

# **Unit 6 : Data Mining Approaches and Methods**

**Lecturer : Bijay Mishra**

# Types of Data Mining Models

## Predictive Model

- (a) Classification -Data is mapped into predefined groups or classes. Also termed as supervised learning as classes are established prior to examination of data.
- (b) Regression- Mapping of data item into known type of functions. These may be linear, logistic functions etc.
- (c) Time Series Analysis- Value of an attribute are examined at evenly spaced times, as it varies with time.
- (d) Prediction- It means fore telling future data states based on past and current data.

## Descriptive Models

- (a) Clustering- It is referred as unsupervised learning or segmentation/partitioning. In clustering groups are not pre-defined.
- (b) Summarization- Data is mapped into subsets with simple descriptions . Also termed as Characterization or generalization.
- (c) Sequence Discovery- Sequential analysis or sequence discovery utilized to find out sequential patterns in data. Similar to association but relationship is based on time.
- (d) Association Rules- A model which identifies specific types of data associations.

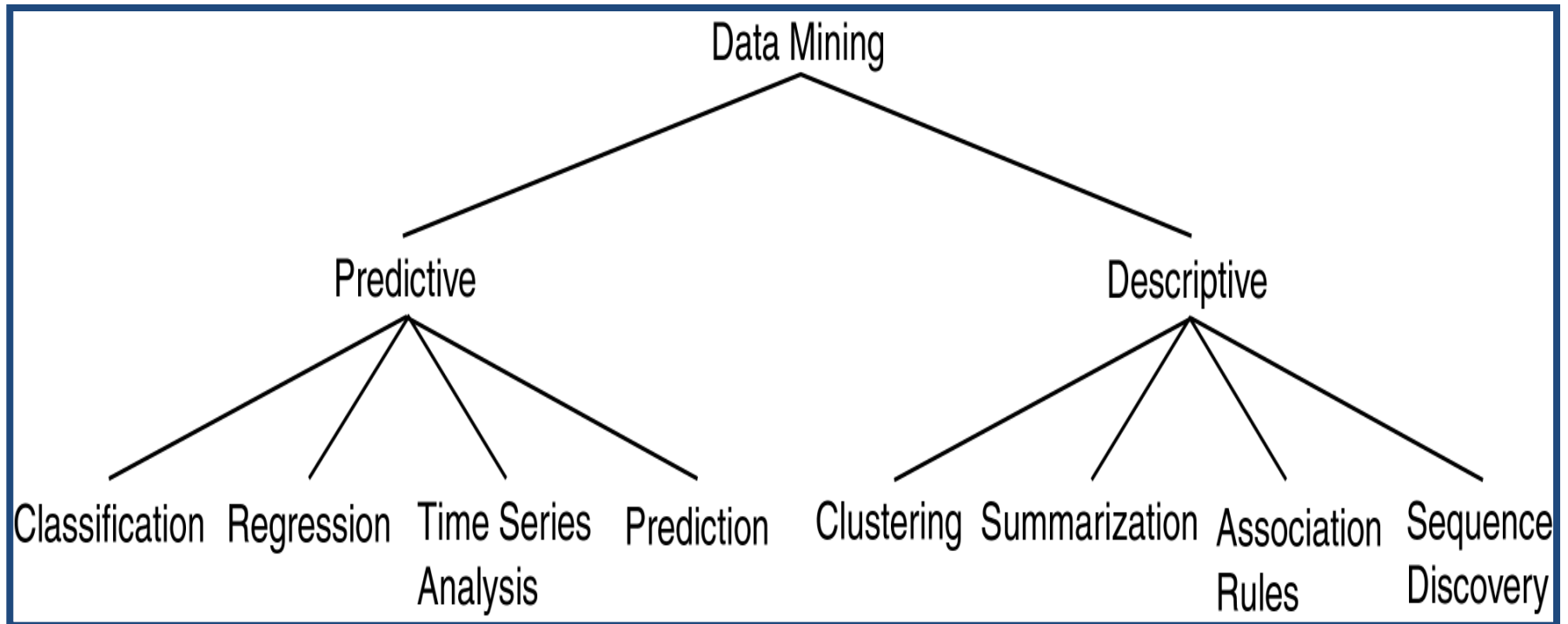
# Descriptive vs. Predictive Data Mining

## Descriptive Mining:

It describes concepts or task-relevant data sets in concise, summarative, informative, discriminative forms.

## Predictive Mining:

It is based on data and analysis, constructs models for the database, and predicts the trend and properties of unknown data.



# Supervised and Unsupervised learning

## Supervised learning:

- The network answer to each input pattern is directly compared with the desired answer and a feedback is given to the network to correct possible errors

## Unsupervised learning:

- The target answer is unknown. The network groups the input patterns of the training sets into clusters, based on correlation and similarities.

## Supervised

- Bayesian Modeling
- Decision Trees
- Neural Networks

Type and number of classes are known in advance

## Unsupervised

- One-way Clustering
- Two-way Clustering

Type and number of classes are **NOT** known in advance

# Concept/Class Description

A concept typically refers to a collection of data such as *frequent buyers*, *graduate students*, and so on. As a data mining task, concept description is not a simple enumeration of the data. Instead, concept description generates descriptions for the *characterization* and *comparison* of the data. It is sometimes called class description, when the concept to be described refers to a class of objects.

Concept description is the most basic form of descriptive data mining. It describes a given set of task-relevant data in a concise and summarative manner, presenting interesting general properties of the data.

Data can be associated with classes or concepts. For example, in the *AllElectronics* store, classes of items for sale include *computers* and *printers*, and concepts of customers include *bigSpenders* and *budgetSpenders*. It can be useful to describe individual classes and concepts in summarized, concise, and yet precise terms. Such descriptions of a class or a concept are called **class/concept descriptions**.

These descriptions can be derived via

- (1) **Data characterization**, by summarizing the data of the class under study (often called the target class) in general terms, or
- (2) **Data discrimination**, by comparison of the target class with one or a set of comparative classes (often called the contrasting classes), or
- (3) both data characterization and discrimination.

The output of data characterization can be presented in various forms. Examples include pie charts, bar charts, curves, multidimensional data cubes, and multidimensional tables, including crosstabs. The resulting descriptions can also be presented as generalized relations or in rule form (called characteristic rules).

# Data Characterization

## **Data Characterization:**

A data mining system should be able to produce a description summarizing the characteristics of customers.

## **Example:**

The characteristics of customers who spend more than \$1000 a year at XYZ store. The result can be a general profile such as age, employment status or credit ratings.



# Data Discrimination

## **Data Discrimination:**

It is a comparison of the general features of targeting class data objects with the general features of objects from one or a set of contrasting classes. User can specify target and contrasting classes.

## **Example:**

The user may like to compare the general features of software products whose sales increased by 10% in the last year with those whose sales decreased by about 30% in the same duration.

# Concept Description vs. OLAP

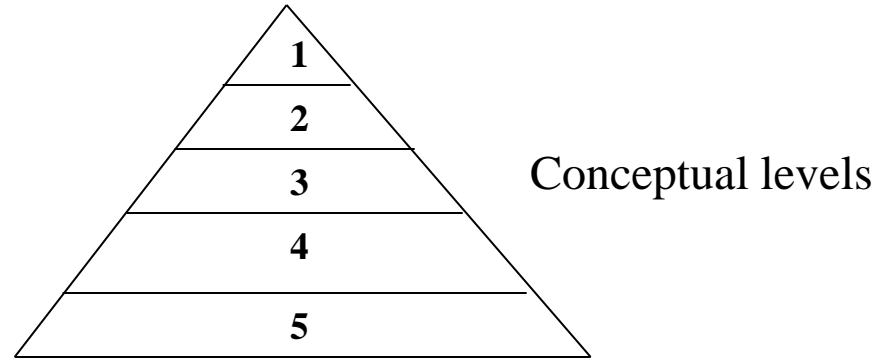
## ❖ Concept Description

- can handle complex data types of the attributes and their aggregations
- a more automated process

## ❖ OLAP

- restricted to a small number of dimension and measure types
- user-controlled process

# Data Generalization



Data generalization is a process that abstracts a large set of task-relevant data in a database from a relatively low conceptual level to higher conceptual levels.

Data generalization summarizes data by replacing relatively low-level values (such as numeric values for an attribute *age*) with higher-level concepts (such as *young*, *middleaged*, and *senior*).

Given the large amount of data stored in databases, it is useful to be able to describe concepts in concise and brief terms at generalized (rather than low) levels of abstraction. Allowing data sets to be generalized at multiple levels of abstraction facilitates users in examining the general behavior of the data.

Data generalization approaches include **data cube approach (OLAP Approach)** and **attribute oriented induction approach**.

# Data Cube Approach (without using Attribute Oriented-Induction)

It perform computations and store results in data cubes

## Strength

- An efficient implementation of data generalization
- Computation of various kinds of measures
  - e.g., count( ), sum( ), average( ), max( )
- Generalization and specialization can be performed on a data cube by *roll-up* and *drill-down*

## Limitations

- handle only dimensions of *simple nonnumeric data* and measures of *simple aggregated numeric values*.
- Lack of intelligent analysis, can't tell which dimensions should be used and what levels should the generalization reach

# Attribute-Oriented Induction

- ❖ Proposed in 1989 (KDD '89 workshop)
- ❖ Not confined to categorical data nor particular measures.
- ❖ How it is done?
  - Collect the task-relevant data( *initial relation*) using a relational database query
  - Perform generalization by *attribute removal* or *attribute generalization*.
  - Apply aggregation by merging identical, generalized tuples and accumulating their respective counts.
  - Interactive presentation with users.

# Classification and Prediction

Classification and prediction are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends. Such analysis can help provide us with a better understanding of the data at large.

Whereas *classification* predicts categorical (discrete, unordered) labels, *prediction* models continuous valued functions.

For example, we can build a classification model to categorize bank loan applications as either safe or risky, or a prediction model to predict the expenditures in dollars of potential customers on computer equipment given their income and occupation.

# Prediction

Prediction is viewed as the construction and use of a model to assess the class of an unlabeled sample or to assess the value ranges of an attribute that a given sample is likely to have.

It is a statement or claim that a particular event will occur in the future in more certain terms than a forecast . It is similar to classification .It constructs a model to predict unknown or missing values. Prediction is the most prevalent grade level expectation on reasoning in state mathematics standards.



Generally it predicts a continuous value rather than categorical label. Numeric prediction predicts the continuous value. The most widely used approach for numeric prediction is regression.

**Regression analysis** is used to model the relationship between one or more independent or predictor variables and a dependent or response variable. In the context of Data Mining, predictor variables are attributes of interest describing the tuple.

# Linear Regression

Regression is a statistical methodology developed by Sir Frances Galton in 1822-1911. Straight line regression analysis involves a response variable  $y$  and a single predictor variable  $x$ .

The simplest form of regression is

$$y = a + bx$$

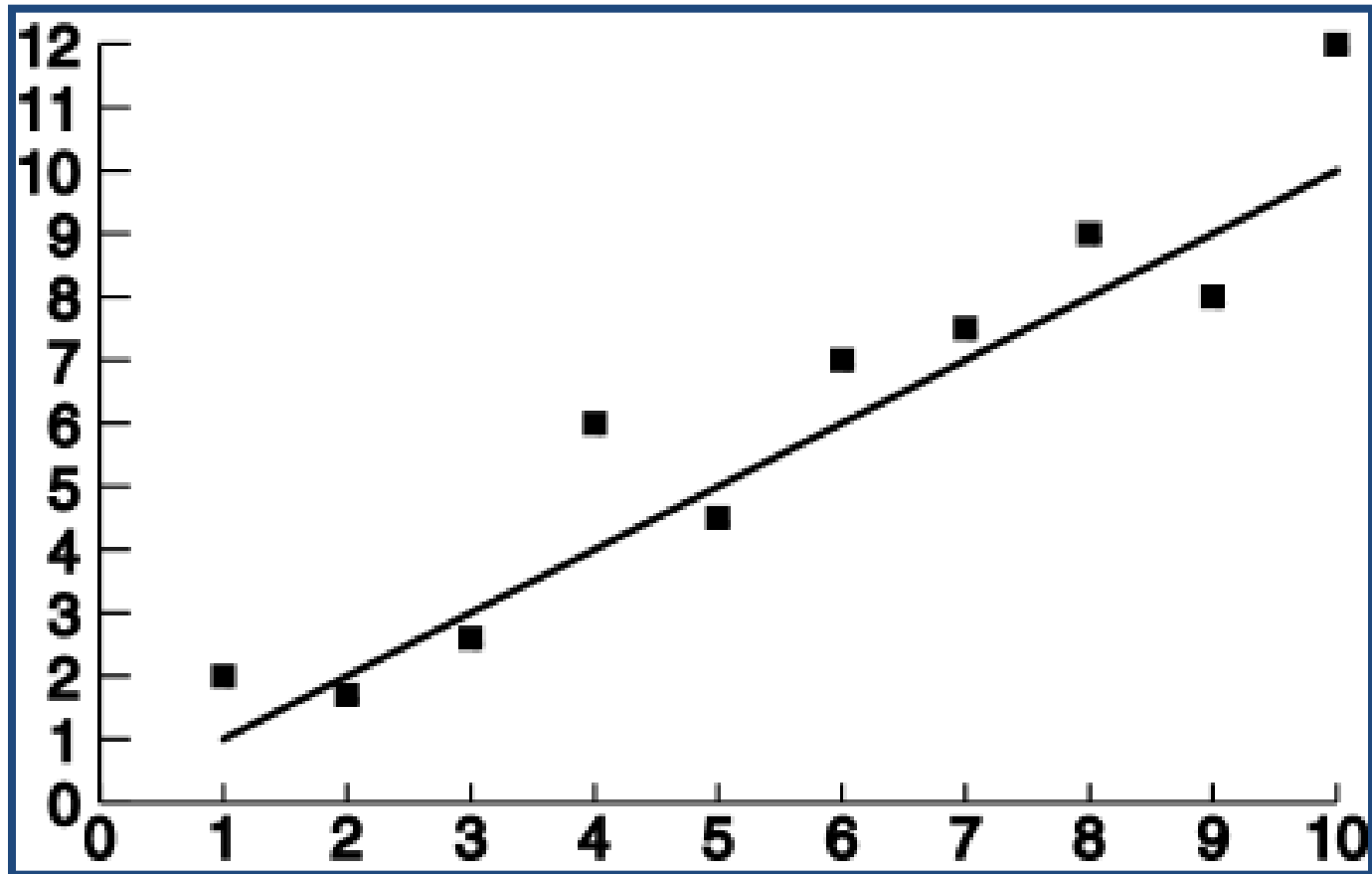
Where  $y$  is response variable and  $x$  is single predictor variable  $y$  is a linear function of  $x$ .  $a$  and  $b$  are regression coefficients.

As the regression coefficients are also considered as weights, we may write the above equation as:

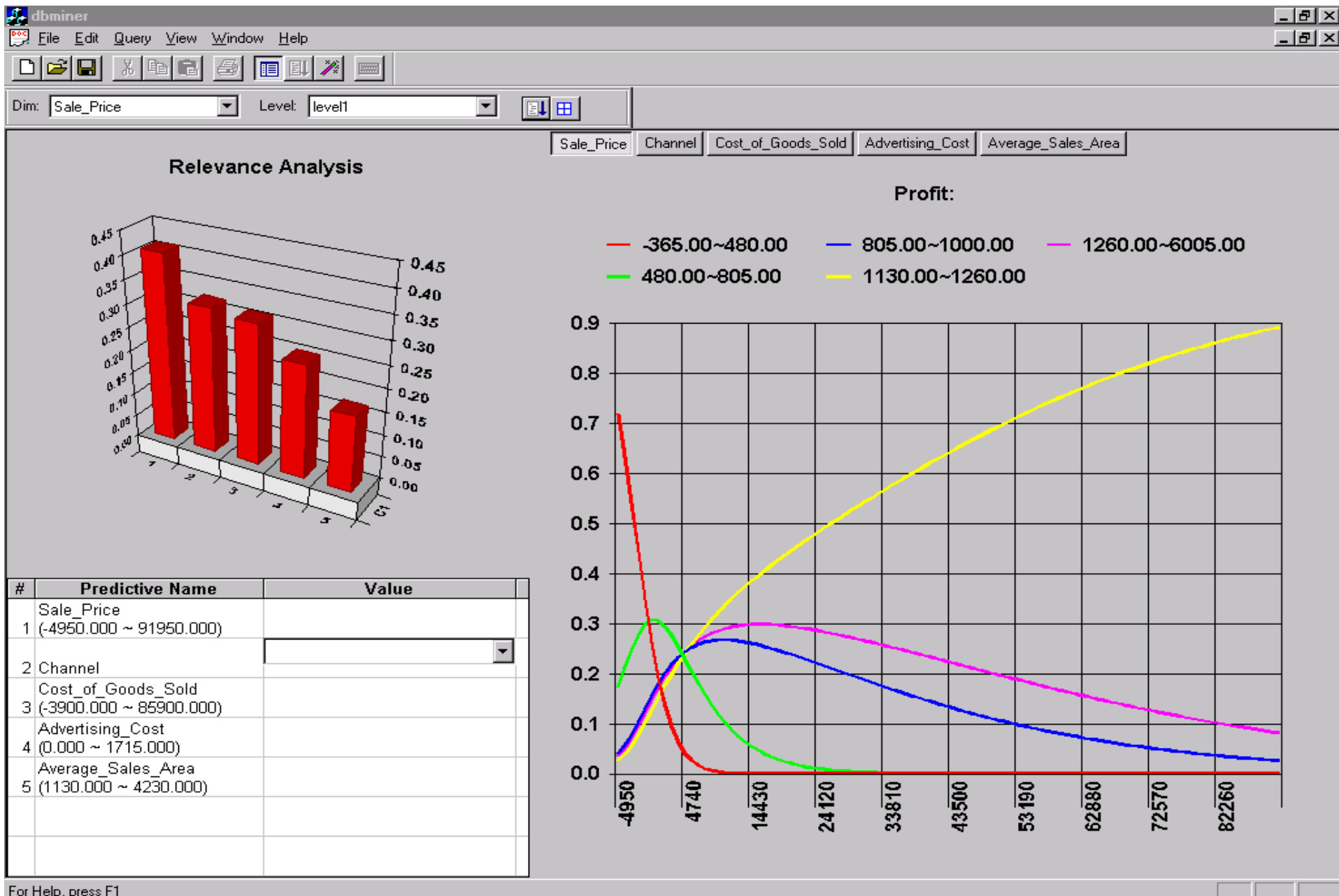
$$y = w + w_1x$$

These coefficients are solved by the method of least squares, which estimates the best fitting straight line as the one that minimizes the error between the actual data and the estimate of the line.

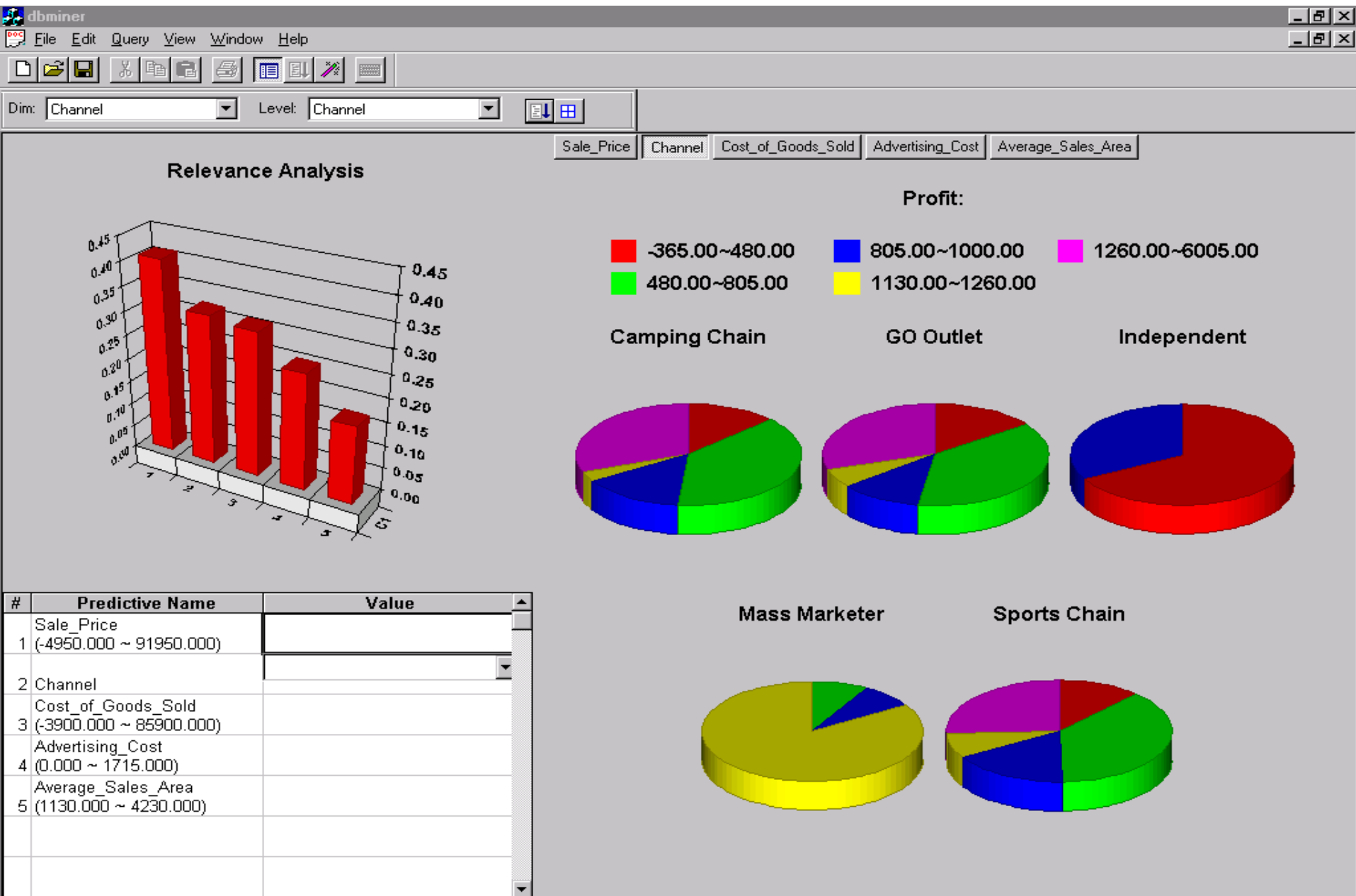
# Linear Regression



# Prediction: Numerical Data

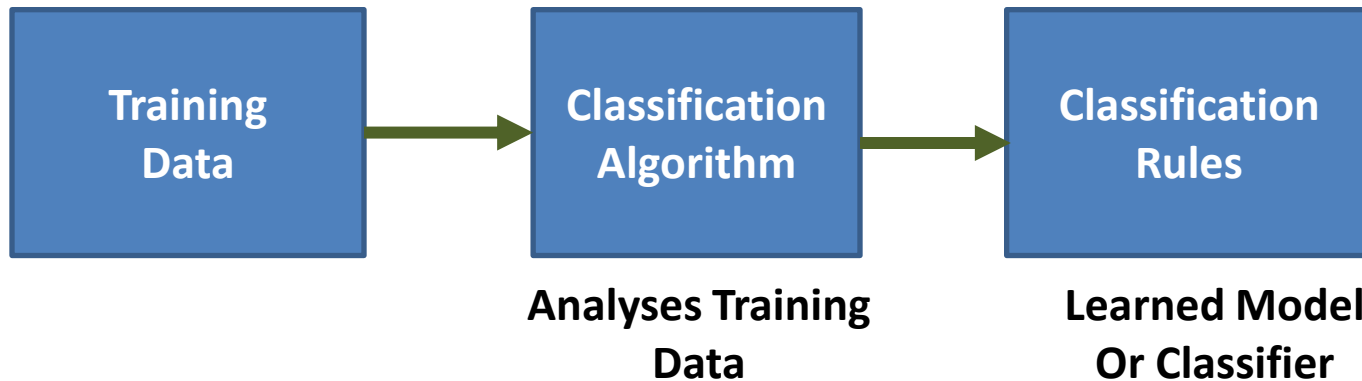


# Prediction: Categorical Data



# Classification

Classification can be described by a two step process given in appended block diagram:

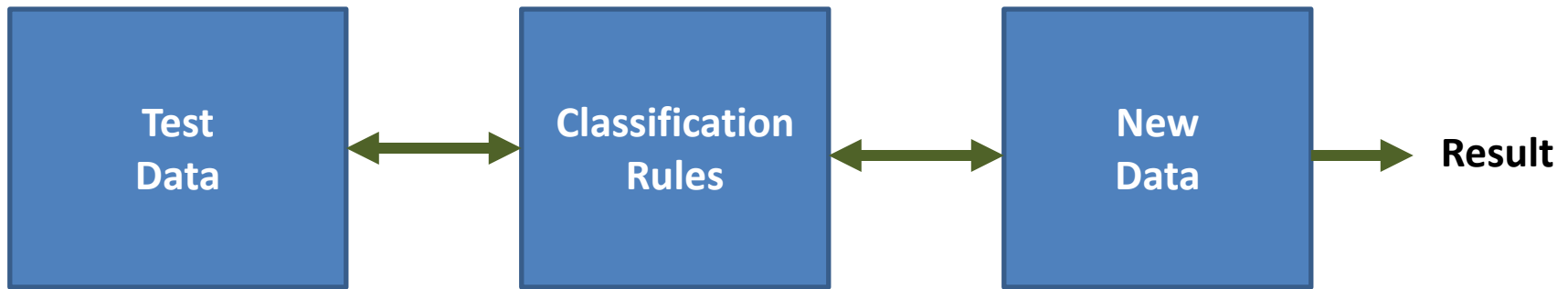


## Step 1

Also known as “supervised learning” as class labels are known. It is different than “Unsupervised learning” or clustering where class labels are not known. A model is built describing a predetermined set of data classes or concepts. The model is constructed by analyzing database tuples described by their attributes. Each tuple is assumed to belong to a predefined class and called as a class label attribute. Data tuples are also referred as Samples, Examples or Objects. Data tuples selected randomly form a training data set and are called training samples. The learning of the model is termed as “Supervised” as it is told which class the training sample belongs. This is in contrast to Clustering which is termed unsupervised learning.

## Step 2

Test data verifies the accuracy of Classification Rules



The model is used for classification. First the predictive accuracy of the model is estimated. If the accuracy is acceptable the model can be used to classify future tuples or objects for which class label is not known.

**Classification** is the process of finding a model (or function) that describes and distinguishes data classes or concepts, for the purpose of being able to use the model to predict the class of objects whose class label is unknown. The derived model is based on the analysis of a set of training data (i.e., data objects whose class label is known).

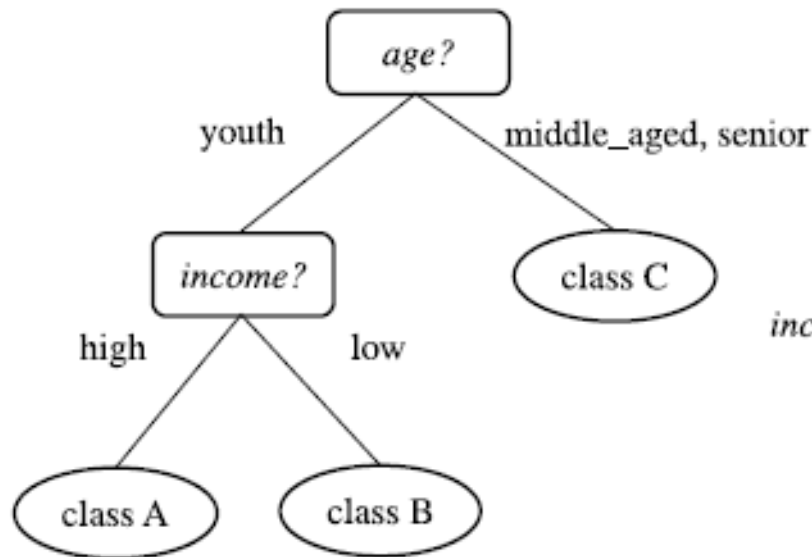
*“How is the derived model presented?”* The derived model may be represented in various forms, such as *classification (IF-THEN) rules, decision trees, mathematical formulae, or neural networks.*



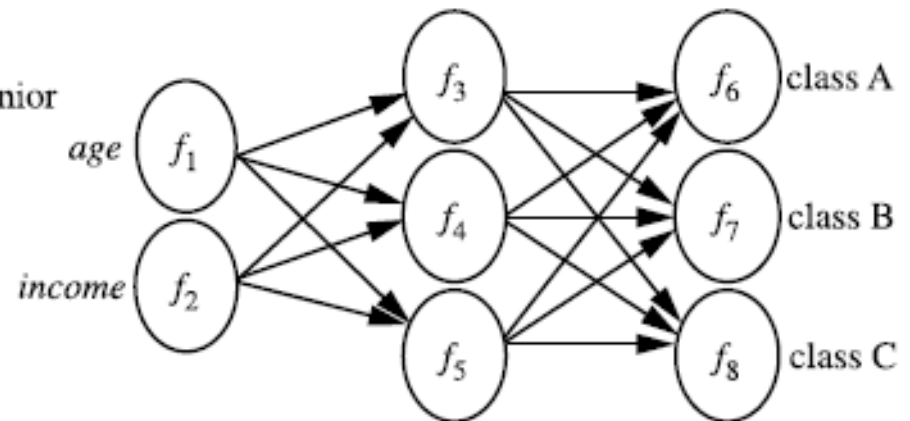
(a)

age(X, "youth") AND income(X, "high")  $\longrightarrow$  class(X, "A")  
age(X, "youth") AND income(X, "low")  $\longrightarrow$  class(X, "B")  
age(X, "middle\_aged")  $\longrightarrow$  class(X, "C")  
age(X, "senior")  $\longrightarrow$  class(X, "C")

(b)



(c)



**Figure :** A classification model can be represented in various forms, such as (a) IF-THEN rules, (b) a decision tree, or a (c) neural network.

# Classification : Example of Grading

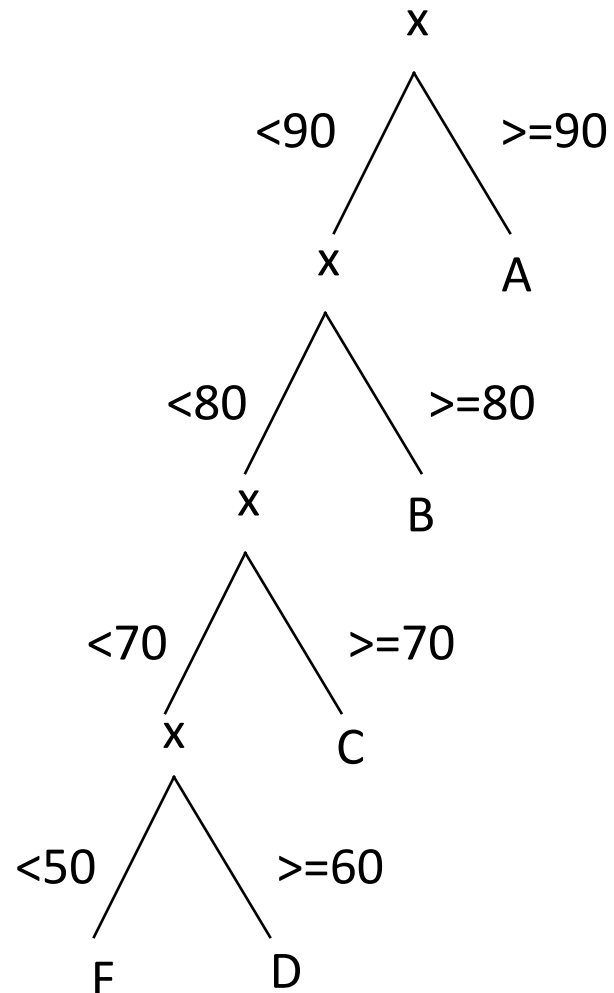
If  $x \geq 90$  then grade = A.

If  $80 \leq x < 90$  then grade = B.

If  $70 \leq x < 80$  then grade = C.

If  $60 \leq x < 70$  then grade = D.

If  $x < 50$  then grade = F.



# How does classification work?

Data classification is a two-step process:

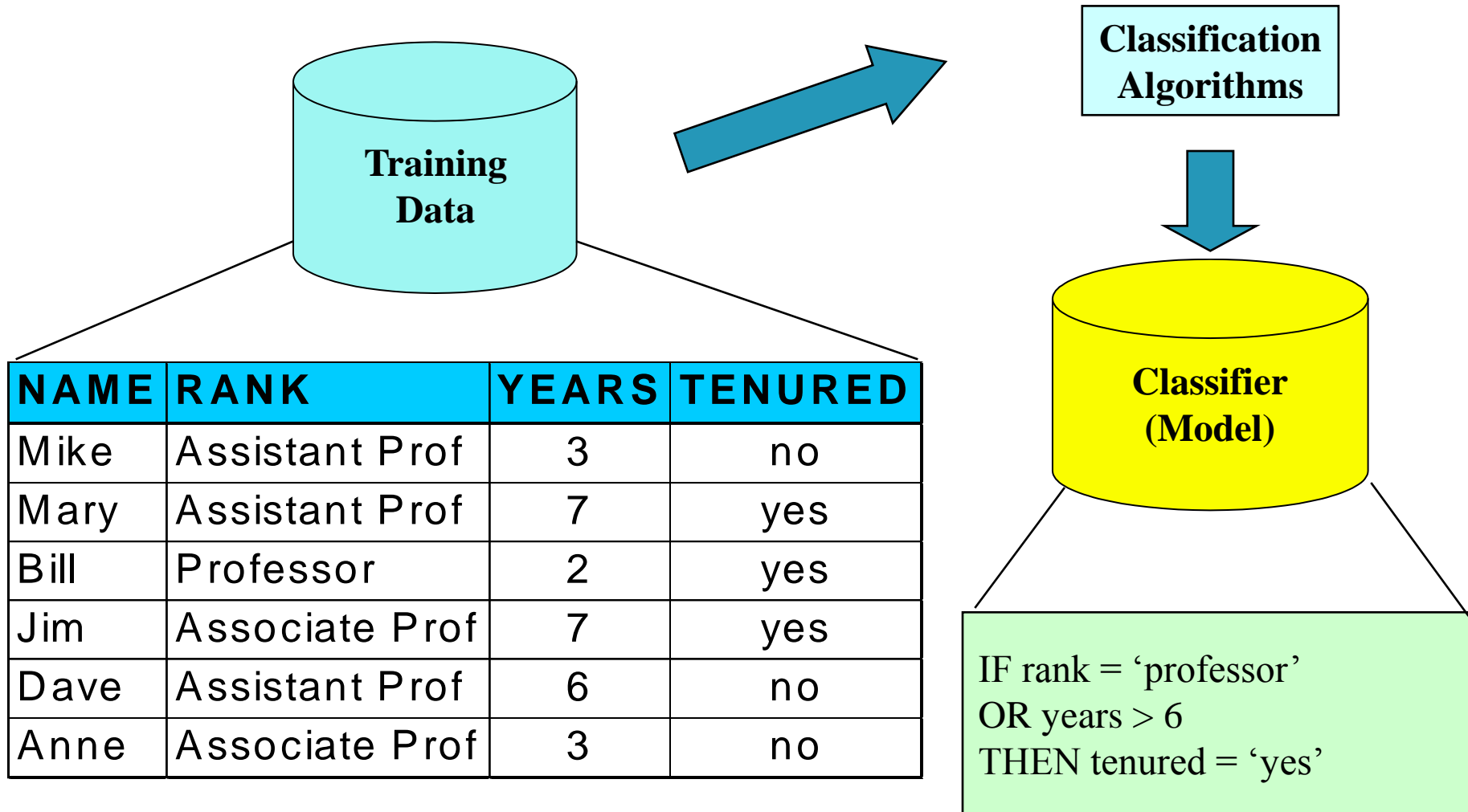
## (1) Model construction

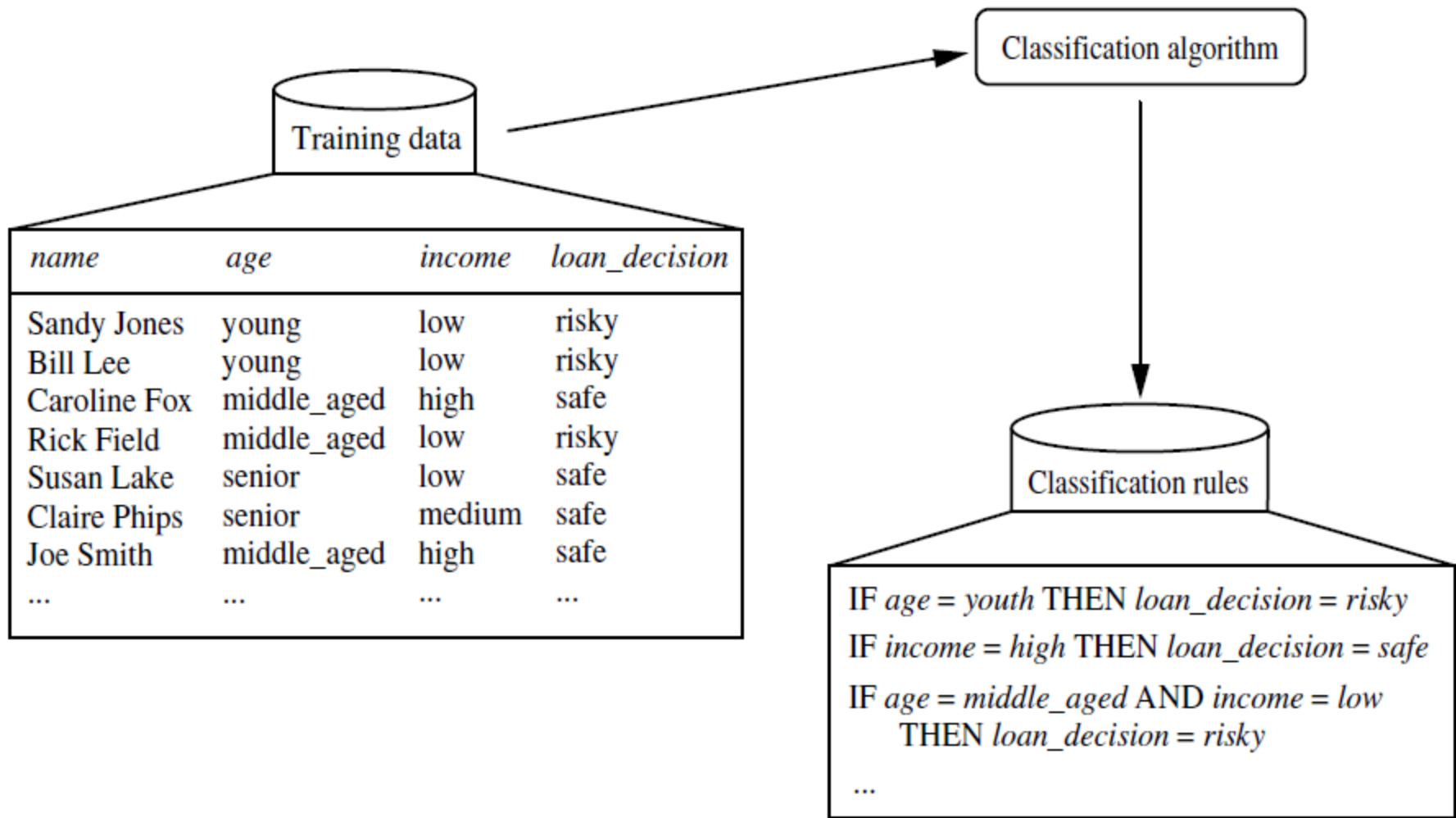
Training data are analyzed by a classification algorithm. A classifier is built describing a predetermined set of data classes or concepts. Also called as training phase or learning stage.

## (2) Model usage

Test data are used to estimate the accuracy of the classification rules. If the accuracy is considered acceptable, the rules can be applied to the classification of new data tuples.

# Model Construction

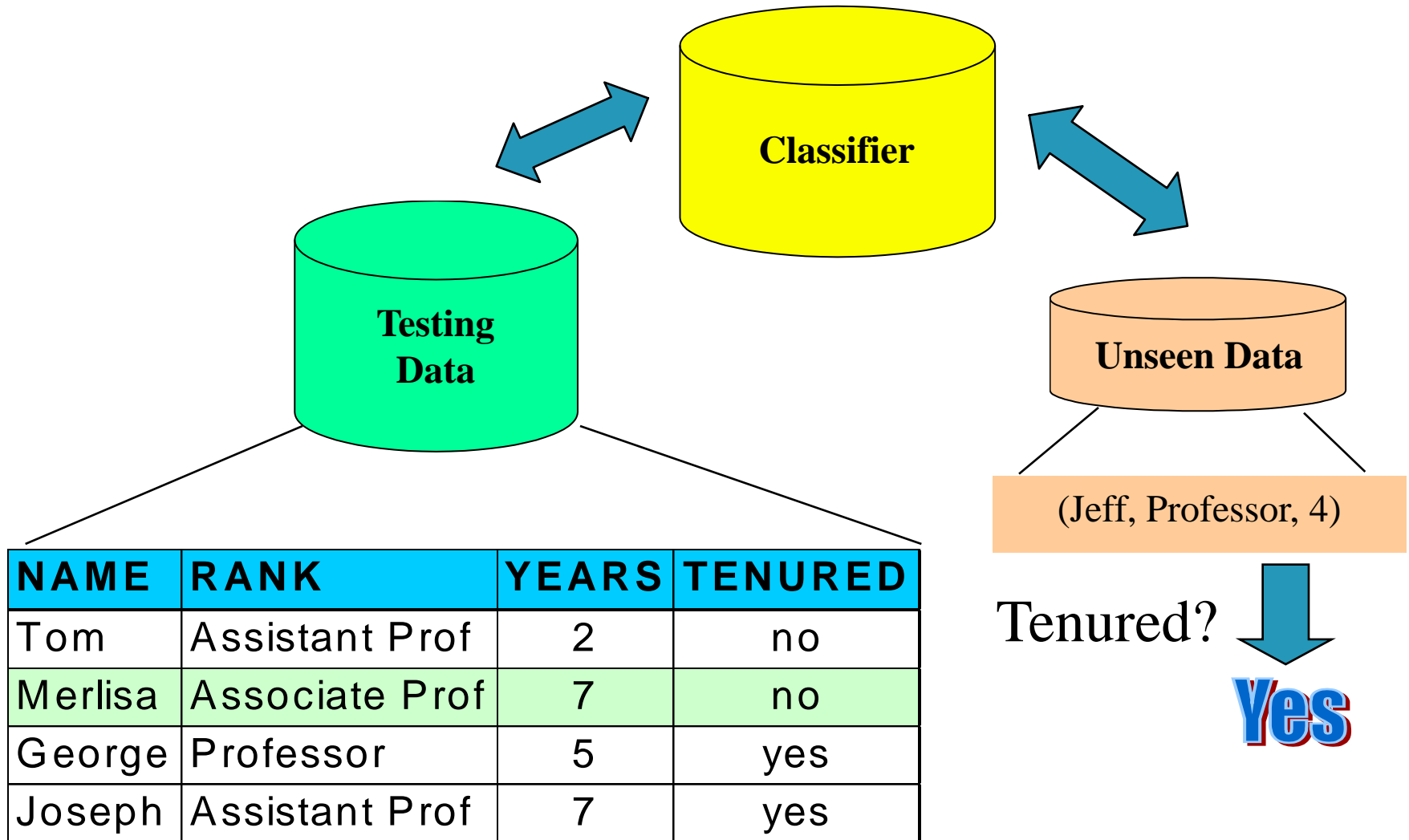


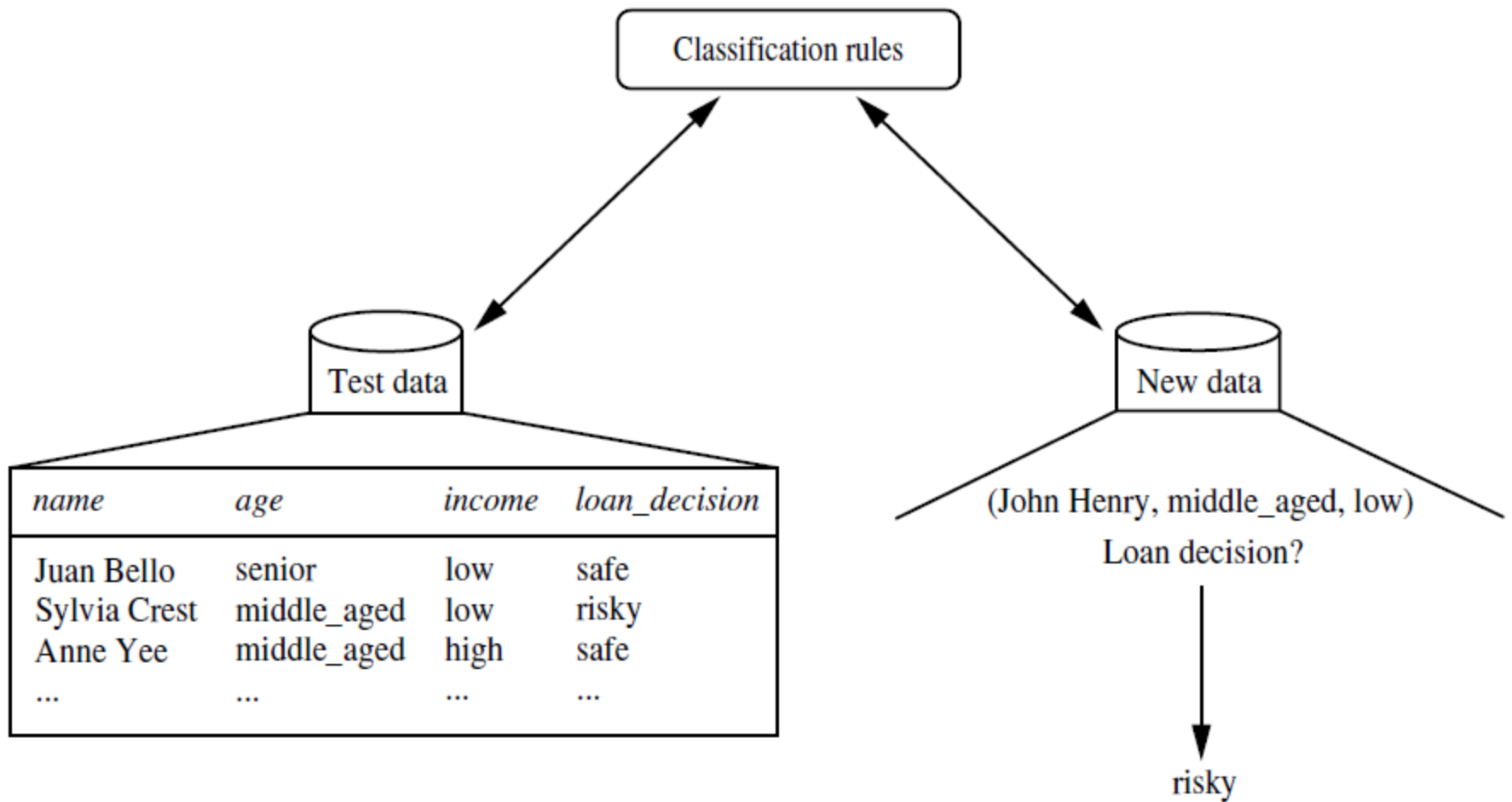


**Figure: Learning**

Here, the class label attribute is *loan decision*, and the learned model or classifier is represented in the form of classification rules.

# Model Usage





**Figure: Classification**

# Examples of Classification Algorithms

- ❖ Decision Trees
- ❖ Neural Networks
- ❖ Bayesian Networks



# Issues regarding classification and prediction

## Issues (1): Data Preparation

- ❖ Data cleaning
  - Preprocess data in order to reduce noise and handle missing values
- ❖ Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes
- ❖ Data transformation
  - Generalize and/or normalize data

## Issues (2): Evaluating Classification Methods

- ❖ Predictive accuracy
- ❖ Speed and scalability
  - time to construct the model
  - time to use the model
- ❖ Robustness
  - handling noise and missing values
- ❖ Scalability
  - efficiency in disk-resident databases
- ❖ Interpretability
  - understanding and insight provided by the model
- ❖ Goodness of rules
  - decision tree size
  - compactness of classification rules

# Decision Trees

A decision tree is a predictive model that as its name implies can be viewed as a tree. Specifically each branch of the tree is a classification question and the leaves are partitions of data set with their classification.

A decision tree makes a prediction on the basis of a series of decisions. The decision trees are being built on historical data and are a part of the supervised learning. The machine learning technique for inducing a decision tree from data is called **decision tree learning**.

- Internal node denotes a test on an attribute
- Branch represents an outcome of the test
- Leaf nodes represent class labels or class distribution

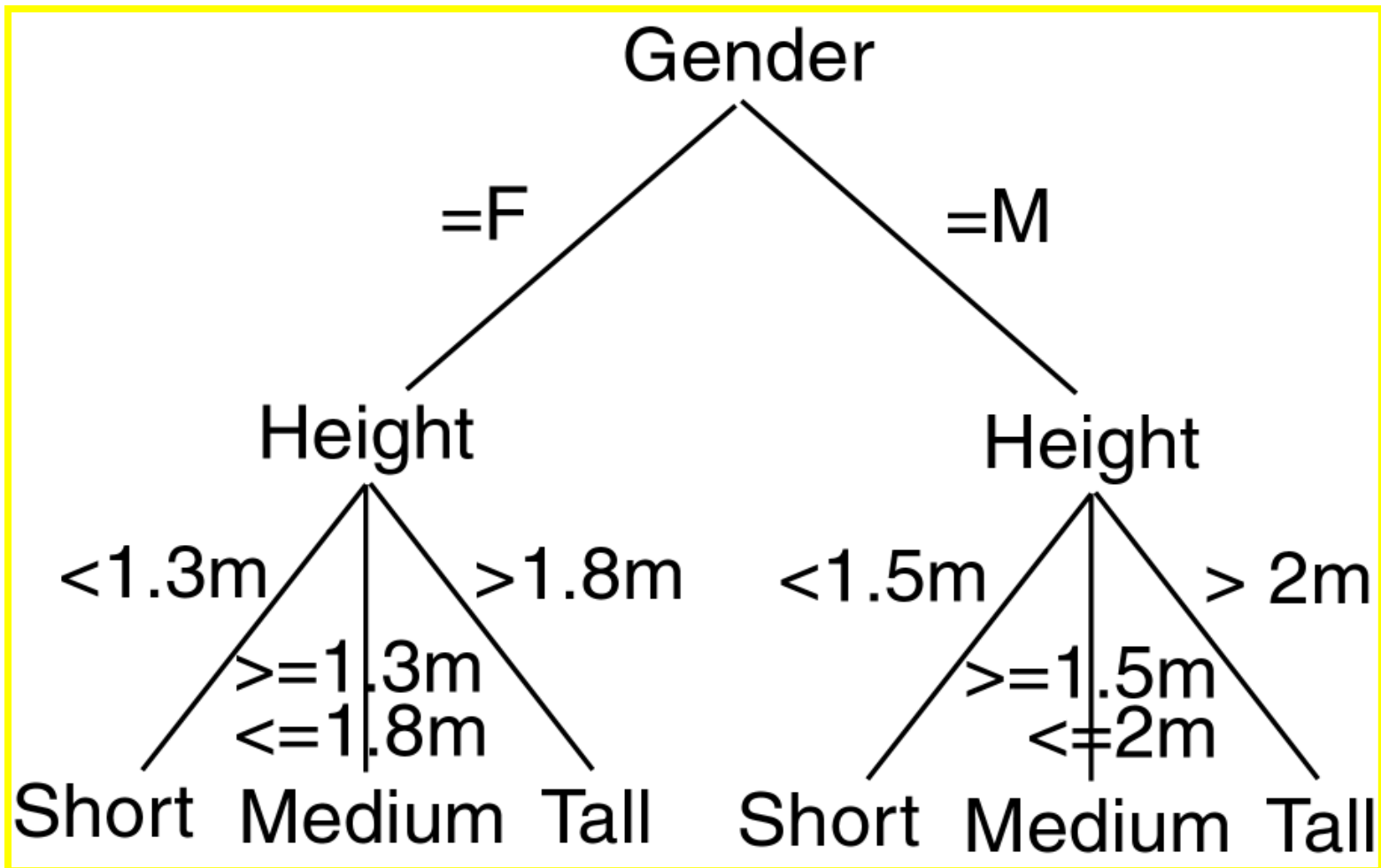
**In data mining, trees have three more descriptive categories/names:**

- ❖ *Classification tree* analysis - when the predicted outcome is the class to which the data belongs.
- ❖ *Regression tree* analysis - when the predicted outcome can be considered a real number (e.g. the price of a house, or a patient's length of stay in a hospital).
- ❖ *Classification And Regression Tree (CART)* analysis - when both of the above procedures are referred.

Decision tree generation consists of two phases

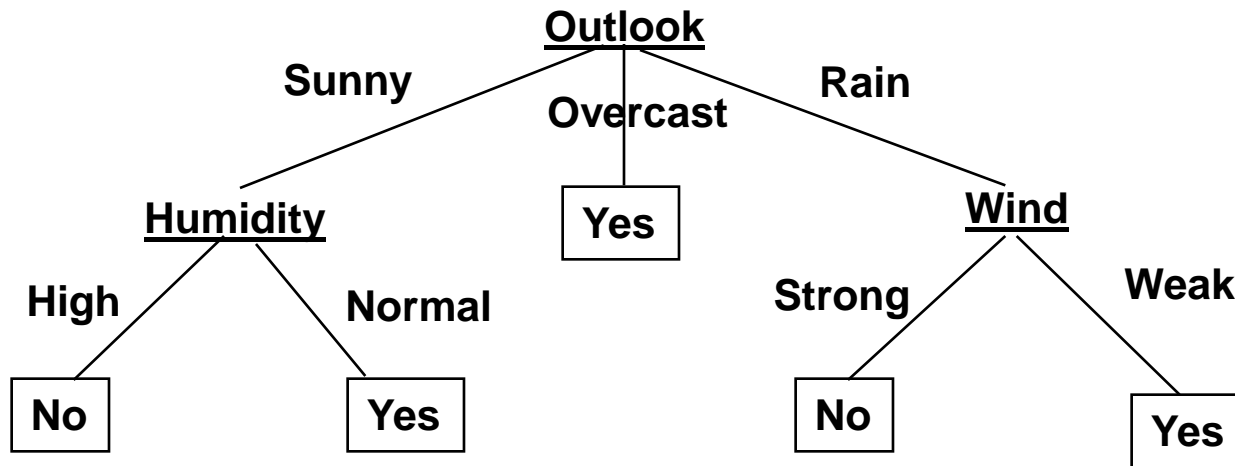
- ❖ Tree construction
  - At start, all the training examples are at the root
  - Partition examples recursively based on selected attributes
- ❖ Tree pruning
  - Identify and remove branches that reflect noise or outliers

# Decision Tree Example



# Decision Tree: Example

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No



Attributes = {Outlook, Temperature, Humidity, Wind}

Play Tennis = {yes, no}

# 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
- Examples are partitioned recursively to *maximize purity*

## Conditions for stopping partitioning

- All samples belong to the same class
- Leaf node smaller than a specified threshold
- Tradeoff between complexity and generalizability

## Predictions for new data:

- Classification by **majority voting** is employed for classifying all members of the leaf
- Probability based on training data that ended up in that leaf.
- Class Probability estimates can be used also

# Algorithm for building Decision Trees

Decision trees are a popular structure for supervised learning. They are constructed using attributes best able to differentiate the concepts to be learned.

A decision tree is built by initially selecting a subset of instances from a training set. This subset is then used by the algorithm to construct a decision tree. The remaining training set instances test the accuracy of the constructed tree.

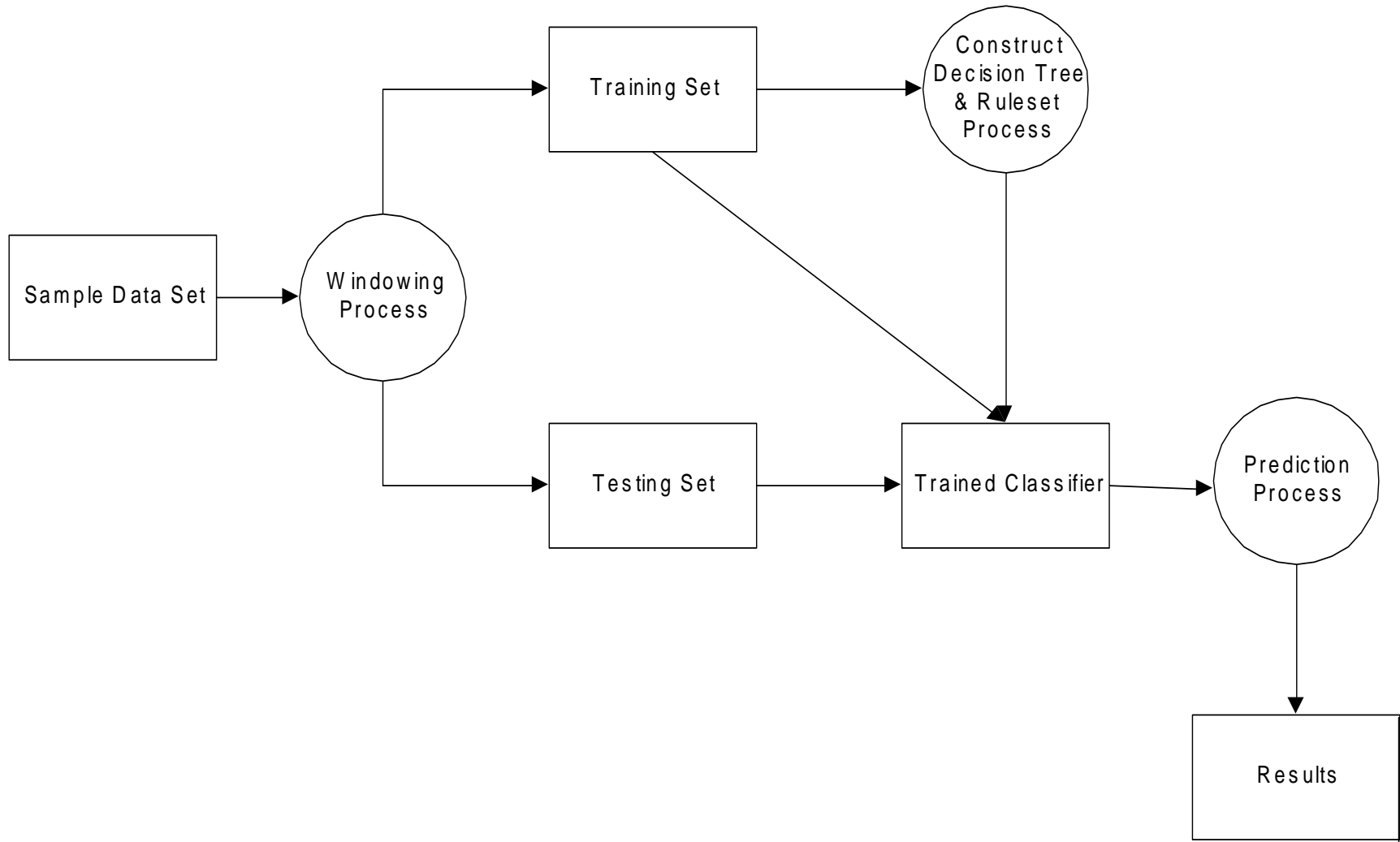
If the decision tree classified the instances correctly, the procedure terminates.

If an instance is incorrectly classified, the instance is added to the selected subset of training instances and a new tree is constructed.

This process continues until a tree that correctly classify all non-selected instances is created or the decision tree is built from the entire training set.



|----- Data Preparation Stage -----|----- Tree Building Stage -----|--- Prediction Stage ---|



**Block Diagram of Training Process**

# Decision Tree Algorithm

Input:

$T$      // *Decision Tree*

$D$      // *Input Database*

Output:

$M$      // *Model Prediction*

DTProc Algorithm:

      // *Illustrates Prediction Technique using DT*

for each  $t \in D$  do

$n = \text{root node of } T;$

    while  $n$  not leaf node do

*Obtain answer to question on  $n$  applied  $t$ ;*

*Identify arc from  $t$  which contains correct answer;*

$n = \text{node at end of this arc};$

*Make prediction for  $t$  based on labeling of  $n$ ;*

# Decision Tree Pseudocode

```
node = tree-design(Data = {X,C})
```

```
  For i = 1 to d
```

```
    quality_variable(i) = quality_score( $X_i$ , C)
```

```
  end
```

```
  node = {X_split, Threshold } for max{quality_variable}
```

```
  {Data_right, Data_left} = split(Data, X_split, Threshold)
```

```
  if node == leaf?
```

```
    return(node)
```

```
  else
```

```
    node_right = tree-design(Data_right)
```

```
    node_left = tree-design(Data_left)
```

```
  end
```

```
end
```

# Basic algorithm for inducing a decision tree

Algorithm:   Generate\_decision\_tree.   Generate   a decision tree from the given training data.

Input: The training samples, represented by discrete-valued attributes; the set of candidate attributes, attribute-list;

Output: A decision tree

Begin

Partition (S)

If (all records in S are of the same class or only 1 record found in S)  
then return;

For each attribute  $A_i$  do

    evaluate splits on attribute  $A_i$ ;

Use best split found to partition S into S1 and S2 to grow a tree  
with two      Partition (S1) and Partition (S2);

Repeat partitioning for Partition (S1) and (S2) until it meets tree  
stop growing criteria;

End;

# Decision Tree Algorithms & their main Issues

1. **Tree Structure** - Selection of a tree structure like Balanced tree for improving performance.
2. **Training Data** - Structure of a tree depends on the training data. Selecting adequate data prevents either the tree to overfit and on the other hand good enough to work on a general data
3. **Stopping Criteria** - Construction of a tree stops on a Stopping criteria. It is essential to achieve a balance between too early or late to create a tree with right level.
4. **Pruning** - After constructing a tree, modify it to remove duplication or subtrees.
5. **Splitting** - Selection of the best splitting attribute and size of the training set are important factors in creating a decision tree algorithm. For example, Splitting attributes in the case of students may be gender, marks scored and electives chosen. The order in which splitting attributes are chosen are important for avoiding redundancy and unnecessary comparisons at different levels.

# Decision Tree Learning Algorithm - ID3

**ID3 (Iterative Dichotomiser)** is a simple decision tree learning algorithm developed by Ross Quinlan (1983). ID3 follows a non-backtracking approach in which decision trees are constructed in a top-down recursive “divide and conquer” manner to test each attribute at every tree node. This approach starts with a training set of tuples and their associated class labels. The training set is recursively partitioned into smaller subsets as the tree is being built.

## ID3 Algorithm:

1. create a node N;
2. if tuples in D are all of the same class C then
3. return N as a leaf node labeled with the class C;
4. if attribute\_list is empty then
5. return N as a leaf node labeled with the majority class in D;
6. apply Attribute\_selection\_method(D, attribute\_list) to find the "best" splitting\_criterion;
- label node N with splitting\_criterion;
7. if splitting\_attribute is discrete-valued and multiway splits allowed then //not restricted to binary trees  
attribute\_list (arrow mark) attribute\_list - splitting\_attribute;
8. for each outcome j of splitting\_criterion
9. let (symbol) be the set of data tuples in D satisfying outcome j; // partition
10. if (symbol) is empty then  
attach a leaf labeled with the majority class in D to node N;
11. else attach the node returned by  
Generate\_decision\_tree(symbol, attribute\_list) to node N;
- endfor
- return N;



## **Explanation of Algorithm:**

The above algorithm has three parameters  $D$ , `attribute_list` and `attribute_selection_method`.  $D$  is data partition. It is a set of training tuples and their associated class labels. `Attribute_list` contains a list of attributes describing the tuples.

Now tree starts as a single node  $N$ . It represents the training tuples in  $D$ . If the tuples in  $D$  are all of the same class then node  $N$  is considered as leaf. It is labeled with that class. It is occurring in step 2 and 3. Step 4 and 5 are terminating conditions. IF this condition does not follow then algorithm calls `Attribute_selection_method` to determine the splitting criterion. This criterion determines the best way to partition the tuples in  $D$  into individual classes(step 6). Step 7 serves as a test at the node. In steps 10 and 11, tuples in  $D$  are partitioned.

# Advantages of using ID3

- ❖ Understandable prediction rules are created from the training data.
- ❖ Builds the fastest tree.
- ❖ Builds a short tree.
- ❖ Only need to test enough attributes until all data is classified.
- ❖ Finding leaf nodes enables test data to be pruned, reducing number of tests.
- ❖ Whole dataset is searched to create tree.

# Disadvantages of using ID3

- ❖ Data may be over-fitted or over-classified, if a small sample is tested.
- ❖ Only one attribute at a time is tested for making a decision.
- ❖ Classifying continuous data may be computationally expensive, as many trees must be generated to see where to break the continuum.

# Pros and Cons of Decision Tree

## Pros

- no distributional assumptions
- can handle real and nominal inputs
- speed and scalability
- robustness to outliers and missing values
- interpretability
- compactness of classification rules
- They are easy to use.
- Generated rules are easy to understand .
- Amenable to scaling and the database size.

## Cons

- several tuning parameters to set with little guidance
- decision boundary is non-continuous
- Cannot handle continuous data.
- Incapable of handling many problems which cannot be divided into attribute domains.
- Can lead to over-fitting as the trees are constructed from training data.

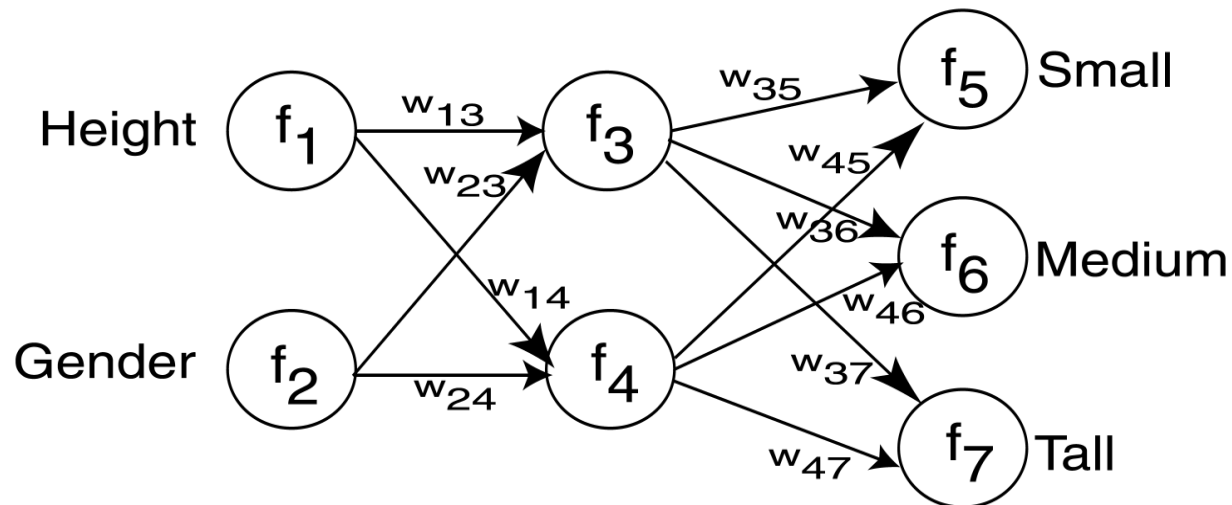
# Neural Networks

Neural Network is a set of connected INPUT/OUTPUT UNITS, where each connection has a WEIGHT associated with it. It is a case of SUPERVISED, INDUCTIVE or CLASSIFICATION learning.

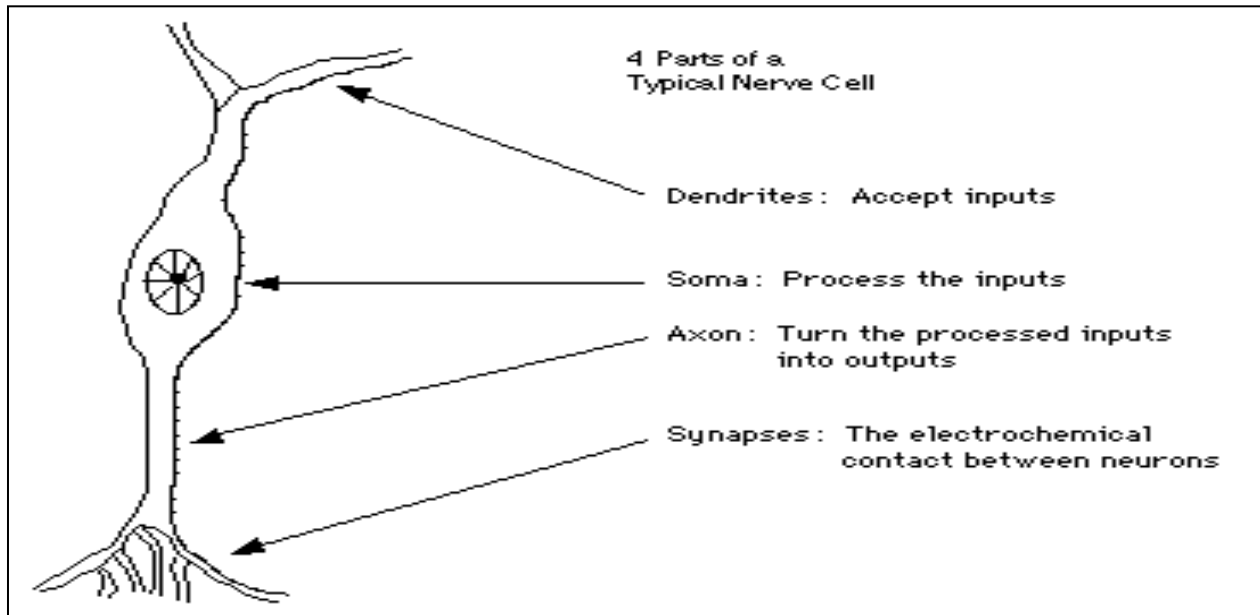
Neural Network learns by adjusting the weights so as to be able to correctly classify the training data and hence, after testing phase, to classify unknown data. Neural Network needs long time for training. Neural Network has a high tolerance to noisy and incomplete data.

**A Neural Network (NN)** is a directed graph  $F=\langle V,A\rangle$  with vertices  $V=\{1,2,\dots,n\}$  and arcs  $A=\{\langle i,j\rangle \mid 1\leq i,j\leq n\}$ , with the following restrictions:

- $V$  is partitioned into a set of input nodes,  $V_I$ , hidden nodes,  $V_H$ , and output nodes,  $V_O$ .
- The vertices are also partitioned into layers
- Any arc  $\langle i,j\rangle$  must have node  $i$  in layer  $h-1$  and node  $j$  in layer  $h$ .
- Arc  $\langle i,j\rangle$  is labeled with a numeric value  $w_{ij}$ .
- Node  $i$  is labeled with a function  $f_i$ .

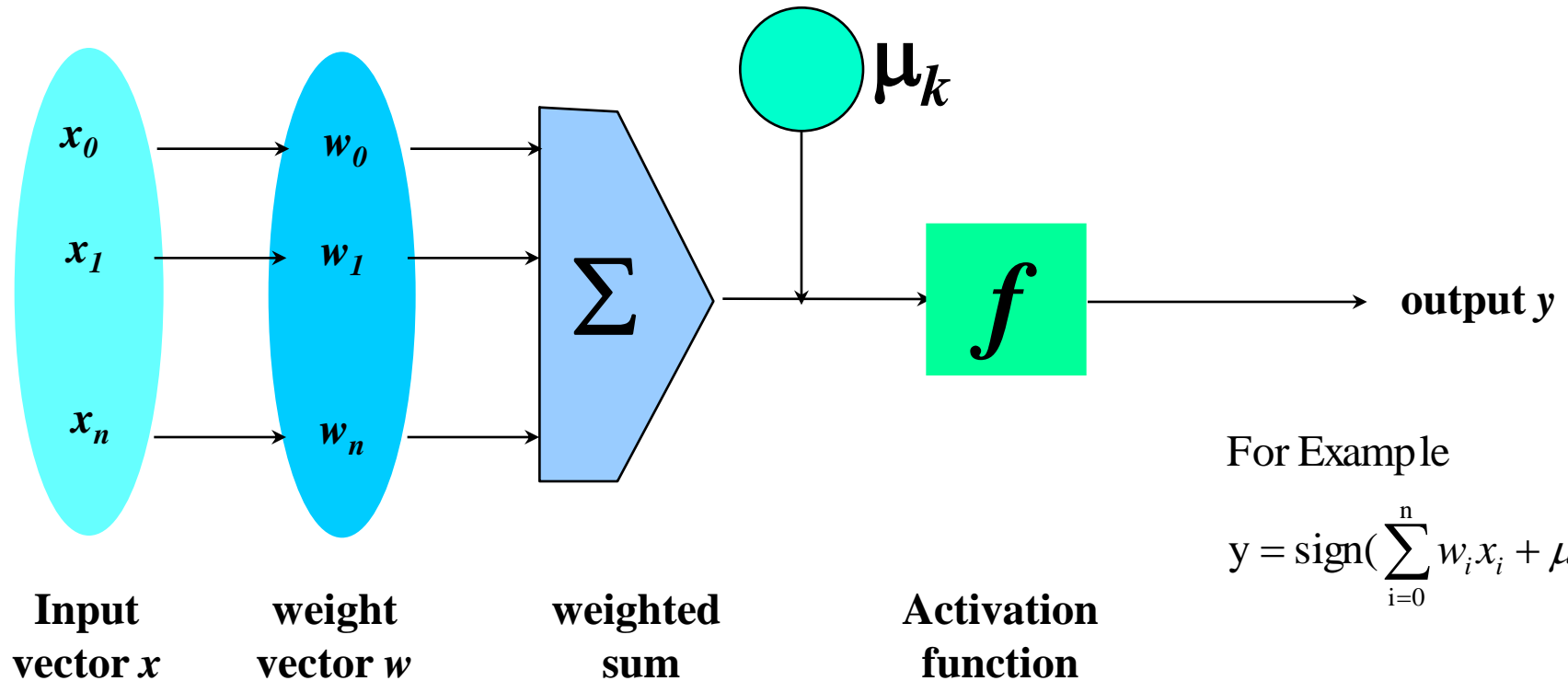


# Similarity with Biological Network



- ❖ Fundamental processing element of a neural network is a neuron
- ❖ A human brain has 100 billion neurons
- ❖ An ant brain has 250,000 neurons

# A Neuron (= a Perceptron)



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



# Network Training

The ultimate objective of training is to obtain a set of weights that makes almost all the tuples in the training data classified correctly.

## Steps

Initialize weights with random values

Feed the input tuples into the network one by one

For each unit

- Compute the net input to the unit as a linear combination of all the inputs to the unit
- Compute the output value using the activation function
- Compute the error
- Update the weights and the bias

# Backpropagation

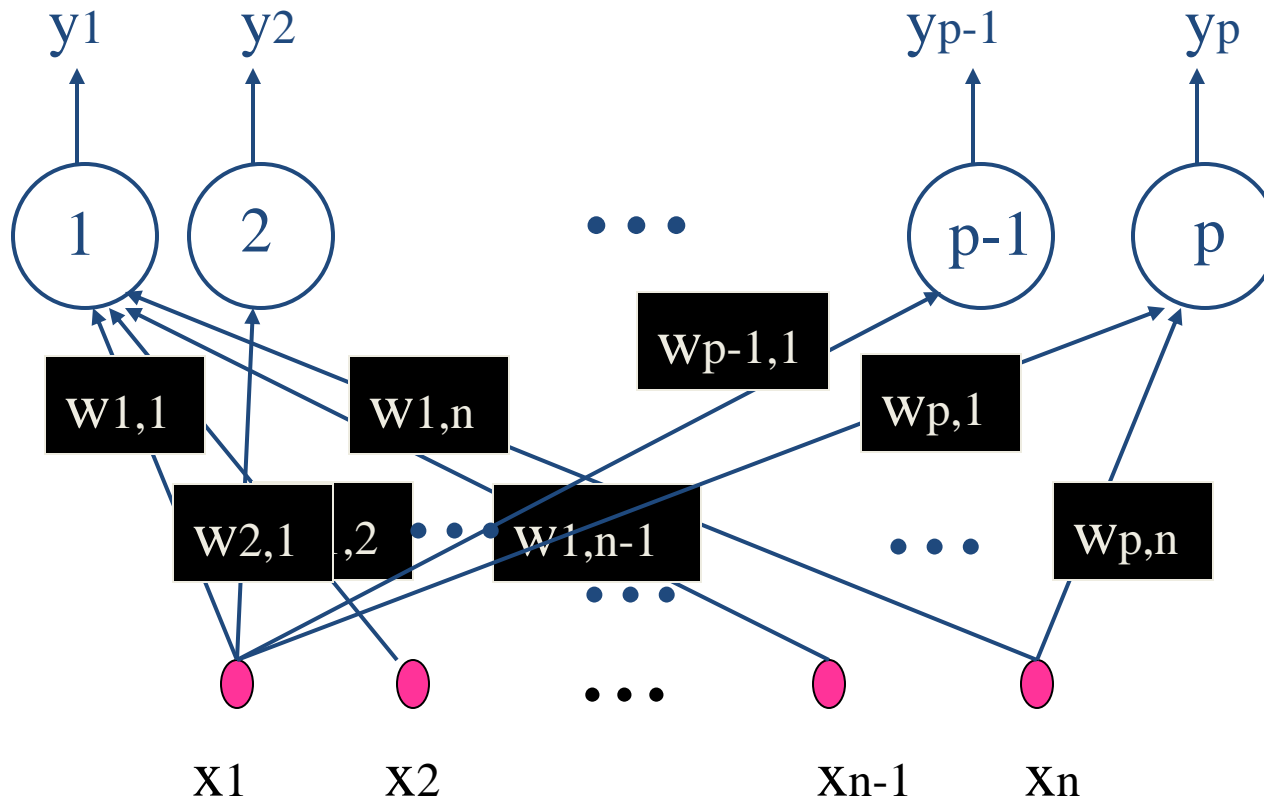
- ❖ Backpropagation is a neural network learning algorithm which gained reputation in the 1980s.
- ❖ Backpropagation learns by iteratively processing a data set of training tuples, comparing the network's prediction for each tuple with the actual known *target* value.
- ❖ The target value may be the known class label of the training tuple (for classification problems) or a continuous value (for prediction).
- ❖ For each training tuple, the weights are modified so as to minimize the mean squared error between the network's prediction and the actual target value.

- ❖ These modifications are made in the “backwards” direction, that is, from the output layer, through each hidden layer down to the first hidden layer (hence the name *backpropagation*).
- ❖ Although it is not guaranteed, in general the weights will eventually converge, and the learning process stops.

## **Steps in Backpropagation:**

- Initialize weights (to small random numbers) and biases in the network
- Propagate the inputs forward (by applying activation function)
- Backpropagate the error (by updating weights and biases)
- Terminating condition (when error is very small, etc.)

# Perceptron



$$y_i(t+1) = f\left(\sum_{k=0}^n w_{ik} x_k(t)\right) \quad i = 1, 2, \dots, p$$

# Multi-Layer Perceptron

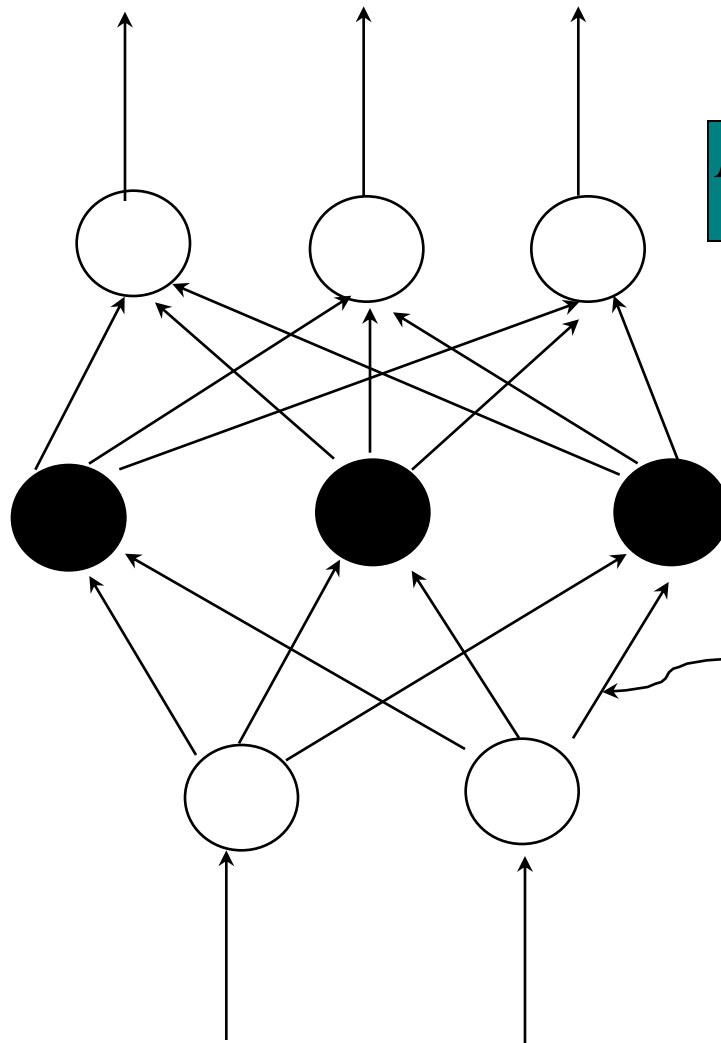
Output vector

Output nodes

Hidden nodes

Input nodes

Input vector:  $x_i$



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

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

$$w_{ij} = w_{ij} + (l) Err_j O_i$$

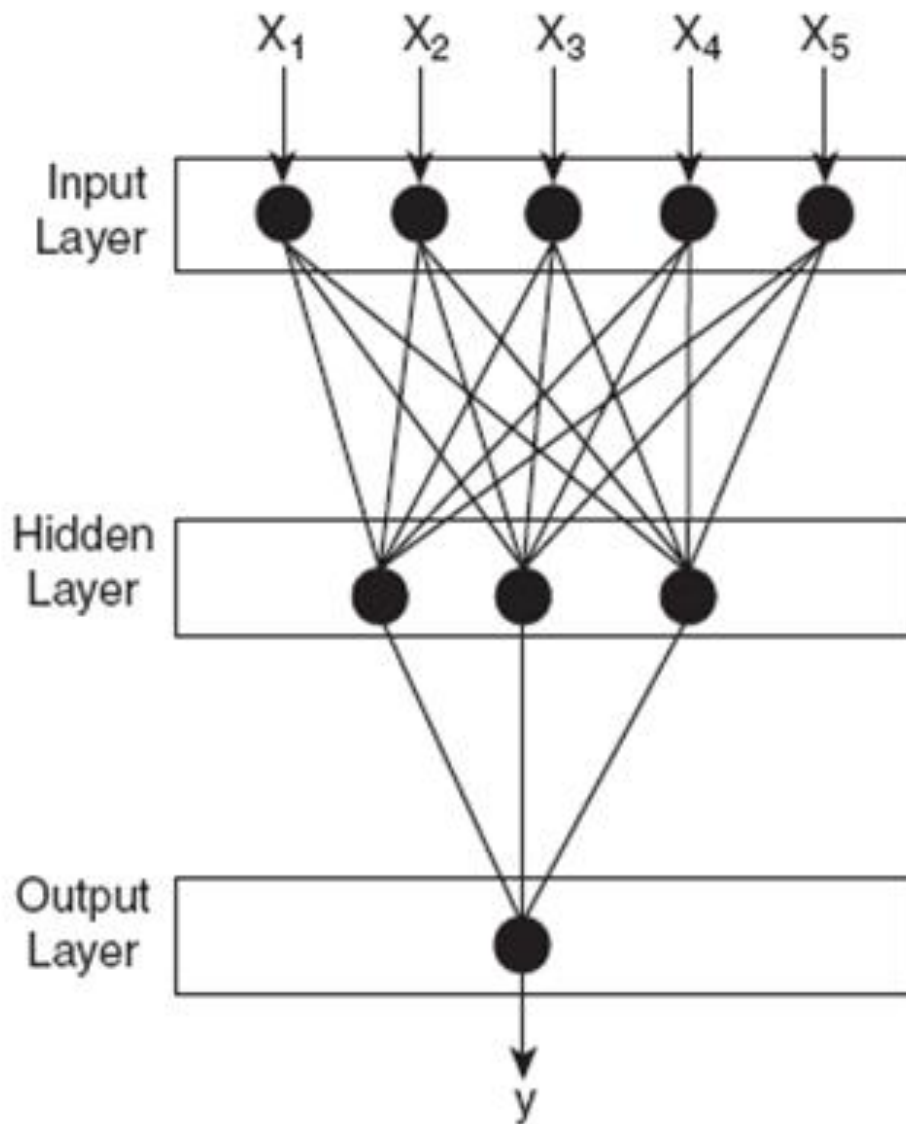
$$Err_j = O_j(1 - O_j)(T_j - O_j)$$

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

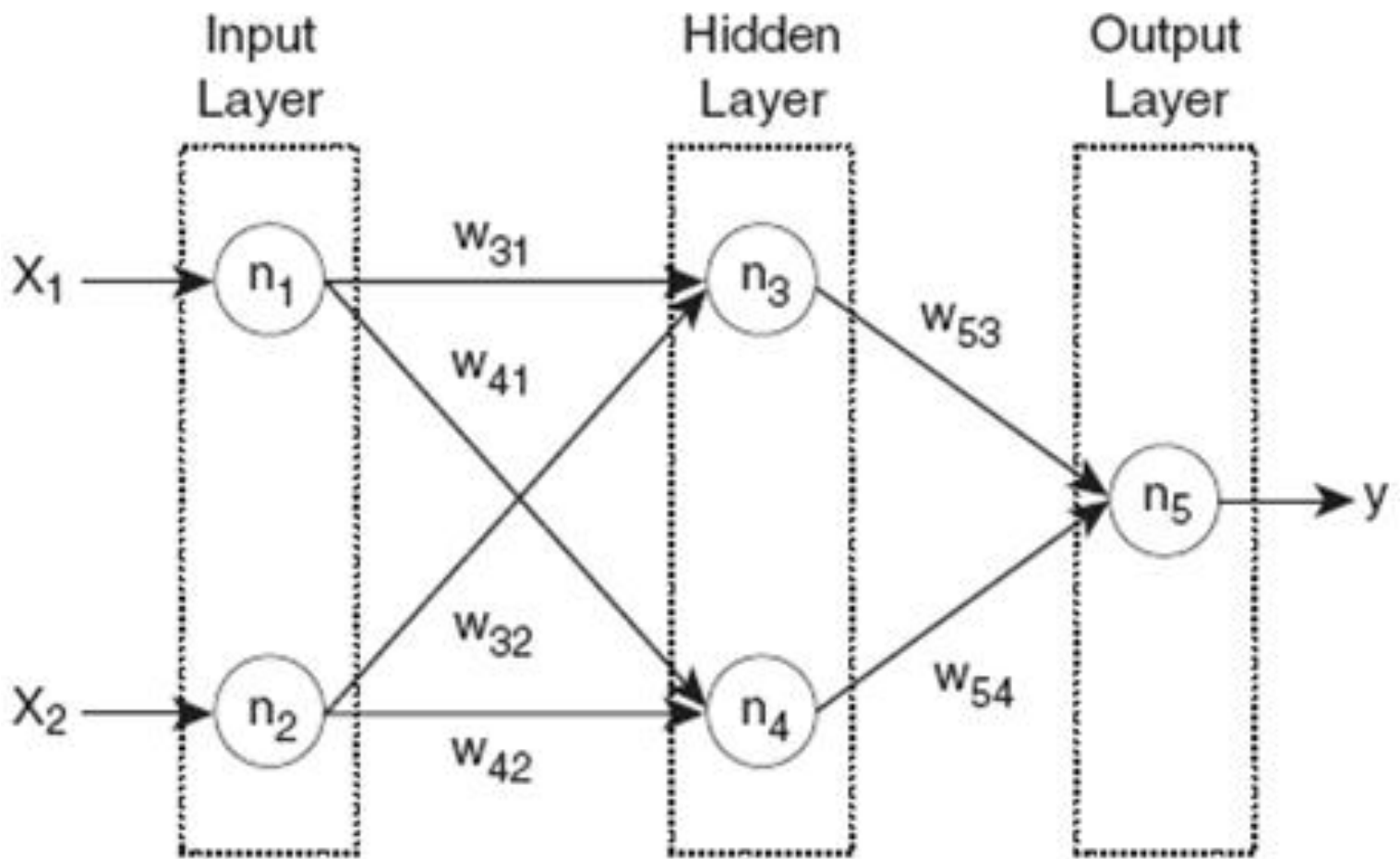
$$I_j = \sum_i w_{ij} O_i + \theta_j$$

# How A Multi-Layer Neural Network Works?

- ❖ The **inputs** to the network correspond to the attributes measured for each training tuple
- ❖ Inputs are fed simultaneously into the units making up the **input layer**
- ❖ They are then weighted and fed simultaneously to a **hidden layer**
- ❖ The number of hidden layers is arbitrary, although usually only one
- ❖ The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction
- ❖ The network is **feed-forward** in that none of the weights cycles back to an input unit or to an output unit of a previous layer
- ❖ From a statistical point of view, networks perform **nonlinear regression**: Given enough hidden units and enough training samples, they can closely approximate any function



Example of a multilayer feed-forward artificial neural network (ANN).



A two-layer, feed-forward neural network



# Advantages of Neural Network

- prediction accuracy is generally high
- robust, works when training examples contain errors
- output may be discrete, real-valued, or a vector of several discrete or real-valued attributes
- fast evaluation of the learned target function
- High tolerance to noisy data
- Ability to classify untrained patterns
- Well-suited for continuous-valued inputs and outputs
- Successful on a wide array of real-world data
- Algorithms are inherently parallel
- Techniques have recently been developed for the extraction of rules from trained neural networks

# Disadvantages of Neural Network

- long training time
- difficult to understand the learned function (weights)
- not easy to incorporate domain knowledge
- Require a number of parameters typically best determined empirically, e.g., the network topology or ``structure."
- *Poor interpretability*: Difficult to interpret the symbolic meaning behind the learned weights and of ``hidden units" in the network

# Association Rule

- ❖ Proposed by **Agrawal** et al in 1993.
- ❖ It is an important data mining model studied extensively by the database and data mining community.
- ❖ Assume all data are categorical.
- ❖ No good algorithm for numeric data.
- ❖ Initially used for **Market Basket Analysis** to find how items purchased by customers are related.
- ❖ Given a set of records each of which contain some number of items from a given collection;
  - Produce dependency rules which will predict occurrence of an item based on occurrences of other items.

<i>TID</i>	<i>Items</i>
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Rules Discovered:

**{Milk} --> {Coke}**

**{Diaper, Milk} --> {Beer}**

# Applications:

Basket data analysis, cross-marketing, catalog design, loss-leader analysis, clustering, classification, etc.

E.g., *98% of people who purchase tires and auto accessories also get automotive services done*

# Concepts:

**An item:** an item/article in a basket

**I:** the set of all items sold in the store

**A transaction:** items purchased in a basket; it may have TID (transaction ID)

**A transactional dataset:** A set of transactions

# The model: rules

A transaction **t** contains **X**, a set of items (**itemset**) in **I**, if  $X \subseteq t$ .

An **association rule** is an implication of the form:

$$X \rightarrow Y, \text{ where } X, Y \subset I, \text{ and } X \cap Y = \emptyset$$

An **itemset** is a set of items.

E.g.,  $X = \{\text{milk, bread, cereal}\}$  is an itemset.

A **k-itemset** is an itemset with **k** items.

E.g.,  $\{\text{milk, bread, cereal}\}$  is a 3-itemset

# Rule Strength Measures

## Support:

The rule holds with **support**  $sup$  in  $T$  (the transaction data set) if  $sup\%$  of transactions contain  $X \cup Y$ .

$$sup = \Pr(X \cup Y)$$

## Confidence:

The rule holds in  $T$  with **confidence**  $conf$  if  $conf\%$  of transactions that contain  $X$  also contain  $Y$ .

$$conf = \Pr(Y \mid X)$$

An association rule is a pattern that states when  $X$  occurs,  $Y$  occurs with certain probability.

## Support and Confidence

- ❖ *support* of  $X$  in  $D$  is  $\text{count}(X)/|D|$
- ❖ For an association rule  $X \Rightarrow Y$ , we can calculate
  - $\text{support}(X \Rightarrow Y) = \text{support}(XY)$
  - $\text{confidence}(X \Rightarrow Y) = \text{support}(XY)/\text{support}(X)$
- ❖ Relate Support (S) and Confidence (C) to Joint and Conditional probabilities
- ❖ There could be exponentially many A-rules
- ❖ Interesting association rules are (for now) those whose S and C are greater than minSup and minConf (some thresholds set by data miners)

# Support and Confidence

## Support count:

The support count of an itemset  $X$ , denoted by  $X.count$ , in a data set  $T$  is the number of transactions in  $T$  that contain  $X$ . Assume  $T$  has  $n$  transactions. Then,

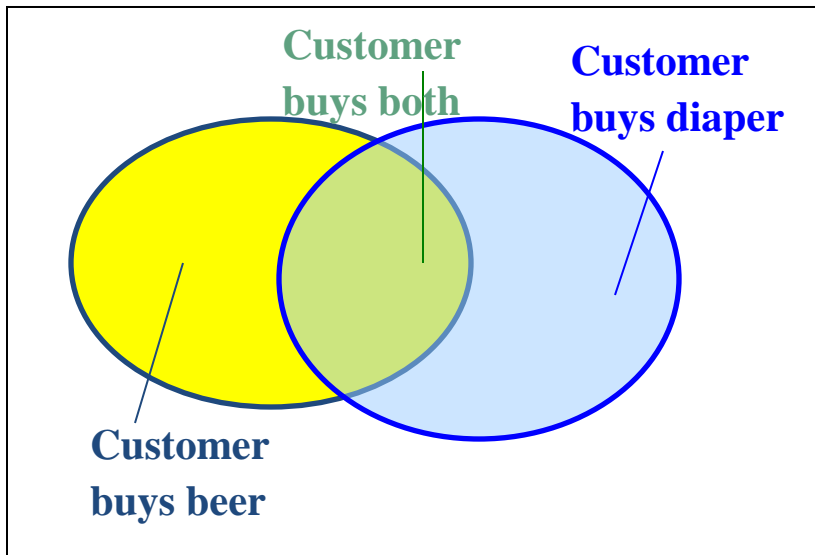
$$support = \frac{(X \cup Y).count}{n}$$

$$confidence = \frac{(X \cup Y).count}{X.count}$$

# Basic Concepts: Association Rules

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F

- ❖ Itemset  $X = \{x_1, \dots, x_k\}$
- ❖ Find all the rules  $X \rightarrow Y$  with min confidence and support
  - **support**,  $s$ , **probability** that a transaction contains  $X \cup Y$
  - **confidence**,  $c$ , **conditional probability** that a transaction having  $X$  also contains  $Y$ .



*Let minimum support 50%, and minimum confidence 50%, we have*

$A \rightarrow C$  (50%, 66.7%)

$C \rightarrow A$  (50%, 100%)

# Example

## Data set $D$

TID	Itemsets
T100	1 3 4
T200	2 3 5
T300	1 2 3 5
T400	2 5

*Count, Support, Confidence:*

$Count(13)=2$

$|D| = 4$

$Support(13)=0.5$

$Support(3 \rightarrow 2)=0.5$

$Confidence(3 \rightarrow 2)=0.67$



# Mining Association Rules: Example

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F

Min. support 50%  
Min. confidence 50%

Frequent pattern	Support
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

For rule  $A \Rightarrow C$ :

support =  $\text{support}(\{A\} \cup \{C\}) = 50\%$

confidence =  $\text{support}(\{A\} \cup \{C\}) / \text{support}(\{A\}) = 66.6\%$

The **Apriori** principle:

Any subset of a frequent itemset must be frequent

# Example of Association Rule

TID	Items
T1	bread, jelly, peanut-butter
T2	bread, peanut-butter
T3	bread, milk, peanut-butter
T4	beer, bread
T5	beer, milk

## Examples:

bread  $\Rightarrow$  peanut-butter

beer  $\Rightarrow$  bread

**Frequent itemsets:** Items that frequently appear together

$I = \{\text{bread, peanut-butter}\}$

$I = \{\text{beer, bread}\}$

**Support count ( $\sigma$ ):** Frequency of occurrence of an itemset

$$\sigma(\{\text{bread, peanut-butter}\}) = 3$$

$$\sigma(\{\text{beer, bread}\}) = 1$$

**Support:** Fraction of transactions that contain an itemset

$$s(\{\text{bread, peanut-butter}\}) = 3/5$$

$$s(\{\text{beer, bread}\}) = 1/5$$

**Frequent itemset:** An itemset whose support is greater than or equal to a minimum support threshold (**minsup**)

# What's an Interesting Rule?

An association rule is an implication of two itemsets:

$$X \Rightarrow Y$$

Many measures of interest. The two most used are:

**Support (s):** The occurring frequency of the rule, i.e., number of transactions that contain both X and Y

$$s = \frac{\sigma(X \cup Y)}{\# \text{ of trans.}}$$

**Confidence (c):** The strength of the association, i.e. measures of how often items (X)

$$c = \frac{\sigma(X \cup Y)}{\sigma(X)}$$

TID	s	c
bread $\Rightarrow$ peanut-butter	0.60	0.75
peanut-butter $\Rightarrow$ bread	0.60	1.00
beer $\Rightarrow$ bread	0.20	0.50
peanut-butter $\Rightarrow$ jelly	0.20	0.33
jelly $\Rightarrow$ peanut-butter	0.20	1.00
jelly $\Rightarrow$ milk	0.00	0.00

# Mining Association Rules:

What We Need to Know

- ❖ Goal: Rules with high support/confidence

- ❖ How to compute?

  - Support: Find sets of items that occur frequently

  - Confidence: Find frequency of subsets of supported itemsets

*If we have all frequently occurring sets of items (frequent itemsets), we can compute support and confidence!*

# The Apriori Algorithm

## Pseudo-code:

$C_k$ : Candidate itemset of size  $k$

$L_k$ : frequent itemset of size  $k$

$L_1 = \{\text{frequent items}\};$

**for** ( $k = 1; L_k \neq \emptyset; k++$ ) **do begin**

$C_{k+1}$  = candidates generated from  $L_k$ ;

**for each** transaction  $t$  in database **do**

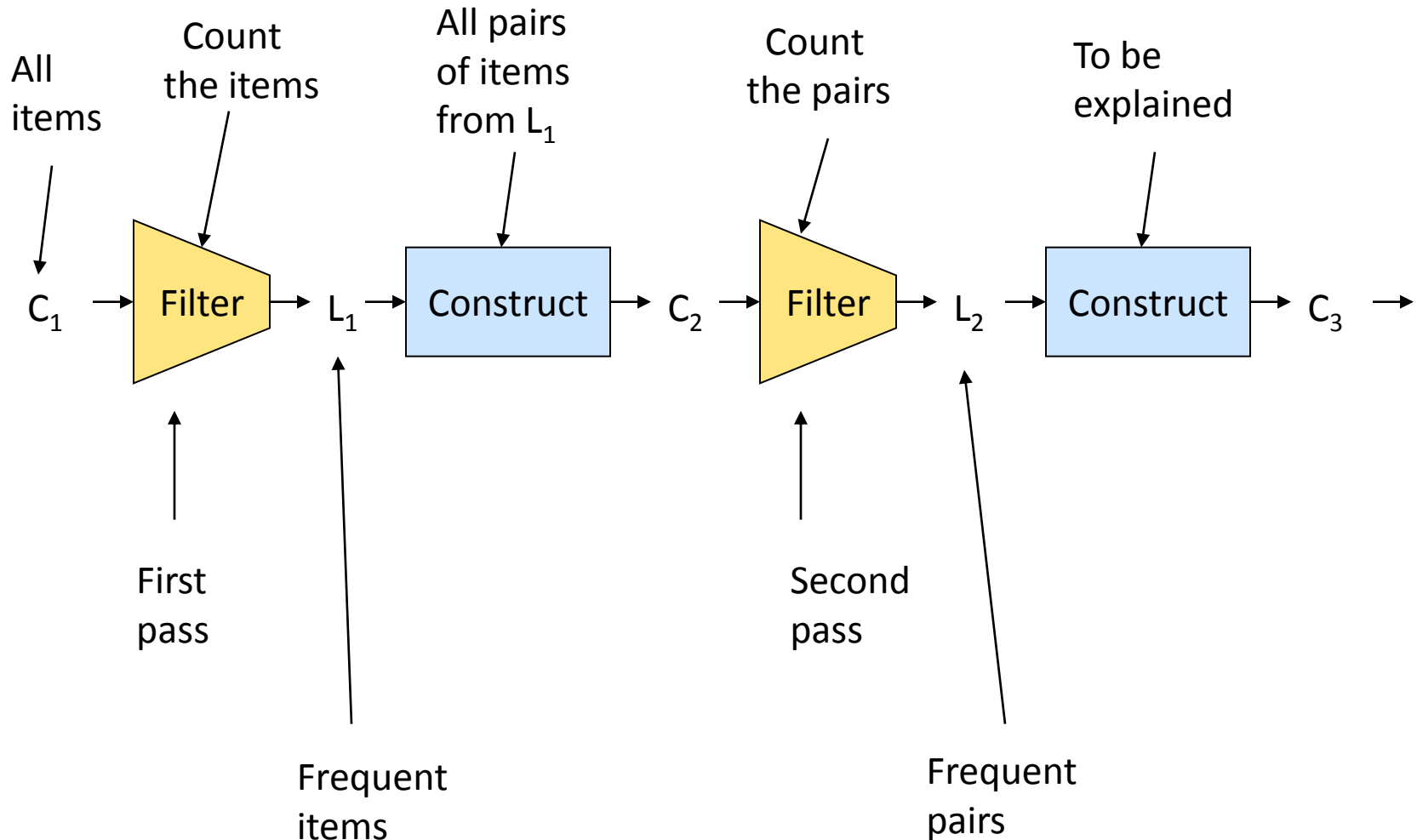
        increment the count of all candidates in  $C_{k+1}$   
        that are contained in  $t$

$L_{k+1}$  = candidates in  $C_{k+1}$  with min\_support

**end**

**return**  $\cup_k L_k$ ;

# A-Priori Algorithm (in nutshell)





# The Apriori Algorithm—An Example

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

Min\_sup=2

1<sup>st</sup> scan

1-candidates

$C_1$

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

Freq 1-itemsets

$L_1$

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

Freq 2-itemsets

$L_2$

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

Counting

$C_2$

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

2<sup>nd</sup> scan

2-candidates

$C_2$

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

3-candidates

$C_3$

Itemset
{B, C, E}

3<sup>rd</sup> scan

Freq 3-itemsets

$L_3$

Itemset	sup
{B, C, E}	2

# Clustering and Cluster Analysis

A **cluster** is a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters.

**Clustering** is “the process of organizing objects into groups whose members are similar in some way”.

**“Cluster Analysis** is a set of methods for constructing a (hopefully) sensible and informative classification of an initially unclassified set of data, using the variable values observed on each individual.”

- B. S. Everitt (1998), *“The Cambridge Dictionary of Statistics”*

# Applications of Cluster Analysis

- ❖ Pattern Recognition

- ❖ Spatial Data Analysis

- Create thematic maps in GIS by clustering feature spaces
- Detect spatial clusters or for other spatial mining tasks

- ❖ Image Processing

- ❖ Economic Science (especially market research)

- ❖ WWW

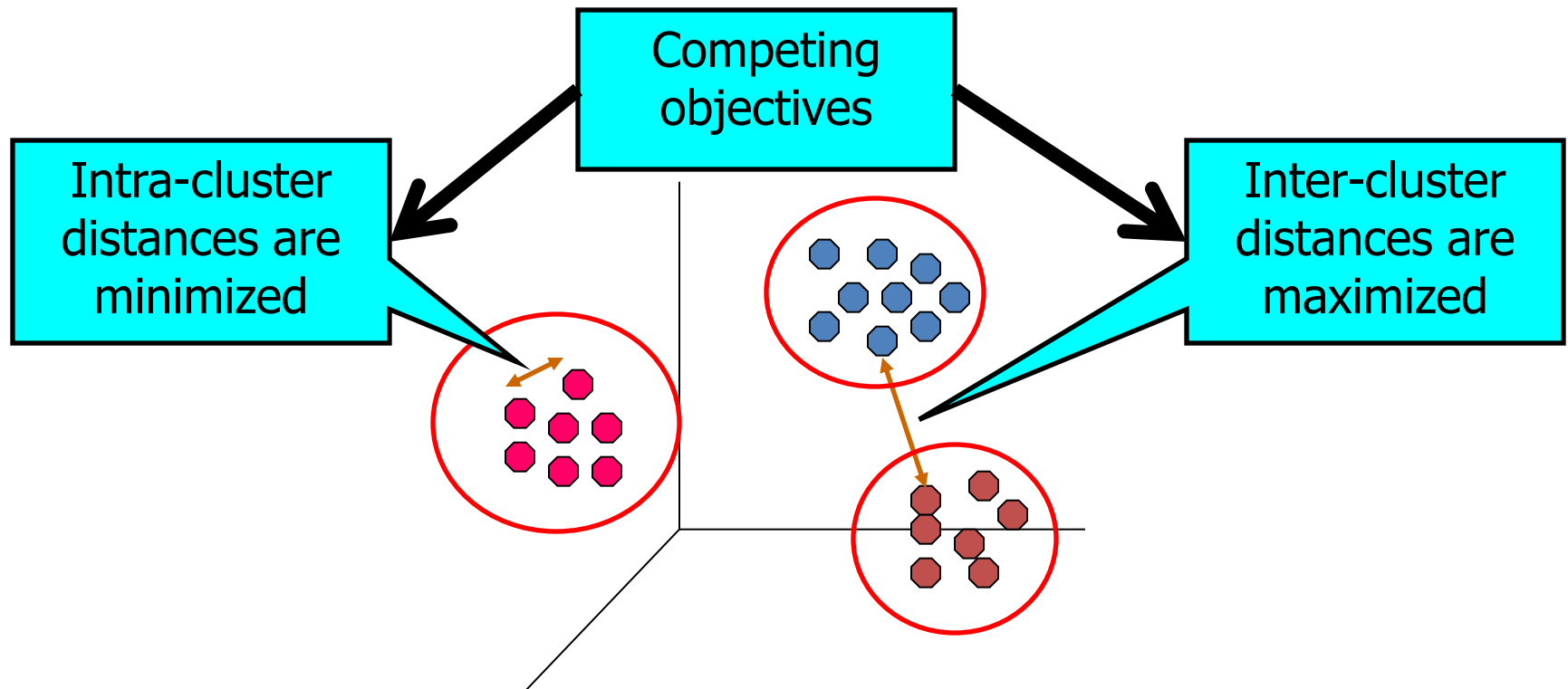
- Document classification
- Cluster Weblog data to discover groups of similar access patterns

# Applications of Cluster Analysis

- ❖ Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- ❖ Land use: Identification of areas of similar land use in an earth observation database
- ❖ Insurance: Identifying groups of motor insurance policy holders with a high average claim cost
- ❖ City-planning: Identifying groups of houses according to their house type, value, and geographical location
- ❖ Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults

# Objectives of Cluster Analysis

Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



# Types of Clusterings

## ❖ Partitioning Clustering

- A division data objects into non-overlapping subsets (clusters) such that each data object is in exactly one subset
- Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
- Typical methods: k-means, k-medoids, CLARA (Clustering LARge Applications)

## ❖ Hierarchical clustering

- A set of nested clusters organized as a hierarchical tree
- Create a hierarchical decomposition of the set of data (or objects) using some criterion
- Typical methods: DiAna (Divisive Analysis), AgNes (Agglomerative Nesting), BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies), ROCK (RObust Clustering using linKs), CAMELEON

## ❖ Density-based Clustering

- Based on connectivity and density functions
- Typical methods: DBSACN (Density Based Spatial Clustering of Applications with Noise), OPTICS (Ordering Points To Identify the Clustering Structure), DenClue (DENSity-based CLUstEring )

## ❖ Grid-based Clustering

- based on a multiple-level granularity structure
- Typical methods: STING (STatistical INformation Grid ), WaveCluster, CLIQUE (Clustering In QUEst)

## ❖ Model-based Clustering

- A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
- Typical methods: EM (Expectation Maximization), SOM (Self-Organizing Map), COBWEB

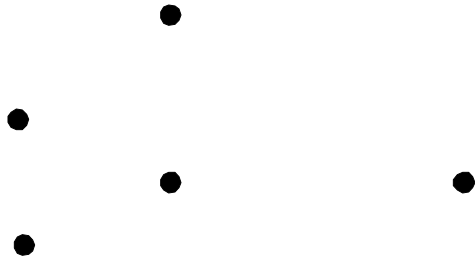
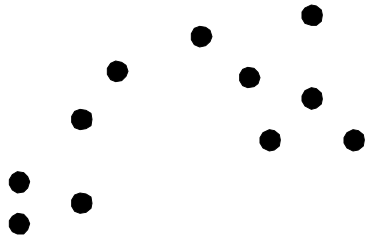
## ❖ Frequent pattern-based Clustering

- Based on the analysis of frequent patterns
- Typical methods: pCluster

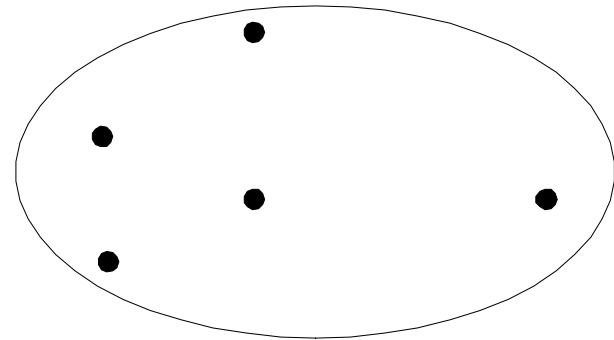
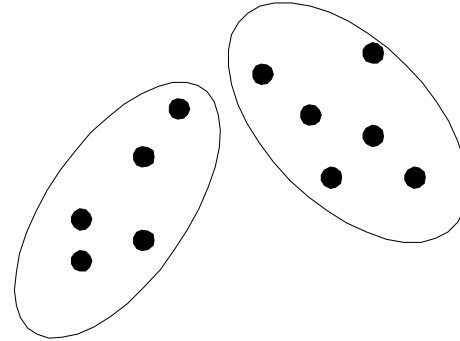
## ❖ User-guided or constraint-based Clustering

- Clustering by considering user-specified or application-specific constraints
- Typical methods: COD, constrained clustering

# Partitioning Clustering



**Original Points**

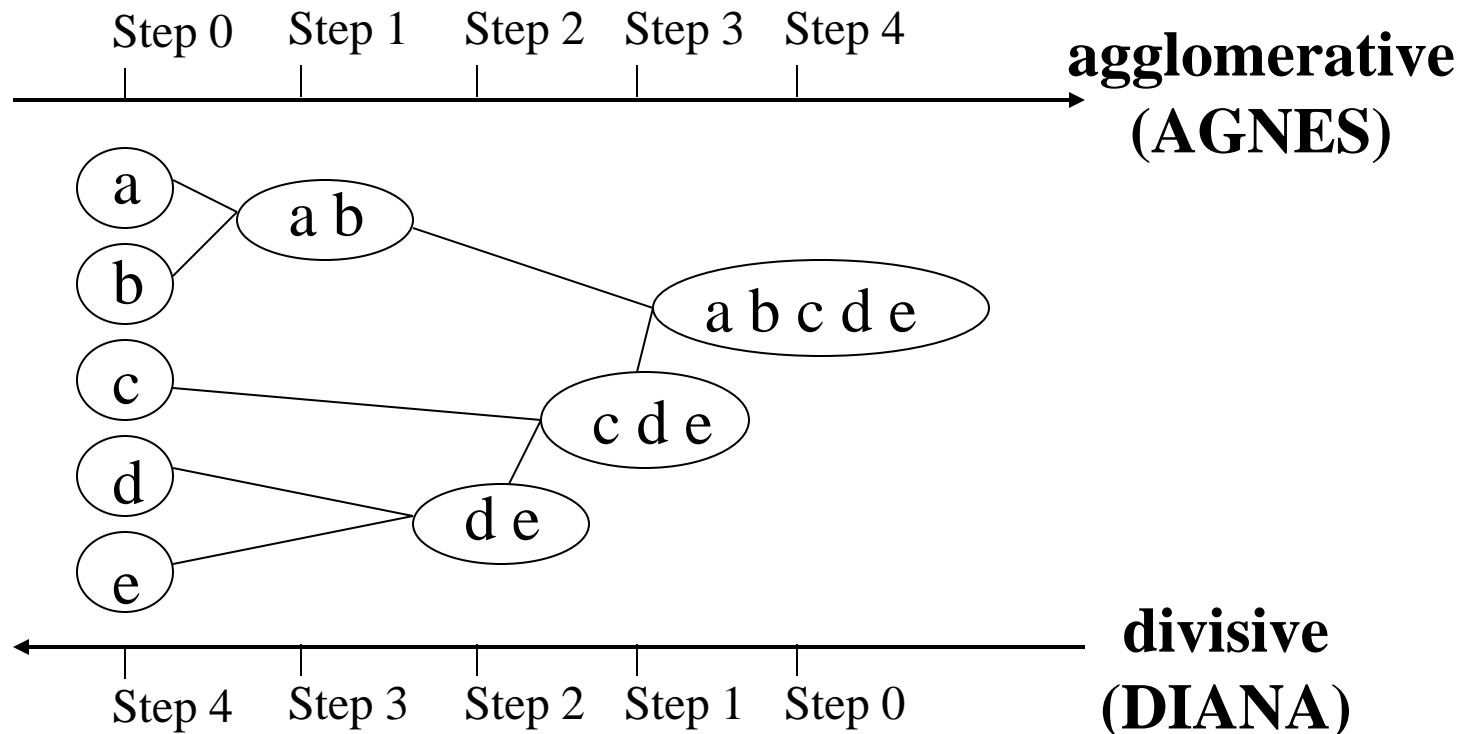


**A Partitional Clustering**



# Hierarchical Clustering

Use distance matrix as clustering criteria. This method does not require the number of clusters  $k$  as an input, but needs a termination condition



## **Agglomerative (bottom up)**

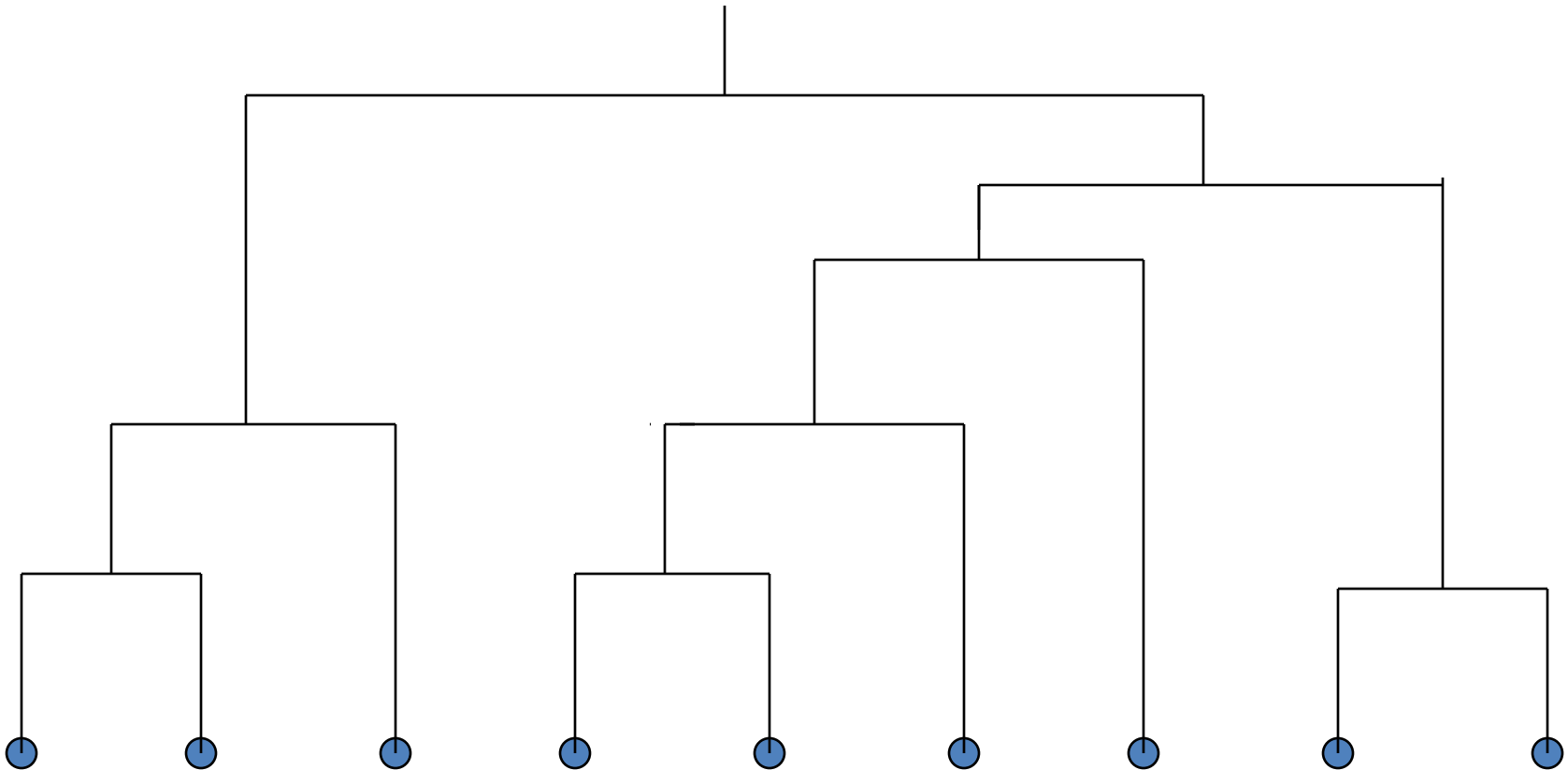
1. start with 1 point (singleton)
2. recursively add two or more appropriate clusters
3. Stop when k number of clusters is achieved.

## **Divisive (top down)**

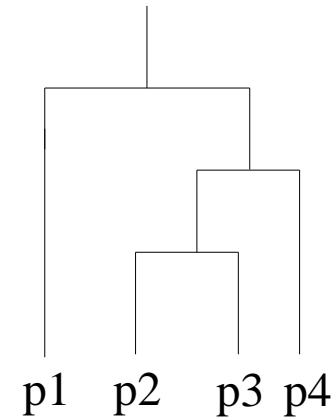
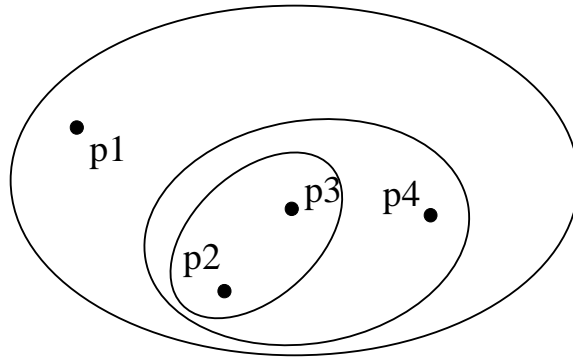
1. Start with a big cluster
2. Recursively divide into smaller clusters
3. Stop when k number of clusters is achieved.

# Dendrogram

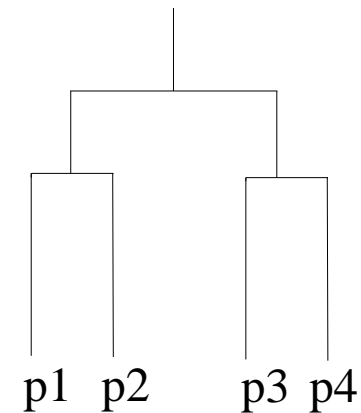
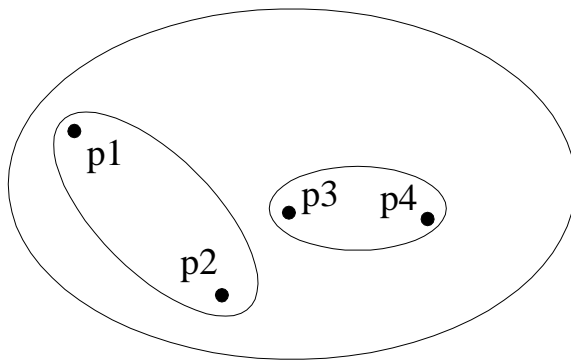
- ❖ Decompose data objects into a several levels of nested partitioning (tree of clusters), called a dendrogram.
- ❖ A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.



# Hierarchical Clustering



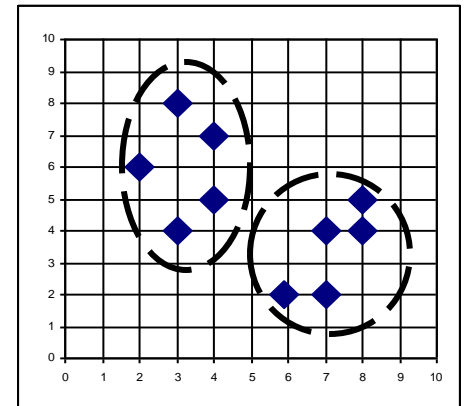
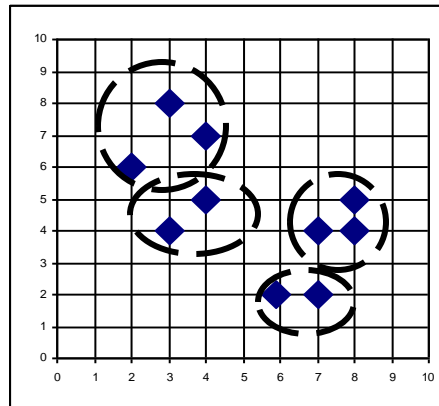
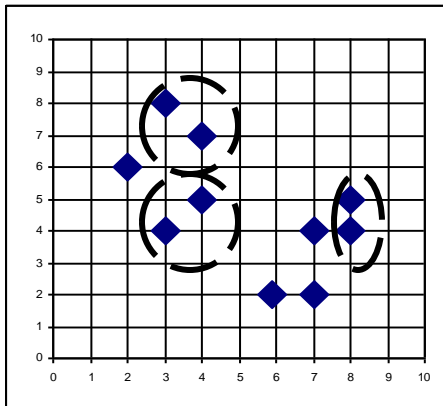
**Dendrogram 1**



**Dendrogram 2**

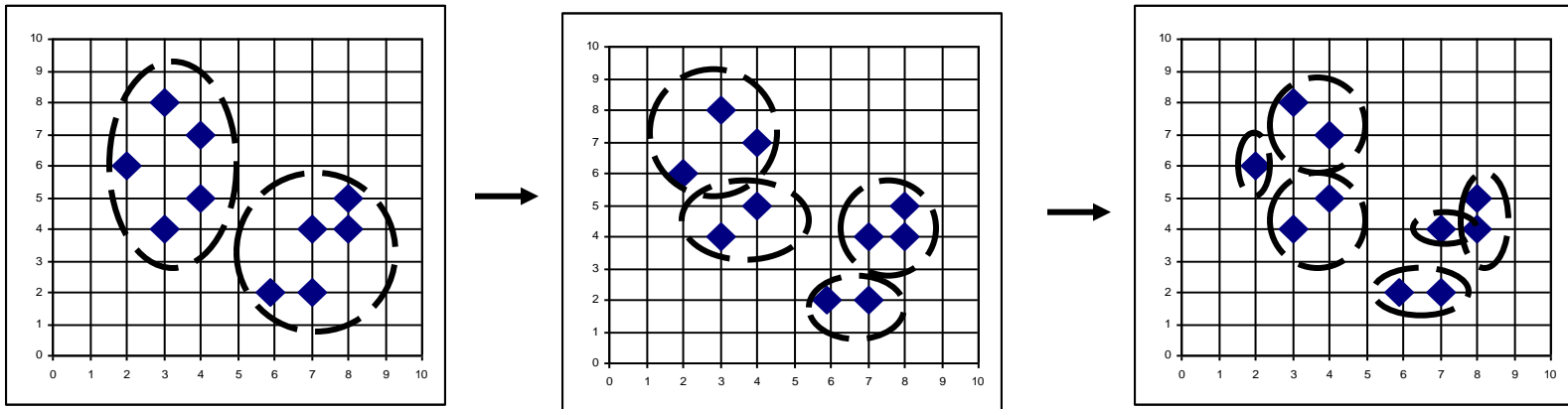
# AGNES (Agglomerative Nesting)

- ❖ Introduced in Kaufmann and Rousseeuw (1990)
- ❖ Implemented in statistical analysis packages, e.g., Splus
- ❖ Use the Single-Link method and the dissimilarity matrix.
- ❖ Merge nodes that have the least dissimilarity
- ❖ Go on in a non-descending fashion
- ❖ Eventually all nodes belong to the same cluster



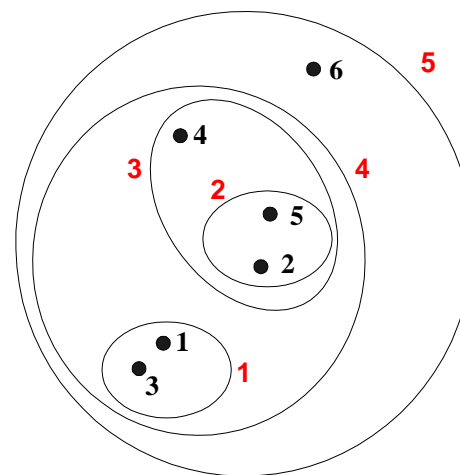
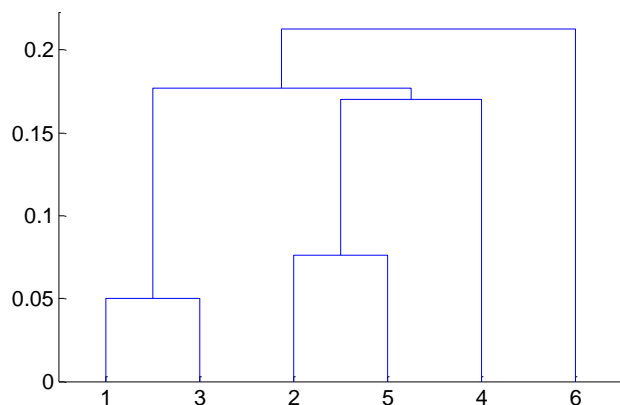
# DIANA (Divisive Analysis)

- ❖ Introduced in Kaufmann and Rousseeuw (1990)
- ❖ Implemented in statistical analysis packages, e.g., Splus
- ❖ Inverse order of AGNES
- ❖ Eventually each node forms a cluster on its own



# Hierarchical Clustering

- ❖ Produces a set of nested clusters organized as a hierarchical tree
- ❖ Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits



# Strengths of Hierarchical Clustering

- ❖ Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by 'cutting' the dendrogram at the proper level
- ❖ They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, ...)



# Hierarchical Clustering

## ❖ Two main types of hierarchical clustering

### ➤ Agglomerative (bottom up):

- Start with the points as individual clusters
- At each step, merge the closest pair of clusters until only one cluster (or  $k$  clusters) left

### ➤ Divisive (top down):

- Start with one, all-inclusive cluster
- At each step, split a cluster until each cluster contains a point (or there are  $k$  clusters)

# Agglomerative Clustering Algorithm

More popular hierarchical clustering technique

Basic algorithm

- Compute the proximity matrix

- Let each data point be a cluster

- Repeat**

  - Merge the two closest clusters

  - Update the proximity matrix

- Until** only a single cluster remains

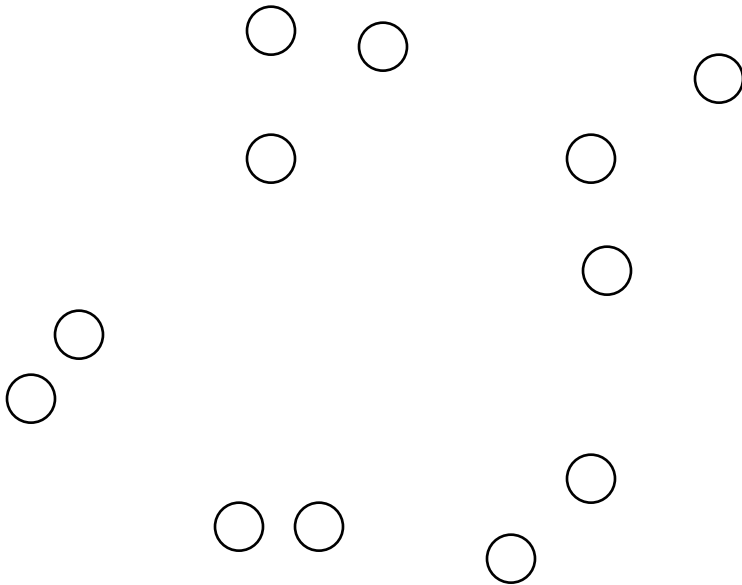
**Note:**

Key operation is the computation of the proximity of two clusters

- Different approaches to defining the distance between clusters distinguish the different algorithms

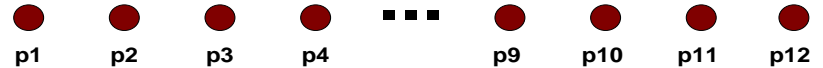
# Starting Situation

❖ Start with clusters of individual points and a proximity matrix



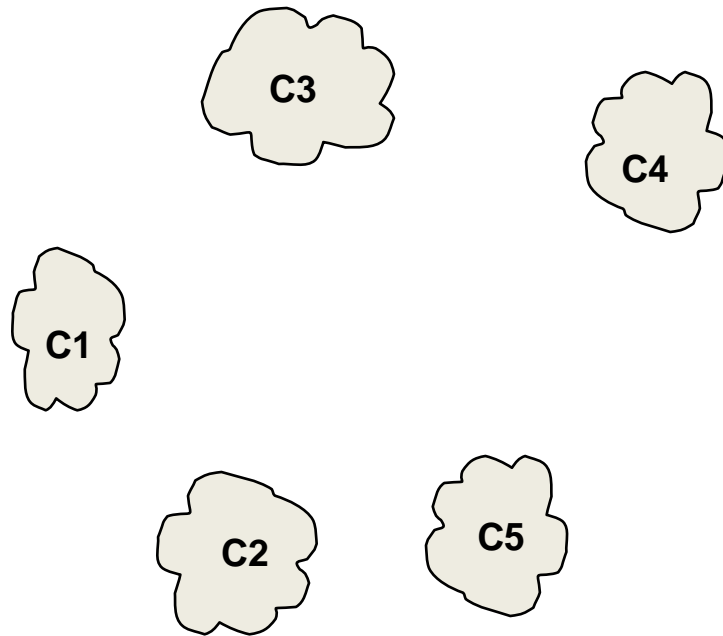
	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

**Proximity Matrix**



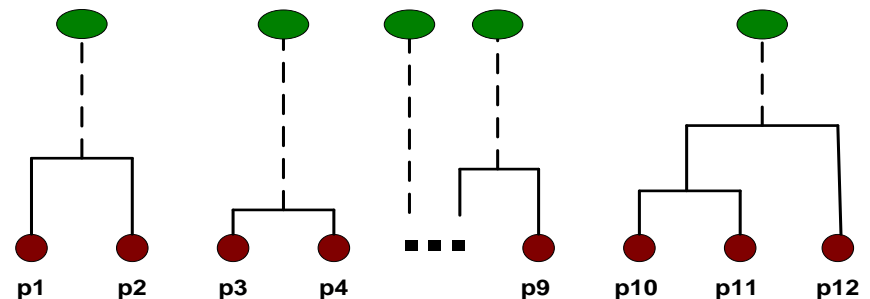
# Intermediate Situation

❖ After some merging steps, we have some clusters



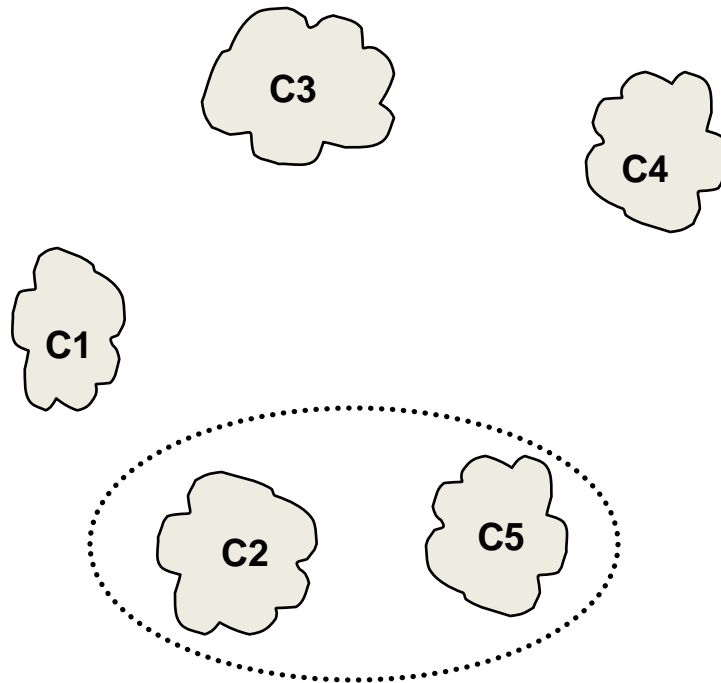
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

**Proximity Matrix**



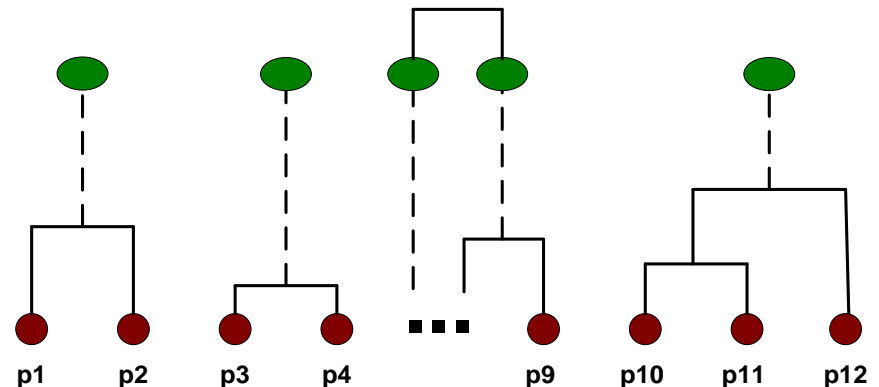
# Intermediate Situation

- ❖ We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



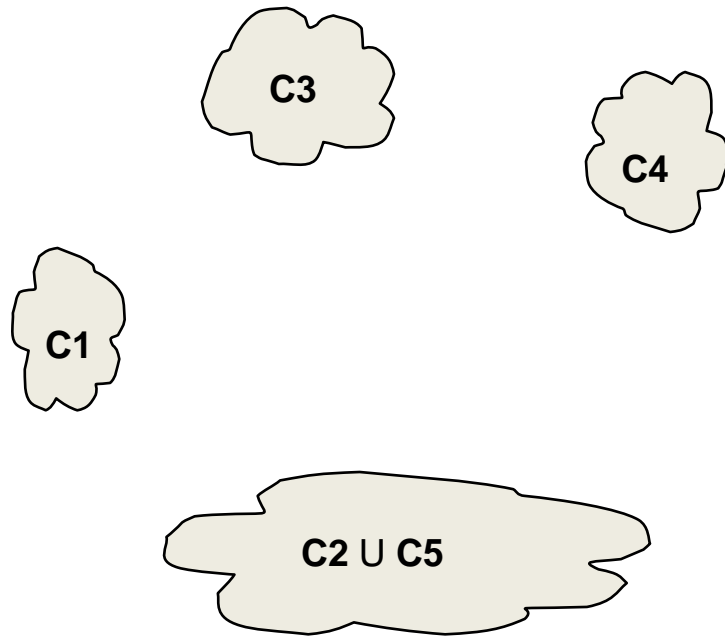
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

**Proximity Matrix**



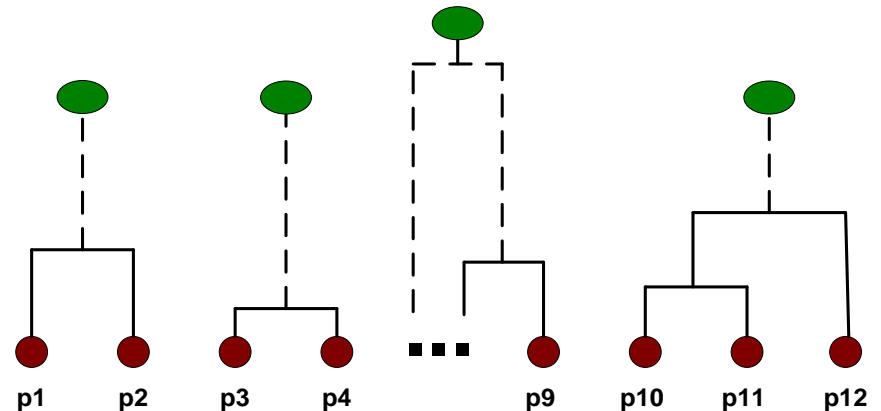
# After Merging

❖ The question is “How do we update the proximity matrix?”

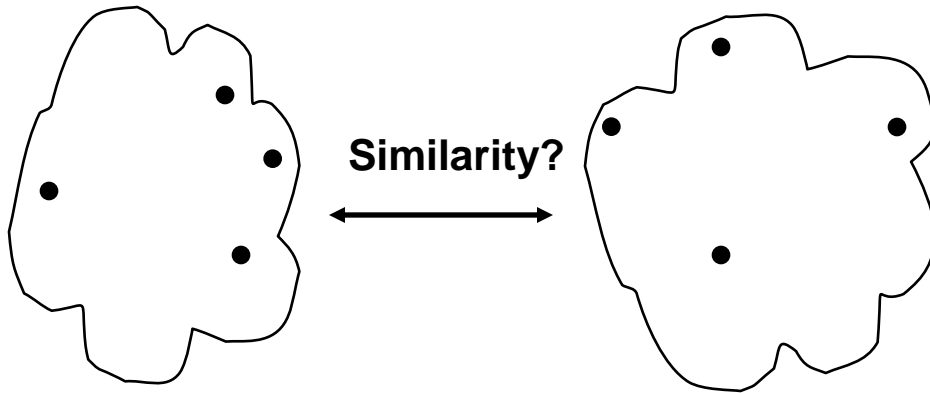


			<b>C2</b>		
			<b>U</b>		
		<b>C1</b>	<b>C5</b>	<b>C3</b>	<b>C4</b>
<b>C2 U</b>	<b>C1</b>		?		
	<b>C5</b>	?	?	?	?
	<b>C3</b>		?		
	<b>C4</b>		?		

## Proximity Matrix



# How to Define Inter-Cluster Similarity

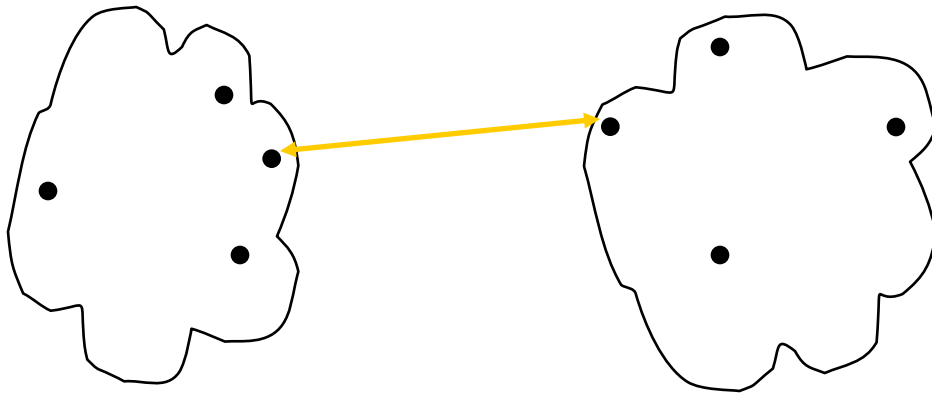


- ❖ MIN
- ❖ MAX
- ❖ Group Average
- ❖ Distance Between Centroids

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

**Proximity Matrix**

# How to Define Inter-Cluster Similarity



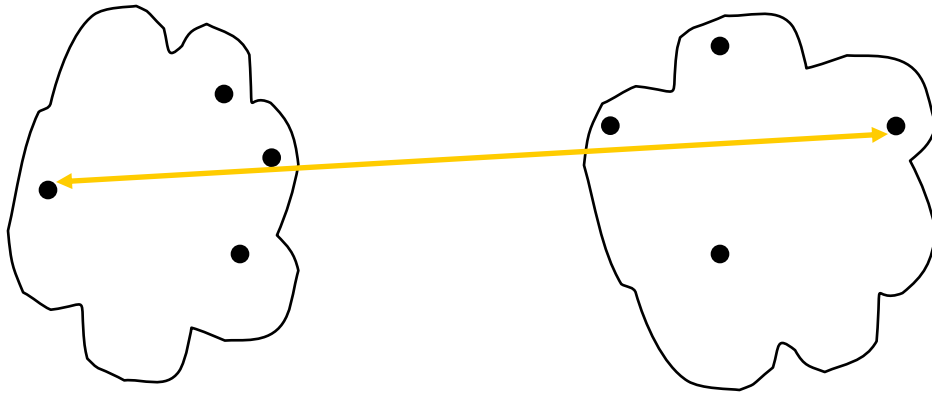
	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

**Proximity Matrix**

- ❖ **MIN**
- ❖ **MAX**
- ❖ **Group Average**
- ❖ **Distance Between Centroids**



# How to Define Inter-Cluster Similarity

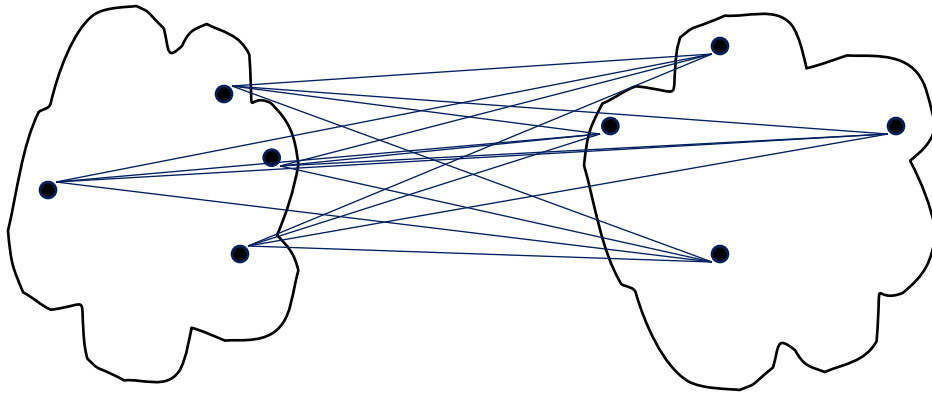


- ❖ MIN
- ❖ MAX
- ❖ Group Average
- ❖ Distance Between Centroids

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						

· **Proximity Matrix**

# How to Define Inter-Cluster Similarity

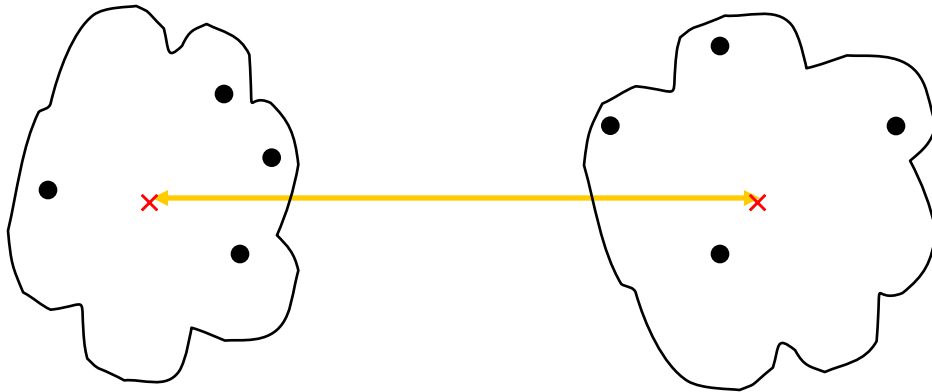


- ❖ MIN
- ❖ MAX
- ❖ **Group Average**
- ❖ Distance Between Centroids

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

**Proximity Matrix**

# How to Define Inter-Cluster Similarity



- ❖ MIN
- ❖ MAX
- ❖ Group Average
- ❖ Distance Between Centroids

	p1	p2	p3	p4	p5	...
p1						
p2						
p3						
p4						
p5						
.						

Proximity Matrix

# Hierarchical Clustering: Problems and Limitations

- ❖ Once a decision is made to combine two clusters, it cannot be undone
- ❖ Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Breaking large clusters (divisive)
- ❖ Dendrogram correspond to a given hierarchical clustering is not unique, since for each merge one needs to specify which subtree should go on the left and which on the right
- ❖ They impose structure on the data, instead of revealing structure in these data.

# K-means Algorithm

- ❖ Partitioning clustering approach
- ❖ Each cluster is associated with a **centroid** (center point or mean point)
- ❖ Each point is assigned to the cluster with the closest centroid
- ❖ Number of clusters,  $K$ , must be specified

The basic algorithm is very simple:

---

- 1: Select  $K$  points as the initial centroids.
  - 2: **repeat**
  - 3:   Form  $K$  clusters by assigning all points to the closest centroid.
  - 4:   Recompute the centroid of each cluster.
  - 5: **until** The centroids don't change
-

# The *k*-means partitioning algorithm.

**Algorithm: *k*-means.** The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

## **Input:**

$k$ : the number of clusters,

$D$ : a data set containing  $n$  objects.

**Output:** A set of  $k$  clusters.

## **Method:**

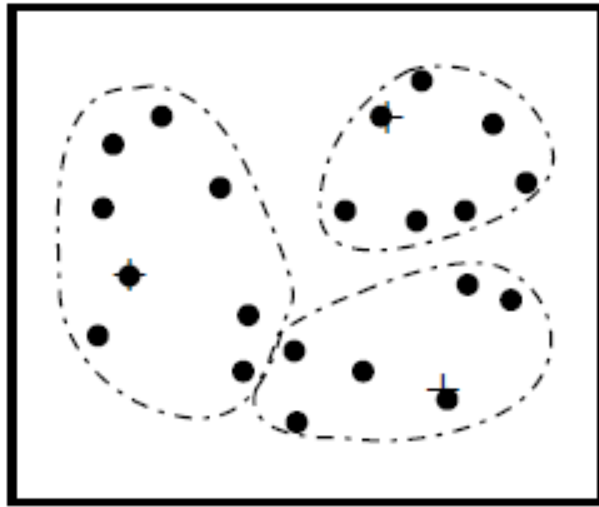
(1) arbitrarily choose  $k$  objects from  $D$  as the initial cluster centers;

(2) repeat

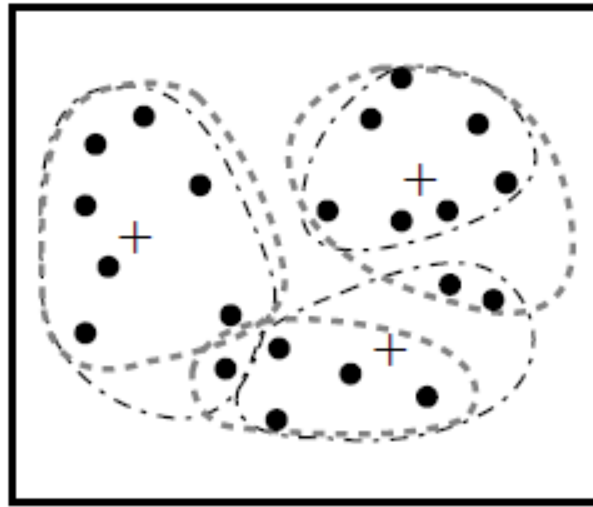
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;

(4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;

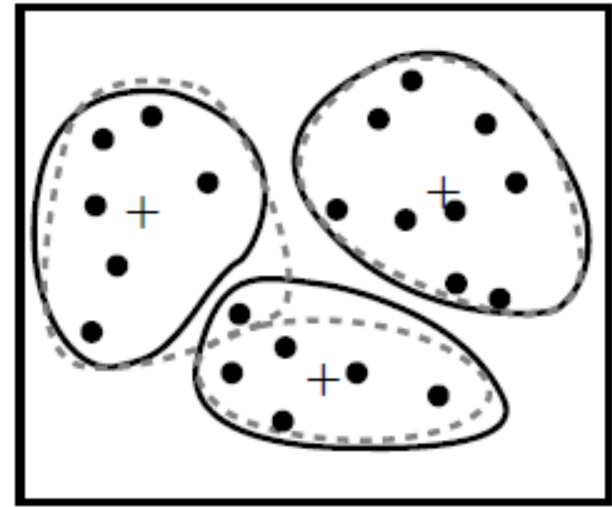
(5) until no change;



(a)



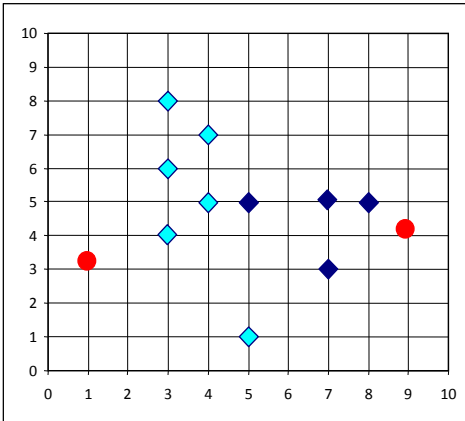
(b)



(c)

**Figure:** Clustering of a set of objects based on the  $k$ -means method. (The mean of each cluster is marked by a "+".)

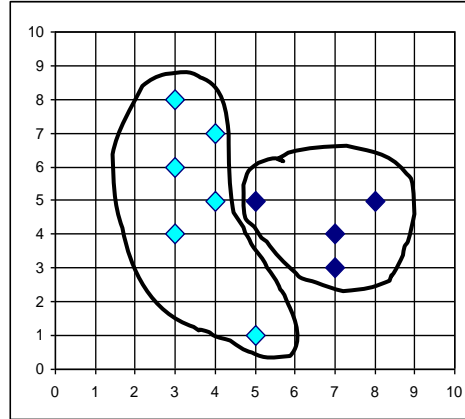
# Example



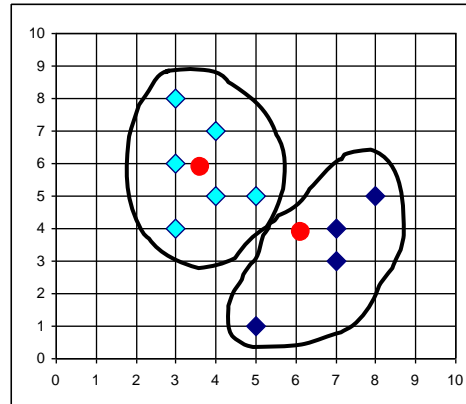
$K=2$

Arbitrarily choose  $K$  object as initial cluster center

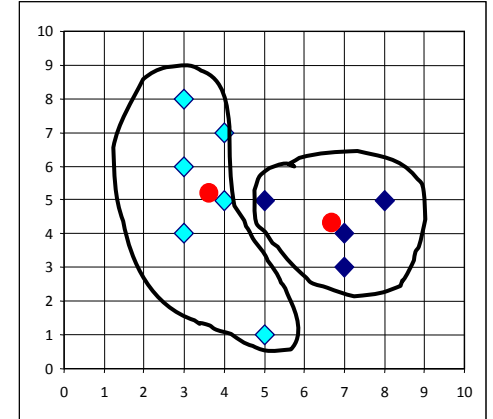
Assign each object to most similar center



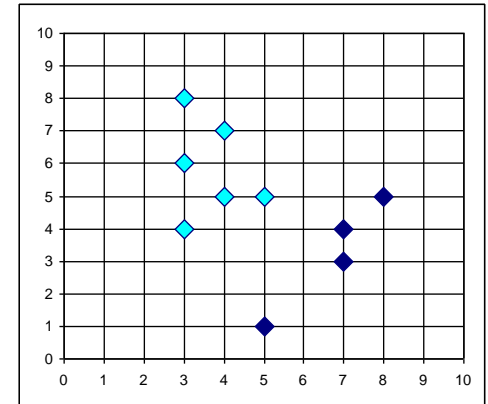
↑ reassign



Update the cluster means

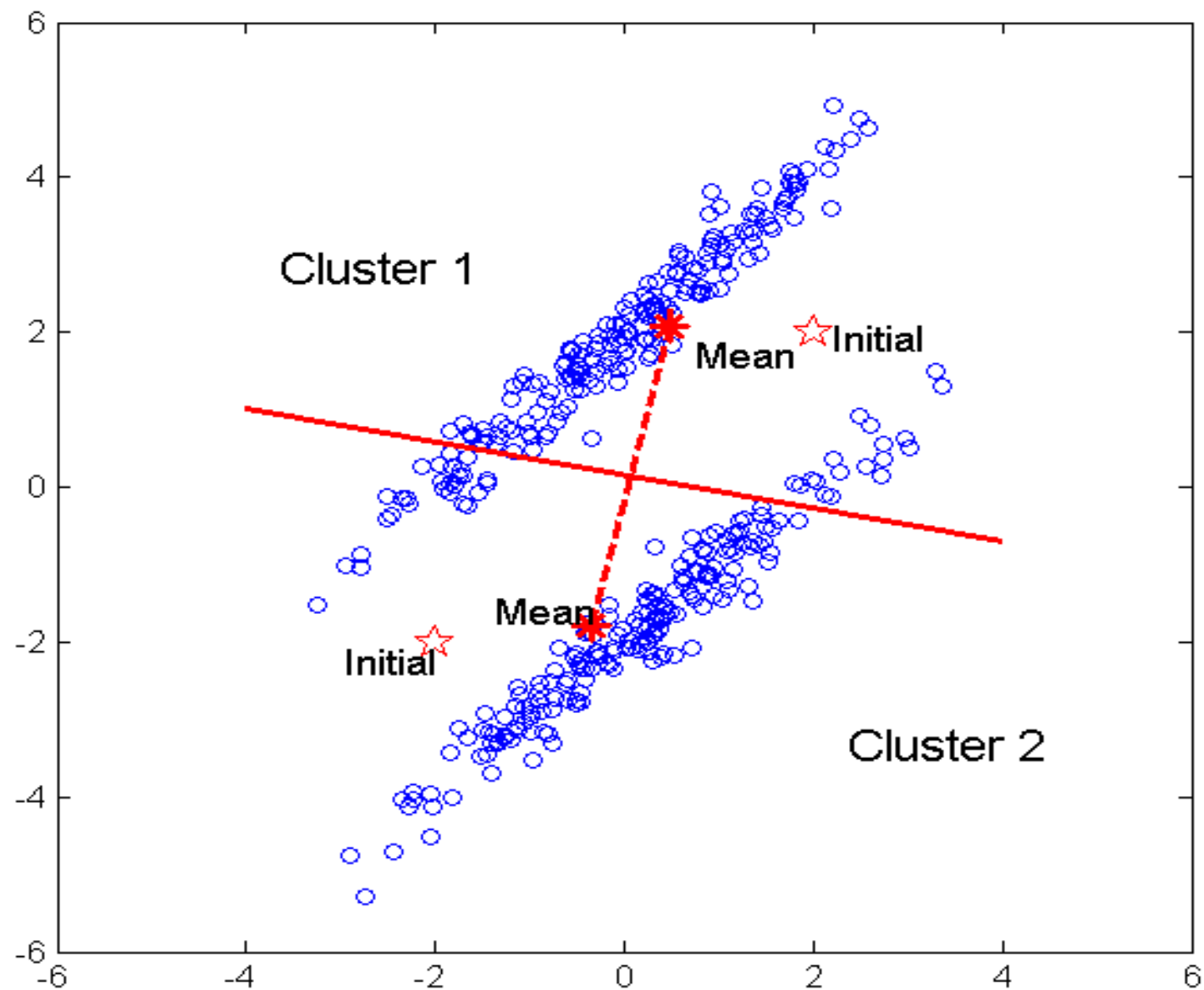


↓ reassign




Update the cluster means





# K-means Clustering – Details

- ❖ Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- ❖ The centroid is (typically) the mean of the points in the cluster.
- ❖ ‘Closeness’ is measured mostly by **Euclidean distance**, cosine similarity, correlation, etc.
- ❖ K-means will converge for common similarity measures mentioned above.
- ❖ Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to ‘Until relatively few points change clusters’
- ❖ Complexity is  $O(n * K * I * d)$ 
  - $n$  = number of points,  $K$  = number of clusters,
  - $I$  = number of iterations,  $d$  = number of attributes

# Issues and Limitations for K-means

- ❖ How to choose initial centers?
- ❖ How to choose K?
- ❖ How to handle Outliers?
- ❖ Clusters different in
  - Shape
  - Density
  - Size
- ❖ Assumes clusters are spherical in vector space
  - Sensitive to coordinate changes

# K-means Algorithm

## Pros

- ❖ Simple
- ❖ Fast for low dimensional data
- ❖ It can find pure sub clusters if large number of clusters is specified

## Cons

- ❖ K-Means cannot handle non-globular data of different sizes and densities
- ❖ K-Means will not identify outliers
- ❖ K-Means is restricted to data which has the notion of a center (centroid)
- ❖ Applicable only when *mean* is defined, then what about categorical data?
- ❖ Need to specify  $k$ , the *number* of clusters, in advance
- ❖ Unable to handle noisy data and *outliers*
- ❖ Not suitable to discover clusters with *non-convex shapes*

# Outliers

## What are outliers?

The set of objects are considerably dissimilar from the remainder of the data

Example: Sports: Michael Jordon, Randy Orton, Sachin Tendulkar ...

## Applications:

- Credit card fraud detection

- Telecom fraud detection

- Customer segmentation

- Medical analysis

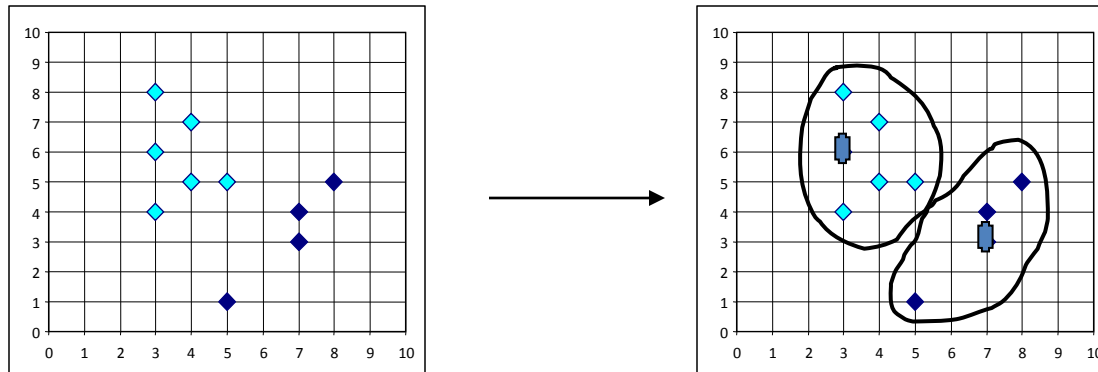
Outlier detection and analysis are very useful for fraud detection, etc. and can be performed by statistical, distance-based or deviation-based approaches

# How to handle Outliers?

❖ The k-means algorithm is sensitive to outliers !

- Since an object with an extremely large value may substantially distort the distribution of the data.

**K-Medoids:** Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster.



**Example:**

Use in finding Fraudulent usage of credit cards. Outlier Analysis may uncover Fraudulent usage of credit cards by detecting purchases of extremely large amounts for a given account number in comparison to regular charges incurred by the same account. Outlier values may also be detected with respect to the location and type of purchase or the purchase frequency.

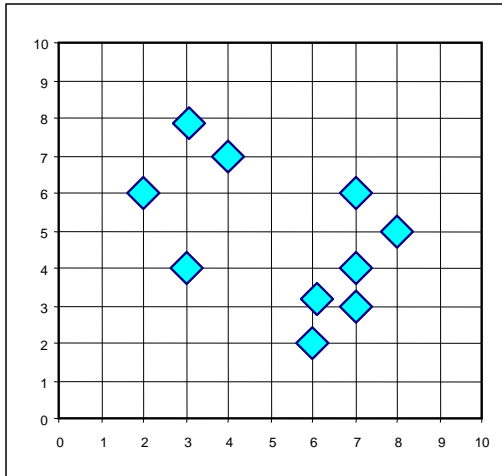
# The *K-Medoids* Clustering Method

- ❖ Find *representative* objects, called medoids, in clusters
- ❖ *PAM* (Partitioning Around Medoids, 1987)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - *PAM* works effectively for small data sets, but does not scale well for large data sets
- ❖ *CLARA* (Kaufmann & Rousseeuw, 1990)
- ❖ *CLARANS* (Ng & Han, 1994): Randomized sampling
- ❖ Focusing + spatial data structure (Ester et al., 1995)

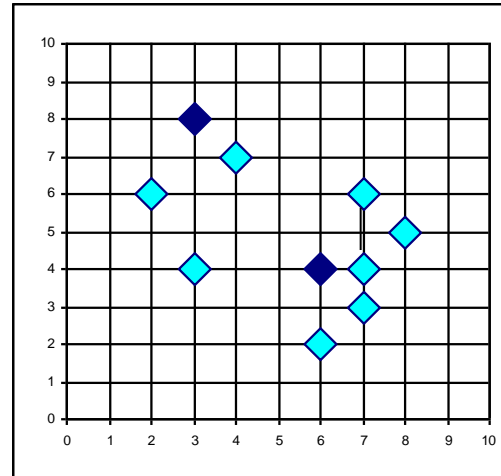


# A Typical K-Medoids Algorithm (PAM)

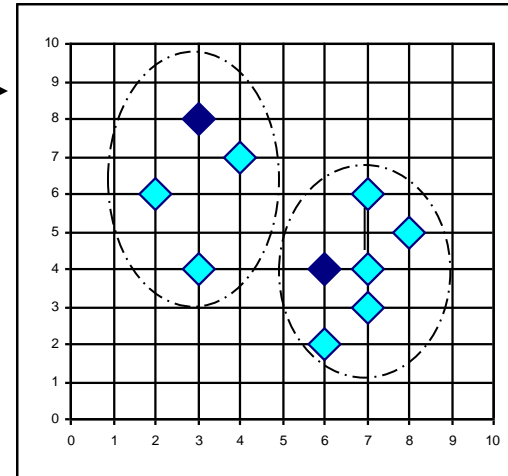
Total Cost = 20



Arbitrary  
choose  $k$   
object as  
initial  
medoids



Assign  
each remainin  
g object to  
nearest  
medoids



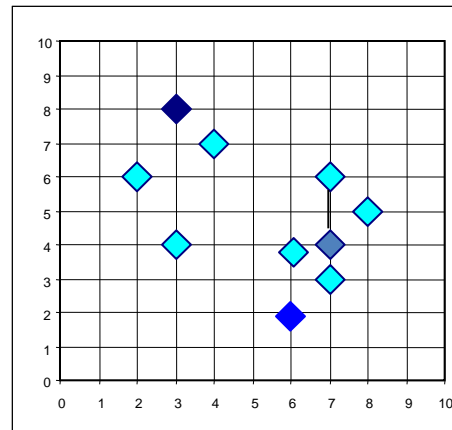
$K=2$

Do loop

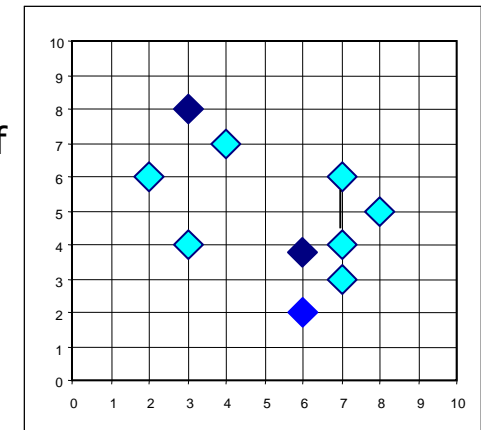
Until no change

Swapping  $O$   
and  $O_{\text{random}}$   
If quality is  
improved.

Total Cost = 26



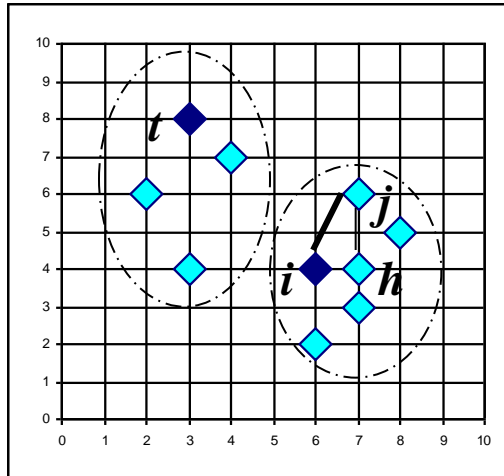
Compute  
total cost of  
swapping



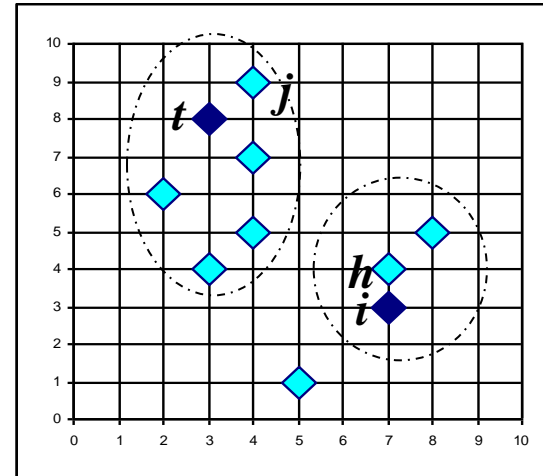
# PAM (Partitioning Around Medoids)

- ❖ PAM (Kaufman and Rousseeuw, 1987), built in Splus
- ❖ Use real object to represent the cluster
  - Select  $k$  representative objects arbitrarily
  - For each pair of non-selected object  $h$  and selected object  $i$ , calculate the total swapping cost  $TC_{ih}$
  - For each pair of  $i$  and  $h$ ,
    - If  $TC_{ih} < 0$ ,  $i$  is replaced by  $h$
    - Then assign each non-selected object to the most similar representative object
  - repeat steps 2-3 until there is no change

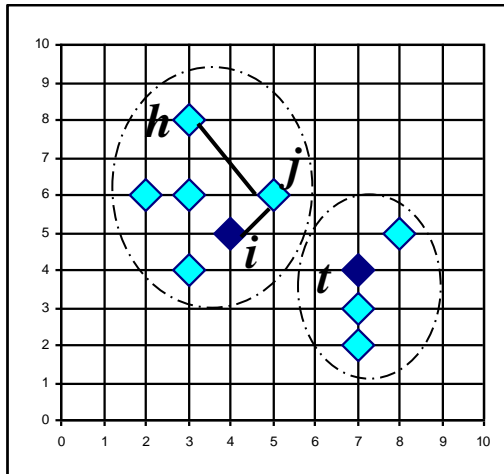
# PAM Clustering: Total swapping cost $TC_{ih} = \sum_j C_{jih}$



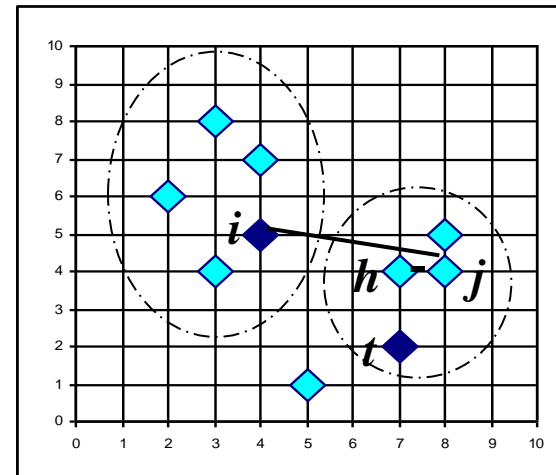
$$C_{jih} = d(j, h) - d(j, i)$$



$$C_{jih} = 0$$



$$C_{jih} = d(j, t) - d(j, i)$$



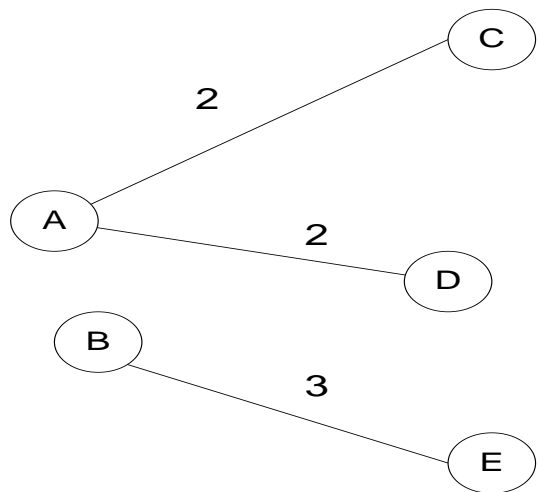
$$C_{jih} = d(j, h) - d(j, t)$$

# Example of K-medoids

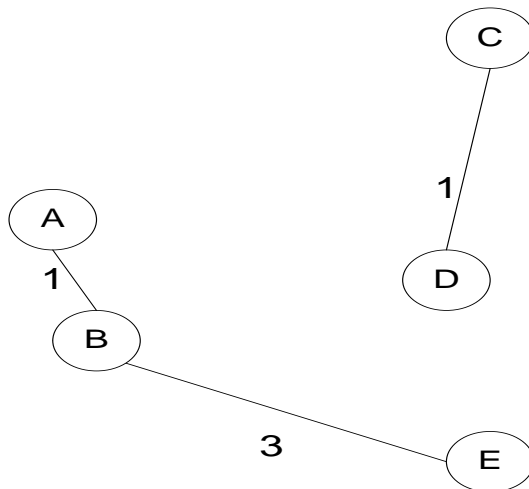
Given the two medoids that are initially chosen are A and B. Based on the following table and randomly placing items when distances are identical to the two medoids, we obtain the clusters {A, C, D} and {B, E}. The three non-medoids {C, D, E} are examined to see which should be used to replace A or B. We have six costs to determine:  $TC_{AC}$  (the cost change by replacing medoid A with medoid C),  $TC_{AD}$ ,  $TC_{AE}$ ,  $TC_{BC}$ ,  $TC_{BD}$  and  $TC_{BE}$ .

$$TC_{AC} = C_{AAC} + C_{BAC} + C_{CAC} + C_{DAC} + C_{EAC} = 1 + 0 - 2 - 1 + 0 = -2$$

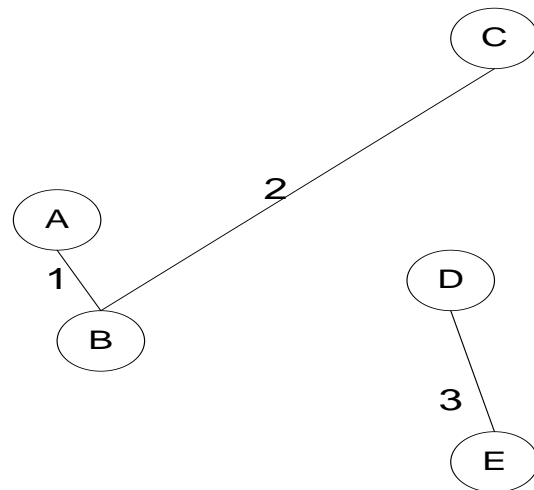
Where  $C_{AAC}$  = the cost change of object A after replacing medoid A with medoid C



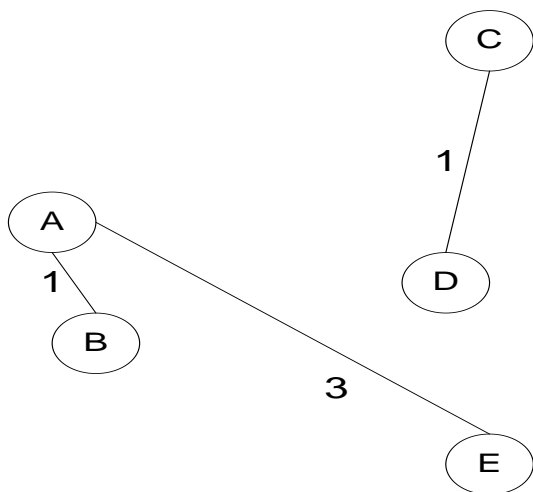
(a) Start: Medoids: A, B



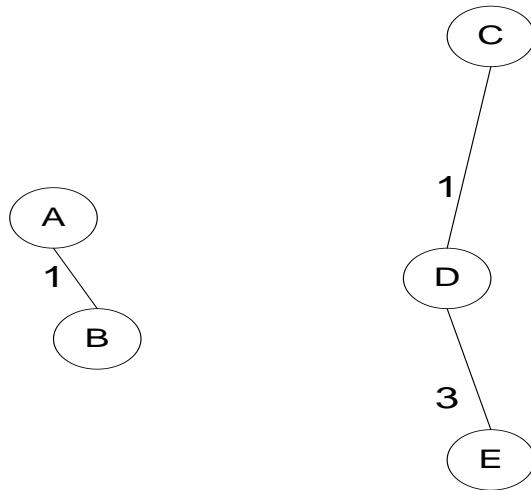
(b)  $TC_{AC}$ : Change -2  
Medoids: B, C



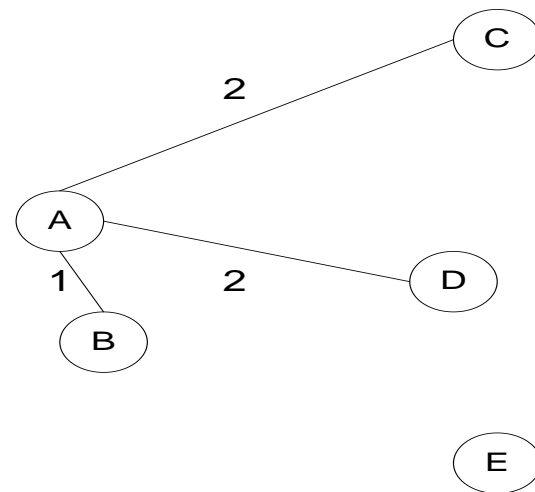
(c)  $TC_{AE}$ : Change -1  
Medoids: B, E



(d)  $TC_{BC}$ : Change -2  
Medoids: A, C



(e)  $TC_{BD}$ : Change -2  
Medoids: A, D



(f)  $TC_{BE}$ : Change -2  
Medoids: A, E

# Cost calculations for example

The diagram illustrates the calculation of these six costs. We see that the minimum cost is 2 and that there are several ways to reduce this cost. Arbitrarily choosing the first swap, we get C and B as the new medoids with the clusters being {C, D} and {B, A, E}

# An example

Initial five objects A, B, C, D, E, two clusters (A, C, D), (B, E), and centers {A, B}.  
Evaluate swap enter A to center C.

Consider the new cost (new centers {B, C})

$$TC_{AC} = C_{AAC} + C_{BAC} + C_{CAC} + C_{DAC} + C_{EAC}$$

$$C_{AAC} = C_{AB} - C_{AA} = 1 - 0 = 1$$

$$C_{BAC} = C_{BB} - C_{BB} = 0 - 0 = 0$$

$$C_{CAC} = C_{CC} - C_{CA} = 0 - 2 = -2$$

$$C_{DAC} = C_{DC} - C_{DA} = 1 - 2 = -1$$

$$C_{EAC} = C_{EB} - C_{EB} = 3 - 3 = 0$$

$$\text{As a result, } TC_{AC} = 1 + 0 - 2 - 1 + 0 = -2$$

The new center {B, C} is less costly. As a result, we should swap {A, B} to {B, C} by Medoid method

# Comparison between K-means and K-medoids

The k-medoids method is more robust than k-means in the presence of noise and outliers because a medoid is less influenced by outliers or other extreme values than a mean. However, its processing is more costly than the k-means method. Both methods require the user to specify  $k$ , the number of clusters.



# Questions?



# References

1. S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 26:65-74, 1997
2. J. Han and M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2000.
3. W. Cleveland. Visualizing Data. Hobart Press, Summit NJ, 1993
4. T. M. Mitchell. Generalization as search. Artificial Intelligence, 18:203-226, 1982.
5. J. Hertz, A. Krogh, R.G. Palmer, “Introduction to the theory of Neural Computation”, Addison-Wesley, 1991.
6. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. VLDB'94.
7. C. Apte and S. Weiss. Data mining with decision trees and decision rules. Future Generation Computer Systems, 13, 1997.
8. I. H. Witten and E. Frank. Data Mining: Practical Machine Learning Tools and Techniques, 2ed. Morgan Kaufmann, 2005.

# End of Unit 6





**Thank you !!!**