**Unit 8**

# Cluster Analysis, Classification and Predication

# Classification vs. Prediction

- **Classification**
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
  - E.g., grouping patients based on their medical history.
  - E.g., grouping students based on their test scores.
- **Prediction**
  - Prediction is the use of existing data values to guess a future value.
  - models continuous-valued functions, i.e., predicts unknown or missing values
  - E.g., using a patient's medical history to guess the effect of a treatment.
  - E.g., using a student's past test scores to guess chance of success in a future exam.

# Classification

**Classification**

- It 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.

# Classification

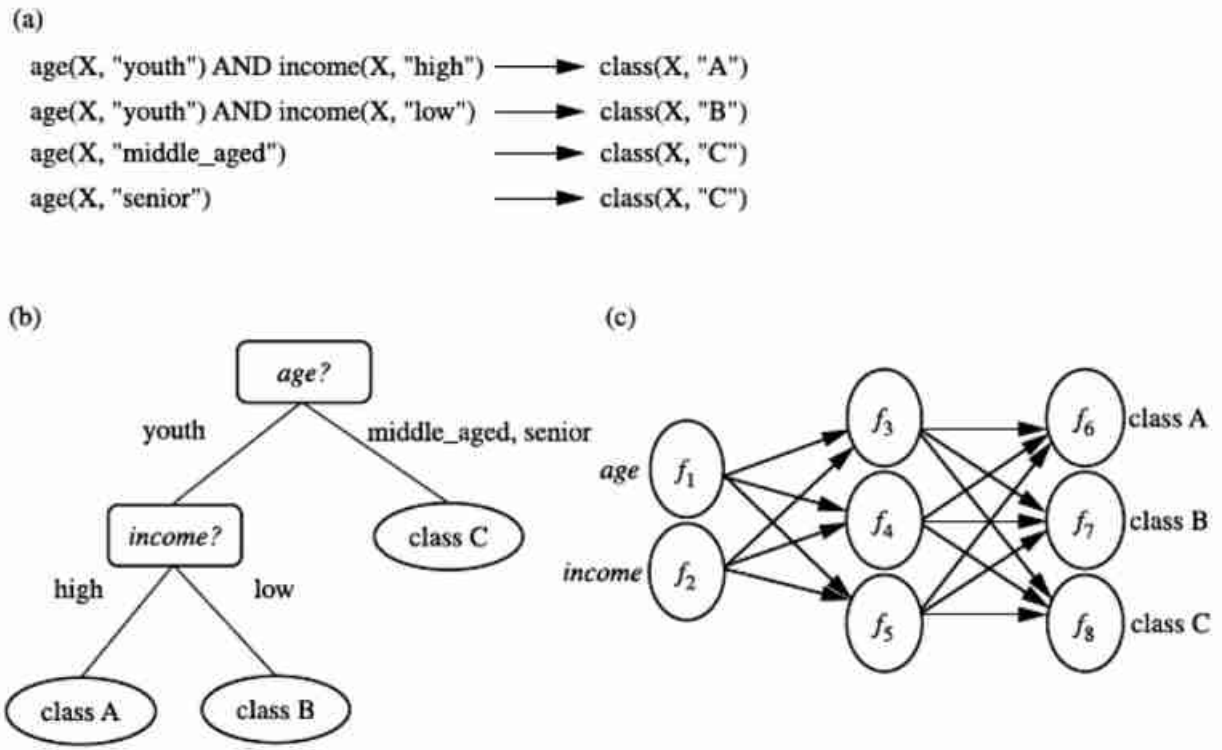



**Figure : A classification model can be represented in various forms, such as (a) IF-THEN rules, (b) a decision tree, or a (c) neural network.**

# Classification → Example of Grading

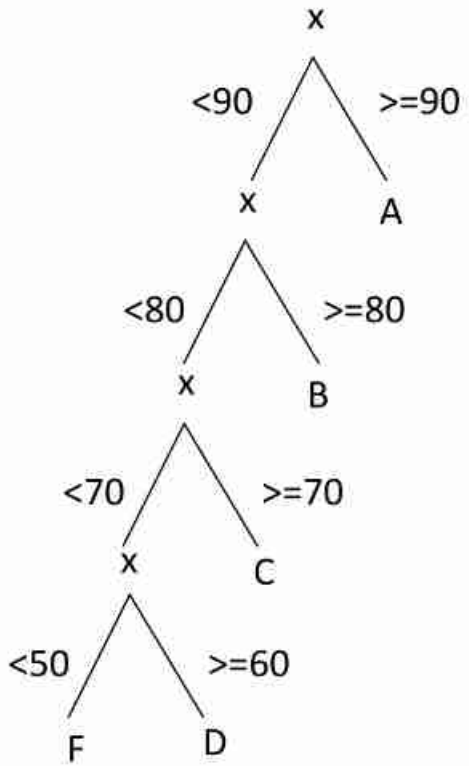
If x >= 90 then grade =A.

If 80<=x<90 then grade =B.

If 70<=x<80 then grade =C.

If 60<=x<70 then grade =D.

If x<50 then grade =F.

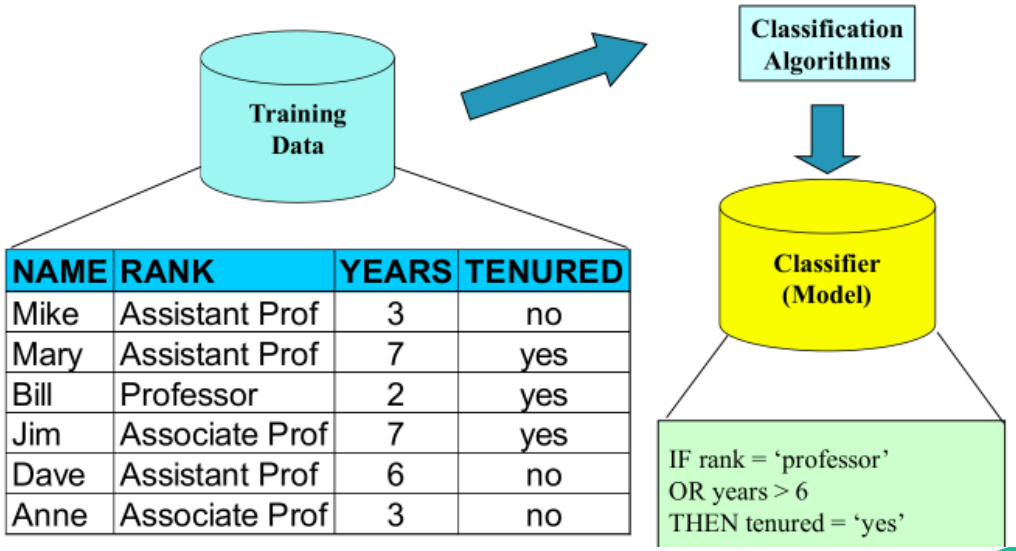


# Classification

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

**Step 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.

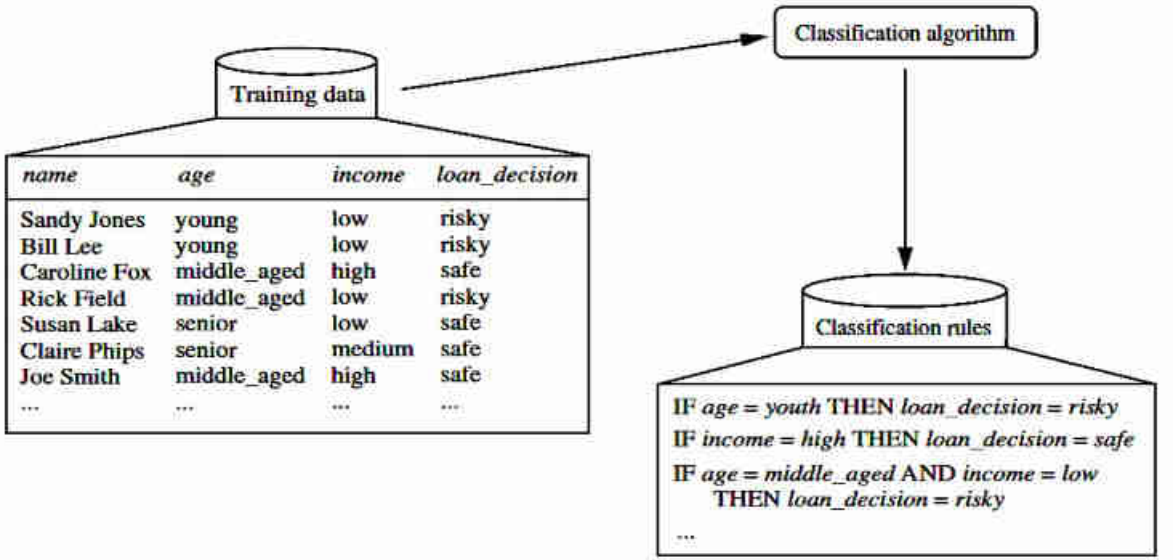

| NAME | RANK | YEARS | TENURED |
|---|---|---|---|
| Mike | Assistant Prof | 3 | no |
| Mary | Assistant Prof | 7 | yes |
| Bill | Professor | 2 | yes |
| Jim | Associate Prof | 7 | yes |
| Dave | Assistant Prof | 6 | no |
| Anne | Associate Prof | 3 | no |

# Classification

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

**Step 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.

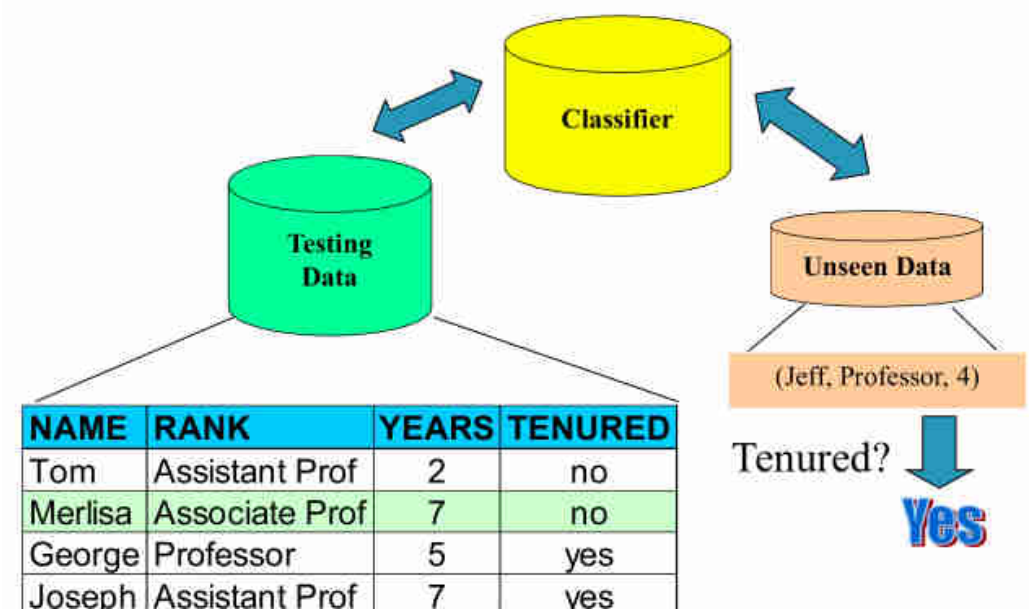



# Classification

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

**Step 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.

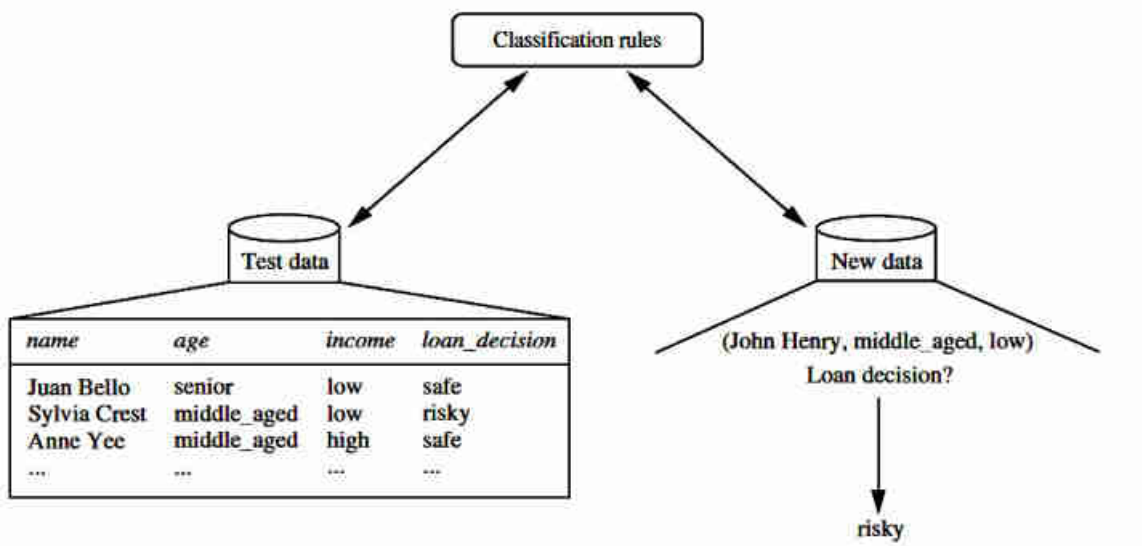

| NAME | RANK | YEARS | TENURED |
|---|---|---|---|
| Tom | Assistant Prof | 2 | no |
| Merlisa | Associate Prof | 7 | no |
| George | Professor | 5 | yes |
| Joseph | Assistant Prof | 7 | yes |

# Classification

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

**Step 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.

# Supervised vs. Unsupervised Learning

- **Supervised learning (classification)**
  - The training data (observations, measurements, etc) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set

- **Unsupervised learning (clustering)**
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

# Issues in Classification and Prediction

- The major issue is preparing the data for classification and Prediction are:
  - Data Cleaning
  - Relevance Analysis
  - Data Transformation and reduction
    a) Normalization
    b) Generalization

- Data cleaning involves removing the noise and treatment of missing values.

- The noise is removed by applying smoothing techniques and the problem of missing values is solved by replacing a missing value with most commonly occurring value for that attribute.

# Classification and Prediction → Relevance Analysis

- Database may also have the irrelevant attributes. Correlation analysis is used to know whether any two given attributes are related.

- **Normalization**
  - Data transformation process
  - Normalization involves scaling all values for given attribute in order to make them fall within a small specified range.
  - Normalization is used when in the learning step, the neural networks or the methods involving measurements are used.
- **Generalization**
  - The data can also be transformed by generalizing it to the higher concept.
  - For this purpose we can use the concept hierarchies.

# Evaluating Classification Methods → Issues

- **Accuracy**
  - Classifier Accuracy: predicting class label
  - Predictor Accuracy: guessing value of predicted attributes
- **Speed**
  - Time to construct the model (training time)
  - Time to use the model (classification/prediction time)
- **Robustness**
  - Handling noise and missing values
- **Scalability**
  - Efficiency in disk-resident databases
- **Interpretability**
  - Understanding and insight provided by the model

# Assignment … !!!!

Compare different classification algorithms

# 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 inducting a decision tree from data is called decision tree learning.

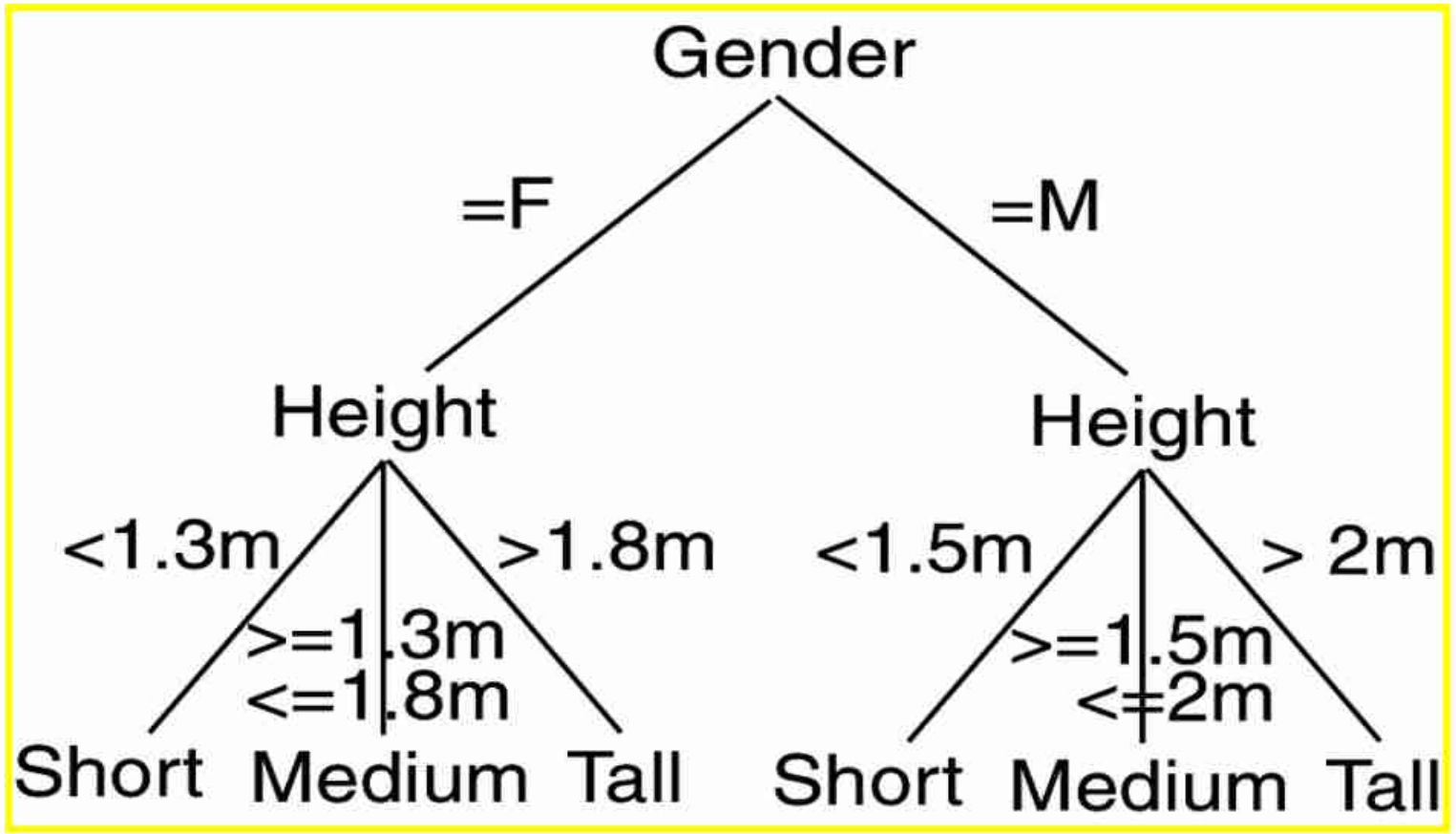
# Decision Trees → Descriptive categories

- **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).

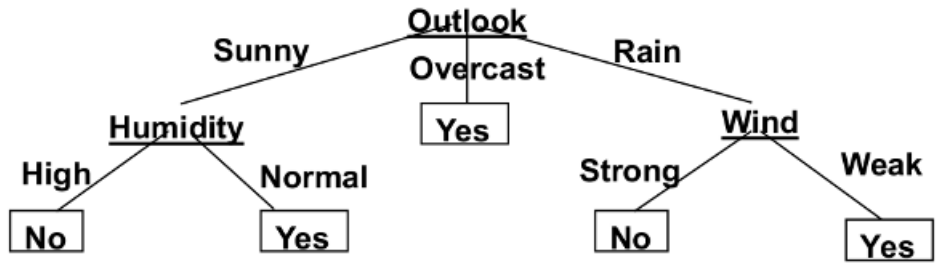
- **Classification And Regression Tree (CART) analysis**
  When both of the above procedures are referred.

# Decision Trees → Example



# Decision Trees → 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}

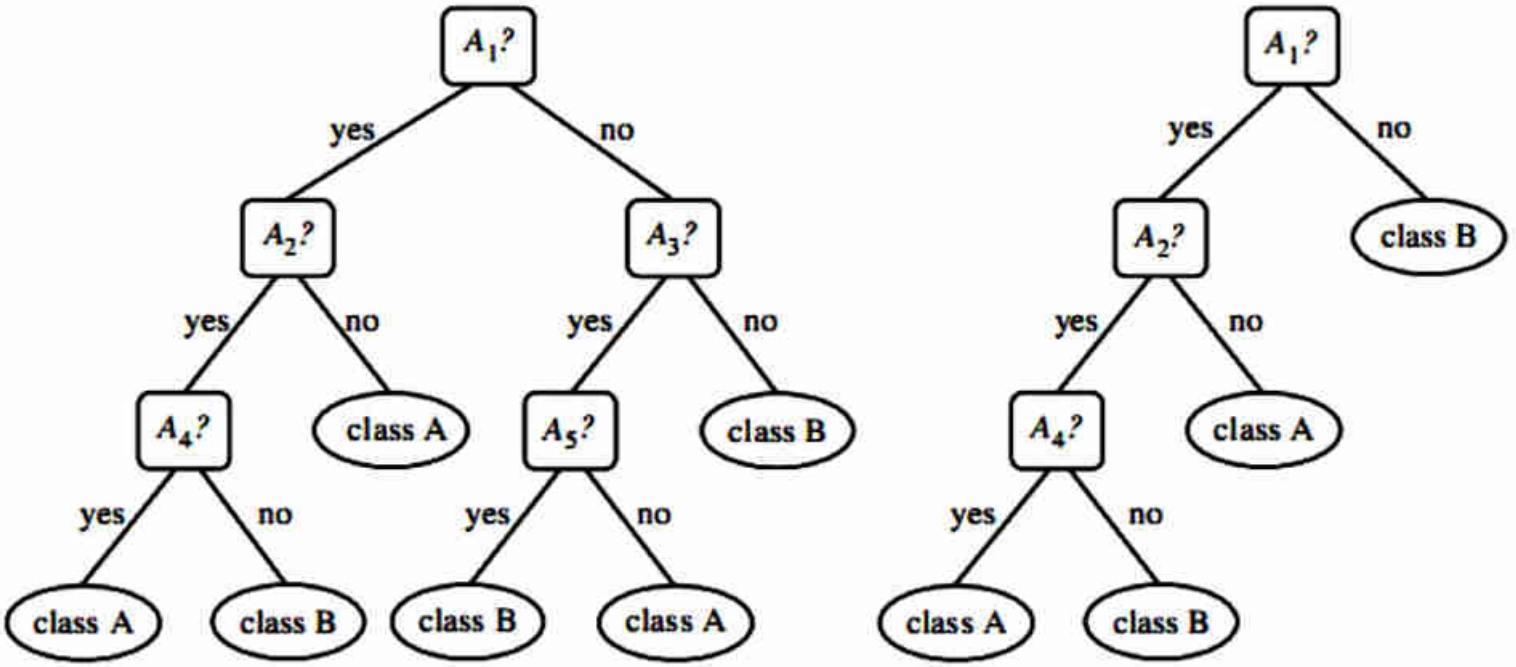
# Classification by Decision Tree Induction

- **Decision tree**
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution
- **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
- **Use of decision tree: Classifying an unknown sample**
  - Test the attribute values of the sample against the decision tree

# Decision Tree Induction → Pruning Approaches

- Pruning is a technique in machine learning that reduces the size of decision trees by removing sections of the tree that provide little power to classify instances.
- The dual goal of pruning is reduced complexity of the final classifier as well as better predictive accuracy by the reduction of over fitting and removal of sections of a classifier that may be based on noisy data.
- Approaches
  a) **Pre pruning** (The tree is pruned by halting its construction early)
     *Based on statistical significance test, Stop growing the tree when there is no statistically significant association between any attribute and the class at a particular node, Most popular test: chi-squared test*
  b) **Post pruning** (This approach removes subtree from fully grown tree)
     First, build full tree then, prune it. Fully-grown tree shows all attribute interactions
     *Problem*: some subtrees might be due to chance effects

# Decision Tree Induction → Pruning Approaches

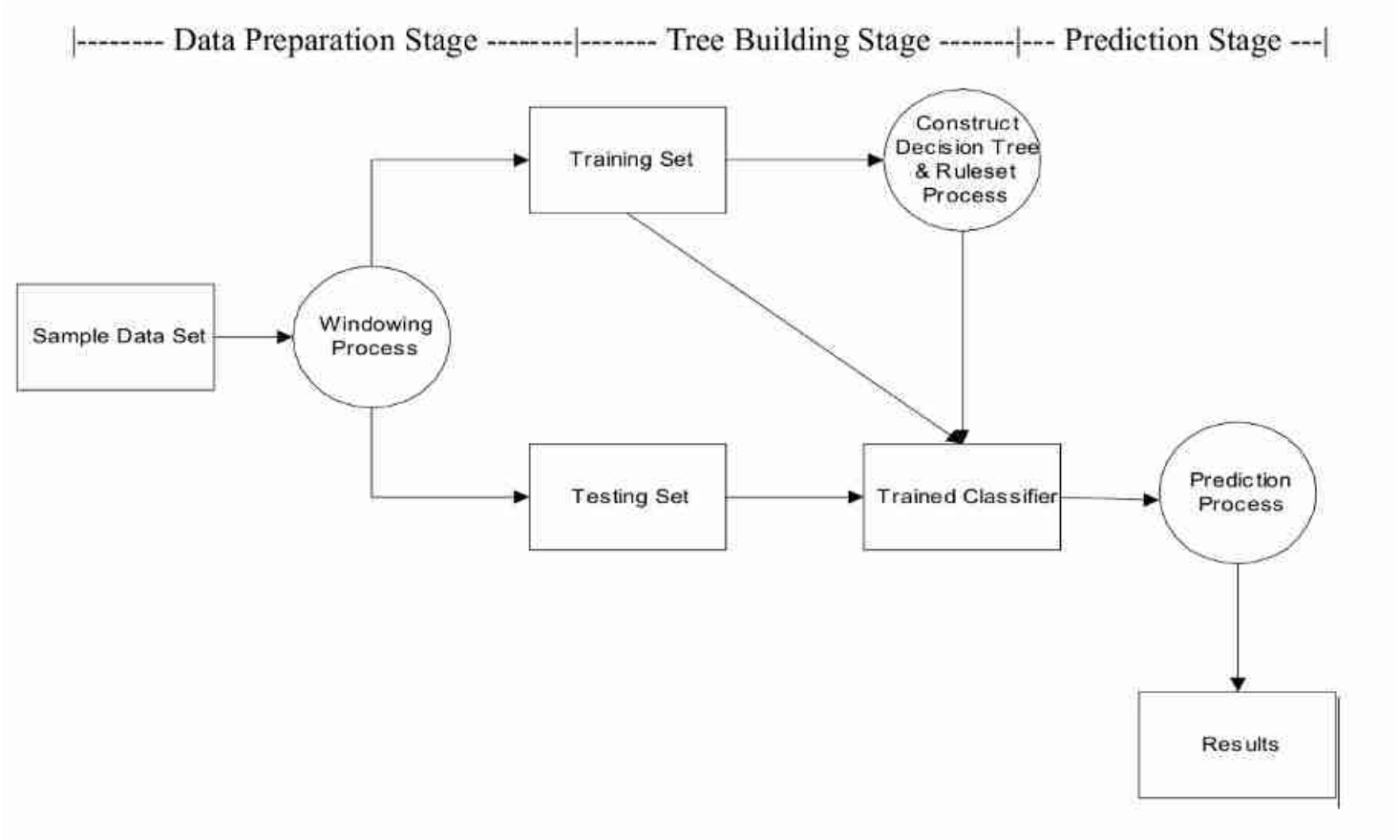



An unpruned decision tree and a pruned version of it.

# 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

# Decision Tree → Block Diagram



# Algorithm for building Decision Trees

```
Input:
   T      //Decision Tree
   D      //Input Database
Output:
   M      //Model Prediction
DTProc Algorithm:
                //Illustrates Prediction Technique using DT
   for each t ∈ D do
      n = 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 = node at end of this arc;
      Make prediction for t based on labeling of n;
```

# 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 over fit 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 sub-trees.
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.*

# Attribute Selection Measures

- An attribute selection measure is a heuristic for selecting the splitting criterion that "best" separates a given data partition, D, of class-labeled training tuples into individual classes.
- If we were to split D into smaller partitions according to the outcomes of the splitting criterion, ideally each partition would be pure (i.e., all of the tuples that fall into a given partition would belong to the same class). Conceptually, the "best" splitting criterion is the one that most closely results in such a scenario.
- Attribute selection measures are also known as splitting rules because they determine how the tuples at a given node are to be split.
- There are three popular attribute selection measures—information gain, gain ratio, and gini index.

# Information gain

- This measure is based on pioneering work by Claude Shannon on information theory, which studied the value or "information content" of messages.
- Let node N represents or hold the tuples of partition D. The attribute with the highest information gain is chosen as the splitting attribute for node N. This attribute minimizes the information needed to classify the tuples in the resulting partitions and reflects the least randomness or "impurity" in these partitions.

# Information gain

**The expected information needed to classify a tuple in D is given by**

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

*where $p_i$ is the probability that an arbitrary tuple in D belongs to class $C_i$*

**How much more information would we still need (after the partitioning) in order to arrive at an exact classification? This amount is measured by**

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j).$$

InfoA(D) is the expected information required to classify a tuple from D based on the partitioning by A

**Information gain is defined as the difference between the original information requirement (i.e., based on just the proportion of classes) and the new requirement (i.e., obtained after partitioning on A). That is**

$$Gain(A) = Info(D) - Info_A(D).$$

# Iterative Dichotomiser (ID3)

- ID3 algorithm selects the best feature at each step while building a Decision tree
- Information Gain calculates the reduction in the entropy and measures how well a given feature separates or classifies the target classes. The feature with the highest Information Gain is selected as the best one.

# Iterative Dichotomiser (ID3)

Assignment → Pros and cons of ID3

# Decision tree using information gain → sample dataset of COVID-19 infection

| ID | Fever | Cough | Breathing issues | Infected |
|---|---|---|---|---|
| 1 | NO | NO | NO | NO |
| 2 | YES | YES | YES | YES |
| 3 | YES | YES | NO | NO |
| 4 | YES | NO | YES | YES |
| 5 | YES | YES | YES | YES |
| 6 | NO | YES | NO | NO |
| 7 | YES | NO | YES | YES |
| 8 | YES | NO | YES | YES |
| 9 | NO | YES | YES | YES |
| 10 | YES | YES | NO | YES |
| 11 | NO | YES | NO | NO |
| 12 | NO | YES | YES | YES |
| 13 | NO | YES | YES | NO |
| 14 | YES | YES | NO | NO |

The columns are self-explanatory. Infected and Not Infected stand for Yes and No respectively. The values or classes in Infected column Y and N represent Infected and Not Infected respectively.

The columns used to make decision nodes viz. Breathing Issues, Cough and Fever are called feature columns or just features and the column used for leaf nodes i.e. Infected is called the target column.

# Decision tree using information gain → sample dataset of COVID-19 infection

- We denote our data set as S, entropy is calculated as:

$$\textbf{Entropy(S)} = -\sum p_i * \log_2(p_i) \; ; \; \textbf{i = 1 to n}$$

Where, n is the total number of classes in the target column (in our case n = 2 i.e YES and NO) $p_i$ is the probability of class 'i' or the ratio of "number of rows with class i in the target column" to the "total number of rows" in the dataset.

- Information Gain for a feature column A is calculated as:

$$\textbf{IG(S,A) = Entropy(S) - } \sum((|S_v| / |S|) * \textbf{Entropy}(S_v))$$

where $S_v$ is the set of rows in S for which the feature column A has value v, $|S_v|$ is the number of rows in $S_v$ and likewise |S| is the number of rows in S.

# Decision tree using information gain → sample dataset of COVID-19 infection

From the total of 14 rows in our data set S, there are 8 rows with the target value YES and 6 rows with the target value NO. The entropy of S is calculated as:

**Entropy(S) = − (8/14) * $\log_2$(8/14) − (6/14) * $\log_2$(6/14) = 0.99**

# Decision tree using information gain → sample dataset of COVID-19 infection

**Considering 'Infected' as the target column(IG for Fever)**

- Total rows |S| = 14
- For v = YES, $|S_v| = 8$

  **$Entropy(S_v) = -(6/8) * \log_2(6/8) - (2/8) * \log_2(2/8) = 0.81$**
- For v = NO, $|S_v| = 6$

  **$Entropy(S_v) = -(2/6) * \log_2(2/6) - (4/6) * \log_2(4/6) = 0.91$**
- Expanding the summation in the IG formula:

  **$IG(S, Fever) = Entropy(S) - (|S_{YES}| / |S|) * Entropy(S_{YES}) - (|S_{NO}| / |S|) * Entropy(S_{NO})$**

  ∴ IG(S, Fever) = 0.99 − (8/14) * 0.81 − (6/14) * 0.91 = 0.13
- Next, we calculate the IG for the features "Cough" and "Breathing issues".

  ∴ IG(S, Cough) = 0.04

  ∴ IG(S, BreathingIssues) = 0.40

# Decision tree using information gain → sample dataset of COVID-19 infection

- Next, from the remaining two unused features, namely, Fever and Cough, we decide which one is the best for the left branch of Breathing Issues.
- Since the left branch of Breathing Issues denotes YES, we will work with the subset of the original data i.e the set of rows having YES as the value in the Breathing Issues column. These 8 rows are shown below:

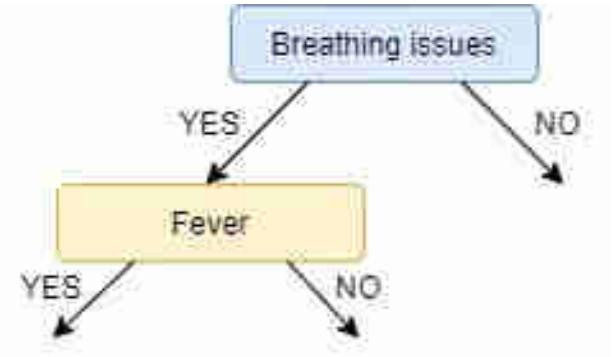
| Fever | Cough | Breathing issues | Infected |
|---|---|---|---|
| YES | YES | YES | YES |
| YES | NO | YES | YES |
| YES | YES | YES | YES |
| YES | NO | YES | YES |
| YES | NO | YES | YES |
| NO | YES | YES | YES |
| NO | YES | YES | YES |
| NO | YES | YES | NO |

# Decision tree using information gain → sample dataset of COVID-19 infection

Next, we calculate the IG for the features Fever and Cough using the subset $S_{BY}$ (Set Breathing Issues Yes) which is shown above :

$IG(S_{BY}, Fever) = 0.20$

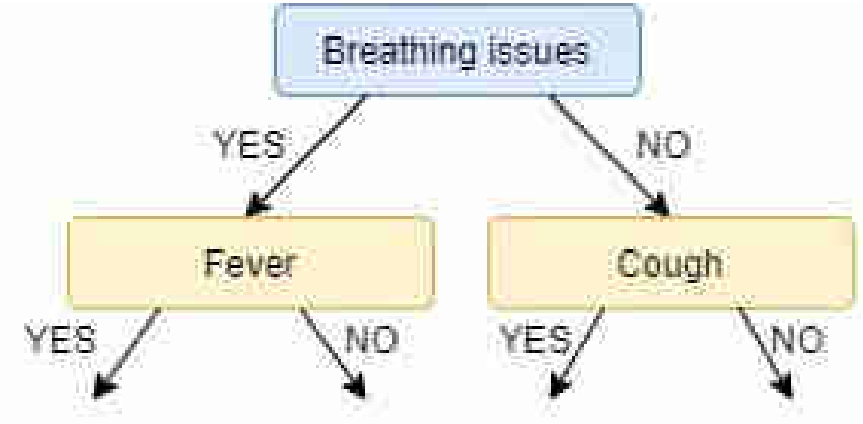
$IG(S_{BY}, Cough) = 0.09$



# Decision tree using information gain → sample dataset of COVID-19 infection

Next, we find the feature with the maximum IG for the right branch of Breathing Issues. But, since there is only one unused feature left we have no other choice but to make it the right branch of the root node.

So our tree now looks like this:



# Decision tree using information gain → sample dataset of COVID-19 infection

There are no more unused features, so we stop here and jump to the final step of creating the leaf nodes.

For the left leaf node of Fever, we see the subset of rows from the original data set that has Breathing Issues and Fever both values as YES.

| Fever | Cough | Breathing issues | Infected |
|---|---|---|---|
| YES | YES | YES | YES |
| YES | NO | YES | YES |
| YES | YES | YES | YES |
| YES | NO | YES | YES |
| YES | NO | YES | YES |

# Decision tree using information gain → sample dataset of COVID-19 infection

Since all the values in the target column are YES, we label the left leaf node as YES, but to make it more logical we label it Infected.

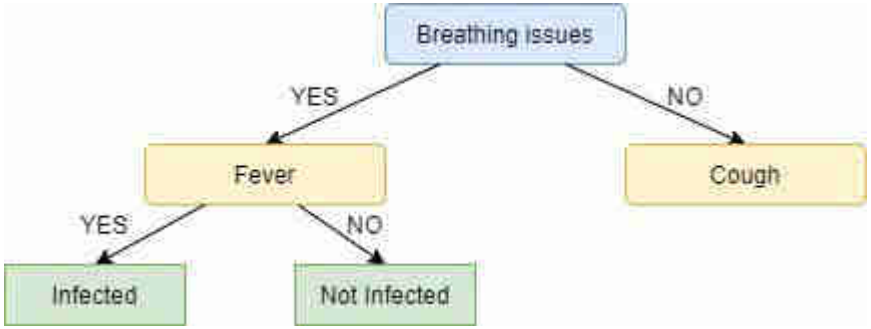
Similarly, for the right node of Fever we see the subset of rows from the original data set that have Breathing Issues value as YES and Fever as NO.

| Fever | Cough | Breathing issues | Infected |
|---|---|---|---|
| NO | YES | YES | YES |
| NO | YES | YES | NO |
| NO | YES | YES | NO |

# Decision tree using information gain → sample dataset of COVID-19 infection

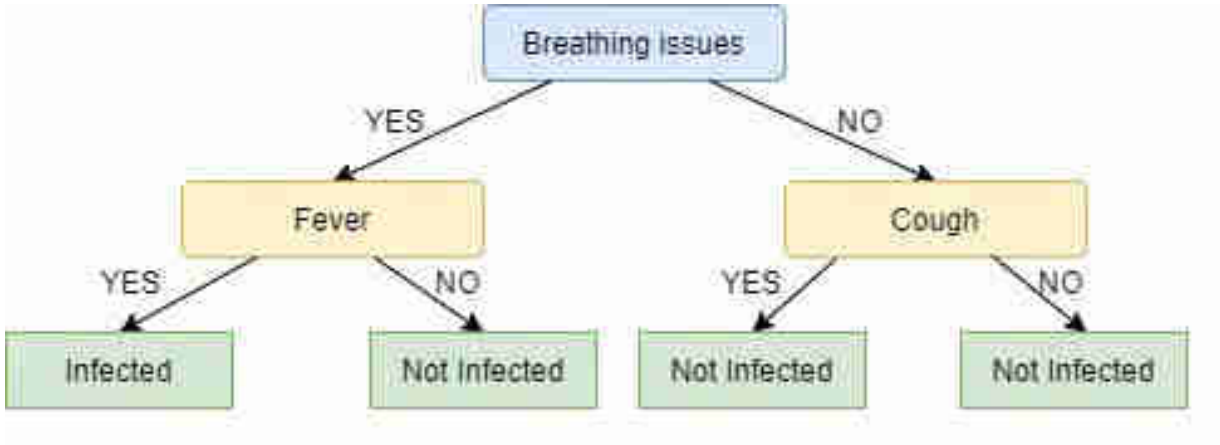
Here not all but most of the values are NO, hence NO or Not Infected becomes our right leaf node.

Our tree, now, looks like this:



# Decision tree using information gain → sample dataset of COVID-19 infection

We repeat the same process for the node Cough, however here both left and right leaves turn out to be the same i.e. NO or Not Infected as shown below:



# Rules Based Classification

- Rule Based Classification has learned model as a set of IF-THEN rules
- We need to
  - Generate the model
  - Examine how to use model to classify data
- IF-THEN rule is an expression of the form
  - IF *condition* THEN *conclusion*

  Example
- *R1:IF age=youth and student=yes THEN buys_computer=yes*

  *Or*
- *R1:(age=youth) Λ (student=yes)=>(buys_computer=yes)*

# Using If-Then rules for classification

- IF part is called rule antecedent or precondition, it can consist of one or more attributes test
- THEN part is called rule consequent, it consist a class prediction
- A rule R can be assessed by its coverage and accuracy
  - Given a **tuple X from a data D**
  - Let **$n_{cover}$: Number of tuples covered by R**
  - **$n_{correct}$ : Number of tuples correctly classify by R**
  - **|D|: Number of tuples in D**

$$coverage(R) = \frac{n_{covers}}{|D|}$$

$$accuracy(R) = \frac{n_{correct}}{n_{covers}}.$$

# Using If-Then rules for classification

Class-labeled training tuples from the *AllElectronics* customer database.

| RID | age | income | student | credit_rating | Class: buys_computer |
|---|---|---|---|---|---|
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle_aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle_aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle_aged | medium | no | excellent | yes |
| 13 | middle_aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

# Using If-Then rules for classification

R: IF age=youth AND student=yes THEN buys_computer=yes

➔ |D| = 14

$n_{cover} = 2$

$n_{correct} = 2$

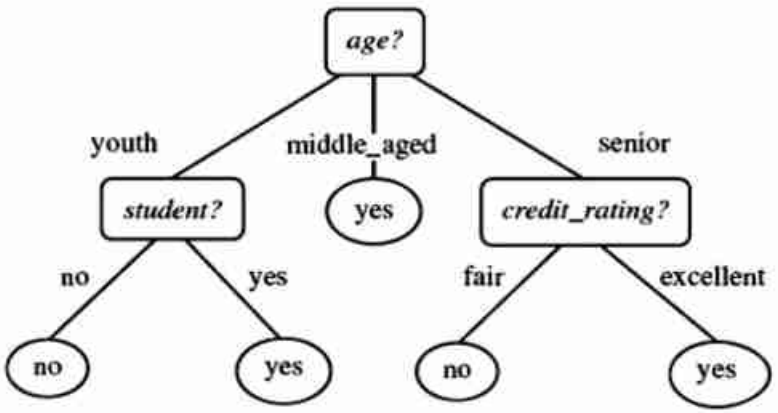
$$coverage(R) = \frac{2}{14} = 14.28\%$$

$$accuracy(R) = \frac{2}{2} = 100\%$$

# Rule Extraction from a Decision Tree

- One rule is created for each path from the root to a leaf node
- Each splitting criterion is logically AND to form the rule antecedent (IF part)
- Leaf node holds the class prediction for rule consequent (THEN part)
- Logical OR is implied between each of the extracted rules

# Rule Extraction from a Decision Tree → Example




*R*1: IF *age* = *youth* AND *student* = *no* THEN *buys_computer* = *no*
*R*2: IF *age* = *youth* AND *student* = *yes* THEN *buys_computer* = *yes*
*R*3: IF *age* = *middle_aged* THEN *buys_computer* = *yes*
*R*4: IF *age* = *senior* AND *credit_rating* = *excellent* THEN *buys_computer* = *yes*
*R*5: IF *age* = *senior* AND *credit_rating* = *fair* THEN *buys_computer* = *no*

# Bayesian Classification

- Bayesian classification is based on Baye's Theorem.
- It is a statistical classifier that predicts class membership probabilities such as the probability that a given tuple belongs to a particular class.
- Baye's Law
  - Formula : P(A/B) = (P(B/A)* P(A))/ P(B)
  - Has high accuracy and speed for large databases.
  - Has minimum error rate in comparison to all other classifier

# Bayesian Classification

Given training evidence X, posterior probability of a hypothesis H, P(H|X), follows the Bayes theorem

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

Informally, this can be written as

**posterior = (likelihood * prior) / evidence**

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

# Bayesian Classification: Why?

- **A statistical classifier:**

  Performs probabilistic prediction, i.e., predicts class membership probabilities

- **Foundation:**

  Based on Bayes' Theorem.

- **Performance:**

  A simple Bayesian classifier, naïve Bayesian classifier, has comparable performance with decision tree and selected neural network classifiers

- **Incremental:**

  Each training example can incrementally increase/decrease the probability that a hypothesis is correct– prior knowledge can be combined with observed data

- **Standard:**

  Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

# Towards Naïve Bayesian Classifier

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-dim. attribute vector $\mathbf{X} = (x_1, x_2, \ldots, x_n)$
- Suppose there are *m* classes $C_1, C_2, \ldots, C_m$.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since prior prob. $P(\mathbf{X})$ is constant for all classes,

maximizing $P(C_i|\mathbf{X})$ is equivalent to maximizing $P(\mathbf{X}|C_i)P(C_i)$

# Naïve Bayesian Classifier: Training Dataset

Class:
$C_1$: buys_computer = 'yes'
$C_2$: buys_computer = 'no'

Data sample
X = (age = youth,
income = medium,
student = yes
credit_rating = Fair)

Class-labeled training tuples from the *AllElectronics* customer database.

| RID | age | income | student | credit_rating | Class: buys_computer |
|---|---|---|---|---|---|
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle_aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle_aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle_aged | medium | no | excellent | yes |
| 13 | middle_aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

# Naïve Bayesian Classifier → An Example

- $P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643
  P(buys_computer = "no") = 5/14= 0.357

- Compute $P(X|C_i)$ for each class
  P(age = "youth" | buys_computer = "yes") = 2/9 = 0.222
  P(age = "youth" | 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 = youth, income = medium, student = yes, credit_rating = fair)**

**$P(X|C_i)$ :** P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019
**$P(X|C_i)*P(C_i)$ :** P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, **X** belongs to class ("buys_computer = yes")

# Prediction

“What if we would like to predict a continuous value, rather than a categorical label?”

# Prediction

- Numeric prediction is the task of predicting continuous (or ordered) values for given input.

  *For example, we may wish to predict the salary of college graduates with 10 years of work experience, or the potential sales of a new product given its price.*

- By far, the most widely used approach for numeric prediction is regression

# Prediction

- Numeric prediction is the task of predicting continuous (or ordered) values for given input.

  *For example, we may wish to predict the salary of college graduates with 10 years of work experience, or the potential sales of a new product given its price.*

- By far, the most widely used approach for numeric prediction is regression

- In fact, many texts use the terms “regression” and “numeric prediction” synonymously.

  However, as we have seen, some classification techniques (such as back-propagation, support vector machines, and k-nearest-neighbor classifiers) can be adapted for prediction

# Regression Analysis

- Regression analysis can be used to model the relationship between one or more independent or predictor variables and a dependent or response variable (which is continuous-valued).
- In the context of data mining, the predictor variables are the attributes of interest describing the tuple

# Regression Analysis → Types

- Linear and multiple regression
- Non-linear regression
- Other regression methods:

  generalized linear model, Poisson regression, log-linear models, regression trees

# Linear Regression

- Straight-line regression analysis involves a response variable, y, and a single predictor variable, x. It is the simplest form of regression, and models y as a linear function of x.
- That is,

$$y = b + wx,$$

where the variance of y is assumed to be constant, and b and w are regression coefficients specifying the Y-intercept and slope of the line, respectively. The regression coefficients, w and b, can also be thought of as weights, so that we can equivalently write,

$$y = w_0 + w_1 x.$$

# Linear Regression

The regression coefficients can be estimated using this method with the following equations:

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

# Linear Regression

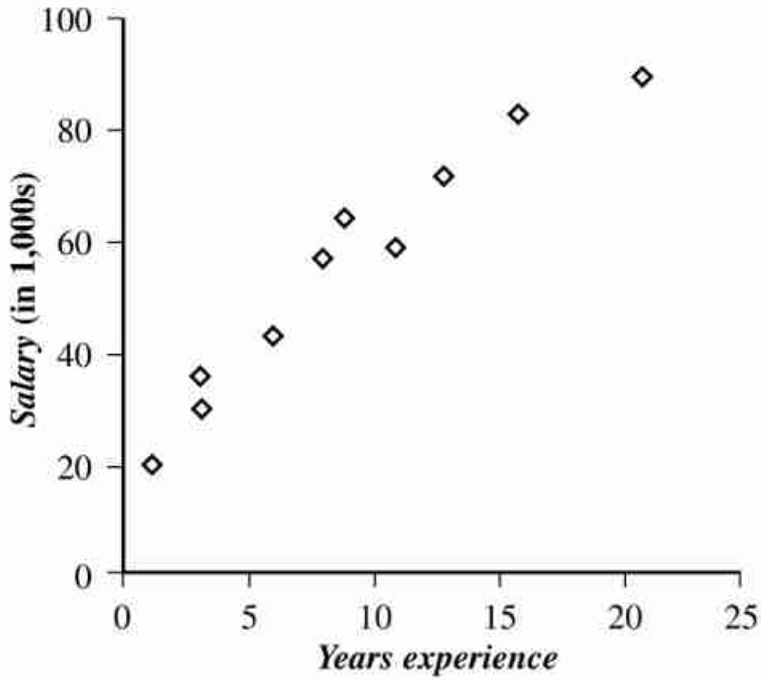
Since the $X_i$ here are one-dimensional, we use the notation $x_i$ over $X_i$ in this case.

$$w_0 = \bar{y} - w_1 \bar{x}$$

where $\bar{x}$ is the mean value of $x_1, x_2, \ldots, x_{|D|}$, and $\bar{y}$ is the mean value of $y_1, y_2, \ldots, y_{|D|}$. The coefficients $w_0$ and $w_1$ often provide good approximations to otherwise complicated regression equations.

# Linear Regression → Example

| *x* years experience | *y* salary (in $1000s) |
|---|---|
| 3 | 30 |
| 8 | 57 |
| 9 | 64 |
| 13 | 72 |
| 3 | 36 |
| 6 | 43 |
| 11 | 59 |
| 21 | 90 |
| 1 | 20 |
| 16 | 83 |



# Linear Regression → Example

Given the above data, we compute $\bar{x} = 9.1$ and $\bar{y} = 55.4$. Substituting these values into Equations (6.50) and (6.51), we get

$$w_1 = \frac{(3-9.1)(30-55.4)+(8-9.1)(57-55.4)+\cdots+(16-9.1)(83-55.4)}{(3-9.1)^2+(8-9.1)^2+\cdots+(16-9.1)^2} = 3.5$$

$$w_0 = 55.4-(3.5)(9.1) = 23.6$$

Thus, the equation of the least squares line is estimated by $y = 23.6 + 3.5x$. Using this equation, we can predict that the salary of a college graduate with, say, 10 years of experience is \$58,600.

# Multiple linear regression

Multiple linear regression is an extension of straight-line regression so as to involve more than one predictor variable. It allows response variable y to be modeled as a linear function of, say, n predictor variables or attributes, $A_1$ , $A_2$ , . . . , $A_n$ , describing a tuple, X. (That is, X = ($x_1$ , $x_2$ , . . . , $x_n$).)

Our training data set, D, contains data of the form$(X_1, y_1)$,$(X_2, y_2)$, . . . ,$(X_{|D|}, y_{|D|})$, where the $X_I$ are the n-dimensional training tuples with associated class labels, $y_i$ . An example of a multiple linear regression model based on two predictor attributes or variables, $A_1$ and $A_2$ is

$$y = w_0 + w_1 x_1 + w_2 x_2,$$

# Nonlinear Regression

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

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

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

  $y = w_0 + w_1 x + w_2 x_2 + w_3 x_3$
- Other functions, such as power function, can also be transformed to linear model
- Some models are intractable nonlinear (e.g., sum of exponential terms)
  - possible to obtain least square estimates through extensive calculation on more complex formulae

# Other Regression-Based Models

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

# Assignment ….. !!

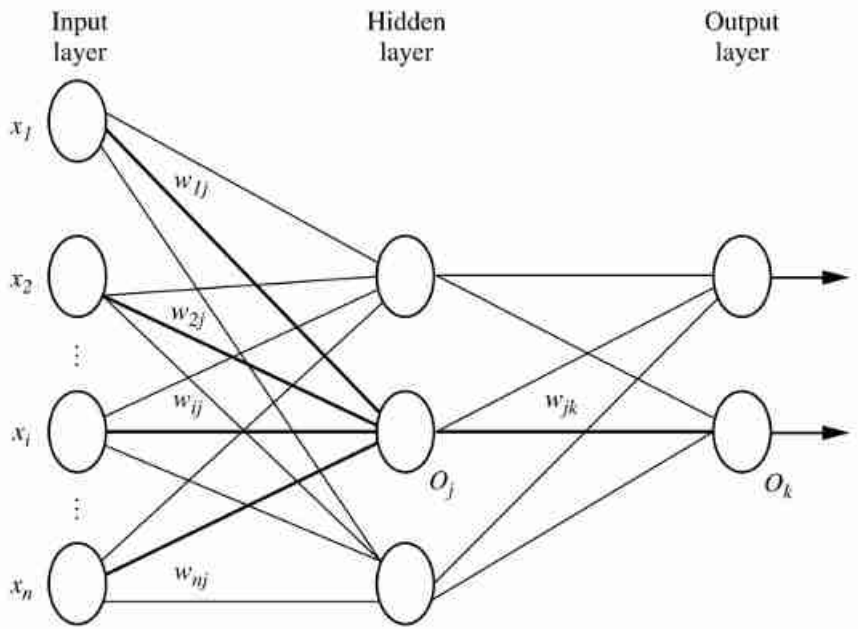
- Define neurons and neural network
- List Pros and Cons of Neural Network.

# Classification by Backpropagation

- Backpropagation is a neural network learning algorithm.
- A neural network is a set of connected input/output units in which each connection has a weight associated with it.
- During the learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of the input tuples.
- Neural network learning is also referred to as connectionist learning due to the connections between units.

# A Multilayer Feed-Forward Neural Network

- Each layer is made up of units. The inputs to the network correspond to the attributes measured for each training tuple.
- The inputs are fed simultaneously into the units making up the input layer. These inputs pass through the input layer and are then weighted and fed simultaneously to a second layer of “neuron like” units, known as a hidden layer.
- The outputs of the hidden layer units can be input to another hidden layer, and so on. The number of hidden layers is arbitrary, although in practice, usually only one is used.
- The weighted outputs of the last hidden layer are input to units making up the output layer, which emits the network’s prediction for given tuples.
- The units in the input layer are called input units. The units in the hidden layers and output layer are sometimes referred to as neurodes, due to their symbolic biological basis, or as output units.



# Backpropagation → Algorithm

**Input:**

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

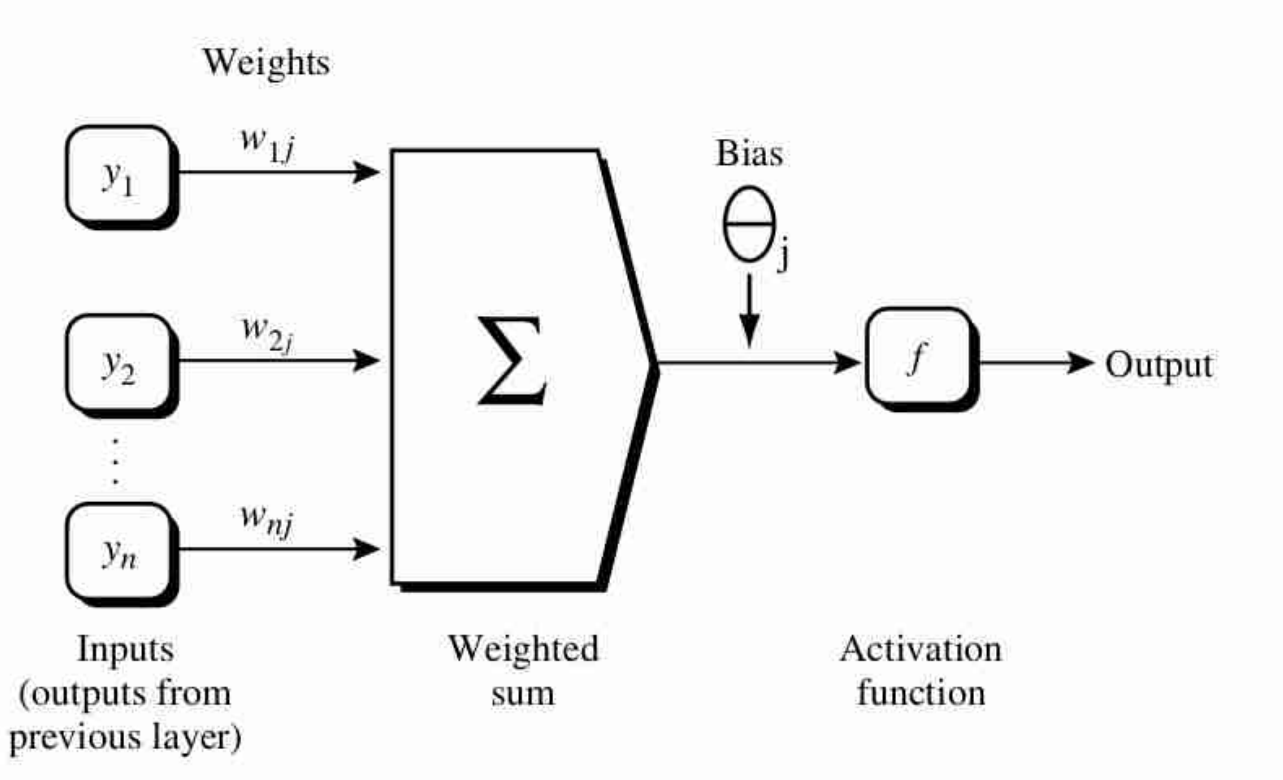
**Output:** A trained neural network.

# Backpropagation → Algorithm

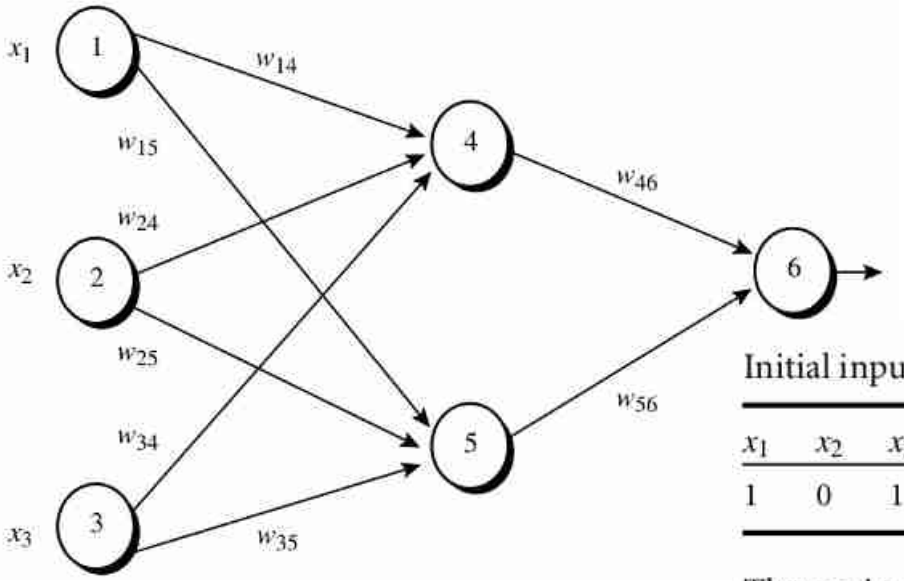
**Method:**

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

# Multilayer Feed-Forward NN



# Multilayer Feed-Forward NN → Example




Initial input, weight, and bias values.

| $x_1$ | $x_2$ | $x_3$ | $w_{14}$ | $w_{15}$ | $w_{24}$ | $w_{25}$ | $w_{34}$ | $w_{35}$ | $w_{46}$ | $w_{56}$ | $\theta_4$ | $\theta_5$ | $\theta_6$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 0.2 | −0.3 | 0.4 | 0.1 | −0.5 | 0.2 | −0.3 | −0.2 | −0.4 | 0.2 | 0.1 |

The net input and output calculations.

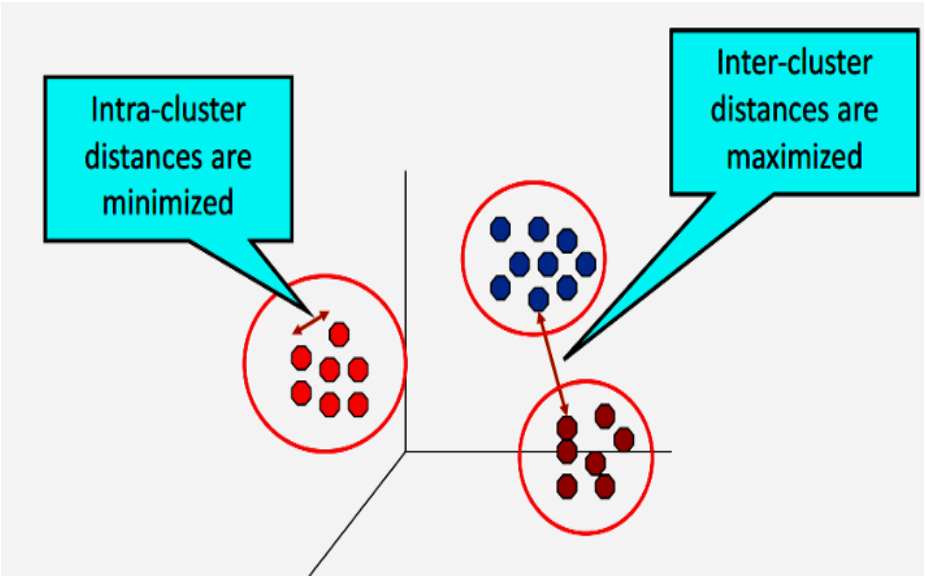
| *Unit $j$* | *Net input, $I_j$* | *Output, $O_j$* |
|---|---|---|
| 4 | $0.2+0-0.5-0.4=-0.7$ | $1/(1+e^{0.7})=0.332$ |
| 5 | $-0.3+0+0.2+0.2=0.1$ | $1/(1+e^{-0.1})=0.525$ |
| 6 | $(-0.3)(0.332)-(0.2)(0.525)+0.1=-0.105$ | $1/(1+e^{0.105})=0.474$ |

# Cluster Analysis

- Clustering involves the grouping of similar objects into a set known as cluster. Objects in one cluster are likely to be different when compared to objects grouped under another cluster.
- Clustering is one of the main tasks in exploratory data mining and is also a technique used in statistical data analysis. While clustering is not one specific algorithm, it is a general task that can be solved by means of several algorithms.

# Quality: What Is Good Clustering?

- A good clustering method will produce high quality clusters
  - High intra-class similarity: cohesive within clusters
  - Low inter-class similarity: distinctive between clusters
- The quality of a clustering method depends on
  - The similarity measure used by the method
  - It's implementation, and
  - Its ability to discover some or all of the hidden patterns



# Measure the Quality of Clustering

- **Dissimilarity/Similarity metric**
  - Similarity is expressed in terms of a distance function, typically metric: $d(i, j)$
  - The definitions of distance functions are usually rather different for interval-scaled, boolean, categorical, ordinal ratio, and vector variables
  - Weights should be associated with different variables based on applications and data semantics
- Quality of clustering:
  - There is usually a separate "quality" function that measures the "goodness" of a cluster.
  - It is hard to define "similar enough" or "good enough"

# Considerations for Cluster Analysis

- Partitioning criteria
  - Single level vs. hierarchical partitioning (often, multi-level hierarchical partitioning is desirable)
- Separation of clusters
  - Exclusive (e.g., one customer belongs to only one region) vs. non-exclusive (e.g., one document may belong to more than one class)
- Similarity measure
  - Distance-based (e.g., Euclidian, road network, vector) vs. connectivity-based (e.g., density or contiguity)
- Clustering space
  - Full space (often when low dimensional) vs. subspaces (often in high-dimensional clustering)

# Similarity and Dissimilarity Between Objects

- Distances are normally used to measure the similarity or dissimilarity between two data objects
- Some popular ones include: *Minkowski distance*:

$$d(i,j) = \sqrt[q]{(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \ldots + |x_{ip} - x_{jp}|^q)}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jp})$ are two $p$-dimensional data objects, and $q$ is a positive integer

- If $q = 1$, $d$ is Manhattan distance

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ip} - x_{jp}|$$

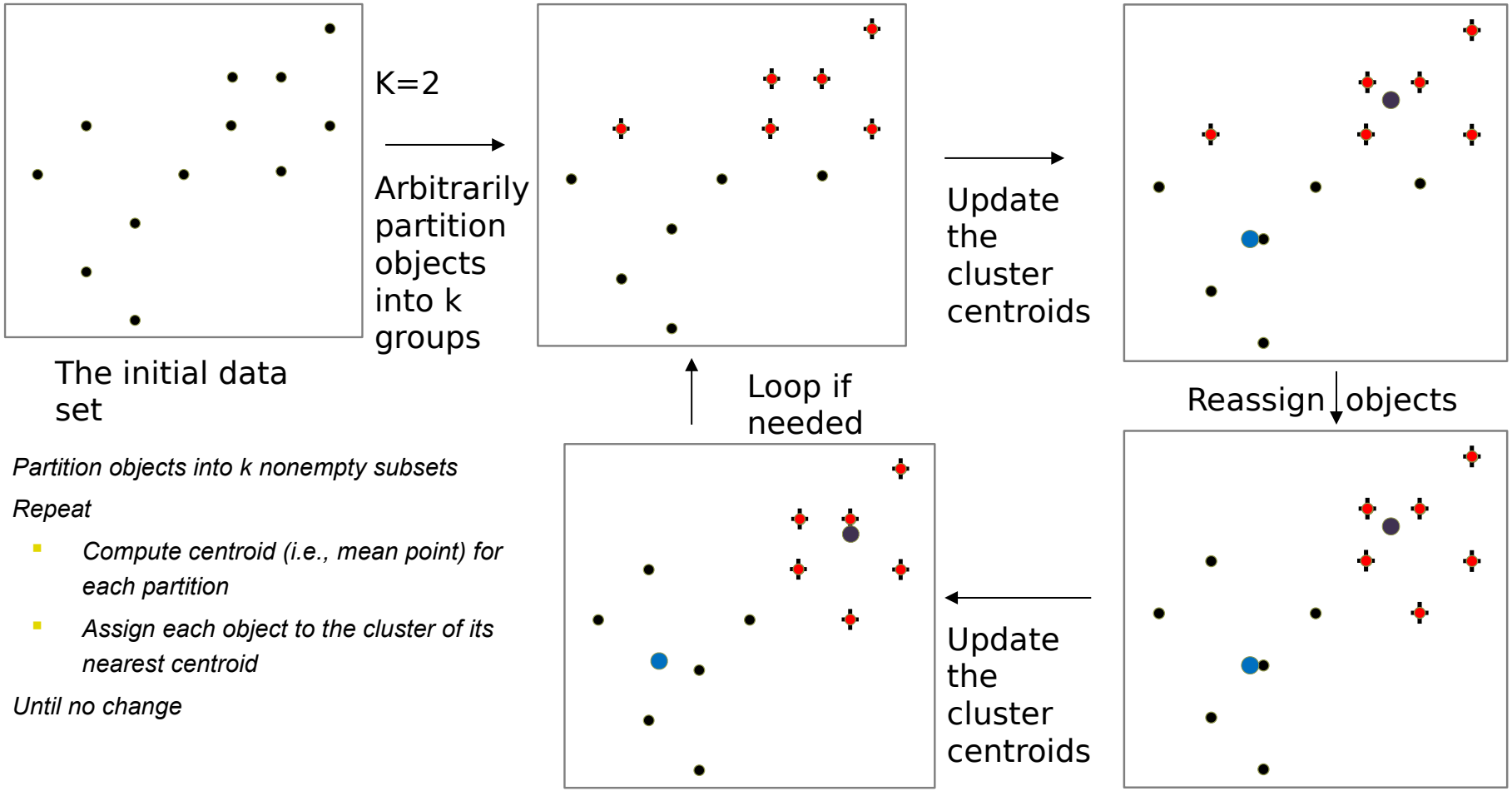
# Major Clustering Approaches

- Partitioning approach:
  - Construct various partitions and then evaluate them by some criterion, e.g., minimizing the sum of square errors
  - Typical methods: k-means, k-medoids, CLARANS
- Hierarchical approach:
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion
  - Typical methods: Diana, Agnes, BIRCH, CAMELEON
- Density-based approach:
  - Based on connectivity and density functions
  - Typical methods: DBSACN, OPTICS, DenClue
- Grid-based approach:
  - based on a multiple-level granularity structure
  - Typical methods: STING, WaveCluster, CLIQUE

# The K-Means Clustering Method

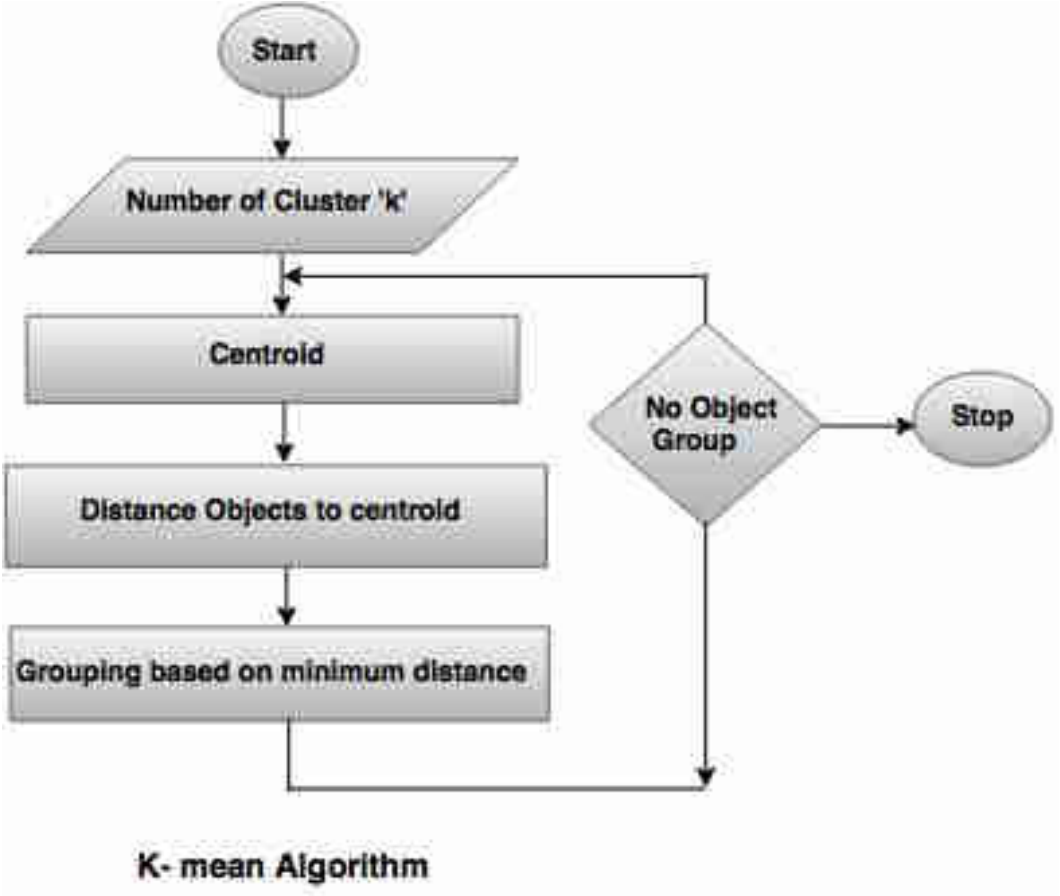
- Given *k*, the *k-means* algorithm is implemented in four steps:
  - Partition objects into *k* nonempty subsets
  - Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., *mean point*, of the cluster)
  - Assign each object to the cluster with the nearest seed point
  - Go back to Step 2, stop when the assignment does not change

# An Example of K-Means Clustering




- *Partition objects into k nonempty subsets*
- *Repeat*
  - *Compute centroid (i.e., mean point) for each partition*
  - *Assign each object to the cluster of its nearest centroid*
- *Until no change*

# K-Means Method




K- mean Algorithm

# K-Means Method → Example

Using K-means clustering, cluster the following data into two clusters and show each step.

{2, 4, 10, 12, 3, 20, 30, 11, 25}

# K-Means Method → Example

Step 1: Randomly assign the means: m1 = 3, m2 = 4

Step 2: Group the numbers close to mean m1 = 3 are grouped into cluster k1 and m2 = 4 are grouped into cluster k2

Step 3: k1 = {2, 3}, k2 = {4, 10, 12, 20, 30, 11, 25}, m1= 2.5, m2 = 16

Step 4: k1 = {2, 3, 4}, k2 = {10, 12, 20, 30, 11, 25}, m1= 3 , m2 = 18

Step 5: k1 = {2, 3, 4, 10}, k2= {12, 20, 30, 11, 25}, m1= 4.75, m2 = 19.6

Step 6: k1 = {2, 3, 4, 10, 11, 12}, k2 = {20, 30, 25}, m1 = 7, m2 = 25

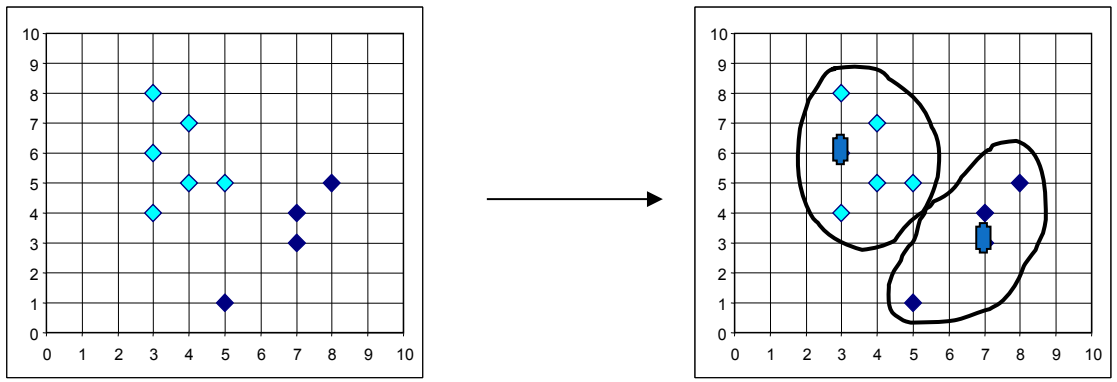
Step 7: k1 = {2, 3, 4, 10, 11, 12}, k2 = {20, 30, 25}, m1 = 7, m2 = 25

Step 8: Stop since The clusters in step 6 and 7 are same.

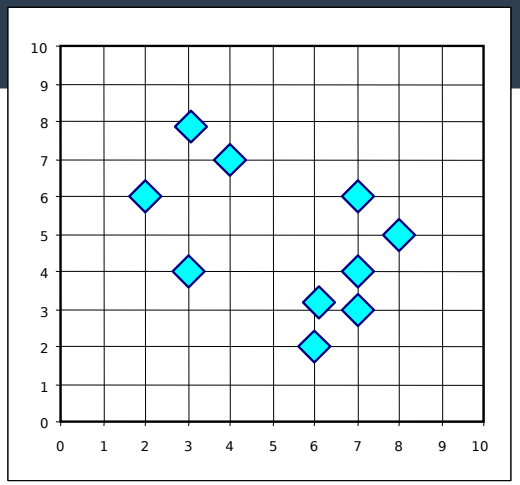
Final answer: k1 = {2, 3, 4, 10, 11, 12} and k2 = {20, 30, 25}

# What Is the Problem of the K-Means Method?

- 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

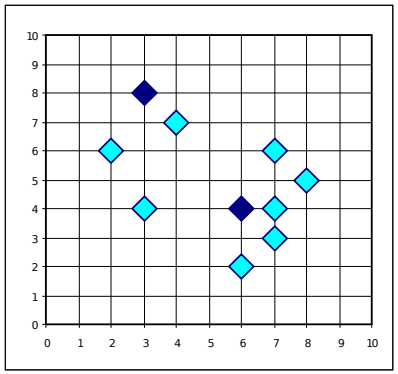


# A Typical K-Medoids Algorithm

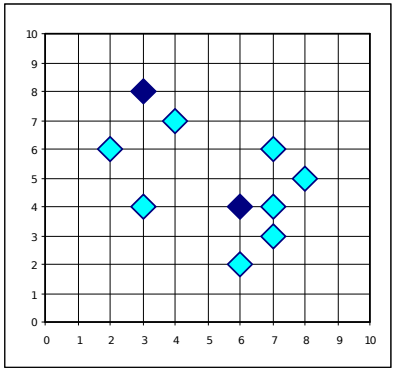


K=2

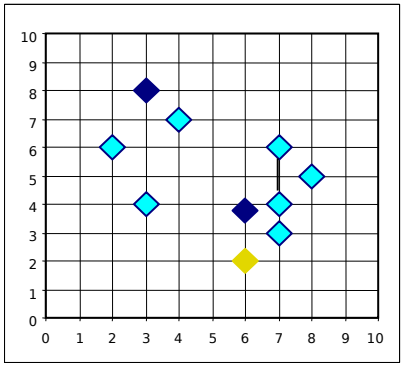
Arbitrary choose k object as initial medoids



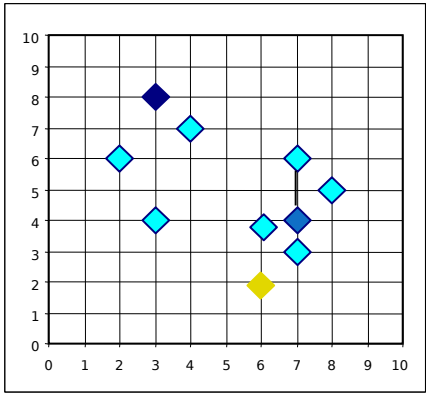
Assign each remaining object to nearest medoids



Randomly select a nonmedoid object, $O_{random}$



Compute total cost of swapping



Total Cost = 26

Swapping O and $O_{random}$

If quality is improved.

**Do loop**

**Until no change**

# Assignment ….!!!

- Explain k-mediod clustering with example
- Practice some numerical related to k-means and k-mediod
- Explain the application areas of clustering.
- Explain DBSCAN