# UNIT 5

## CHALLENGES THE WEB POSES FOR EFFECTIVE RESOURCE & KNOWLEDGE DISCOVERY

→ The web seems to be too big for effective data warehousing & data mining
The size of the web is in the order of hundreds of terabytes & is rapidly growing. It is barely possible to replicate, store & integrate all the data on the web.

→ The complexity of web pages is far greater than that of any traditional text document collection.
Web pages lack a unifying structure. The web is like a digital library, but the documents are not sorted in any order.

→ The web page is a highly dynamic information source.
Not only does the web grow rapidly, but its info. is also constantly updated.

→ The web serves a broad diversity of user communi-ties. The web connects more than 100 million workstations & its user communities are rapidly expanding.

→ Only a portion of the info. on the web is useful or relevant. It is said that 99% of the web info. is useless to 99% of the users.

- Compared to DOM-based methods, the segments obtained by VIPS are more semantically aggregated.
- Contents w/ different topics are distinguished as seperate blocks.

## MINING THE WEB PAGE LAYOUT STRUCTURE

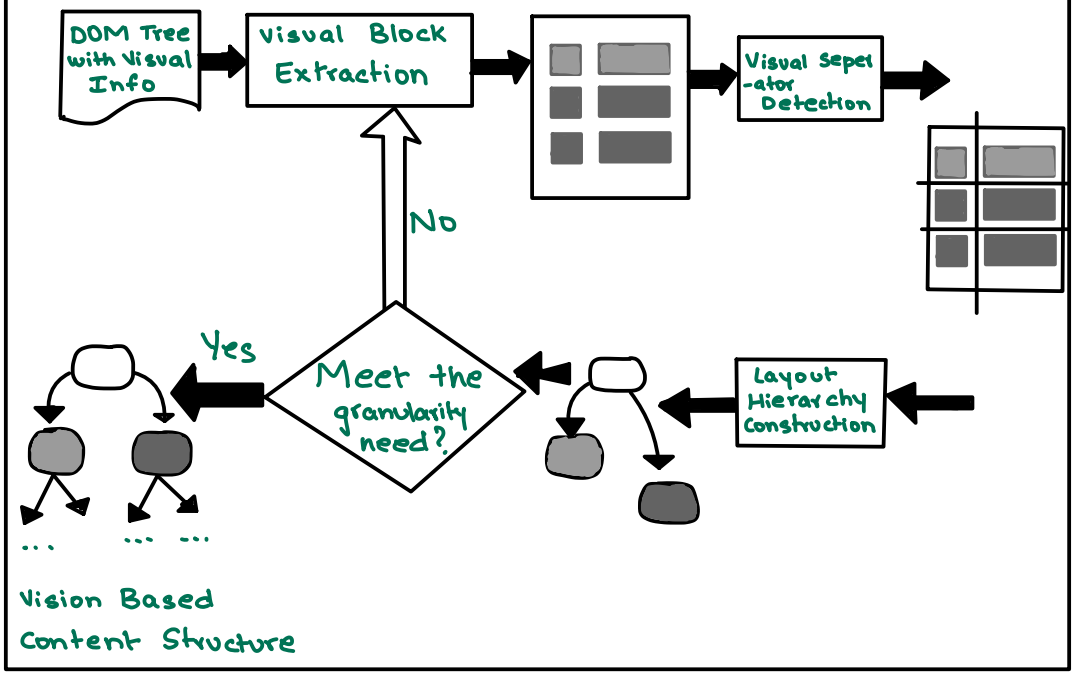
- A web page has more structure than tradit-ional plain text. Web pages are also regarded as semi-structured data.
- The basic structure of a webpage is its DOM (Document Object Model) structure.
- The DOM structure of a webpage is a tree structure, where every HTML tag corresponds to a node in the tree.
- Due to the flexibility of HTML syntax, many web pages don't obey the W3C specifications, which may result in errors in the DOM tree structure.
- Initially, the DOM tree was introduced for presentation in the browser rather than description of the semantic structure of the webpage.
- In the sense of human perception, people always view a web page as different semantic objects rather than a single object.
- Research efforts show that users always expect that certain parts of a webpage appear at certain positions on the page.
- VIPS (VIsion-based Page Segmentation) is an algo that aims to extract the semantic structure of a webpage based on its visual presentation.
- The algo 1st extracts all the suitable blocks from the HTML DOM tree, then finds seperators b/w these blocks.

contd.

# MINING THE WEB'S LINK STRUCTURES

- In addition to retrieving pages that are relevant, you also hope that the pages retrieved (by a web search) will be of high quality, or authoritative on the topic.
- The secrecy of authority is hiding in webpage linkages.
- The web consists of not only pages, but also of hyperlinks pointing to another page
- when an author of a web page includes a hyperlink on their webpage to another page, it is considered the author's endorsement of that page.
  The collective endorsement of a page by diff. authors may indicate the importance of the page & may naturally lead to the discovery of authoritative web pages.
- ∴ the tremendous amount of web linkage info provides rich info about the relevance, the quality, & the structure of the web's contents, & thus is a rich source for web mining.
- A hub is 1 or a set of Web pages that provides collections of links to authorities.
  In general, a good hub is a page that points to many good authorities; a good authority is a page pointed to by many good hubs.
- HITS (Hyperlink-Induced Topic Search) – an algo. that uses hubs.
  First, HITS uses the query term to collect a starting set of, say, 200 pages from an index-based search engine. These pages form the root set.

The process flow of vision-based page segmentation algo.




Second, a weight-propagation phase is initiated. This iterative process determines numerical estimates of hub & authority weights.
Finally, the algo outputs a short list of the pages with large hub weights, & the pages with large authority weights for the given search topic.

## WEB USAGE MINING

- It mines weblog records to discover user access patterns of web pages.
- Analyzing & exploring regularities in weblog records can identify potential customers for electronic commerce, enhance the quality & delivery of Internet information services to the end user, & improve web server system performance.
- A web server usually registers a log entry for every access of a web page. It includes the URL requested, IP address from which the request originated, & the timestamp.
- In developing techniques for web usage mining, we consider the following:
  - → Although it is encouraging to imagine the various potential applications of weblog file analysis, note that the success of such applications depends on how much valid & reliable knowledge can be discovered from the large raw log data.
  - → With the available URL, time, IP address, & web page content info, a multidimensional view can be constructed on the weblog database, & multidimensional OLAP analysis can be performed to find the top N users, top N accessed webpages, & so on.
  - → Data mining can be performed on weblog records to find association patterns, sequential patterns, & trends of web accessing.

- Because weblog data provides info about what kind of users that will access what kind of web pages, weblog info can be integrated w/ web content & web linkage structure mining to help web page ranking, web document classification, & the construction of a multilayered web Info base.

## DM FOR FINANCIAL DATA ANALYSIS

Financial data collected in the banking & financial industry are often relatively complete, reliable, & of high quality, which facilitates systematic data analysis & data mining. A few cases:

→ Design & construction of data warehous-es for multidimensional data analysis & data mining - MultiD data analysis methods should be used to analyze the general properties of such data.
Eg: To view the debt & revenue changes by month, region or sector along w/ statisti-cal info.

→ Loan payment prediction & customer credit policy analysis - " " " are critical to the business of a bank. Many factors can strongly or weakly influence loan payment performance & customer credit rating. DM methods like attribute selection & attribute relevance ranking can help identify important factors & eliminate irrelevant ones.

# DM FOR THE RETAIL INDUSTRY

The retail industry is a major application area for data mining, since it collects huge amounts of data on sales, customer shopping history, goods transport-ation, consumption, & service. The quantity of data continues to expand rapidly, especially due to the increasing ease & availability of e-commerce.

Retail DM can help identify customer buying behaviors, discover customer shopping patterns & trends, reduce the cost of business, etc.

Some examples:

→ **Multidimensional analysis of sales, customers, products, time & region:** The retail industry requires timely data regarding customer needs, product sales, trends, & fashions, as well as quality, cost & profit. It is ∴ imp. to provide powerful multiD analysis & visualization tools.

→ **Analysis of the effectiveness of sales campaigns:** careful analysis " " can help improve company profits. Association analysis may disclose which items are likely to be purchased together w/ the items on sale, especially in comparison w/ the sales before the campaign.

→ **Product recommendation & cross-referencing of items:** By mining associations from sales records, one may discover that a customer that buys a particular product is likely to buy another set of items. Such info can be used to form product recom.

→ **Customer retention - analysis of customer loyalty:** with customer loyalty card info, one can register sequences of purchases of particular customers. Customer loyalty & purchase trends can be analyzed systematically.

→ **Classification & clustering of customers for targeted marketing** – " " can be used for customer group identification & targeted marketing.
Eg: customers w/ similar behaviors regardi-ng loan payments may be identified by multiD clustering.

→ **Detection of money laundering & other financial crimes** – For this, it is important to integrate info from multiple databases, as long as they are potentially related to the study. Multipe data analysis tools can then be used to detect unusual patterns, such as large amounts of cash flow.

# DM FOR INTRUSION DETECTION

- An intrusion can be defined as any set of actions that threaten the integrity, confidentiality, or availability of a network resource.
- Most commercial IDS are limiting & do not provide a complete solution. Such systems employ a **misuse detection strategy** – it searches for patterns of program or user behaviour that match known intrusion scenarios, which are stored as **signatures**.
- If a pattern match is found, this signals an event for which an alarm is raised.
- Human security analysts evaluate the alarms to decide what action to take.
- **DRAWBACK** – misuse detection can only identify cases that match the signatures
- **DRAWBACK** – As systems are not static, the signatures need to be updated whenever new software versions arrive, or changes in network configuration occur.

# UNIT 4

## REQUIREMENTS OF CLUSTERING

**SCALABILITY** - Many clustering algos work well on small datasets; however, a large database may contain millions of objects. Clustering on a sample of a given large dataset may lead to biased results. ∴ Highly scalable algos are needed

**ABILITY TO DEAL WITH DIFF TYPES OF ATTRIBUTES** - Many algos are designed to cluster numerical data. However, app$^n$s may require clustering other types of data, like binary, categorical & ordinal data or a mixture of these types.

**DISCOVERY OF CLUSTERS WITH ARBITRARY SHAPE** - Many algos determine clusters based on Euclidean or Manhatten distance measures. Algos based on such measures tend to find spherical clusters wl similar size & density. However, a cluster could be of any shape. Algos have to be able to detect clusters of arbitrary shape.

**ABILITY TO DEAL WITH NOISY DATA** - Most databases contain outliers or missing, or erroneous data. Some clustering algos are sensitive to such data & may lead to clusters of poor quality

**MINIMAL REQUIREMENTS FOR DOMAIN KNOWLEDGE TO DETERMINE INPUT PARAMETERS** - Many clustering algos require users to i/p certain parameters in cluster analysis. The clustering results can be quite sensitive to i/p parameters. Parameters are often difficult to determine.

**INCREMENTAL CLUSTERING & INSENSITI-VITY TO THE ORDER OF I/P RECORDS** - Some algos cannot incorporate newly inserted data into clustering structur-es & must determine a new clustering from scratch. Some algos are sensitive to the order of i/p data, such that, it may return dramatically different clusterings depending on the order of the i/p objects.

**HIGH DIMENSIONALITY** - A database can contain several dimensions of attributes. Finding clusters in high-dimensional space is challenging.

**CONSTRAINT-BASED CLUSTERING** - Real world app$^n$s may need to perform clustering under various kinds of constraints.

## TYPES OF DATA IN CLUSTER ANALYSIS

① DATA MATRIX - represents n objects, such as people, with p variables (attributes), such as age, height & so on.
The structure is in the form of a relational table, or n-by-p matrix

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

② DISSIMILARITY MATRIX - This stores a collection of proximities that are available for all pairs of n objects. It is represented by an n-by-n table

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

## * DISTANCE MEASURES *

**Euclidean distance**

$$d(i,j) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

**MANHATTAN (CITY BLOCK) DISTANCE**

$$d(i,j) = |x_2 - x_1| + |y_2 - y_1|$$

**JACCARD COEFFICIENT**

$$J = \frac{f_{11}}{f_{11} + f_{10} + f_{01}}$$

**COSINE SIMILARITY**

$$\cos(x, y) = \frac{x \cdot y}{\|x\| * \|y\|}$$

**CORRELATION**

$$r_{xy} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

**Dissimilarity between binary variables.** Suppose that a patient record table (Table 7.2) contains the attributes *name, gender, fever, cough, test-1, test-2, test-3,* and *test-4*, where *name* is an object identifier, *gender* is a symmetric attribute, and the remaining attributes are asymmetric binary.

For asymmetric attribute values, let the values *Y* (*yes*) and *P* (*positive*) be set to 1, and the value *N* (*no* or *negative*) be set to 0. Suppose that the distance between objects (patients) is computed based only on the asymmetric variables. According to Equation (7.10), the distance between each pair of the three patients, Jack, Mary, and Jim, is

$d(i,j) = \frac{r+s}{q+r+s}$

$$d(Jack, Mary) = \frac{0+1}{2+0+1} = 0.33$$
$$d(Jack, Jim) = \frac{1+1}{1+1+1} = 0.67$$
$$d(Mary, Jim) = \frac{1+2}{1+1+2} = 0.75$$

A relational table where patients are described by binary attributes.

| name | gender | fever | cough | test-1 | test-2 | test-3 | test-4 |
|---|---|---|---|---|---|---|---|
| Jack | M | Y | N | P | N | N | N |
| Mary | F | Y | N | P | N | P | N |
| Jim | M | Y | Y | N | N | N | N |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

These measurements suggest that Mary and Jim are unlikely to have a similar disease because they have the highest dissimilarity value among the three pairs. Of the three patients, Jack and Mary are the most likely to have a similar disease.

until eventually each object is in 1 cluster, or until a termination condition holds. contd.

## CLUSTERING METHODS

### * PARTITIONING:

Given a database of n objects, a partitioning method constructs K partitions of the data, where each partition represents a cluster, & $K \leq n$. That is, it classifies the data into K groups, which satisfy:
(1) each group must contain at least 1 object.
(2) each object must belong to exactly 1 group

The 2nd requirement can be relaxed in some fuzzy partitioning techniques.
The general criterion of a good partitioning is that objects in the same cluster are "close" or related, whereas objects of different clusters are "far apart" or very different.

### * HIERARCHICAL:

This creates a hierarchical decomposition of the given data objects. It can be classified as being either agglomerative or divisive, based on how the hierarchical decomposition is formed.
Agglomerative/bottom-up approach - starts w/ each object forming a seperate group. It successively merges the objects or groups that are close to one another, until all of the groups are merged into 1, or until a termination condition holds.
Divisive/top-down" - starts w/ all objects in the same cluster. In each iteration, a cluster is split up into smaller clusters,

contd.

Hierarchical methods suffer from the fact that once a step (merge or split) is done, it can never be undone. This technique doesn't allow for correction of erraneous decisions.

* **DENSITY-BASED:**

Their general idea is to continue growing the given cluster as long as the density in the "neighborhood" exceeds some threshold. Such a method can be used to filter out noise (outliers) & discover clusters of arbitrary shape. DBSCAN & DENCLUE.

* **GRID-BASED:**

These methods quantize the object space into a finite no. of cells that form a grid structure. All of the clustering operations are performed on the grid structure. The main advantage of this is its fast processing time.

* **MODEL-BASED:**

These methods hypothesize a model for each of the clusters & find the best fit of the data to the given model. This algo may locate clusters by constructing a density function that reflects the spatial distribution of the data points

* **CLUSTERING HIGH-D DATA:**

As the no. of dimensions increases, the data becomes increasingly sparse so that the distance measurement b/w pairs of points become meaningless & the avg density of the points anywhere in the data is likely to be low.
CLIQUE & PROCLUS are 2 influential subspace clustering methods, which search for clusters in subspaces of the data, rather than over the entire data space.

* **CONSTRAINT-BASED CLUSTERING:**

Performs clustering by incorporating user-specified or application-oriented constraints. Semi-supervised clustering employs pairwise constraints to improve the quality of the resulting clustering.

# SUM EXAMPLES

## ① COSINE

$x = \{3, 2, 0, 5\}$

$y = \{1, 0, 0, 0\}$

$$\cos(x,y) = \frac{x \cdot y}{\|x\| * \|y\|}$$

$$= \frac{3*1 + 2*0 + 0*0 + 5*0}{\sqrt{3^2+2^2+5^2} * \sqrt{1^2}}$$

$$= \frac{3}{\sqrt{38}} = 0.486 = 0.49$$

## ② CORRELATION

$x = (1,1,1,0,0,1)$

$y = (0,1,0,1,0,1)$

$\bar{x} = 4/6 = 0.6$

$\bar{y} = 3/6 = 0.5$

| $X = x - \bar{x}$ | $Y = y - \bar{y}$ | $XY$ | $X^2$ | $Y^2$ |
|---|---|---|---|---|
| 0.4 | -0.5 | -0.2 | 0.16 | 0.25 |
| 0.4 | 0.5 | 0.2 | 0.16 | 0.25 |
| 0.4 | -0.5 | -0.2 | 0.16 | 0.25 |
| -0.6 | 0.5 | -0.3 | 0.09 | 0.25 |
| -0.6 | -0.5 | 0.3 | 0.09 | 0.25 |
| 0.4 | 0.5 | 0.2 | 0.16 | 0.25 |
| 0.4 | 0 | 0 | 0.82 | 1.5 |

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

$$= \frac{0}{\sqrt{0.82 \times 1.5}} = 0$$

## ③ JACCARD

$x = (1, 1, 1, 0, 0, 1)$

$y = (0, 1, 0, 1, 0, 1)$

$$sim(x,y) = \frac{f_{11}}{f_{11} + f_{10} + f_{01}}$$

$$= \frac{2}{2+2+1}$$

$$= \frac{2}{5}$$

$$= 0.4$$

# PARTITIONING METHODS

## ① KMEANS

Algo:

- First, it randomly selects k of the objects, each of which represents a cluster center/mean.
- For each of the remaining objects, an object is assigned to the cluster to which it is the most similar.
- It then computes the new mean for each cluster.

* **Square-error criterion** : $E = \sum_{i=1}^{k} \sum_{p \in C_i} |p_i - m_i|^2$ (obj; mean)

## ② K-MEDOIDS

→ Instead of using the mean of objects in a cluster
→ as a reference point, we pick actual objects to represent the clusters.
Each remaining object is clustered w/ the representative object to which it is the most similar.
→ The algo iterates until each representative object is actually the medoid, or most centrally located object of its clusters.

* **Absolute-error criterion**: $E = \sum_{j=1}^{k} \sum_{p \in C_j} |p - o_j|$ (obj; representative obj)

# DENSITY-BASED METHODS

① DBSCAN (Density-Based Spatial Clustering

→ ε-neighborhood - neigbourhood wlin a radius ε of a given object
→ Core object - An obj. who's ε-neigbourhood has at least MinPts objects.
→ An obj p is directly density-reachable from q if p is wlin the ε-neigbourhood of q & q is a core obj.

A density-based cluster is a set of density-connected objects that is maximal wrt density-reachability. Every object not contained in any cluster is considered noise.

- DBSCAN searches for clusters by checking the ε-neighborhood of each point.
- If the ε-neighborhood of p contains more than MinPts, a new cluster w/ p as a core object is created.
- DBSCAN then iteratively collects density-reachable objects from the core objects, which may involve the merge of a few clusters.
- The process terminates when no new point can be added to any cluster.

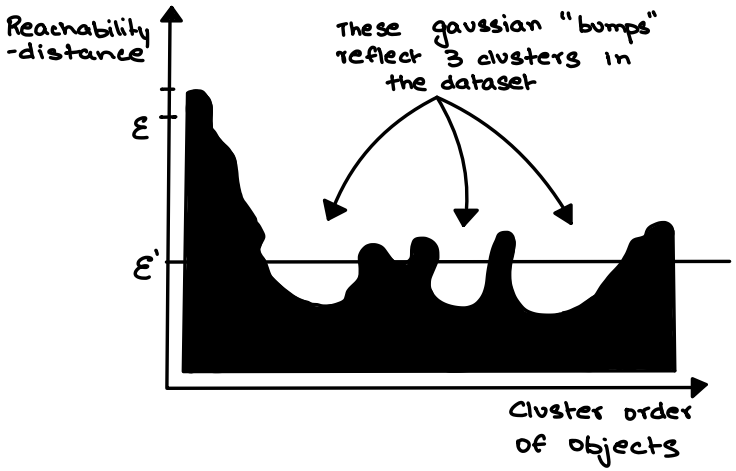
# OPTICS

→ The **core-distance** of an obj p is the smallest ε' value that makes {p} a core object.

→ The **reachability-distance** of an obj q wrt another obj p is the max of the core-distance of p & the Euclidean distance blw p & q.

- OPTICS creates an ordering of the objects, additionally storing the core-distance & reachability-distance for each object.
- An algo was proposed to extract clusters based on the ordering info produced by OPTICS.
- OPTICS doesn't explicitly segment the data into clusters. It produces a visualization of Reachability distances.

(Higher-density points should be processed first, so we find these points first.)




# DENCLUE

→ This method is built on the following ideas:

(1) the influence of each data point can be formally modeled using an **influence function** – describes the impact of a data point wlin its neighborhood.

(2) the overall density of the data space can be modeled analytically as – the sum of the influence function applied to all data points.

(3) clusters can be determined mathematically by identifying density attractors – local maxima of the overall density function.

# CLUSTERING HIGH-D DATA – CLIQUE

→ CLIQUE was the first algo proposed for dimension-growth subspace clustering in highD space.

→ Here, each clustering process starts at singleD subspaces & grows upward to higherD ones.

→ CLIQUE partitions each dimension like a grid structure & determines whether a cell is dense based on the no. of points it contains

CLIQUE performs multiD clustering in **2 steps:**

① CLIQUE partitions the d-D data space into nonoverlapping rectangular units, identifying the dense units among these. This is done for each D. (Dimension)

The subspaces representing these dense units are intersected to form a candidate search space for dense units dense units of highD.

↓

# OUTLIER ANALYSIS

-> Data objects that do not comply with the general behavior of the data are called **outliers**.
-> Outliers can be caused by **measurement** or **execution errors**, or can represent **valid data**.
-> Many DM algos try to minimize the influence of outliers, or eliminate them. This, however, could result in **loss of important info** — like in the case of fraud detection, where outliers may indicate fraudulent activity.
-> Outlier mining - outlier detection & analysis.
-> The outlier mining prob can be viewed as:
(1) define **what data** can be considered as inconsistent
(2) find an efficient **method to mine the outliers**
-> **Data visualization** methods are weak in detecting outliers in data wl many categorical attributes or in data of highD.
-> Approaches for outlier detection:

- statistical approach
- distance-based approach
- density-based local outlier approach
- deviation-based approach

(2) CLIQUE generates a minimal description for each cluster:
For each cluster it determines the maximal region that covers the cluster of connected dense units.
It then determines a minimal cover (logic description) for each cluster.

## SUMS :

K-MEANS

Q1. {2, 3, 4, 10, 11, 12, 20, 25, 30}, K = 2

Assume: Mean1 = 4, Mean2 = 20

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

Calc. mean: $\text{Mean1} = \frac{42}{6} = 7$, $\text{Mean2} = \frac{75}{3} = 25$

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

M1 = 7, M2 = 25

∴ K1 = {2, 3, 4, 10, 11, 12}
K2 = {20, 25, 30}

$|x_2 - x_1| + |y_2 - y_1|$

K = 3 Manhattan D

Q2. A1 = (2,10) A2 = (2,5) A3 = (8,4) A4 = (5,8)
A5 = (7,5) A6 = (6,4) A7 = (1,2) A8 = (4,9) Initial - A1, A4, A7

(1)

| | A1(2,10) Ce1 | A4(5,8) Ce2 | A7(1,2) Ce3 | Clusters |
|---|---|---|---|---|
| A1 | (0) | 3+2 = 5 | 1+8 = 9 | Ce1 |
| A2 | 0+5 = 5 | 3+3 = 6 | 1+3 = (4) | Ce3 |
| A3 | 6+6 = 12 | 3+4 = (7) | 7+2 = 9 | Ce2 |
| A4 | 3+2 = 5 | (0) | 4+6 = 10 | Ce2 |
| A5 | 5+5 = 10 | 2+3 = (5) | 6+3 = 9 | Ce2 |
| A6 | 4+6 = 10 | 1+4 = (5) | 5+2 = 7 | Ce2 |
| A7 | 1+8 = 9 | 4+6 = 10 | (0) | Ce3 |
| A8 | 2+1 = 3 | 1+1 = (2) | 3+7 = 10 | Ce2 |

C1 = {A1}, C2 = {A3, A4, A5, A6, A8}, C3 = {A2, A7}

Ce1 = (2,10) Ce2 = $\left(\frac{8+5+7+6+4}{5}, \frac{4+8+5+4+9}{5}\right) = \left(\frac{30}{5}, \frac{30}{5}\right) = (6,6)$ Ce3 = $\frac{2+1}{2}, \frac{5+2}{2} = (2,4)$

After 1 more iteration,
ans -> Ce1 = (3,10)
Ce2 = (7,5)
Ce3 = (2,4)

*incomplete

contd

(2) K-MEDOID.

# UNIT-3

## DECISION TREE INDUCTION

**Hunt's Algo:**

A decision tree is grown in a recursive fashion by partitioning the training records into successively purer subsets.

$D_t$ → set of training records

$y = \{y_1, y_2, \ldots y_c\}$ → class labels

Step1 → If all the nodes in $D_t$ belong to the same class $y_t$, the t is a leaf node labelled $y_t$.

Step2 → If $D_t$ contains records that belong to more than 1 class, an attribute test condition is selected to partition the records into smaller subsets. A child node is created for each outcome of the test condition. The algo is then recursively applied to each algo.

## DESIGN ISSUES OF DT INDUCTION

A learning algo for decision trees must address the following 2 issues:

① **How should the training records be split?**

Each recursive step of the tree-growing process must select an attribute test cond. to divide the records. To implement this, the algo must provide a method for for specifying the test condition for diff attribute types & a measure for evaluating the goodness of fit for each cond.

② **How should the splitting procedure stop?**

A stopping condition is needed to terminate the tree-growing process.

A possible strategy is to continve expanding a node until all records belong to the same class

### GINI INDEX

$$= 1 - \sum_{i=1}^{c} (p_i)^2$$

($p_i$ → prob. of class i)

| class | Var1 | Var2 |
|---|---|---|
| A | 0 | 33 |
| A | 0 | 54 |
| A | 0 | 56 |
| A | 0 | 42 |
| A | 1 | 50 |
| B | 1 | 55 |
| B | 1 | 31 |
| B | 0 | -4 |
| B | 1 | 77 |
| B | 0 | 49 |

Calc. Gini index for Var1

Var1 → 1 : 4/10, 0 : 6/10

Var1 == 1 → A : 1/4, B : 3/4

$G.I = 1 - \left(\left(\frac{1}{4}\right)^2 + \left(\frac{3}{4}\right)^2\right) = 0.375$

Var1 == 0 → A : 4/6, B : 2/6

$G.I = 1 - \left(\left(\frac{4}{6}\right)^2 + \left(\frac{2}{6}\right)^2\right) = 0.444$

weighted sum $= \frac{4}{10} * 0.375 + \frac{6}{10} * 0.444 = 0.41667$

* calc. Gini index for all Vars (here → Var1 & Var2). The attribute with the lower Gini index value is the "best attribute."

### INFORMATION GAIN

(same table)

① For Var1

Var1 → 1 : 4/10, 0 : 6/10

**Entropy:**

Var1 == 1 : $\frac{4}{10} * \log_2\left(\frac{4}{10}\right)$

Var1 == 0 : $\frac{6}{10} * \log_2\left(\frac{6}{10}\right)$

$$\Rightarrow -\left[\left(\frac{4}{10} * \log_2 \frac{4}{10}\right) + \left(\frac{6}{10} * \log_2 \frac{6}{10}\right)\right]$$

$$= -(-0.159 - 0.133)$$

$$= 0.292$$

* calc. info gain for all Vars (here → Var1 & Var2). The attribute with the lower info gain valve is the best attribute."

## DECISION TREE INDUCTION ALGO

E -> training records, F -> Attribute set

```
TreeGrowth (E, F)
 if stopping_cond (E, F) = true then
      leaf = createNode();
      leaf.label = Classify (E)
      return leaf
else
      root = createNode()
      root.test_cond = find_best_split
                                (E, F)
      let V = {v | v is a possible
            outcome of root.test_cond}
    for each v ∈ V do
       Ev = {e | root.test_cond (e) = v
                          & e ∈ E}
       child = TreeGrowth (Ev, F)
       add child as descendent of
          root & label the edge
          (root -> child) as v
    end for
 end if
return root
```

## CHARACTERISTICS OF DT INDUCTION

* It is a nonparametric approach for building classification models.
* Finding an optimal decision tree is an NP-Complete problem.
* Techniques developed for constructing decision trees are computationally inexpensive.
* DTs, especially smaller-sized trees, are relatively easy to interpret.
* DTs provide an expressive representation for learning discrete-valued functions
* DT algos are quite robust to the presence of noise
* The presence of redundant var's doesn't adversely affect the accuracy of DTs.
* Since most DTs employ a top-down approach, the no. of records become smaller as we traverse down the tree.
* A subtree can be replicated multiple times in a DT — this makes the DT more complex than necessary & difficult to interpret.

--

learning algos that continue to refine their models even when few training records are available.

## MODEL OVERFITTING

- The errors committed by a classification model

  **Training errors**: No. of miscalculation errors committed on training records.

  **Generalization errors**: Expected error of the model on previously unseen records

- A good model must have low training error & low Generalization error.
- A model that fits the training data too well can have a poorer generalization error than a model wl a higher training error. This is called **model overfitting**.

### CAUSES

* Due to **Presence of Noise**

  Errors due to exceptional cases are often unavoidable & establish the min error rate achievable by any classifier.

* Due to **Lack of Representative Samples**

  Models that make their classification decisions based on a small no. training records are susceptible to overfitting.

  Such models can be generated because of lack of representative samples in the training data & --

# HANDLING OVERFITTING IN DT INDUCTION

## PREPRUNING

- The tree-growing algo is halted before generating a full-grown tree that perfectly fits the entire training data.
- For this, a more restrictive stopping condition must be used
- The advantage is that it avoids generating overly complex subtrees that overfit the training data.
- It is difficult to choose the right threshold for early termination
- Too high of a threshold will result in underfitted models, while a threshold that is set too low may not be sufficient to overcome the model overfitting problem.

## POST PRUNING

- The DT is initially grown to its max size.
- This is followed by a tree-pruning step which proceeds to trim the fully grown tree in a bottom-up fashion.
- The tree-pruning step terminates when no further improvement is observed.
- Gives better results than prepruning as it makes pruning decisions based on a fully grown tree.
- However, here, the additional computations needed to grow the full tree may be wasted when the subtree is pruned.

# RULE-BASED CLASSIFIER

- It is a technique for classifying records using a collection of "if...then" rules.
- The rules for the model are represented in a disjunctive normal form, $R = (r_1 \vee r_2 \vee \dots r_k)$
  - $R$: Rule set
  - $r_1 \vee r_2 \vee \dots r_k$: classification rules
- Each classification rule can be expressed as:

$$r_i : (\text{condition}_i) \rightarrow y_i$$

  - $(\text{condition}_i)$: precondition / rule antecedent
  - $y_i$: rule consequent
- $\text{condition}_i = (A_1 \ op \ v_1) \wedge (A_2 \ op \ v_2) \wedge \dots (A_k \ op \ v_k)$
- $(A_j, v_j)$ is an attribute-value pair
- $op = \{=, \neq, \leq, \geq, <, >\}$

- A rule $r$ covers a record $x$ if the precondition of $r$ matches the attributes of $x$.
  $r$ is said to be fired or triggered whenever it covers a given record.

**Example**

$r_1$: (Gives Birth = no) $\wedge$ (Aerial Creature = yes) $\rightarrow$ Birds

$r_2$: (Gives Birth = no) $\wedge$ (Aquatic Creature = yes) $\rightarrow$ Fish

- It classifies a test record based on the rule triggered by the record.

* Mutually Exclusive Rules:

The rules in a rule set R are mutually exclusive if no 2 rules in R are triggered by the same record. Ensures that every record is covered by at most 1 rule in R.

contd.

* Exhaustive Rules:

R has exhaustive coverage if there is a rule for each combination of attribute values. Ensures every record is covered by at least 1 rule.

* Ordered Rules:

Rules are ordered in decreasing order of priority. (can be defined in many ways)

* Unordered Rules:

Allows a test record to trigger multiple rules & considers the consequent of each rule as a vote for that class. Votes are tallied to determine the label.