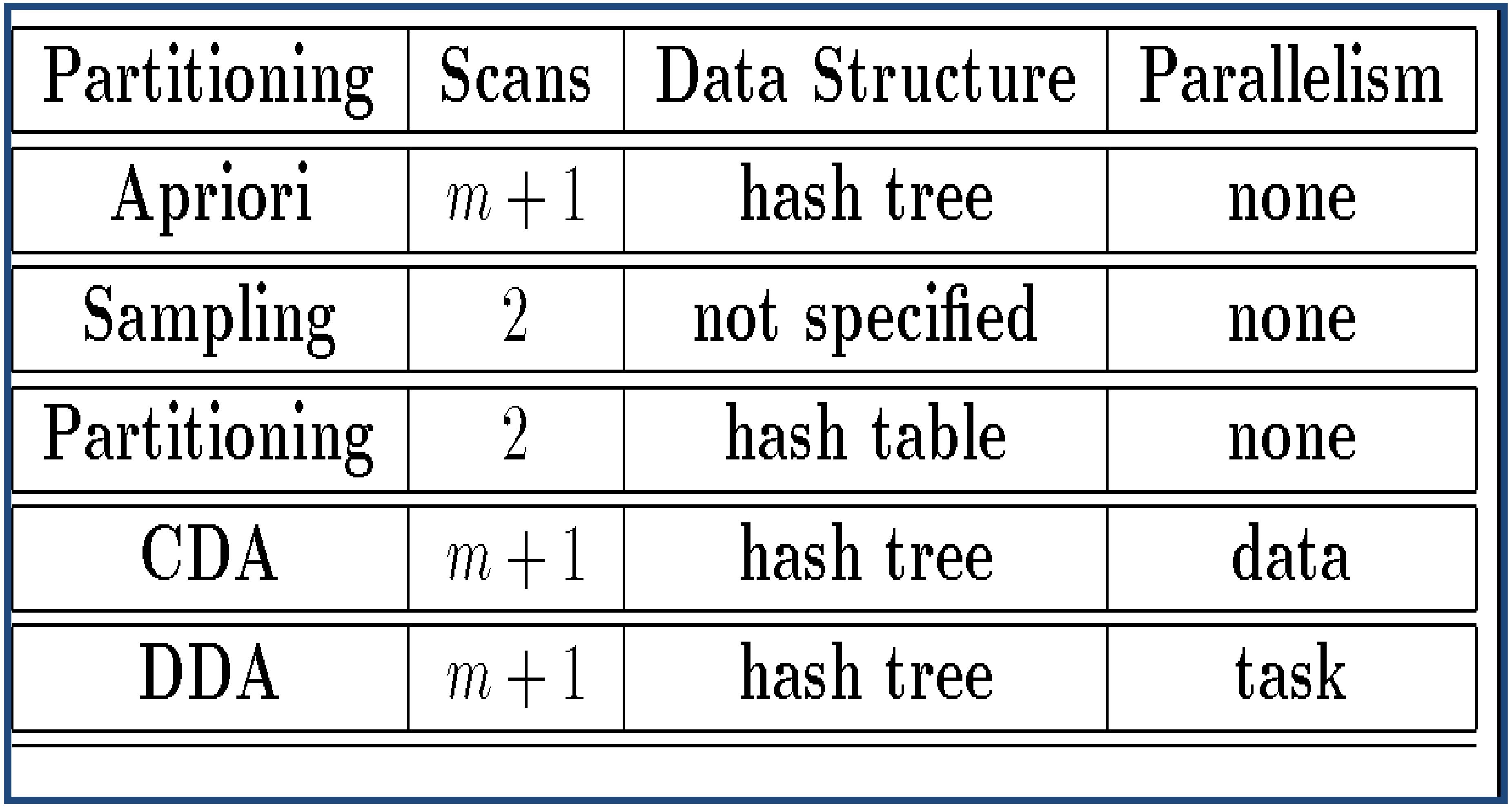
DDA Example


P^1
P^2
P^3
D^1:
t_1 , t_2
Counts:
Beer 0
Bread 2
D^2:
t_3 , t_4
Counts:
Jelly 0
Milk 1
D^3:
t_5
Counts:
PeanutButter 0
Broadcast Database Partition

Comparing AR Techniques

- Target
- Type
- Data Type
- Data Source
- Technique
- Itemset Strategy and Data Structure
- Transaction Strategy and Data Structure
- Optimization
- Architecture
- Parallelism Strategy

Comparison of AR Techniques



Partitioning	Scans	Data Structure	Parallelism
Apriori	$m+1$	hash tree	none
Sampling	2	not specified	none
Partitioning	2	hash table	none
CDA	$m+1$	hash tree	data
DDA	$m+1$	hash tree	task

Incremental Association Rules

- Generate ARs in a dynamic database.
- Problem: algorithms assume static database
- Objective:
  - Know large itemsets for D
  - Find large itemsets for D $\cup$ {Δ D}
- Must be large in either D or Δ D
- Save L_i and counts

Note on ARs

- Many applications outside market basket data analysis
  - Prediction (telecom switch failure)
  - Web usage mining
- Many different types of association rules
  - Temporal
  - Spatial
  - Causal

Association rules: Evaluation

- Association rule algorithms tend to produce too many rules
  - many of them are uninteresting or redundant
  - Redundant if {A,B,C} $\rightarrow$ {D} and {A,B} $\rightarrow$ {D} have same support & confidence
- In the original formulation of association rules, support & confidence are the only measures used
- Interestingness measures can be used to prune/rank the derived patterns

Measuring Quality of Rules

- Support
- Confidence
- Interest
- Conviction
- Chi Squared Test

Advanced AR Techniques

- Generalized Association Rules
- Multiple minimum supports
- Multiple-Level Association Rules
- Quantitative Association Rules
- Using multiple minimum supports
- Correlation Rules
- Sequential patterns mining
- Graph mining
- Mining association rules in stream data
- Fuzzy association rules
- Anomalous association rules

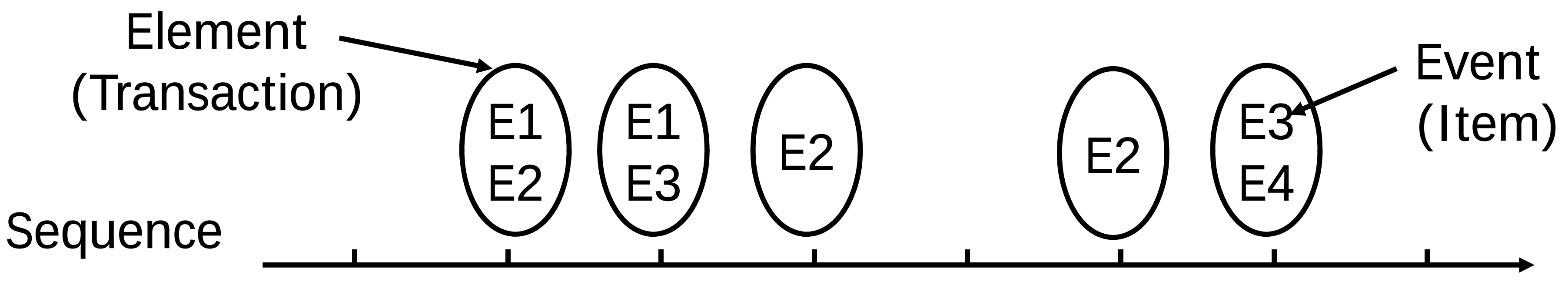
Extensions: Handling Continuous Attributes

- Different kinds of rules:
  - $Age \in [21,35) \wedge Salary \in [70k,120k) \rightarrow Buy$
  - $Salary \in [70k,120k) \wedge Buy \rightarrow Age: \mu=28, \sigma=4$

- Different methods:
  - Discretization-based
  - Statistics-based
  - Non-discretization based
    - minApriori

Extensions: Sequential pattern mining

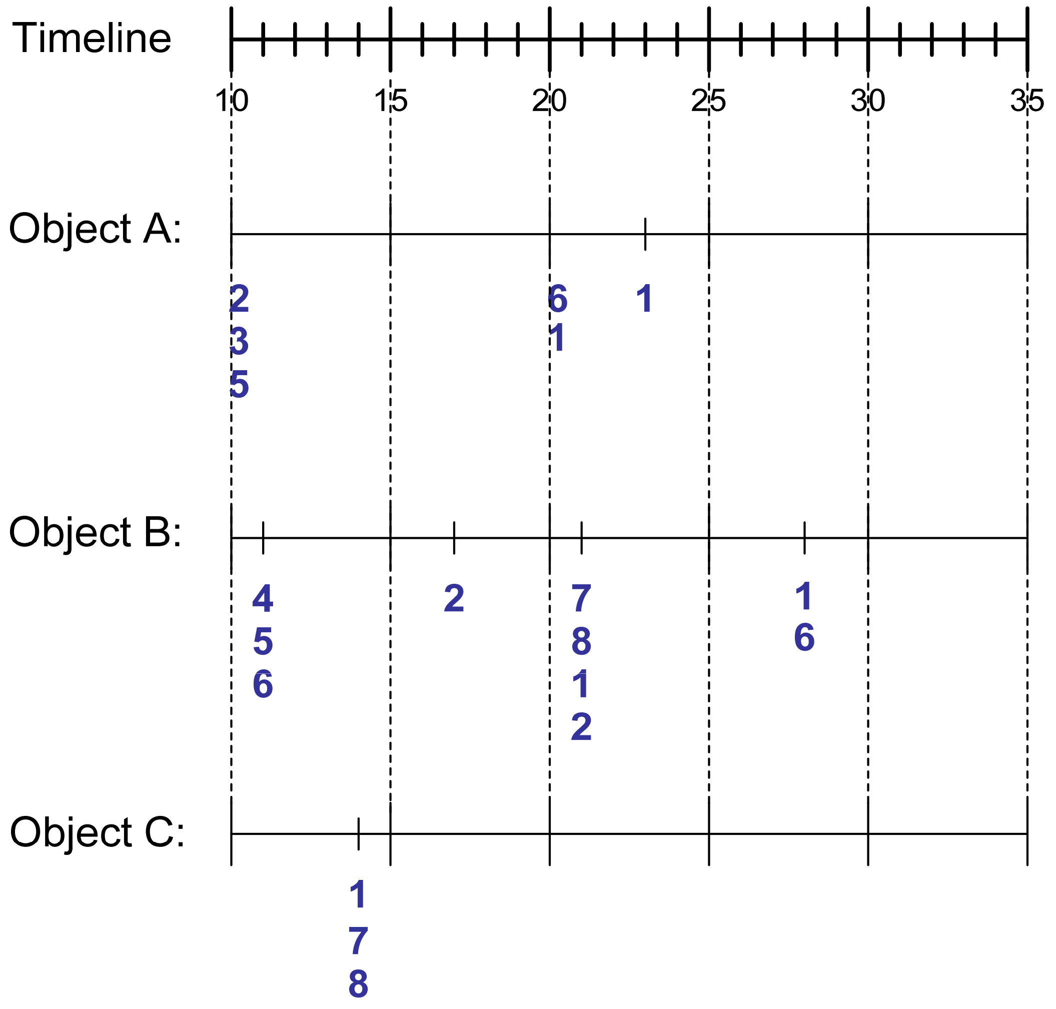
Sequence Database	**Sequence**	**Element (Transaction)**	**Event (Item)**
Customer	Purchase history of a given customer	A set of items bought by a customer at time t	Books, diary products, CDs, etc
Web Data	Browsing activity of a particular Web visitor	A collection of files viewed by a Web visitor after a single mouse click	Home page, index page, contact info, etc
Event data	History of events generated by a given sensor	Events triggered by a sensor at time t	Types of alarms generated by sensors
Genome sequences	DNA sequence of a particular species	An element of the DNA sequence	Bases A,T,G,C



Extensions: Sequential pattern mining

Sequence Database:

Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 7, 8



Extensions: Sequential pattern mining

- A sequence $<a_1\ a_2 \dots a_n>$ is contained in another sequence $<b_1\ b_2 \dots b_m>$ $(m \geq n)$ if there exist integers $i_1 < i_2 < \dots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i1}$, …, $a_n \subseteq b_{in}$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {8} >	< {2} {3,5} >	Yes
< {1,2} {3,4} >	< {1} {2} >	No
< {2,4} {2,4} {2,5} >	< {2} {4} >	Yes

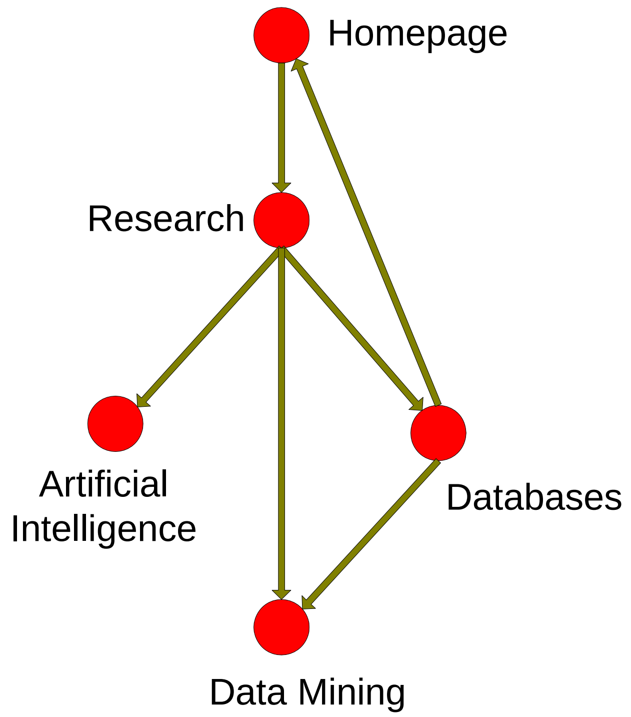
- The support of a subsequence w is defined as the fraction of data sequences that contain w
- A *sequential pattern* is a frequent subsequence (i.e., a subsequence whose support is ≥ *minsup*)

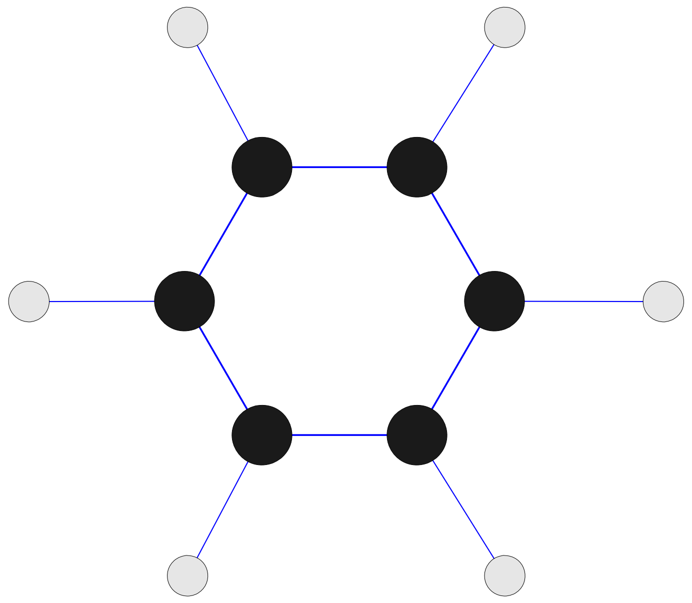
Outline

- Top 10 Algorithms in Data Mining Research
  - Introduction
  - Classification
  - Statistical Learning
  - Bagging and Boosting
  - Clustering
  - Association Analysis
  - Link Mining – Text Mining
  - Top 10 Algorithms
- 10 Challenging Problems in Data Mining Research
- Concluding Remarks

Extensions: Graph mining

- Extend association rule mining to finding frequent subgraphs
- Useful for Web Mining, computational chemistry, bioinformatics, spatial data sets, etc





Outline

- Top 10 Algorithms in Data Mining Research
  - Introduction
  - Classification
  - Statistical Learning
  - Bagging and Boosting
  - Clustering
  - Association Analysis
  - Link Mining – Text Mining
  - Top 10 Algorithms
- 10 Challenging Problems in Data Mining Research
- Concluding Remarks

ICDM'06 Panel on "Top 10 Algorithms in Data Mining"

IEEE COMPUTER SOCIETY

The 18 Identified Candidates

Link Mining

- #9. PageRank: Brin, S. and Page, L. 1998. The anatomy of a large-scale hypertextual Web search engine. In WWW-7, 1998.
- #10. HITS: Kleinberg, J. M. 1998. Authoritative sources in a hyperlinked environment. In Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, 1998.

Link Mining

- Traditional machine learning and data mining approaches assume:
  - A random sample of homogeneous objects from single relation
- Real world data sets:
  - Multi-relational, heterogeneous and semi-structured
- Link Mining
  - newly emerging research area at the intersection of research in social network and link analysis, hypertext and web mining, relational learning and inductive logic programming and graph mining.

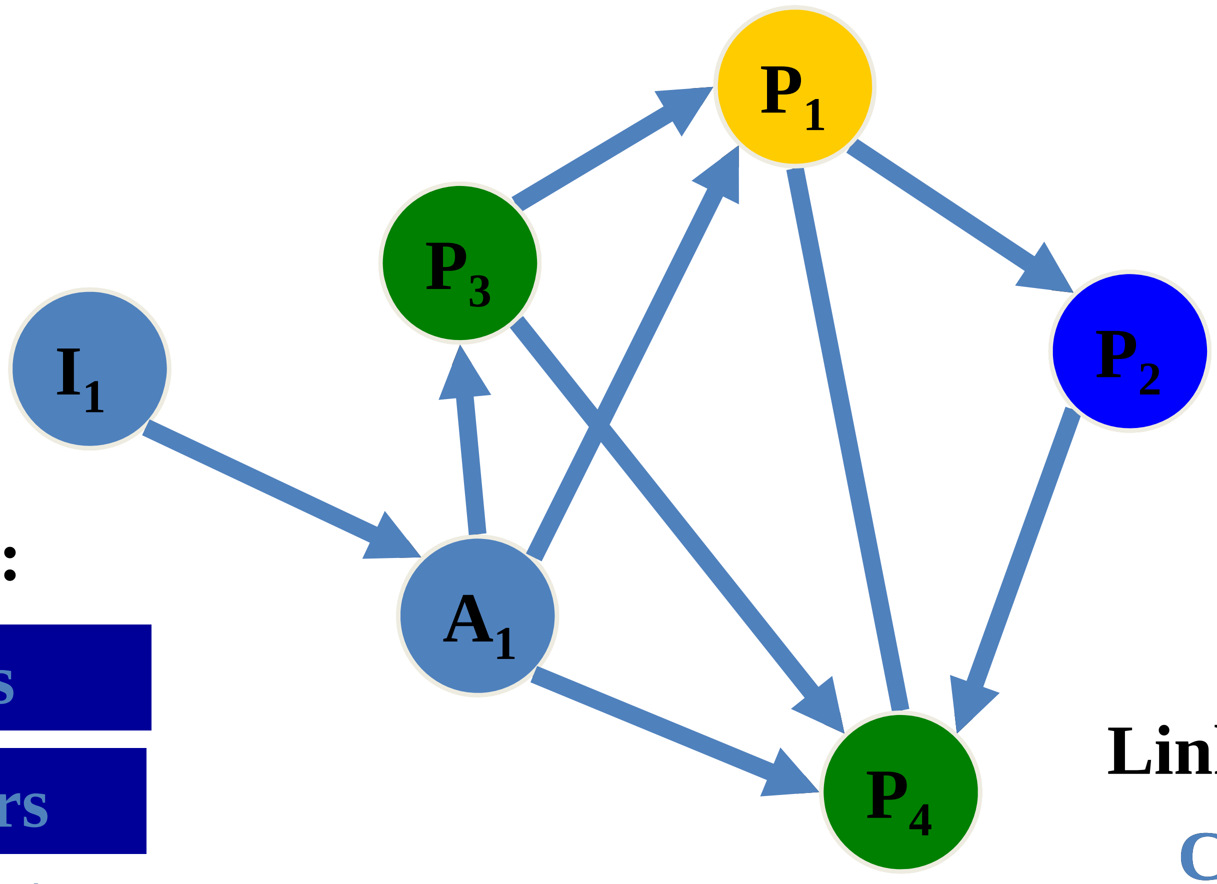
Linked Data

- Heterogeneous, multi-relational data represented as a graph or network
  - Nodes are objects
    - May have different kinds of objects
    - Objects have attributes
    - Objects may have labels or classes
  - Edges are links
    - May have different kinds of links
    - Links may have attributes
    - Links may be directed, are not required to be binary

Sample Domains

- web data (**web**)
- bibliographic data (**cite**)
- epidimiological data (**epi**)

Example: Linked Bibliographic Data



Objects:

Papers

Authors

Institutions

Attributes:

Categories 🔴 🟢 🟡

Links:

Citation

Co-Citation

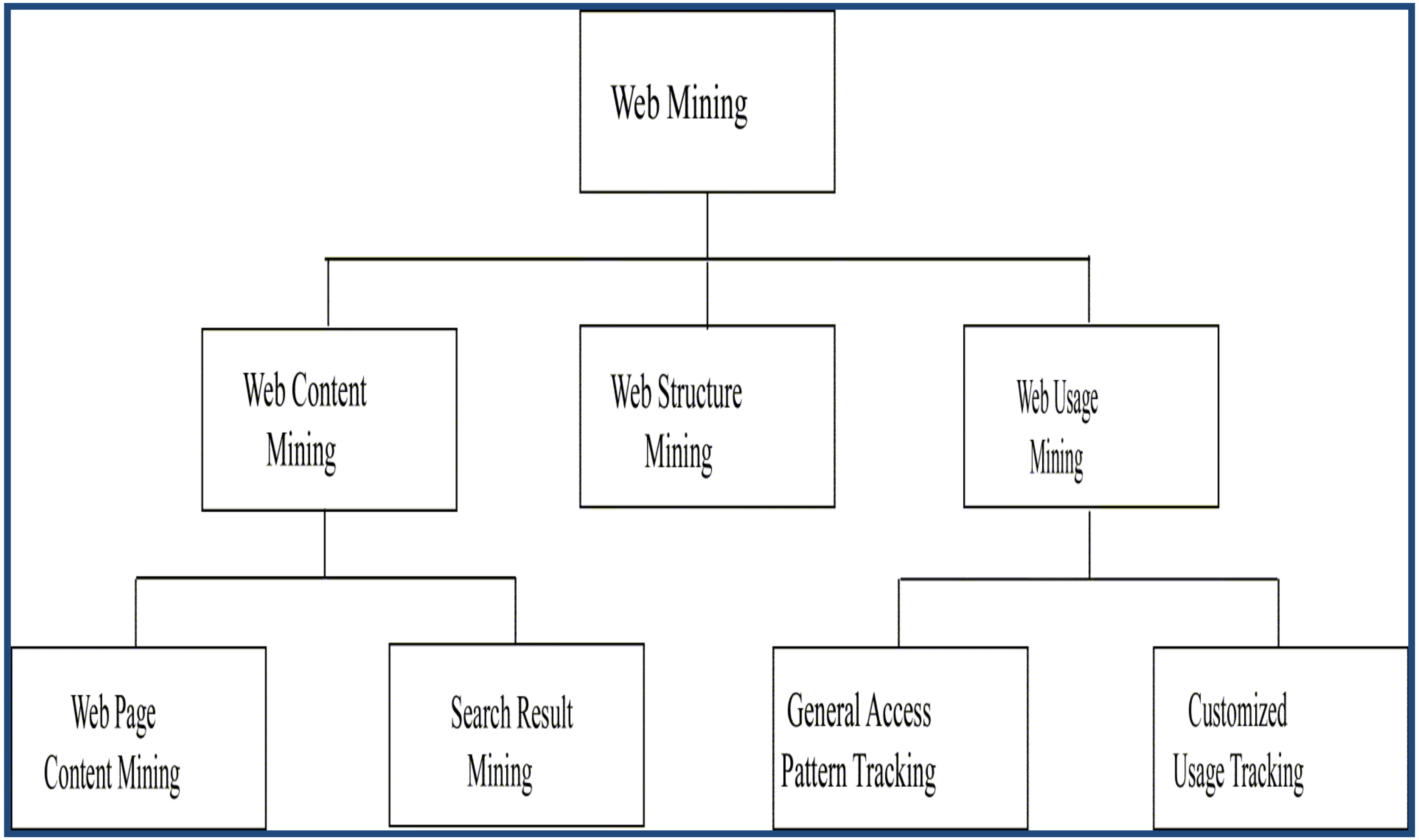
Author-of

Author-affiliation

Link Mining Tasks

- Link-based Object Classification
- Link Type Prediction
- Predicting Link Existence
- Link Cardinality Estimation
- Object Identification
- Subgraph Discovery

Web Mining Outline



Web Data

- Web pages
- Intra-page structures
- Inter-page structures
- Usage data
- Supplemental data
  - Profiles
  - Registration information
  - Cookies

Web Content Mining

- Extends work of basic search engines
- Search Engines
  - IR application
  - Keyword based
  - Similarity between query and document
  - Crawlers
  - Indexing
  - Profiles
  - Link analysis

Web Structure Mining

- Mine structure (links, graph) of the Web
- Techniques
  - PageRank
  - CLEVER
- Create a model of the Web organization.
- May be combined with content mining to more effectively retrieve important pages.

PageRank (Larry **Page** and Sergey **Brin)**

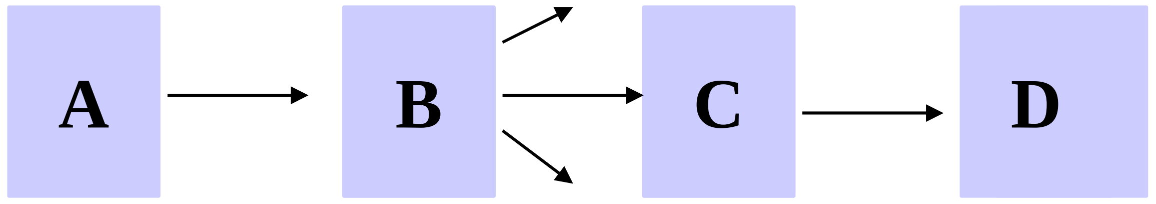
- Used by Google
- Prioritize pages returned from search by looking at Web structure.
- Importance of page is calculated based on number of pages which point to it – ***Backlinks***.
- Weighting is used to provide more importance to backlinks coming form important pages.

PageRank (cont'd)

- $PR(p) = c\ (PR(1)/N_1 + \dots + PR(n)/N_n)$
  - PR(i): PageRank for a page i which points to target page p.
  - N_i: number of links coming out of page i

PageRank (cont'd)

General Principle



- Every page has some number of Outbounds links (forward links) and Inbounds links (backlinks).

- A page X has a high rank if:
  - It has many Inbounds links
  - It has highly ranked Inbounds links
  - Page linking to has few Outbounds links.

HITS

- Hyperlink-Induces Topic Search
- Based on a set of keywords, find set of relevant pages – R.
- Identify hub and authority pages for these.
  - Expand R to a base set, B, of pages linked to or from R.
  - Calculate weights for authorities and hubs.
- Pages with highest ranks in R are returned.
    - ***Authoritative Pages*** :
      - Highly important pages.
      - Best source for requested information.
    - ***Hub Pages*** :
      - Contain links to highly important pages.

HITS Algorithm

```
Input:
    $W$        //WWW viewed as a directed graph.
    $q$        //Query.
    $s$        // Support.
Output:
    $A$        // Set of authority pages.
    $H$        // Set of hub pages.
HITS Algorithm
    $R = SE(W, q)$;
    $B = R \cup \{pages\ linked\ to\ from\ R\} \cup \{pages\ which\ link\ to\ pages\ in\ R\}$;
    $G(B, L) = Subgraph\ of\ W\ induced\ by\ B$;
    $G(B, L^1) = Delete\ links\ in\ G\ within\ same\ site$;
    $x_p = \sum_{q\ where\ <q,p> \in L^1} y_q$;        // Find authority weights.
    $y_p = \sum_{q\ where\ <p,q> \in L^1} x_q$;        // Find hub weights.
    $A = \{p \mid p\ has\ one\ of\ the\ highest\ x_p\}$;
    $H = \{p \mid p\ has\ one\ of\ the\ highest\ y_p\}$;
```

Web Usage Mining

- Extends work of basic search engines
- Search Engines
  - IR application
  - Keyword based
  - Similarity between query and document
  - Crawlers
  - Indexing
  - Profiles
  - Link analysis

Web Usage Mining Applications

- Personalization
- Improve structure of a site’s Web pages
- Aid in caching and prediction of future page references
- Improve design of individual pages
- Improve effectiveness of e-commerce (sales and advertising)

Web Usage Mining Activities

- Preprocessing Web log
  - Cleanse
  - Remove extraneous information
  - Sessionize

    Session: Sequence of pages referenced by one user at a sitting.
- Pattern Discovery
  - Count patterns that occur in sessions
  - ***Pattern*** is sequence of pages references in session.
  - Similar to association rules
    - Transaction: session
    - Itemset: pattern (or subset)
    - Order is important
- Pattern Analysis

ICDM'06 Panel on "Top 10 Algorithms in Data Mining"

IEEE COMPUTER SOCIETY

The 18 Identified Candidates

Integrated Mining

- #16. CBA: Liu, B., Hsu, W. and Ma, Y. M. Integrating classification and association rule mining. KDD-98.

Rougth Sets

- #17. Finding reduct: Zdzislaw Pawlak, Rough Sets: Theoretical Aspects of Reasoning about Data, Kluwer Academic Publishers, Norwell, MA, 1992

Integrated Mining

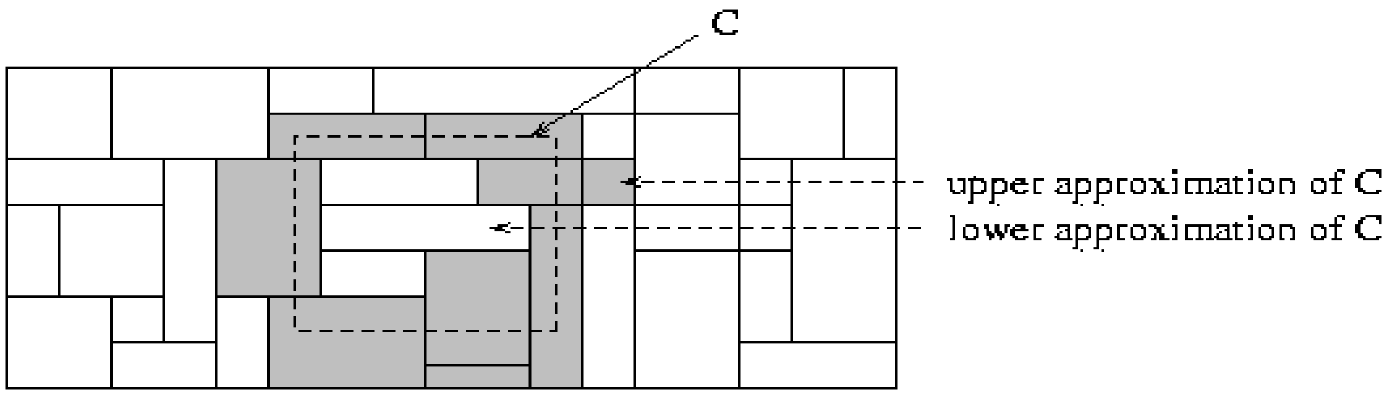
- On the use of association rule algorithms for classification: CBA

- Subgroup Discovery: Caracterization of classes

  "Given a population of individuals and a property of those individuals we are interested in, find population subgroups that are statistically 'most interesting', e.g., are as large as possible and have the most unusual statistical charasteristics with respect to the property of interest".

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 is used to reduce the computation intensity

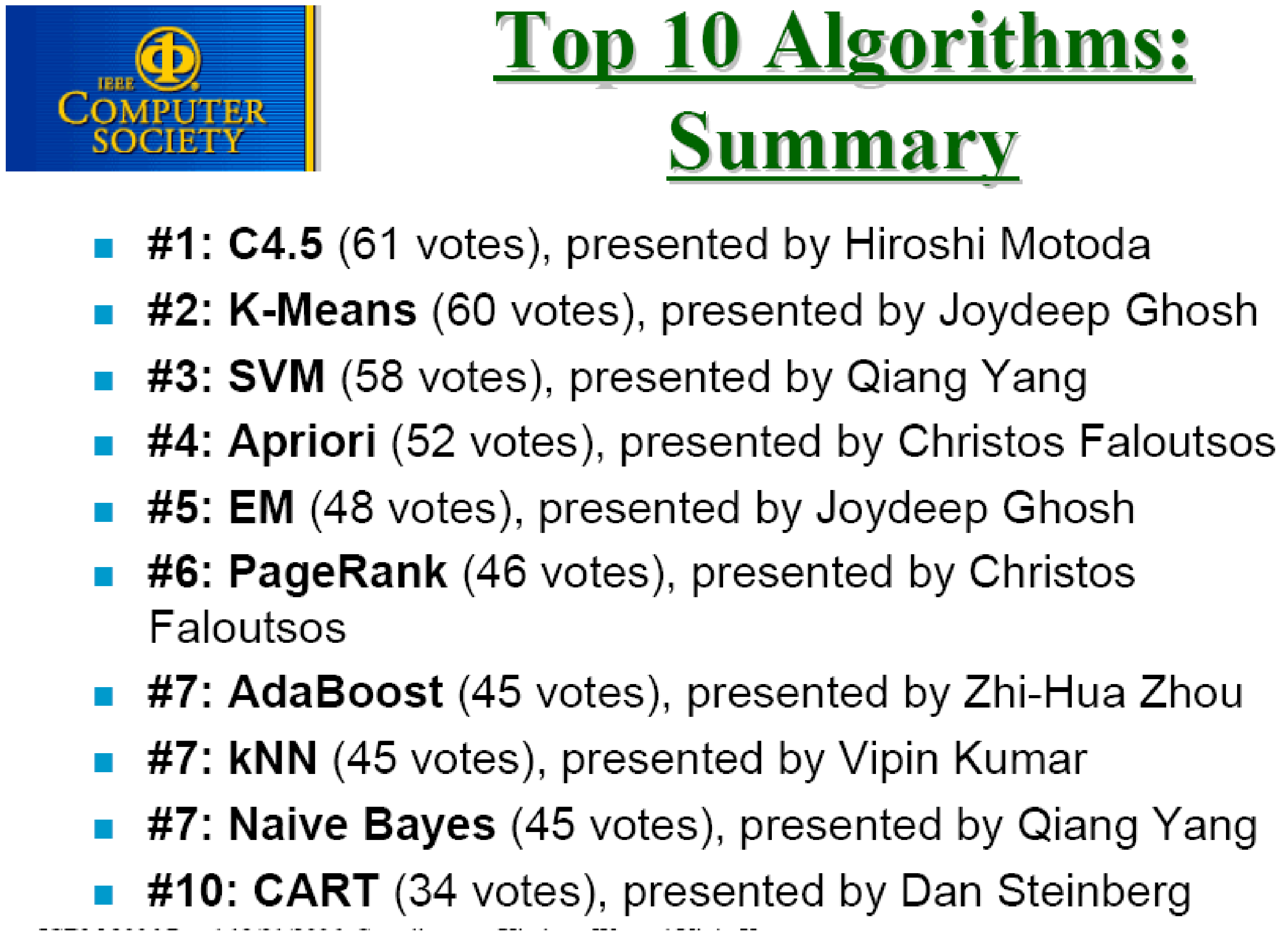


ICDM'06 Panel on "Top 10 Algorithms in Data Mining"

Agenda

1. The 3-step identification process
2. The 18 identified candidates
3. Algorithm presentations
4. Top 10 algorithms: summary
5. Open discussions

ICDM’06 Panel on “Top 10 Algorithms in Data Mining”



Top 10 Algorithms: Summary

- **#1: C4.5** (61 votes), presented by Hiroshi Motoda
- **#2: K-Means** (60 votes), presented by Joydeep Ghosh
- **#3: SVM** (58 votes), presented by Qiang Yang
- **#4: Apriori** (52 votes), presented by Christos Faloutsos
- **#5: EM** (48 votes), presented by Joydeep Ghosh
- **#6: PageRank** (46 votes), presented by Christos Faloutsos
- **#7: AdaBoost** (45 votes), presented by Zhi-Hua Zhou
- **#7: kNN** (45 votes), presented by Vipin Kumar
- **#7: Naive Bayes** (45 votes), presented by Qiang Yang
- **#10: CART** (34 votes), presented by Dan Steinberg

ICDM'06 Panel on "Top 10 Algorithms in Data Mining"

IEEE COMPUTER SOCIETY

Open Votes for Top Algorithms

- Top 3 Algorithms:
  - C4.5: 52 votes
  - SVM: 50 votes
  - Apriori: 33 votes
- Top 10 Algorithms
  - The top 10 algorithms voted from the 18 candidates at the panel are the same as the voting results from the 3-step identification process.

ICDM'06 Panel on "Top 10 Algorithms in Data Mining"

- A survey paper has been generated

Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, Dan Steinberg.

Top 10 algorithms in data mining.

Knowledge Information Systems (2008) 14:1–37.

Outline

- Top 10 Algorithms in Data Mining Research
  - Introduction
  - Classification
  - Statistical Learning
  - Bagging and Boosting
  - Clustering
  - Association Analysis
  - Link Mining – Text Mining
  - Top 10 Algorithms
- 10 Challenging Problems in Data Mining Research
- Concluding Remarks

Outline

- Top 10 Algorithms in Data Mining Research
  - Introduction
  - Classification
  - Statistical Learning
  - Bagging and Boosting
  - Clustering
  - Association Analysis
  - Link Mining – Text Mining
  - Top 10 Algorithms
- 10 Challenging Problems in Data Mining Research
- Concluding Remarks