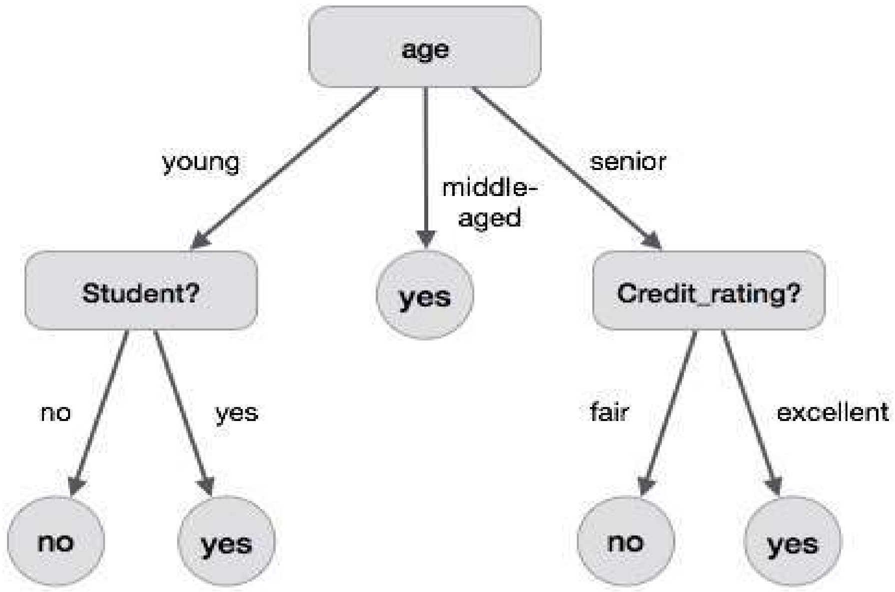
SCS5623-DATA MINING &WAREHOUSING

UNIT III

DECISION TREE INDUCTION

A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.

The following decision tree is for the concept buys_computer that indicates whether a customer at a company is likely to buy a computer or not. Each internal node represents a test on an attribute. Each leaf node represents a class.



The benefits of having a decision tree are as follows :

- It does not require any domain knowledge.
- It is easy to comprehend.
- The learning and classification steps of a decision tree are simple and fast.

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

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

Then, for each attribute A,

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

where Dj / D is the weight of the jth partition.

Info A (D) is the expected information required to classify a tuple from D based on the partitioning by A. The smaller the expected information, the greater the purity of the partitions.

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,

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

//Generating a decision tree form training tuples of data partition D

<u>Algorithm</u> : Generate_decision_tree

```
Input:
Data partition, D, which is a set of training tuples
and their associated class labels.
attribute_list, the set of candidate attributes.
Attribute selection method, a procedure to determine the
splitting criterion that best partitions that the data
tuples into individual classes. This criterion includes a
splitting_attribute and either a splitting point or splitting subset.

Output:
 A Decision Tree

Method
create a node N;

if tuples in D are all of the same class, C then
   return N as leaf node labeled with class C;

if attribute_list is empty then
   return N as leaf node with labeled
   with majority class in D;|| majority voting
```

```
apply attribute_selection_method(D, attribute_list)
to find the best splitting_criterion;
label node N with splitting_criterion;

if splitting_attribute is discrete-valued and
   multiway splits allowed then  // no restricted to binary trees

attribute_list = splitting attribute; // remove splitting attribute
for each outcome j of splitting criterion

   // partition the tuples and grow subtrees for each partition
   let Dj be the set of data tuples in D satisfying outcome j; // a partition

   if Dj is empty then
      attach a leaf labeled with the majority
      class in D to node N;
   else
      attach the node returned by Generate
      decision tree(Dj, attribute list) to node N;
   end for
return N;
```

Tree Pruning

Tree pruning is performed in order to remove anomalies in the training data due to noise or outliers. The pruned trees are smaller and less complex.

Tree Pruning Approaches

Here is the Tree Pruning Approaches listed below −

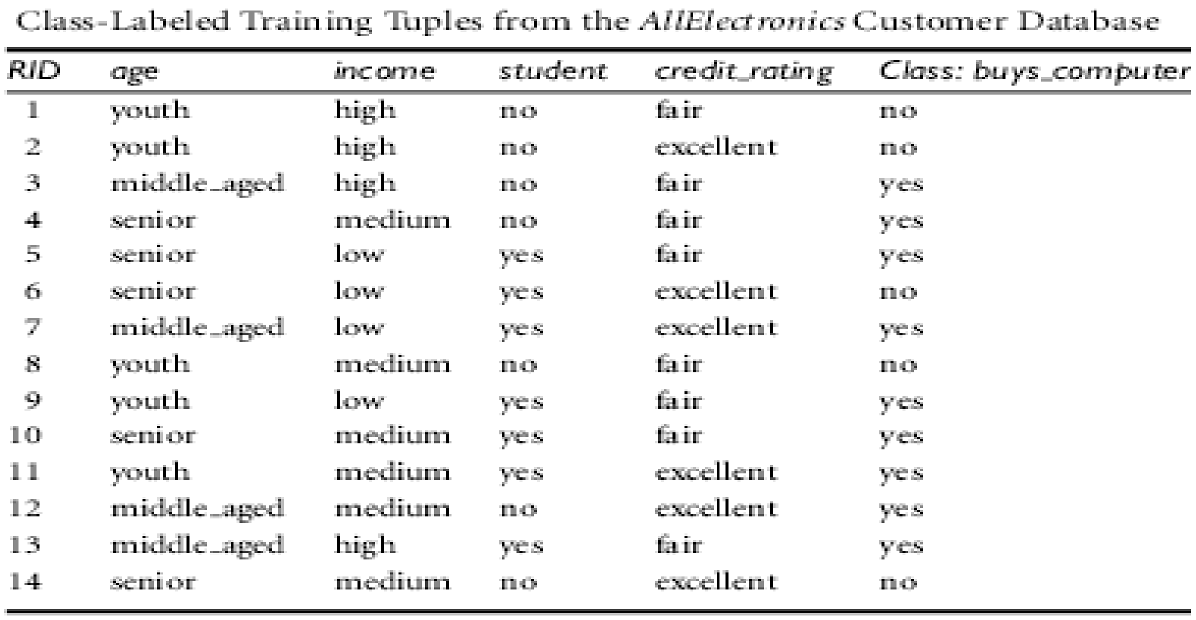
- **Pre-pruning** − The tree is pruned by halting its construction early.
- **Post-pruning -** This approach removes a sub-tree from a fully grown tree.

Cost Complexity

The cost complexity is measured by the following two parameters −

- Number of leaves in the tree, and
- Error rate of the tree.

<u>**Example:**</u>


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

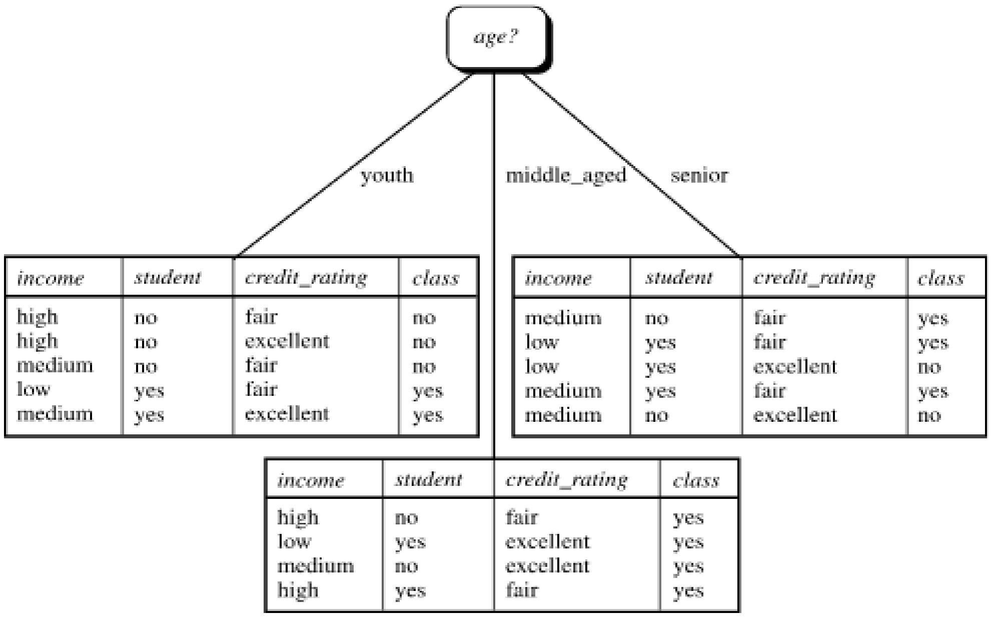
$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

$$\begin{aligned} Info_{age}(D) = {} & \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) \\ & + \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4}\right) \\ & + \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) \\ & = 0.694 \text{ bits.} \end{aligned}$$

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

Similarly, Gain(income)=0.029
Gain(Credit_rating)=0.151
Gain(Student)=0.048

Therefore out of all Gain values obtained, the attribute Age has gained a higher value, and hence it proves itself to be the best splliting attribute. Hence, the decision tree would look like the one given below:



income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	*student*	*credit_rating*	*class*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

income	*student*	*credit_rating*	*class*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

GINI INDEX

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

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

  where p_j is the relative frequency of class *j* in *D*

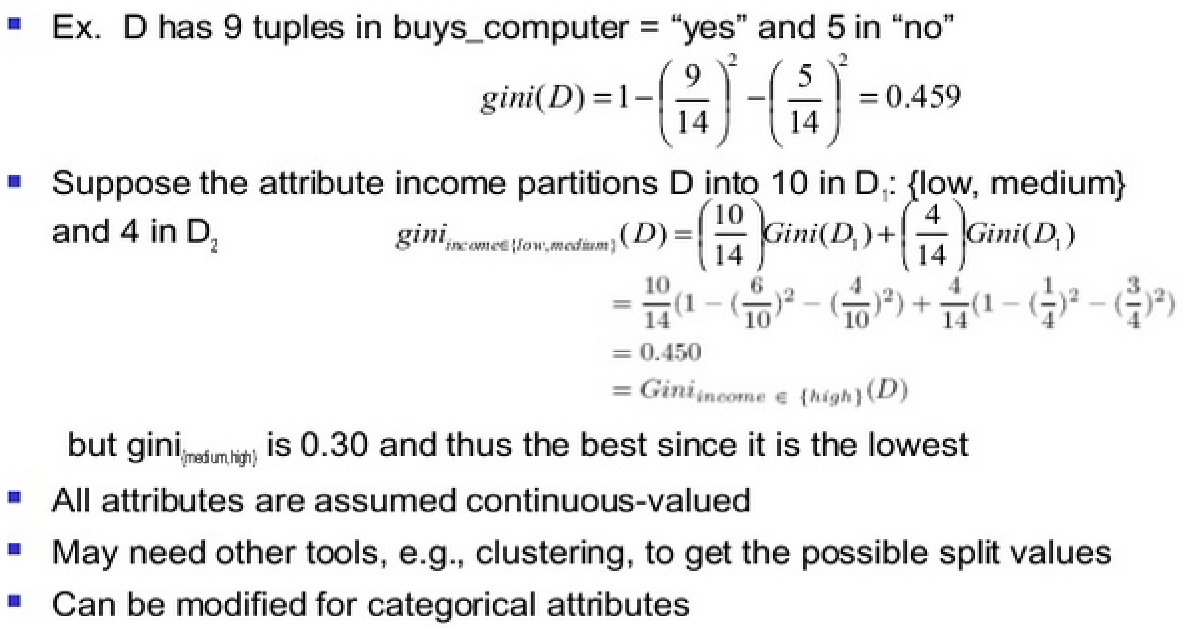
- If a data set *D* is split on A into two subsets D_1 and D_2, the *gini* index *gini*(*D*) is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)



- Ex. D has 9 tuples in buys_computer = “yes” and 5 in “no”

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2

$$gini_{income \in \{low, medium\}}(D) = \left(\frac{10}{14}\right)Gini(D_1) + \left(\frac{4}{14}\right)Gini(D_1)$$

$$= \frac{10}{14}(1 - (\frac{6}{10})^2 - (\frac{4}{10})^2) + \frac{4}{14}(1 - (\frac{1}{4})^2 - (\frac{3}{4})^2)$$

$$= 0.450$$

$$= Gini_{income \in \{high\}}(D)$$

but $gini_{\{medium, high\}}$ is 0.30 and thus the best since it is the lowest

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

BAYESIAN CLASSIFICATION

Bayesian classification is based on Bayes' Theorem. Bayesian classifiers are the statistical classifiers. Bayesian classifiers can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.

Bayesian classification is based on Bayes' Theorem. Bayesian classifiers are the statistical classifiers. Bayesian classifiers can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.

Baye's Theorem

Bayes' Theorem is named after Thomas Bayes. There are two types of probabilities −

- Posterior Probability [P(H/X)]
- Prior Probability [P(H)]

where X is data tuple and H is some hypothesis.

According to Bayes' Theorem,

$$P(H/X)= P(X/H)P(H) / P(X)$$

Naïve Bayesian Classification

It is based on the Bayesian theorem It is particularly suited when the dimensionality of the inputs is high. Parameter estimation for naive Bayes models uses the method of maximum likelihood. In spite over-simplified assumptions, it often performs better in many complex realworld situations.

Advantage: Requires a small amount of training data to estimate the parameters

Example

rec	Age	Income	Student	Credit_rating	Buys_computer
r1	<=30	High	No	Fair	No
r2	<=30	High	No	Excellent	No
r3	31…40	High	No	Fair	Yes
r4	>40	Medium	No	Fair	Yes
r5	>40	Low	Yes	Fair	Yes
r6	>40	Low	Yes	Excellent	No
r7	31…40	Low	Yes	Excellent	Yes
r8	<=30	Medium	No	Fair	No
r9	<=30	Low	Yes	Fair	Yes
r10	>40	Medium	Yes	Fair	Yes
r11	<=30	Medium	Yes	Excellent	Yes
r12	31…40	Medium	No	Excellent	Yes
r13	31…40	High	Yes	Fair	Yes
r14	>40	Medium	No	Excellent	No

Given, X = (age= youth, income = medium, student = yes, credit_rating = fair)

A person belonging to tuple X will buy a computer?

Theory:

Derivation:

D : Set of tuples

- Each Tuple is an ‘n’ dimensional attribute vector
- X : (x1,x2,x3,…. xn)

Let there be ‘m’ Classes : C1,C2,C3…Cm

Naïve Bayes classifier predicts X belongs to Class Ci iff

- ❑ P (Ci/X) > P(Cj/X) for 1<= j <= m , j <> i

Maximum Posteriori Hypothesis

- ❑ P(Ci/X) = P(X/Ci) P(Ci) / P(X)
- ❑ Maximize P(X/Ci) P(Ci) as P(X) is constant

With many attributes, it is computationally expensive to evaluate P(X/Ci).
Naïve Assumption of “class conditional independence”

$$P(X / Ci) = \prod_{k=1}^{n} P(x_k / Ci)$$

P(X/Ci) = P(x1/Ci) * P(x2/Ci) *…* P(xn/ Ci)

Theory applied on previous example:

P(C1) = P(buys_computer = yes) = 9/14 =0.643
P(C2) = P(buys_computer = no) = 5/14= 0.357
P(age=youth /buys_computer = yes) = 2/9 =0.222
P(age=youth /buys_computer = no) = 3/5 =0.600
P(income=medium /buys_computer = yes) = 4/9 =0.444
P(income=medium /buys_computer = no) = 2/5 =0.400
P(student=yes /buys_computer = yes) = 6/9 =0.667
P(student=yes/buys_computer = no) = 1/5 =0.200
P(credit rating=fair /buys_computer = yes) = 6/9 =0.667
P(credit rating=fair /buys_computer = no) = 2/5 =0.400

P(X/Buys a computer = yes) = P(age=youth /buys_computer = yes) * P(income=medium /buys_computer = yes) * P(student=yes /buys_computer = yes) * P(credit rating=fair /buys_computer = yes) = 0.222 * 0.444 * 0.667 * 0.667 = 0.044

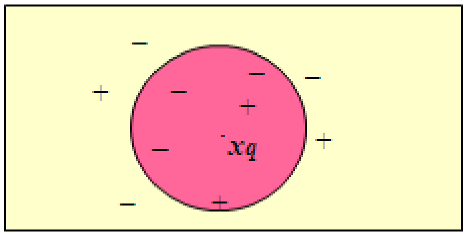
P(X/Buys a computer = No) = 0.600 * 0.400 * 0.200 * 0.400 = 0.019

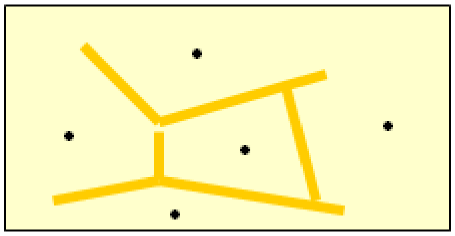
Find class Ci that Maximizes P(X/Ci) * P(Ci)
=>P(X/Buys a computer = yes) * P(buys_computer = yes) = 0.028
=>P(X/Buys a computer = No) * P(buys_computer = no) = 0.007

Prediction : Buys a computer for Tuple X

<u>THE *K*-NEAREST NEIGHBOR CLASSIFIER ALGORITHM</u>

- All instances correspond to points in the n-D space.
- The nearest neighbor are defined in terms of Euclidean distance.
- The target function could be discrete- or real- valued.
- For discrete-valued, the *k*-NN returns the most common value among the k training examples nearest to *xq*.
- Vonoroi diagram: the decision surface induced by 1-NN for a typical set of training examples.





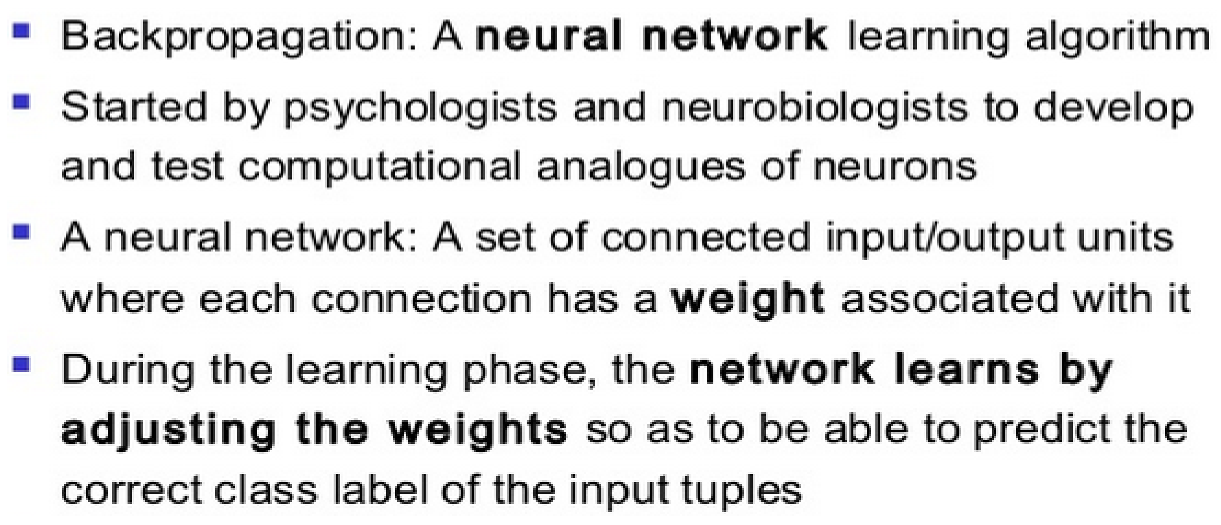
- The k-NN algorithm for continuous-valued target functions

- Calculate the mean values of the *k* nearest neighbors

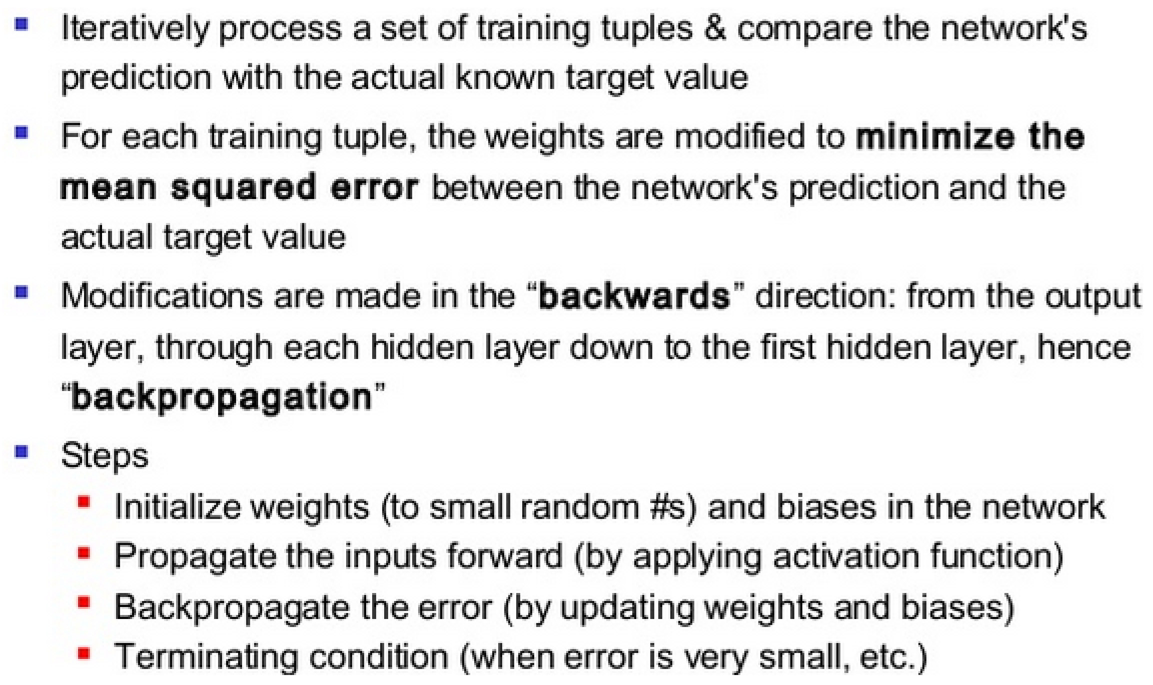
- Distance-weighted nearest neighbor algorithm
    - Weight the contribution of each of the k neighbors according to their distance to the query point x_q
        - giving greater weight to closer neighbors
    - Similarly, for real-valued target functions
- Robust to noisy data by averaging k-nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes.
    - To overcome it, axes stretch or elimination of the least relevant attributes.

OTHER CLASSIFICATION METHODS

1.Classification by Back Propagation Network

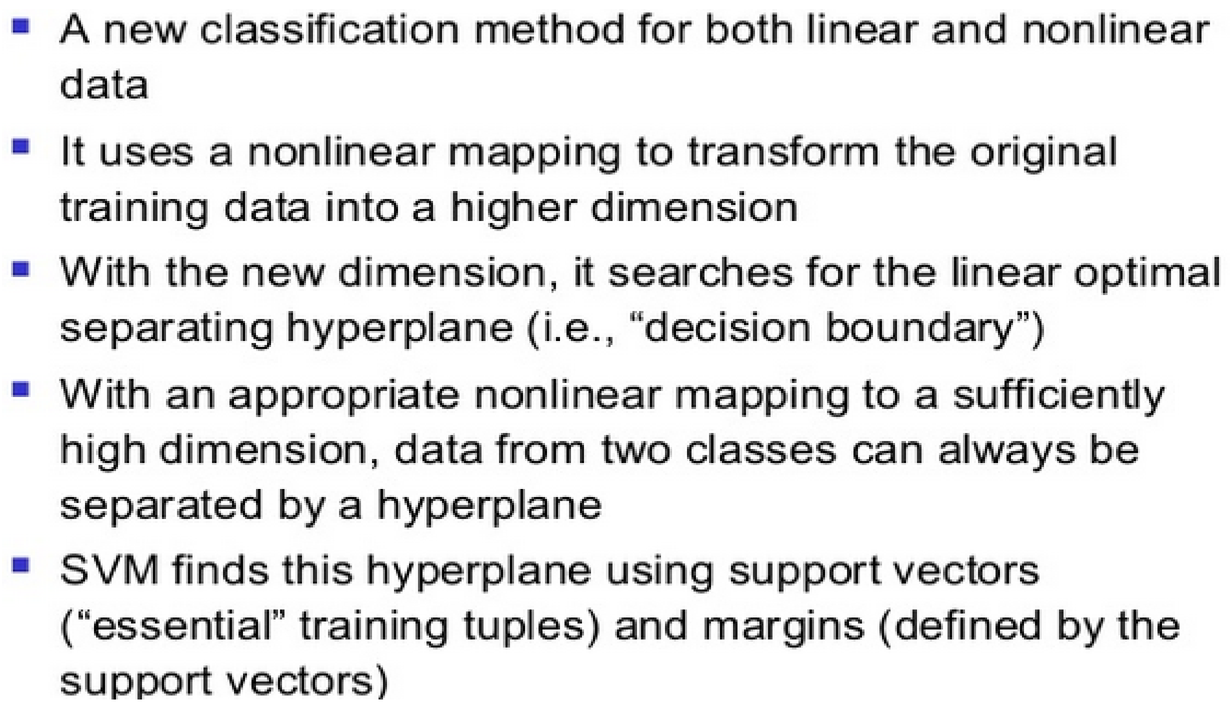


- Backpropagation: A **neural network** learning algorithm
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where each connection has a **weight** associated with it
- During the learning phase, the **network learns by adjusting the weights** so as to be able to predict the correct class label of the input tuples

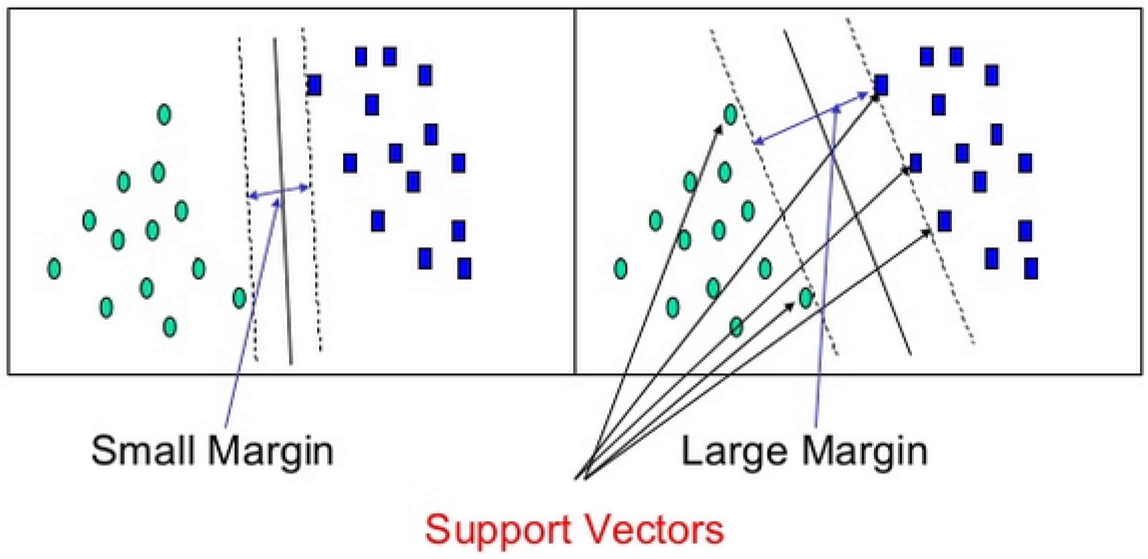


- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to **minimize the mean squared error** between the network's prediction and the actual target value
- Modifications are made in the "**backwards**" direction: from the output layer, through each hidden layer down to the first hidden layer, hence "**backpropagation**"
- Steps
  - Initialize weights (to small random #s) 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.)

2.Support vector machines(SVM)



- A new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyperplane (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using support vectors ("essential" training tuples) and margins (defined by the support vectors)



3.Rule based Classification

IF-THEN Rules

Rule-based classifier makes use of a set of IF-THEN rules for classification. We can express a rule in the following from −

IF condition THEN conclusion

Let us consider a rule R1,

```
R1: IF age=youth AND student=yes THEN buy_computer=yes
```

Points to remember −

- The IF part of the rule is called **rule antecedent** or **precondition**.
- The THEN part of the rule is called **rule consequent**.
- The antecedent part the condition consist of one or more attribute tests and these tests are logically ANDed.
- The consequent part consists of class prediction.

Note − We can also write rule R1 as follows:

```
R1: (age = youth) ^ (student = yes))(buys computer = yes)
```

If the condition holds true for a given tuple, then the antecedent is satisfied.

Rule Extraction

Here we will learn how to build a rule-based classifier by extracting IF-THEN rules from a decision tree.

Points to remember −

- One rule is created for each path from the root to the leaf node.
- To form a rule antecedent, each splitting criterion is logically ANDed.
- The leaf node holds the class prediction, forming the rule consequent.

Rule Induction Using Sequential Covering Algorithm

Sequential Covering Algorithm can be used to extract IF-THEN rules form the training data. We do not require to generate a decision tree first. In this algorithm, each rule for a given class covers many of the tuples of that class.

Some of the sequential Covering Algorithms are AQ, CN2, and RIPPER. As per the general strategy the rules are learned one at a time. For each time rules are learned, a tuple covered by the rule is removed and the process continues for the rest of the tuples. This is because the path to each leaf in a decision tree corresponds to a rule.

Note − The Decision tree induction can be considered as learning a set of rules simultaneously.

The Following is the sequential learning Algorithm where rules are learned for one class at a time. When learning a rule from a class Ci, we want the rule to cover all the tuples from class C only and no tuple form any other class.

```
Algorithm: Sequential Covering

Input:
D, a data set class-labeled tuples,
Att_vals, the set of all attributes and their possible values.

Output:  A Set of IF-THEN rules.
Method:
Rule_set={ }; // initial set of rules learned is empty

for each class c do

   repeat
      Rule = Learn_One_Rule(D, Att_valls, c);
      remove tuples covered by Rule form D;
   until termination condition;

   Rule_set=Rule_set+Rule; // add a new rule to rule-set
end for
return Rule_Set;
```

Rule Pruning

The rule is pruned is due to the following reason −

- The Assessment of quality is made on the original set of training data. The rule may perform well on training data but less well on subsequent data. That's why the rule pruning is required.
- The rule is pruned by removing conjunct. The rule R is pruned, if pruned version of R has greater quality than what was assessed on an independent set of tuples.

FOIL is one of the simple and effective method for rule pruning. For a given rule R,

FOIL_Prune = pos - neg / pos + neg

where pos and neg is the number of positive tuples covered by R, respectively.

Note − This value will increase with the accuracy of R on the pruning set. Hence, if the FOIL_Prune value is higher for the pruned version of R, then we prune R.

4.Genetic Algorithms

The idea of genetic algorithm is derived from natural evolution. In genetic algorithm, first of all, the initial population is created. This initial population consists of randomly generated rules. We can represent each rule by a string of bits.

For example, in a given training set, the samples are described by two Boolean attributes such as A1 and A2. And this given training set contains two classes such as C1 and C2.

We can encode the rule **IF A1 AND NOT A2 THEN C2** into a bit string **100**. In this bit representation, the two leftmost bits represent the attribute A1 and A2, respectively.

Likewise, the rule **IF NOT A1 AND NOT A2 THEN C1** can be encoded as **001**.

Note − If the attribute has K values where K>2, then we can use the K bits to encode the attribute values. The classes are also encoded in the same manner.

Points to remember −

- Based on the notion of the survival of the fittest, a new population is formed that consists of the fittest rules in the current population and offspring values of these rules as well.
- The fitness of a rule is assessed by its classification accuracy on a set of training samples.
- The genetic operators such as crossover and mutation are applied to create offspring.
- In crossover, the substring from pair of rules are swapped to form a new pair of rules.
- In mutation, randomly selected bits in a rule's string are inverted.

5.Rough Set Approach

We can use the rough set approach to discover structural relationship within imprecise and noisy data.

Note − This approach can only be applied on discrete-valued attributes. Therefore, continuous-valued attributes must be discretized before its use.

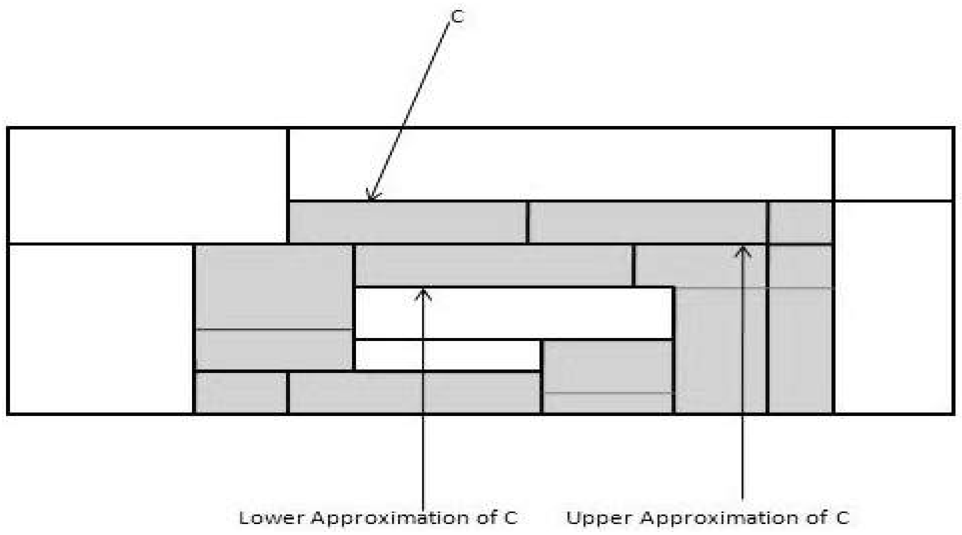
The Rough Set Theory is based on the establishment of equivalence classes within the given training data. The tuples that forms the equivalence class are indiscernible. It means the samples are identical with respect to the attributes describing the data.

There are some classes in the given real world data, which cannot be distinguished in terms of available attributes. We can use the rough sets to **roughly** define such classes.

For a given class C, the rough set definition is approximated by two sets as follows −

- **Lower Approximation of C** − The lower approximation of C consists of all the data tuples, that based on the knowledge of the attribute, are certain to belong to class C.
- **Upper Approximation of C** − The upper approximation of C consists of all the tuples, that based on the knowledge of attributes, cannot be described as not belonging to C.

The following diagram shows the Upper and Lower Approximation of class C:



6.Fuzzy Set Approaches

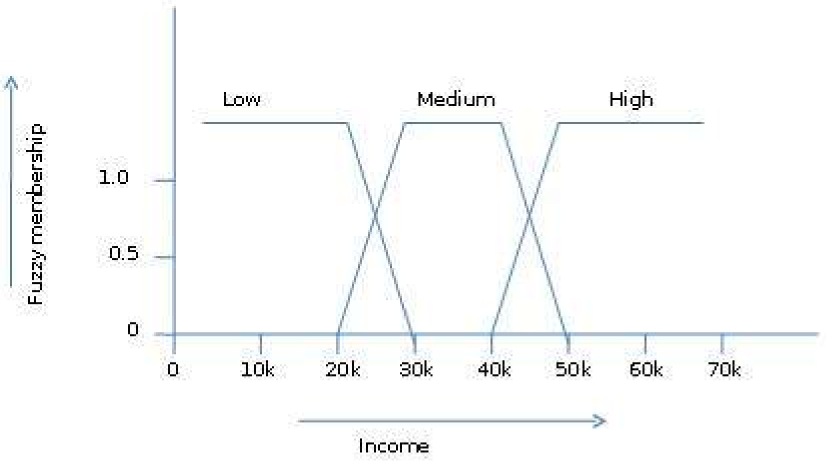
Fuzzy Set Theory is also called Possibility Theory. This theory was proposed by Lotfi Zadeh in 1965 as an alternative the **two-value logic** and **probability theory**. This theory allows us to work at a high level of abstraction. It also provides us the means for dealing with imprecise measurement of data.

The fuzzy set theory also allows us to deal with vague or inexact facts. For example, being a member of a set of high incomes is in exact (e.g. if $50,000 is high then what about $49,000 and $48,000). Unlike the traditional CRISP set where the element either belong to S or its complement but in fuzzy set theory the element can belong to more than one fuzzy set.

For example, the income value $49,000 belongs to both the medium and high fuzzy sets but to differing degrees. Fuzzy set notation for this income value is as follows −

$m_{medium\_income}(\$49k)=0.15$ and $m_{high\_income}(\$49k)=0.96$

where ‘m’ is the membership function that operates on the fuzzy sets of medium_income and high_income respectively. This notation can be shown diagrammatically as follows −



PREDICTION

- Prediction is similar to classification
    - First, construct a model
    - Second, use model to predict unknown value
        - Major method for prediction is regression
            - Linear and multiple regression
            - Non-linear regression
- Prediction is different from classification
    - Classification refers to predict categorical class label
    - Prediction models continuous-valued functions

Predictive Modeling in Databases

- Predictive modeling: Predict data values or construct generalized linear models based on the database data.
- One can only predict value ranges or category distributions
- Method outline:
        - Minimal generalization
        - Attribute relevance analysis
        - Generalized linear model construction
        - Prediction
- Determine the major factors which influence the prediction
    - Data relevance analysis: uncertainty measurement, entropy analysis, expert judgement, etc.
- Multi-level prediction: drill-down and roll-up analysis

Regress Analysis and Log-Linear Models in Prediction

Linear regression: Y = a + b X

- Two parameters , a and b specify the line and are to be estimated by using the data at hand.
- Usesthe least squares criterion to the known values of Y1, Y2, …, X1, X2, ….

Multiple regression: Y = b0 + b1 X1 + b2 X2.

Many nonlinear functions can be transformed into the above.

Log-linear models:

The multi-way table of joint probabilities is approximated by a product of lower-order tables.

Probability: *p(a, b, c, d)* = a*ab* b*accad* d*bcd*

CLUSTER ANALYSIS

Cluster is a group of objects that belongs to the same class. In other words, similar objects are grouped in one cluster and dissimilar objects are grouped in another cluster.

What is Clustering?

Clustering is the process of making a group of abstract objects into classes of similar objects.

Points to Remember

- A cluster of data objects can be treated as one group.
- While doing cluster analysis, we first partition the set of data into groups based on data similarity and then assign the labels to the groups.
- The main advantage of clustering over classification is that, it is adaptable to changes and helps single out useful features that distinguish different groups.

Applications of Cluster Analysis

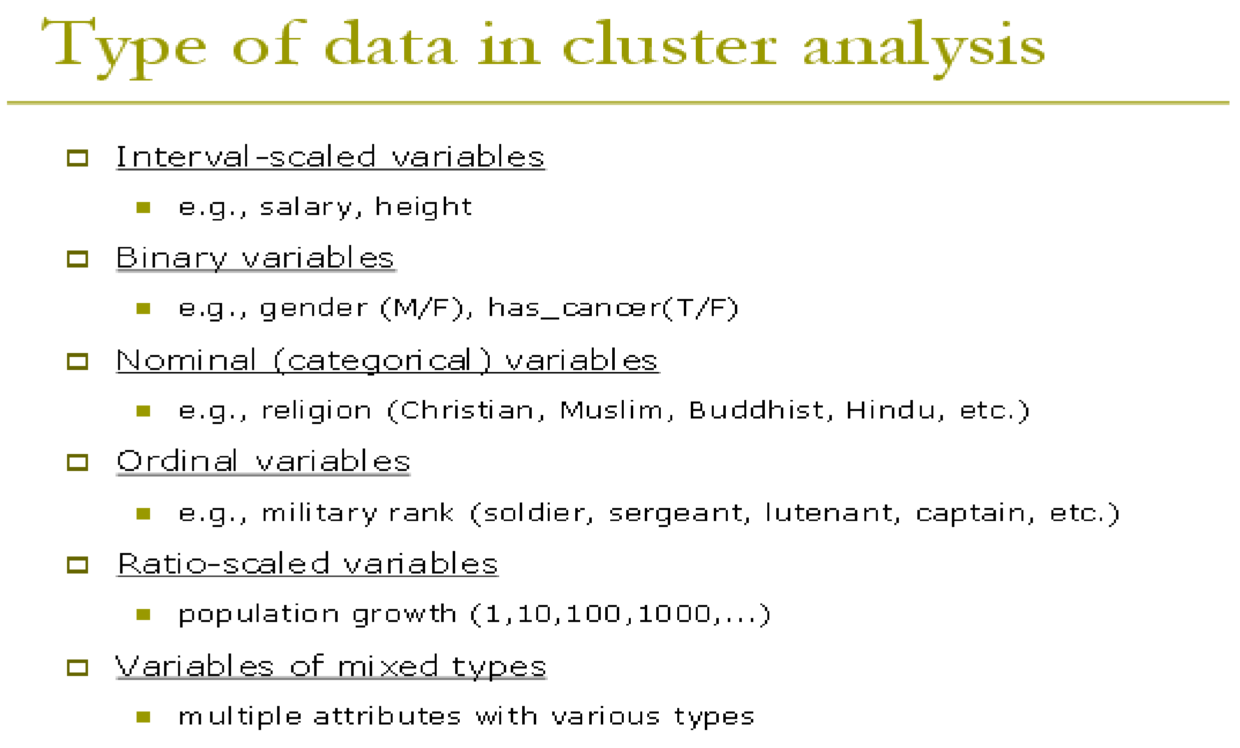
- Clustering analysis is broadly used in many applications such as market research, pattern recognition, data analysis, and image processing.
- Clustering can also help marketers discover distinct groups in their customer base. And they can characterize their customer groups based on the purchasing patterns.
- In the field of biology, it can be used to derive plant and animal taxonomies, categorize genes with similar functionalities and gain insight into structures inherent to populations.
- Clustering also helps in identification of areas of similar land use in an earth observation database. It also helps in the identification of groups of houses in a city according to house type, value, and geographic location.
- Clustering also helps in classifying documents on the web for information discovery.

- Clustering is also used in outlier detection applications such as detection of credit card fraud.
- As a data mining function, cluster analysis serves as a tool to gain insight into the distribution of data to observe characteristics of each cluster.

Requirements of Clustering in Data Mining

The following points throw light on why clustering is required in data mining −

- **Scalability** − We need highly scalable clustering algorithms to deal with large databases.
- **Ability to deal with different kinds of attributes** − Algorithms should be capable to be applied on any kind of data such as interval-based (numerical) data, categorical, and binary data.
- **Discovery of clusters with attribute shape** − The clustering algorithm should be capable of detecting clusters of arbitrary shape. They should not be bounded to only distance measures that tend to find spherical cluster of small sizes.
- **High dimensionality** − The clustering algorithm should not only be able to handle low-dimensional data but also the high dimensional space.
- **Ability to deal with noisy data** − Databases contain noisy, missing or erroneous data. Some algorithms are sensitive to such data and may lead to poor quality clusters.
- **Interpretability** − The clustering results should be interpretable, comprehensible, and usable.



Type of data in cluster analysis

- Interval-scaled variables
  - e.g., salary, height
- Binary variables
  - e.g., gender (M/F), has_cancer(T/F)
- Nominal (categorical) variables
  - e.g., religion (Christian, Muslim, Buddhist, Hindu, etc.)
- Ordinal variables
  - e.g., military rank (soldier, sergeant, lutenant, captain, etc.)
- Ratio-scaled variables
  - population growth (1,10,100,1000,...)
- Variables of mixed types
  - multiple attributes with various types

<u>CATEGORIZATION OF MAJOR CLUSTERING METHODS</u>

Clustering methods can be classified into the following categories −

- Partitioning Method
- Hierarchical Method
- Density-based Method
- Grid-Based Method
- Model-Based Method
- Constraint-based Method

Partitioning Method

Suppose we are given a database of ‘n’ objects and the partitioning method constructs ‘k’ partition of data. Each partition will represent a cluster and k ≤ n. It means that it will classify the data into k groups, which satisfy the following requirements −

- Each group contains at least one object.
- Each object must belong to exactly one group.

Points to remember −

- For a given number of partitions (say k), the partitioning method will create an initial partitioning.
- Then it uses the iterative relocation technique to improve the partitioning by moving objects from one group to other.

Hierarchical Methods

This method creates a hierarchical decomposition of the given set of data objects. We can classify hierarchical methods on the basis of how the hierarchical decomposition is formed. There are two approaches here −

- Agglomerative Approach
- Divisive Approach

Agglomerative Approach

This approach is also known as the bottom-up approach. In this, we start with each object forming a separate group. It keeps on merging the objects or groups that are close to one another. It keep on doing so until all of the groups are merged into one or until the termination condition holds.

Divisive Approach

This approach is also known as the top-down approach. In this, we start with all of the objects in the same cluster. In the continuous iteration, a cluster is split up into smaller clusters. It is down until each object in one cluster or the termination condition holds. This method is rigid, i.e., once a merging or splitting is done, it can never be undone.

Approaches to Improve Quality of Hierarchical Clustering

Here are the two approaches that are used to improve the quality of hierarchical clustering −

- Perform careful analysis of object linkages at each hierarchical partitioning.

- Integrate hierarchical agglomeration by first using a hierarchical agglomerative algorithm to group objects into micro-clusters, and then performing macro-clustering on the micro-clusters.

Density-based Method

This method is based on the notion of density. The basic idea is to continue growing the given cluster as long as the density in the neighborhood exceeds some threshold, i.e., for each data point within a given cluster, the radius of a given cluster has to contain at least a minimum number of points.

Grid-based Method

In this, the objects together form a grid. The object space is quantized into finite number of cells that form a grid structure.

Advantage

- The major advantage of this method is fast processing time.
- It is dependent only on the number of cells in each dimension in the quantized space.

Model-based methods

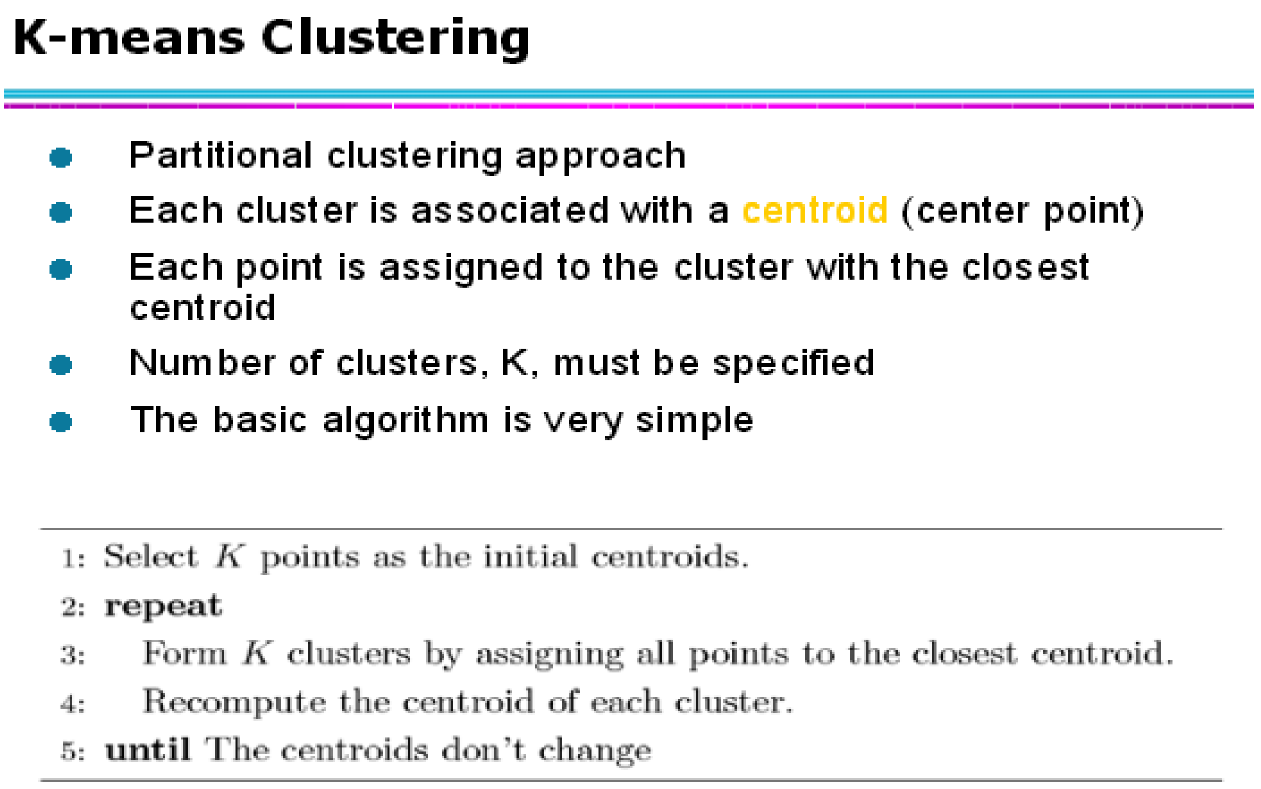
In this method, a model is hypothesized for each cluster to find the best fit of data for a given model. This method locates the clusters by clustering the density function. It reflects spatial distribution of the data points.

This method also provides a way to automatically determine the number of clusters based on standard statistics, taking outlier or noise into account. It therefore yields robust clustering methods.

Constraint-based Method

In this method, the clustering is performed by the incorporation of user or application-oriented constraints. A constraint refers to the user expectation or the properties of desired clustering results. Constraints provide us with an interactive way of communication with the clustering process. Constraints can be specified by the user or the application requirement.

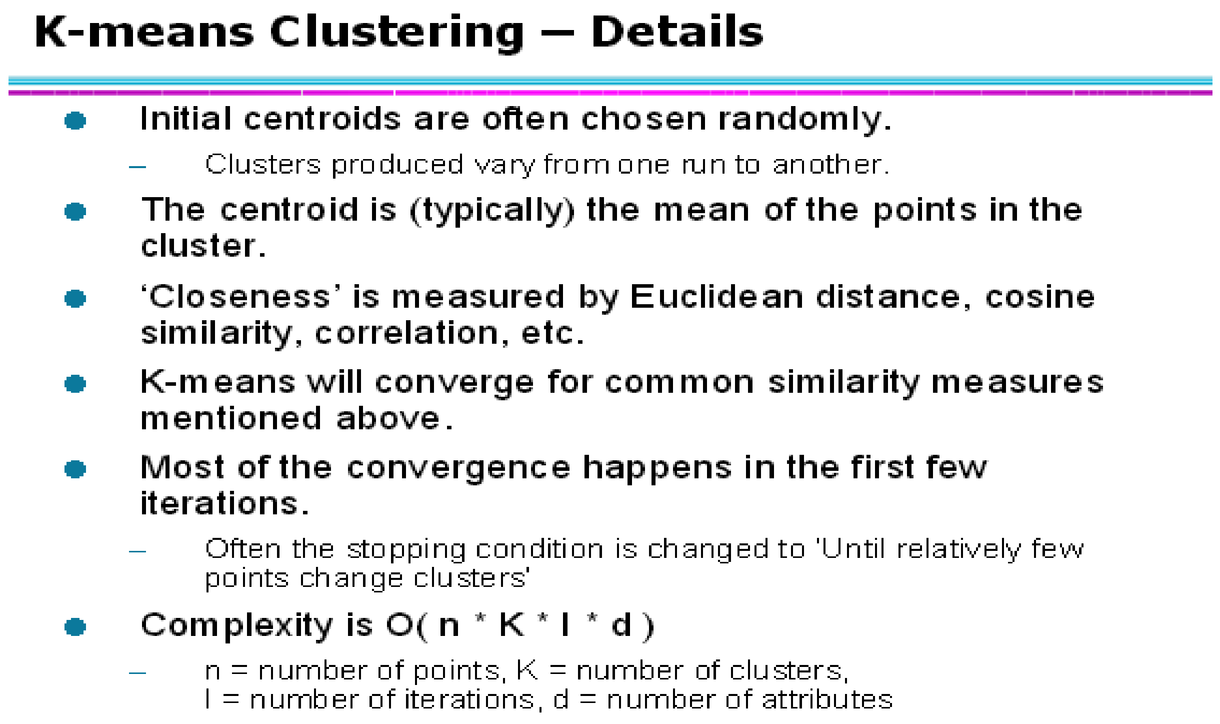
PARTITIONING METHODS



K-means Clustering

- Partitional clustering approach
- Each cluster is associated with a centroid (center 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
```



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 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

<u>HIERARCHICAL METHODS</u>

Hierarchical Clustering

- Two main types of hierarchical clustering
  - Agglomerative:
    - 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:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

Agglomerative Clustering Algorithm

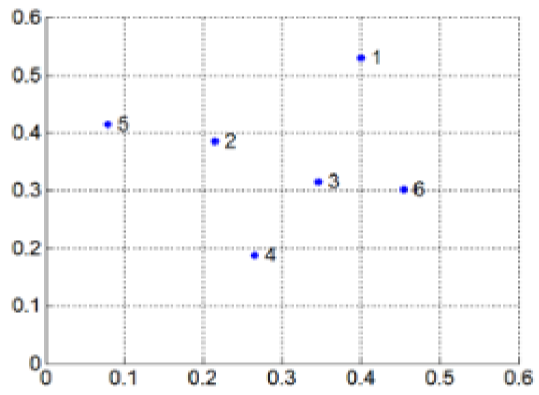
- More popular hierarchical clustering technique
- Basic algorithm is straightforward

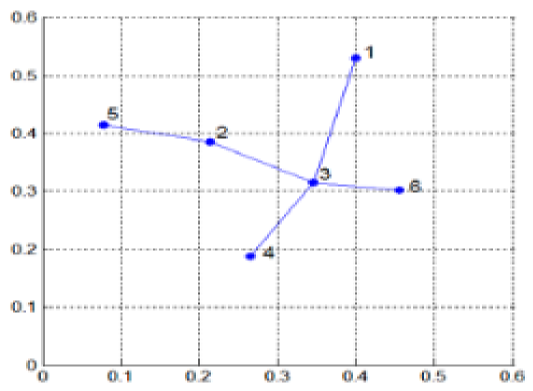
```
1. Compute the proximity matrix
2. Let each data point be a cluster
3. Repeat
4.         Merge the two closest clusters
5.         Update the proximity matrix
6. Until only a single cluster remains
```

- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

MST: Divisive Hierarchical Clustering

- Build MST (Minimum Spanning Tree)
  - Start with a tree that consists of any point
  - In successive steps, look for the closest pair of points (p, q) such that one point (p) is in the current tree but the other (q) is not
  - Add q to the tree and put an edge between p and q





MST: Divisive Hierarchical Clustering

- Use MST for constructing hierarchy of clusters

Algorithm 7.5 MST Divisive Hierarchical Clustering Algorithm

```
1: Compute a minimum spanning tree for the proximity graph.
2: repeat
3:    Create a new cluster by breaking the link corresponding to the largest distance
      (smallest similarity).
4: until Only singleton clusters remain
```