# Unsupervised Learning (CLUSTERING)

Unsupervised Learning: In this type of problems, the dataset does not have any target or outcome variable.

Q) If there is no data on what needs to be predicted, then what can such algorithms do?

⇒ pattern analysis on historical data to do recommendations.

## Association Rule Mining:

→ Example: shopping possibility within (Bread, Butter, Milk, Jam).

So, certain rules will be there which checks for certain occurrance of patterns and does recommendations

→ these rules are automatically inferred from the data (machine-learned), not created by human being.

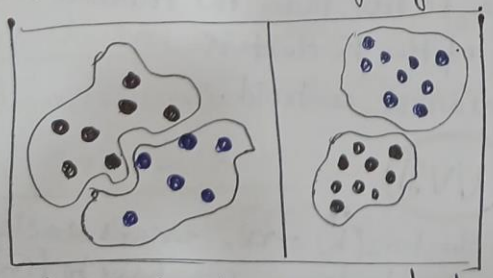
## CLUSTERING

- making groups of objects with meaning/relevancy and objects in any group have similarity in properties.

→ So, when an object (new) is to be placed in any cluster, the similarity has to be checked and placed else work as 'outlier'.

## K-Means Clustering

- K in K-Means stands for number of clusters to be created.
- these K clusters satisfy some tightness criteria.
- points in a cluster are tightly packed and have some associated meaning.



right side clustering is better than the left side.

- it finds local optima but not global optima.

K-Means Clustering Contd.

## • Preliminaries for K-Means

**1) Centroid:** central point of a triangle. Let three points of a triangle are $(x_1, y_1), (x_2, y_2), (x_3, y_3)$.

centroid of the triangle $(x, y)$ is $C(x, y) = \left(\frac{x_1+x_2+x_3}{2}, \frac{y_1+y_2+y_3}{2}\right)$.

It is similarly extended for K clusters and represented as $C_K = (x^k, y^k)$

**2) Within-Cluster Sum of Squares (WCSS)**

$$WCSS(C_K) = \sum_{i=1}^{|C_k|} (x_i - x^k)^2 + (y_i - y^k)^2$$

where $(x^K, y^k)$ = centroid of the points in cluster $C_K$

$|C_k|$ = number of points in cluster

WCSS is basically sum of square of distance between points in cluster and centroid.

(*) ⇒ If points are tightly clustered, then WCSS will be small number.

**3) Inertia** ⇒ It is sum of WCSS values for a group of clusters.

For k clusters, inertia $= \sum_{i=1}^{k} WCSS(C_K)$

For k clusters and n points, it tries to ~~create~~ look that → Sum of WCSS values for K clusters is minimal.

## • Outline of K-Means

K means algo is based on heuristics which reduces the inertia value by redistributing the points among the clusters iteratively.

1) Randomly pick K points from the Sample points as initial cluster centers.
2) Assign each point to the nearest of K points.
   → This creates K clusters
3) Repeat until the change in inertia is less than the tolerance value.
4) Recompute the centroid of each of the K clusters
5) Reassign each point to the nearest centroid.

## • Elbow method for selecting K (VVI)

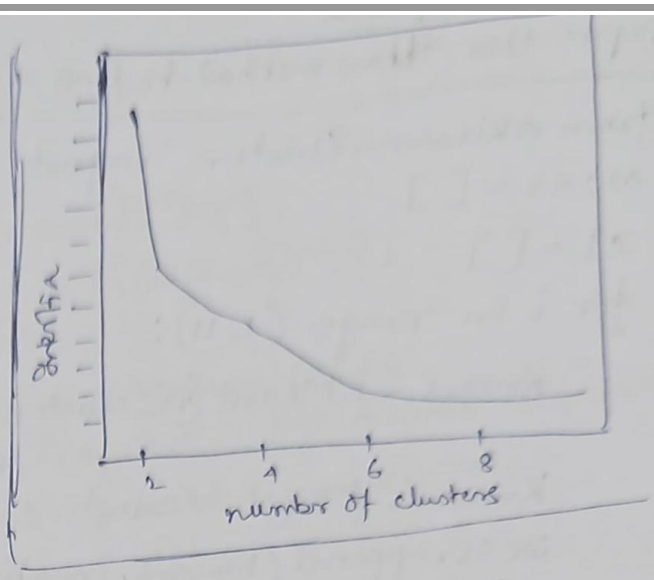
With 'n' points, we can have number of clusters (K) = n, means each point is a cluster. But it is not a good idea. We have to find maximum similarity between points and cluster them.

→ In general, inertia drops steeply when the number of clusters are increased.

...rom 2 to 4, drop is less steep.
4 onwards, steepness of drop reduces.

⇒ When there is a significant reduction in steepness, ANELBOW is formed.

⇒ Elbow points are good choice for value of K.




⇒ most significant elbow formation at K=2
next one at K=4.
So, K may be like K=2,3,4, which may form meaningful clustering.

---

## HANDS ON (Clustering different segment of students)

**Step1: Read the Data** : Each row is a Student. Columns measure a student performance on different dimensions.
ACC → student accuracy of percentage of answers marked correctly.
AVG_ATT → average number of times a student attempted a question.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
% matplotlib inline

dataset = pd.read_csv('.....csv', index_col=0)
dataset.head()
```

**Step2: Analyze the Data** (Let use ACC, AVG_ATT for clustering)

```
dataset.describe().loc[['count','mean','min','std','max']]
```

**Step3: Scale the Data**

```
X = dataset[['ACC','AVG_ATT']].values
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Step4: Use Elbow method to find a good K (let K varies from 2 to 10)

```
from sklearn.cluster import KMeans
wcss = []
cl = []
for i in range (1, 11):
    Kmeans = KMeans (n_clusters = i, init = 'k-means++',
                     random_state = 42)
    Kmeans.fit (X_scaled)
    wcss.append (Kmeans.inertia_)
    cl.append (i)

sns.pointplot (x = cl, y = wcss)
plt.title ("The Elbow Method")
```

Step5: Create Clusters for selected K

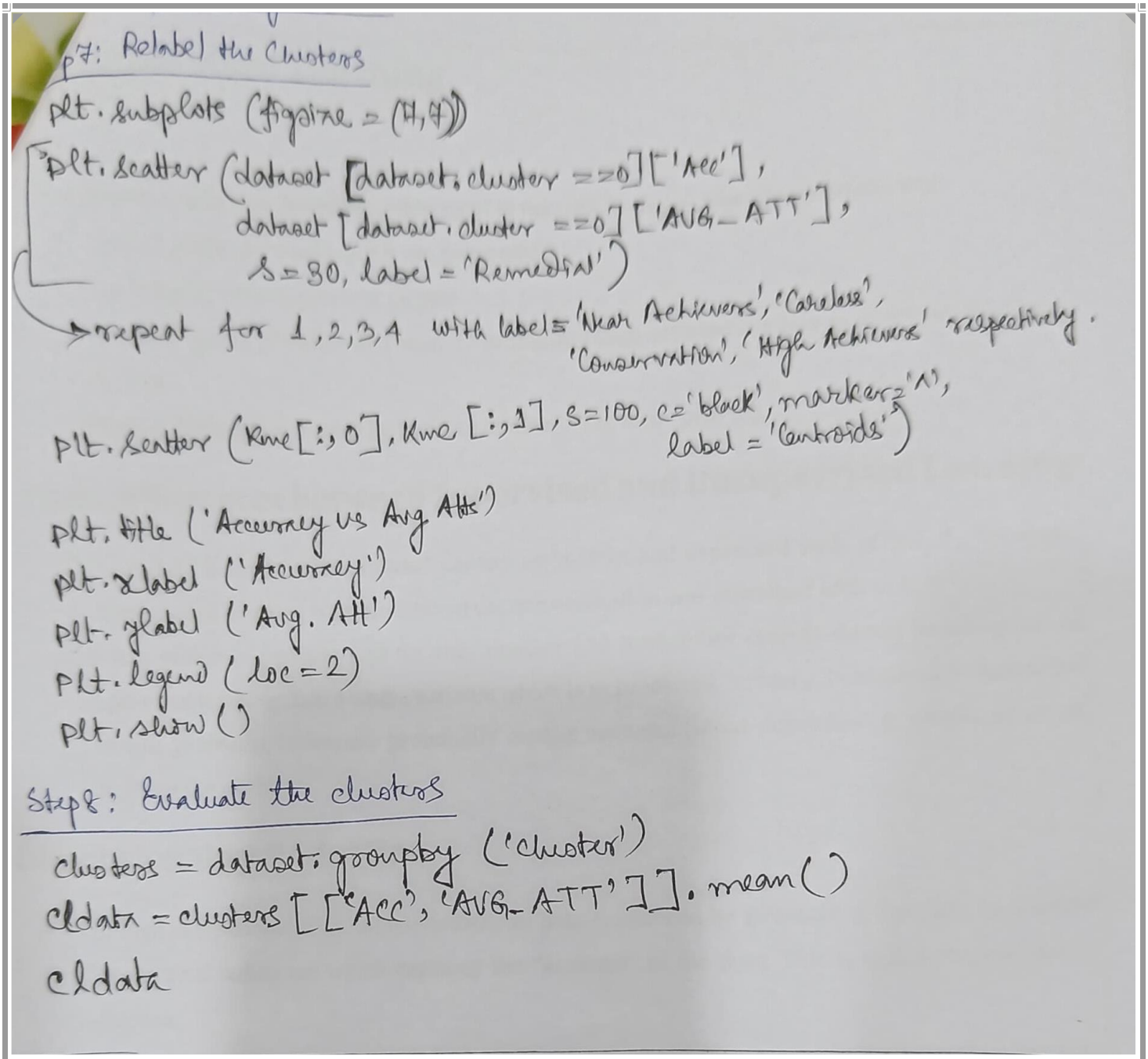
```
Kmeans = KMeans (n_clusters = 5, init = 'k-means++', random_state = 42)
y_kmeans = Kmeans.fit_predict (X_scaled)

dataset ['Cluster'] = y_kmeans          // assign cluster number to the data
dataset.head()
```

Step 6: Evaluate the clusters

```
# plot 5 clusters
plt.subplots (figsize = (7,7))
plt.scatter (dataset[dataset.cluster == 0]['Acc'],
             dataset[dataset.cluster == 0]['AVG_ATT'], s = 30, label = 'Cluster 0')
```

↳ repeat for 1, 2, 3, 4 in place of '0'.

```
# plot cluster centroids, (centroids must be unscaled)
Knc = scaler.inverse_transform (Kmeans.cluster_centers_)
plt.scatter (Knc[:, 0], Knc[:, 1], c = 'black', s = 100, marker = '^', label = 'Centroids')
plt.title ('Accuracy vs Avg Att')
plt.xlabel ('Accuracy')
plt.ylabel ('Avg Att')
plt.legend (loc = 2)
plt.show ()
```




# Un-Supervised Learning

- Goal is to discover "interesting structure" in the data (called as **Knowledge Discovery**)
- Desired output for each input is not being told.
- It is learning without labels or targets.

- It is more typical of human and animal learning and does not need any human intervention to label the data.
- **Destiny Estimation** is done by building models of the form $p(X_i \mid \theta)$.

## Basic differences between Supervised and Unsupervised Learning

(1) Supervised learning is conditional density estimation and expressed with $p(Y_i|X_i, \theta)$. Whereas, unsupervised learning is unconditional destiny estimation and expressed with $p(X_i \mid \theta)$.

(2) $X_i$ is a vector of features and for this, multivariate probability models should be designed. In supervised learning, $Y_i$ is a single variable which is to predicted. It means, in maximum supervised learning problems, univariate probability models are used (input-dependent parameters) which simplifies the problem.

## Dimensionality Reduction

- For high dimensional data, dimensionality of data is reduced by **projecting the data to a lower dimensional subspace** which captures the "**essence**" of the data. This is called dimensionality reduction.
- Low dimensional representations often result in better predictive accuracy, because they focus on "essence" of the object, filtering out **inessential** features.
- Low dimensional representations are useful for enabling **Fast Nearest Neighbour Searches**.
- 2D projections are very useful for **visualizing high dimensional data**.
- Common approach to dimensionality reduction is **Principal Component Analysis (PCA)**. PCA is nothing but an – **Unsupervised version of (multi-output) linear regression**, where high dimensional response (Y) is observed but low dimensional "**cause**" (Z) is not observed. So, the model looks like ($Z \rightarrow Y$).
- **<u>Application Area</u>**:
  - In biology, PCA is used to interpret gene microarray data, to account for the fact that each measurement is usually the result of many genes which are correlated in their behaviour by the fact that they belong to different biological pathways.
  - In NLP, a variant of PCA (**Latent Semantic Analysis**) is used for document retrieval.
  - In signal processing, a variant of PCA (ICA)is used to separate signals into their different sources.

- In computer graphics, motion capture data is projected to a low dimensional space and used to create animations.

# Examples of Unsupervised Learning

## (1) Discovering Clusters

It means clustering data into several groups. In the figure (a) typical height and weight of a group of 210 people are shown and clustering using K = 2 clusters is shown in figure (b). It seems there may be many clusters or subgroups, but that is not clear how many. Let, K denote the number of clusters. Initially, distribution over the number of clusters are estimated by $p(K|D)$, which identifies the presence of subpopulations within the data.

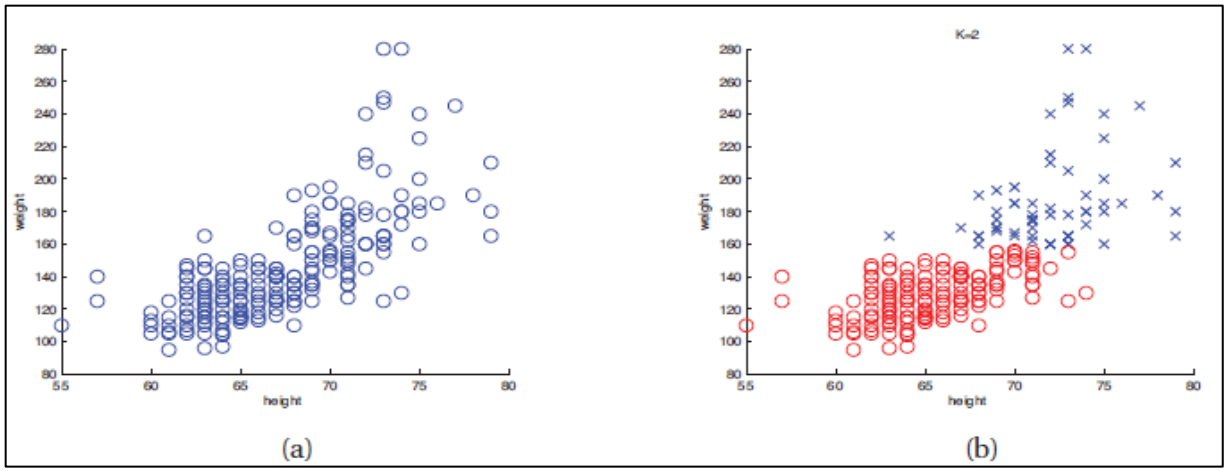
Distribution $p(K|D)$ often approximated by its mode: $K^* = argmax_K\ p(K|D)$. In supervised learning, predefined fact can be – there are two classes (male and female). But, in unsupervised learning, many or few clusters can be created which is not fixed.

Next it has to be estimated that, each point belongs to which cluster. Let $Z_i \in \{1, \dots, K\}$ represent the cluster to which data point ($i$) is assigned. [$Z_i$ is a hidden or latent variable]

Using, $Z_i^* = argmax_K\ p(Z_i = K|X_i, D)$, it can be concluded that each data point belongs to which cluster (shown in figure (b) with K = 2).

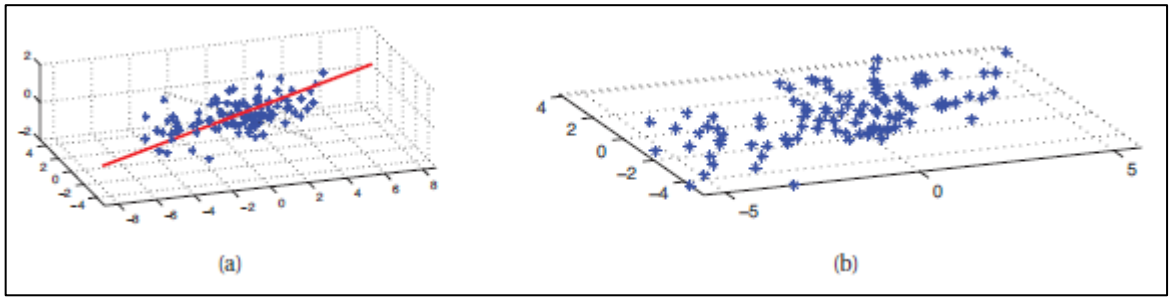
**Applications of Clustering**

- In astronomy, the **AUTOCLASS** system discovered a new type of star, based on clustering astrophysical measurements.
- In e-commerce, users are clustered into groups based on their purchasing or web-surfing behaviour and customized targeted advertisements are sent to each group.
- In biology, **flow-cytometry** data are clustered into groups to discover different sub-populations of cells.




## (2) Discovering Latent Factors

Dimensionality reduction is shown in the figure where some 3D data is projected down to a 2D plane. Solid red line is the first principal component direction and dotted black line is the second principal component direction. 2D approximation is good as most points lie close to the subspace. But, projecting points on the red line using 1D approximation cannot be a good one.

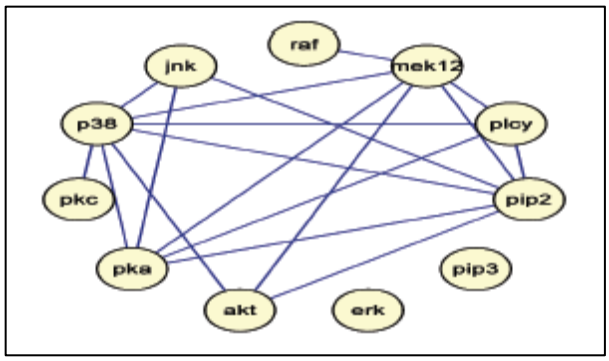



For high dimensional data, degrees of variability are small corresponding to **LATENT FACTORS.** Latent factors used to describe most of the variability for modelling the appearance of face images are lighting, pose, identity etc.

## (3) Discovering Graph Structure

When correlated variables are measured, then it is to discover that which ones are most correlated with which others, represented by a graph($G$), in which nodes represent variables and edges represent direct dependence between variables. The graph structure is learned from data through computation of $\hat{G} = argmax\ p(G|D)$.

Learning Sparse Graphs are applied to discover new knowledge and to get better **joint probability density estimators**.




**Applications**

- To measure the phosphorylation status of proteins in a cell. Upper side figure shows that.
- Recovery of neural "writing diagram" of a certain kind of bird from time-series EEG data.
- Graph structure often can be used to model correlations and to make predictions.
- Port Folio Management, where accurate models of the covariance between large numbers of different stocks is important.
- Predicting traffic flow and jams in any area using graphical model whose structure is learned from data.

## (4) Matrix Completion

When a survey is conducted, it may happen that few people might not have answered certain questions or when sensors got failed or not able to sense environment data, then comes the issue named **missing data** or **variables whose values are unknown**.

At that situation, the corresponding design matrix will have "**HOLES**" in it and the missing entries are represented by "**NaN** (Not a Number)".

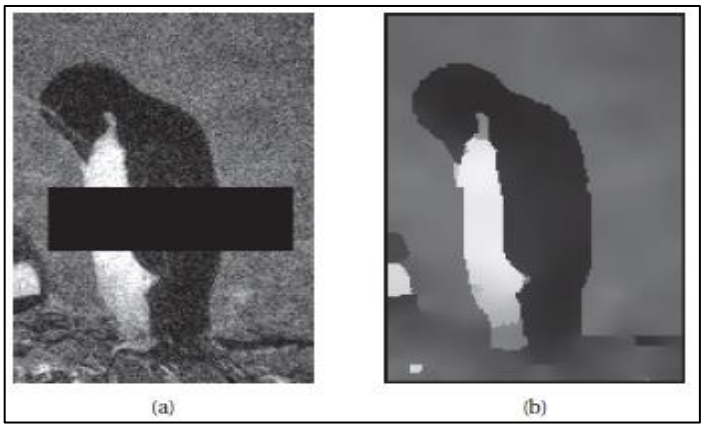
This process **imputation (assignment of a value)** provides the plausible (possible) values for the missing terms, which is called as Matrix Completion.

**Applications**

## (a) IMAGE INPAINTING

Example of imputation like task is Image Inpainting. The goal is to "**fill in**" holes (due to scratches or occlusions) in an image with realistic texture. An occluded noisy image is shown in figure (a) and estimation of pixel intensities are shown in figure (b). The image is denoised and pixels hidden behind the occlusion are imputed. This is done by JOINT PROBABILITY MODEL OF PIXELS, where unknown variables (pixels) are inferred with known variables (pixels).

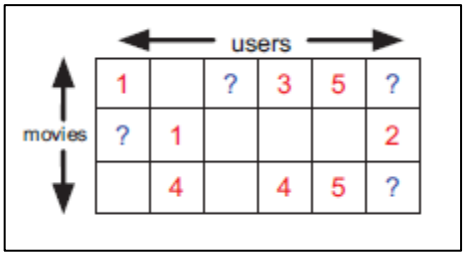
This process is with real-valued structured data and quite similar like MARKET BASKET ANALYSIS.




## (b) COLLABORATIVE FILTERING

Another example of imputation like task is **collaborative filtering**. This can be used for – predicting which movies people will want to watch based on how they and other people have rated movies which they have already seen. Here prediction is not done on the basis of features of the movies or user, but on the basis of ratings matrix.

Like, for a matrix X, where *X (m, u)* is the rating (say an integer between 1 and 5, where 1 is dislike and 5 is like) by user *u* of movie *m*. As most of the users will not have rated the movie, so most of the entries in X will be missing or unknown. For this, a tiny subset of matrix X is observed and a different subset will be predicted on the basis of that. In particular, for any user *u*, prediction will be done for those unrated movies which he / she is most likely want to watch.




## (c) MARKET BASKET ANALYSIS

In commercial data mining, market basket analysis is used. Here data consists of a matrix, where each column represents an item or product and each row represents a transaction. $X_{ij}$ = 1, if item

*j* as purchased on transaction *i*. Many items are purchased together (bread & butter), so there will be correlations among the bits.

For example, partially observed bit vector may represent a subset of items that the consumer has bought. From this, it will be predicted that which other bits will be TURNED ON, representing other items that particular consumer might be likely to buy. Here no chance of having missing data in the training data like collaborative filtering, as past shopping behaviour of a consumer is known.

Another example, dependencies between files in a complex software. Here the task is to predict, given a subset of files that have been updated or changed, which other ones need to be updated to maintain consistency.

For this type of tasks, **Frequent Itemset Mining** (which create **Association Rule**) and **Probabilistic Approach** can be used.

## Branches of Unsupervised Learning

- Distributed Representation (PCA)
- Clustering (K-Means) – The goal of clustering is to assign all data points of clusters which capture their similarity in n-dimensional space.

## Learning Without Labels: K – MEANS (Clustering Algorithm for Unsupervised Learning) or Lloyd-Forgy Algorithm

- K – Means is a clustering algorithm which produce clusters of data but data are not labelled.
- Producing clusters means assigning a cluster name to all data points so that similar data points share the same cluster name. Like, '1', '2', '3', etc.
- A cluster in K – Means is a region around a centroid separated by the HYPERPLPANE.
- Let, two features are there – means working in 2D space. No separate training and testing datasets, all datapoints fall in training dataset and clusters will be made on that.
- Inputs are UNLABELLED but contain only features.
- K – Means algorithm takes "number of CENTROIDS to be used" as input. Each centroid will define a CLUSTER. Initially, the centroids are placed in a random location in the datapoint vector space.
- Algorithm has two phases: ASSIGN and MINIMIZE (these two forms a cycle and the cycle is repeated for number of times).
    - **Assign Phase**: During this phase, each datapoint is assigned to the nearest centroid in terms of Euclidean distance.

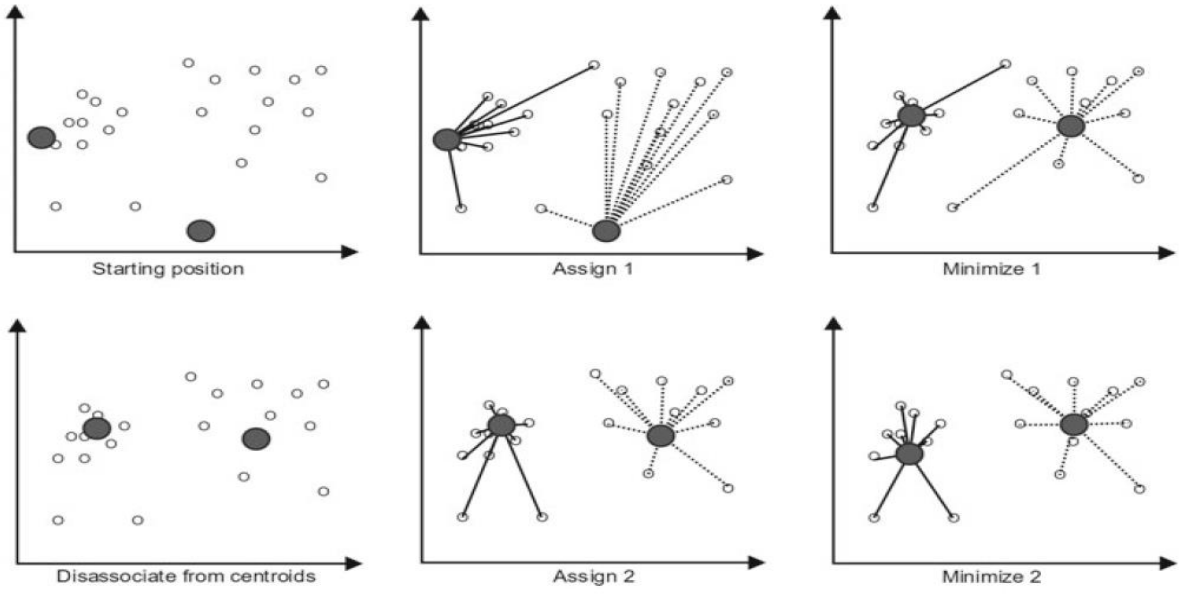
- **Minimize Phase**: In this phase, centroids are moved in a direction that minimizes the sum of the distance of all datapoints assigned to it.

- After completion of one cycle, next cycle begins by disassociating all datapoints from centroids. Centroids stay where they are, but a new assignment phase begins, which may make a different assignment than the previous one.
- After end of a cycle, HYPREPLANE is ready and when a new datapoint arrives, that will be assigned to the closest centroid i.e., that datapoint gest the name of closest centroid as LABEL. Figure shows *Two complete cycles of K-Means with two centroids*.
- As this is without labels, TP, FP, TN and FN calculations are useless.
- When results of clustering are evaluated as if they are classification results, then that is called as *EXTERNAL EVALUATION OF CLUSTERING*.
- *Class of EVALUATION METRICS* are used when data labels are unknown and work should be done without knowing labels, then that is known as *INTERNAL EVALUATION OF CLUSTERING*.
- Dunn Coefficient (an evaluation metrics) measures: "*how dense the clusters are in n-dimensional space*". For each cluster (C), Dunn Coefficient ($D_C$) is calculated by

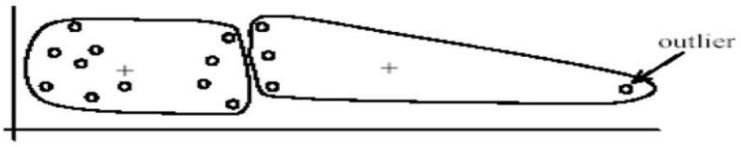
$$\boldsymbol{D_C = \frac{\min\{d(i,j)|i,j \in Centroids\}}{d^{in}(C)}}$$

Where, d(i, j) is the Euclidean Distance between centroids 'i' and 'j' and $d^{in}(C)$ is the intra-cluster distance and taken as $\boldsymbol{d^{in}(C) = max\{d(x,y)|x,y \in C\}}$, where C is the cluster for which Dunn Coefficient is calculated.

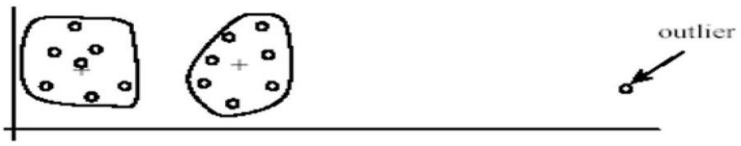
- Dunn Coefficient is calculated for each cluster and the quality of each cluster is assessed by it. Dunn Coefficient can be used to evaluate different clustering's by taking the average of the Dunn Coefficients for each cluster in both clustering's and then comparing them.



- Divide data into K-groups such that each data item is closer to the center of its cluster than to the centre of any other cluster.
- **Explanation:** When a given set of data is to be classified into K number of clusters, K centres are to be defined for each cluster. So, algorithm is to be used to find the middle points for each cluster. Localization of centre points depend on following parameters:
    - **Distance Measure**: It measures distance between a set of points. (different distance measurement techniques can be used for this)
    - **Mean Average**: After calculation of distance measure, central point of the set of data points (**MEAN AVERAGE**).
- **Advantages**
    - It is fast, robust and simple algorithm
    - It is an efficient algorithm.
        - Time Complexity: O(tkn)
            - n = number of data points
            - k = number of clusters
            - t = number of irerations
            - as, both k and t are small, k-means is considered a linear algorithm
    - It gives best results when dataset is well separated from each other.
    - It terminates at a **local optima**.
- **Disadvantages**
    - It requires pre-specification of number of clusters
    - It cannot handle noisy data and outliers
    - It cannot classify highly overlapping data
    - With different data representations, different results can be achieved
    - Random choosing of cluster center does not give good results
    - It is sensitive to **OUTLIERS**.
        - Outliers are data points that are very far away from other data points.
        - Outliers can be errors in the data recording or some special data points with very different values.
    - **To deal with OUTLIERS**
        - Remove some data points in the clustering process that are much further away from the centroids than other data points.
        - Perform random sampling. As in Sampling, small subset of data points is chosen, so the chance of selecting an **OUTLIER** is very small.




(A): Undesirable clusters




(B): Ideal clusters

- **Algorithm**

  Let $X = \{x_1, x_2, \dots\dots, x_n\}$ be the set of data points and $V = \{v_1, v_2, \dots\dots, v_n\}$ be the set of centres, then K-Means Clustering involves following steps:

  Step 1. **Randomly** select "c" as cluster centres.

  Step 2. Compute the **distance** between each data point and cluster centres.

  Step 3. Assign the data point to the cluster whose distance from the cluster centre is **minimum** in the value of cluster centres.

  Step 4. Calculate **the new cluster centre** using the formula:

$$\boldsymbol{v = \frac{1}{c_i} \sum_{j=1}^{c_i} x_i}$$

  where $c_i$ represents the number of data points in the $i^{th}$ cluster.

  Step 5. New **cluster centres** are computed.

  Step 6. If no data point is reassigned, then stop else repeat from Step 3.

**EXAMPLE**

**Initial Cluster Centres: $C_1(2), C_2(16), C_3(38)$.**
**Dataset = {2, 4, 6, 3, 31, 12, 15, 16, 38, 35, 14, 21, 23, 25, 30}**

**Solution**

Initial Cluster Centres: $C_1(2), C_2(16), C_3(38)$.
Dataset = {2, 4, 6, 3, 31, 12, 15, 16, 38, 35, 14, 21, 23, 25, 30}
To compute: Distance between each data point and cluster centres

| Data points | Distance from $C_1(2)$ | Distance from $C_2(16)$ | Distance from $C_3(38)$ |
|---|---|---|---|
| 2 | $(2-2)^2 = (0)^2 = 0$ | $(2-16)^2 = (-14)^2 = 196$ | $(2-38)^2 = (-36)^2 = 1296$ |
| 4 | $(4-2)^2 = (2)^2 = 4$ | $(4-16)^2 = (-12)^2 = 144$ | $(4-38)^2 = (-34)^2 = 1156$ |
| 6 | $(6-2)^2 = (4)^2 = 16$ | $(6-16)^2 = (-10)^2 = 100$ | $(6-38)^2 = (-32)^2 = 1024$ |
| 3 | $(3-2)^2 = (1)^2 = 1$ | $(3-16)^2 = (-13)^2 = 169$ | $(3-38)^2 = (-35)^2 = 1225$ |

| | | | |
|---|---|---|---|
| 31 | $(31-2)^2 = (29)^2 = 841$ | $(31-16)^2 = (15)^2 = 225$ | $(31-38)^2 = (-7)^2 = 49$ |
| 12 | $(12-2)^2 = (10)^2 = 100$ | $(12-16)^2 = (-4)^2 = 16$ | $(12-38)^2 = (-26)^2 = 676$ |
| 15 | $(15-2)^2 = (13)^2 = 169$ | $(15-16)^2 = (-1)^2 = 1$ | $(15-38)^2 = (-23)^2 = 529$ |
| 16 | $(16-2)^2 = (14)^2 = 196$ | $(16-16)^2 = (0)^2 = 0$ | $(16-38)^2 = (-22)^2 = 484$ |
| 38 | $(38-2)^2 = (36)^2 = 1296$ | $(38-16)^2 = (22)^2 = 484$ | $(38-38)^2 = (0)^2 = 0$ |
| 35 | $(35-2)^2 = (33)^2 = 1089$ | $(35-16)^2 = (19)^2 = 361$ | $(35-38)^2 = (-3)^2 = 9$ |
| 14 | $(14-2)^2 = (12)^2 = 144$ | $(14-16)^2 = (-2)^2 = 4$ | $(14-38)^2 = (-24)^2 = 576$ |
| 21 | $(21-2)^2 = (19)^2 = 361$ | $(21-16)^2 = (5)^2 = 25$ | $(21-38)^2 = (-17)^2 = 289$ |
| 23 | $(23-2)^2 = (21)^2 = 441$ | $(23-16)^2 = (7)^2 = 49$ | $(23-38)^2 = (-15)^2 = 225$ |
| 25 | $(25-2)^2 = (23)^2 = 529$ | $(25-16)^2 = (9)^2 = 81$ | $(25-38)^2 = (-13)^2 = 169$ |
| 30 | $(30-2)^2 = (28)^2 = 784$ | $(30-16)^2 = (14)^2 = 196$ | $(30-38)^2 = (-8)^2 = 64$ |

Now, data points are assigned to the cluster-centre whose distance from it is minimum compare to all the cluster centres.

| $C_1(2)$ | $C_2(16)$ | $C_3(38)$ |
|---|---|---|
| $m_1 = 2$ | $m_2 = 16$ | $m_3 = 38$ |
| [2, 3, 4, 6] | [12, 14, 15, 16, 21, 23, 25] | [30, 31, 35, 38] |

**New Cluster-Centres are:** $m_1 = 3.75$, $m_2 = 18$ and $m_3 = 33.5$

Similarly, using the **new cluster centres,** distances can be computed from it and clusters can be allocated based on it. And if, no changes in the clusters found, then process can be stopped otherwise move forward following same process.

| $C_1(3.75)$ | $C_2(18)$ | $C_3(33.5)$ |
|---|---|---|
| $m_1 = 3.75$ | $m_2 = 18$ | $m_3 = 33.5$ |
| [2, 3, 4, 6] | [12, 14, 15, 16, 21, 23, 25] | [30, 31, 35, 38] |

# Matrix Factorization / Matrix Decomposition

Matrix decomposition is a way of reducing a matrix into its constituent parts that make it easier to calculate more complex matrix operations.

Matrix decomposition methods are a foundation of linear algebra in computers, even for basic operations such as solving systems of linear equations, calculating the inverse, and calculating the determinant of a matrix.

It is an approach that can simplify more complex matrix operations that can be performed on the decomposed matrix rather than on the original matrix itself.

A common analogy for matrix decomposition is the factoring of numbers, such as the factoring of 10 into 2 x 5. For this reason, matrix decomposition is also called matrix factorization. Like factoring real values, there are many ways to decompose a matrix, hence there are a range of different matrix decomposition techniques.

Two simple and widely used matrix decomposition methods are the LU matrix decomposition and the QR matrix decomposition.

## LU Matrix Decomposition

The LU decomposition is for square matrices and decomposes a matrix into L and U components.

A = L . U

Where A is the square matrix that we wish to decompose, L is the lower triangle matrix and U is the upper triangle matrix.

The factors L and U are triangular matrices. The factorization that comes from elimination is A = LU.

The LU decomposition is found using an iterative numerical process and can fail for those matrices that cannot be decomposed or decomposed easily.

A variation of this decomposition that is numerically more stable to solve in practice is called the LUP decomposition, or the LU decomposition with partial pivoting.

A = P . L . U

The rows of the parent matrix are re-ordered to simplify the decomposition process and the additional P matrix specifies a way to permute the result or return the result to the original order. There are also other variations of the LU.

The LU decomposition is often used to simplify the solving of systems of linear equations, such as finding the coefficients in a linear regression, as well as in calculating the determinant and inverse of a matrix.

The example below first defines a 3×3 square matrix. The LU decomposition is calculated, then the original matrix is reconstructed from the components.

```
# LU decomposition
from numpy import array
from scipy.linalg import lu
# define a square matrix
```

```
A = array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(A)
# LU decomposition
P, L, U = lu(A)
print(P)
print(L)
print(U)
# reconstruct
B = P.dot(L).dot(U)
print(B)
```

Running the example first prints the defined 3×3 matrix, then the P, L, and U components of the decomposition, then finally the original matrix is reconstructed.

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]

[[ 0.  1.  0.]
[ 0.  0.  1.]
[ 1.  0.  0.]]

[[ 1.          0.        0.      ]
[ 0.14285714  1.        0.      ]
[ 0.57142857  0.5       1.      ]]

[[ 7.00000000e+00  8.00000000e+00  9.00000000e+00]
[ 0.00000000e+00  8.57142857e-01  1.71428571e+00]
[ 0.00000000e+00  0.00000000e+00 -1.58603289e-16]]

[[ 1.  2.  3.]
[ 4.  5.  6.]
[ 7.  8.  9.]]
```

## QR Matrix Decomposition

The QR decomposition is for m x n matrices (not limited to square matrices) and decomposes a matrix into Q and R components.

A = Q . R

Where A is the matrix that we wish to decompose, Q a matrix with the size m x m, and R is an upper triangle matrix with the size m x n.

The QR decomposition is found using an iterative numerical method that can fail for those matrices that cannot be decomposed, or decomposed easily.

Like the LU decomposition, the QR decomposition is often used to solve systems of linear equations, although is not limited to square matrices.

The QR decomposition can be implemented in NumPy using the qr() function. By default, the function returns the Q and R matrices with smaller or 'reduced' dimensions that is more economical. We can change this to return the expected sizes of m x m for Q and m x n for R by specifying the mode argument as 'complete', although this is not required for most applications.

The example below defines a 3×2 matrix, calculates the QR decomposition, then reconstructs the original matrix from the decomposed elements.

```
# QR decomposition

from numpy import array
from numpy.linalg import qr
# define a 3x2 matrix
A = array([[1, 2], [3, 4], [5, 6]])
print(A)
# QR decomposition
Q, R = qr(A, 'complete')
print(Q)
print(R)
# reconstruct
B = Q.dot(R)
print(B)
```

Running the example first prints the defined 3×2 matrix, then the Q and R elements, then finally the reconstructed matrix that matches what we started with.

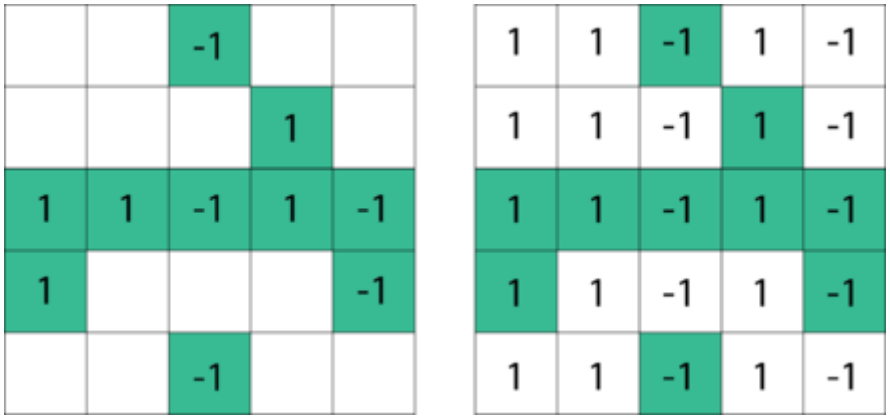
```
1   [[1 2]
```

```
2  [3 4]
3  [5 6]]
4
5  [[-0.16903085  0.89708523  0.40824829]
6  [-0.50709255  0.27602622 -0.81649658]
7  [-0.84515425 -0.34503278  0.40824829]]
8
9  [[-5.91607978 -7.43735744]
10 [ 0.         0.82807867]
11 [ 0.         0.        ]]
12
13 [[ 1.  2.]
14 [ 3.  4.]
15 [ 5.  6.]]
```

## MATRIX COMPLETION

Matrix Completion is a method for recovering lost information. It originates from machine learning and usually deals with highly sparse matrices. Missing or unknown data is estimated using the low-rank matrix of the known data.

Matrix completion is the task of filling in the missing entries of a partially observed matrix. A wide range of datasets are naturally organized in matrix form. One example is the movie-ratings matrix, as appears in the Netflix problem: Given a ratings matrix in which each entry ( i , j ) {\displaystyle (i,j)} (i,j) represents the rating of movie j {\displaystyle j} j by customer i {\displaystyle i} i, if customer i {\displaystyle i} i has watched movie j {\displaystyle j} j and is otherwise missing, we would like to predict the remaining entries in order to make good recommendations to customers on what to watch next.




## LATENT VARIABLE MODEL / LATENT FACTOR MODEL

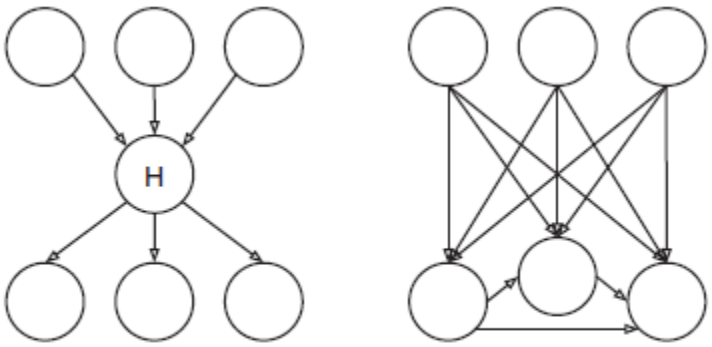
Model with hidden variables are also known as latent variable models or LVMs. Such models are harder to fit than models with no latent variables.

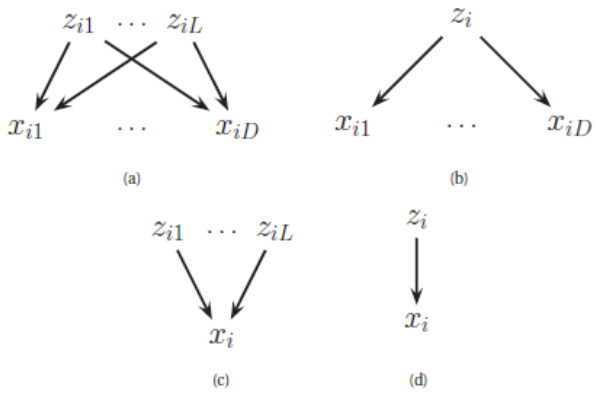
However, they can have significant advantages, for two main reasons.

[1] LVMs often have fewer parameters than models that directly represent correlation in the visible space.

[2] The hidden variables in an LVM can serve as a bottleneck, which computes a compressed representation of the data. This forms the basis of unsupervised learning.

In general there are $L$ latent variables, $z_{i1}, \ldots, z_{iL}$, and $D$ visible variables, $x_{i1}, \ldots, x_{iD}$, where usually $D >> L$. If we have $L > 1$, there are many latent factors contributing to each observation, so there are many-to-many mapping. If $L = 1$, then only a single latent variable is there; in this case, $zi$ is usually discrete, and we have a one-to-many mapping. In many-to-one mapping, representing different competing factors or causes for each observed variable; such models form the basis of probabilistic matrix factorization.






A latent variable model represented as a DGM. (a) Many-to-many. (b) One-to-many. (c) Many-to-one. (d) One-to-one.

# MIXTURE MODELS

The simplest form of LVM is when $zi \in \{1, \ldots, K\}$, representing a discrete latent state. A discrete prior can be used for this, $p(zi) = \text{Cat}(\boldsymbol{\pi})$. For the likelihood, $p(\mathbf{x}i|zi = k) = pk(\mathbf{x}i)$, where $pk$ is the $k$'th **base distribution**. The overall model is known as a **mixture model**, since the $K$ base distributions are mixed together as follows:

$$p(x_i|\theta) = \sum_{k=1}^{K} \pi_k p_k(x_i|\theta)$$

This is a convex combination of the $p_k$'s. Here the weighted sum is calculated.

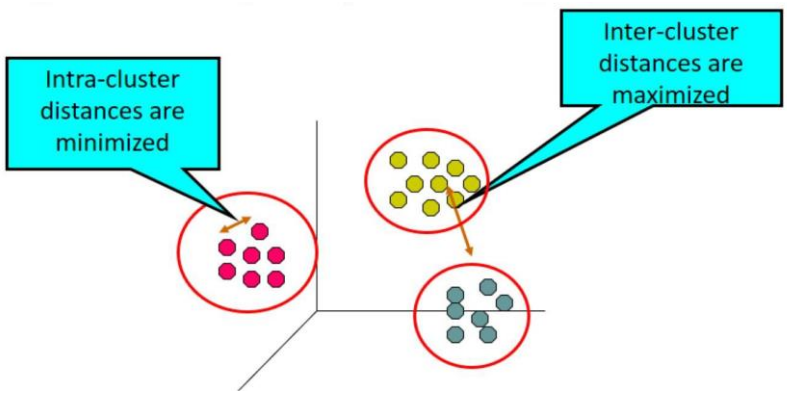
# ASSOCIATION

In this approach a set of rules are discovered which describes large partitions of data. Like, a doctor observes same set of symptoms in several patients suffering from same disease. Then based on his / her experience the doctor can conclude that the patient is suffering from the same disease.

# CLUSTERING

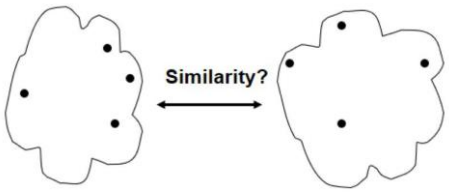
- Clustering is the organization process of grouping of unlabelled data into similar groups.
- Observation in the same cluster is similar.
- Aim is to find the clusters of similar inputs in the data without having prior knowledge that these data points belong to same or different class.
- Example, a collection of 30,000 essays can be automatically grouped into small number on the basis of word frequency, page count, word limit etc.
- **Inputs** used for clustering
    - **Similarity-based clustering** – here input is $N \times N$ **dissimilarity matrix** or **distance matrix (D)**

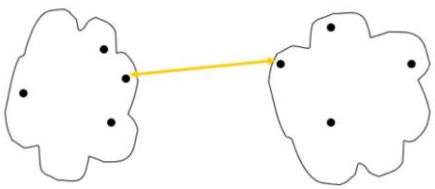
        - Allows inclusion of domain-specific similarity.
    - **Feature-based clustering** – here input is $N \times D$ **feature matrix** or **design matrix (X)**
        - Is applicable to "raw, potentially noisy data".
- Possible types of output
    - **Flat clustering** or **Partitional clustering** – here objects are portioned into disjoint sets.
        - Faster to create $[O(N\ D)]$.
        - Sensitive to initial conditions and require model selection for K.
    - **Hierarchical clustering** – here nested tree of partitions is created.
        - less faster than flat clustering $[O(N^2 \log N)]$.
        - It is deterministic and do not require specification of K (number of clusters)
- Clustering Quality
    - Inter-Clusters distance → Maximized
    - Intra-Clusters distance → Minimized
- The quality of a clustering result depends on the algorithm, the distance function and the application.
- The goal of clustering is to assign points that are similar to the same cluster and to ensure that points that are dissimilar are in different clusters.

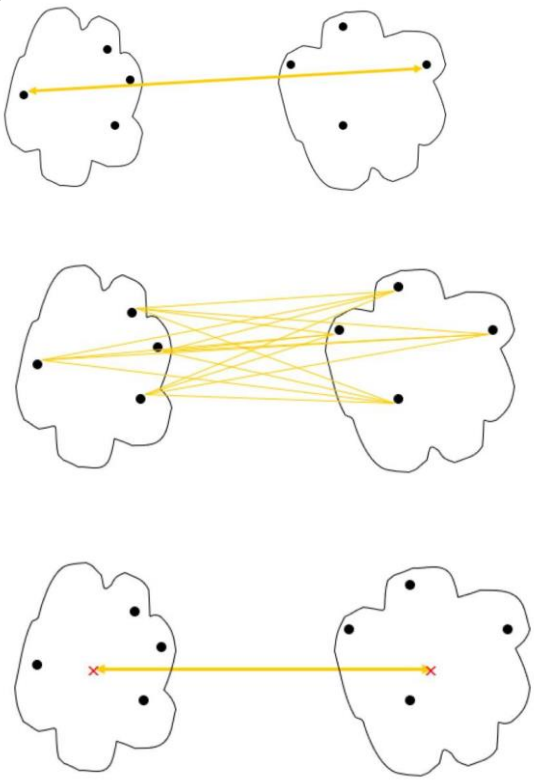



## Inter-Cluster Similarity

- MIN
- MAX
- Group Average
- Distance Between Centroids



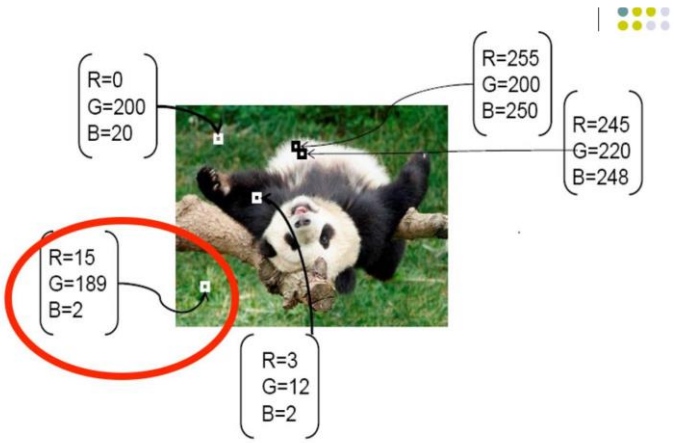





## Feature Space

Every token is identified by a set of salient visual characteristics. Like,

- Position
- Color
- Texture
- Motion Vector
- Size, orientation (if token is larger than a pixel)



# Similarity Measurement for Cluster Identification

Let $DM_{D,q_i}$ represents distance function for database D and query image $q_i$.

"n" represents length of the feature vector.

$F_d^i(j)$ and $F_{q_i}(j)$ are the feature vectors of "i[th]" database image and query image respectively

$$Euclidean\ Distance\ \left(DM_{D,q_i}\right) = \left(\sum_{j=1}^{n} |(F_d^i(j) - F_{q_i}(j))^2|\right)^{1/2}$$

$$Manhattan\ Distance\ (DM_{D,q_i}) = \sum_{j=1}^{n} |F_d^i(j) - F_{q_i}(j)|$$

$$Canberra\ Distance\ (DM_{D,q_i}) = \sum_{j=1}^{n} \left|\frac{F_d^i(j) - F_{q_i}(j)}{F_d^i(j) + F_{q_i}(j)}\right|$$

$$Chi\ Square\ Distance\ (DM_{D,q_i}) = \frac{1}{2}\sum_{j=1}^{n} \frac{(F_d^i(j) - F_{q_i}(j))^2}{F_d^i(j) + F_{q_i}(j)}$$

For ordinal variables, such as {low, medium, high}, it is standard to encode the values as real-valued numbers, say $1/3, 2/3, 3/3$, if there are 3 values. After that squared distance can be applied.

For categorical variables, such as {red, green, blue}, then a distance of "1" is assigned if the features are different and a distance of "0" otherwise. Summing up over all the categorical features gives

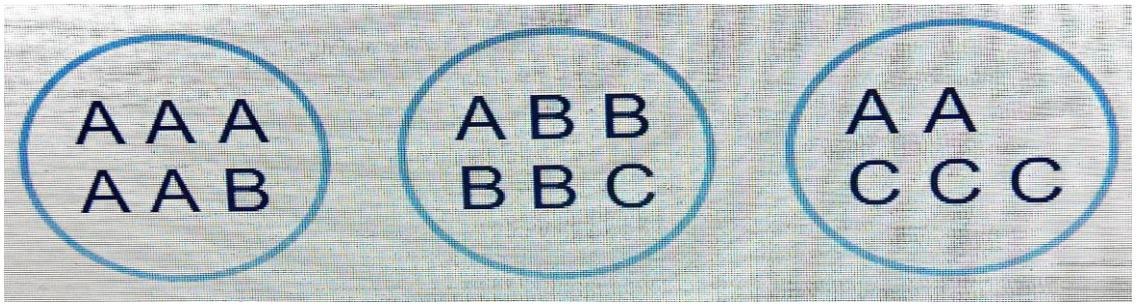
$\Delta(\boldsymbol{x_i}, \boldsymbol{x_i}) = \sum_{j=1}^{p} \mathbb{I}(\boldsymbol{x_{ij}} \neq \boldsymbol{x_{\grave{i}j}})$. ---------- **This is known as Hamming Distance.**

# Cluster Comparison

If objects are labelled properly, then clusters are compared with the labels using various metrics.

## Purity

Let $N_{ij}$ be the number of objects in cluster "i" that belong to class "j", and let $N_i = \sum_{j=1}^{C} N_{ij}$ be the total number of objects in cluster "i".



$\boldsymbol{p_{ij}} = {}^{\boldsymbol{N_{ij}}}\!/_{\boldsymbol{N_i}}$ is the Empirical Distribution over class labels for cluster "i".

Purity of a cluster is defined as $\boldsymbol{p_i} \triangleq \boldsymbol{max_j p_{ij}}$ and

the overall purity of a clustering is defined as $\boldsymbol{purity} \triangleq \sum_i \frac{N_i}{N} \boldsymbol{p_i}$.

For this example,

$$\boldsymbol{purity} = \frac{6}{17}\frac{5}{6} + \frac{6}{17}\frac{4}{6} + \frac{5}{17}\frac{3}{5} = \frac{5+4+3}{17} = \mathbf{0.71}$$

The purity basically changes between 0 (bad) and 1 (good).

Also, trivially purity of 1 can be achieved by putting each object into its own cluster.

## Rand Index

Let U is the estimated clustering and V is reference clustering derived from the class labels. Define a $2 \times 2$ contingency table, containing following values:

TP is the number of pairs that are in the same cluster in both U and V (True Positives)

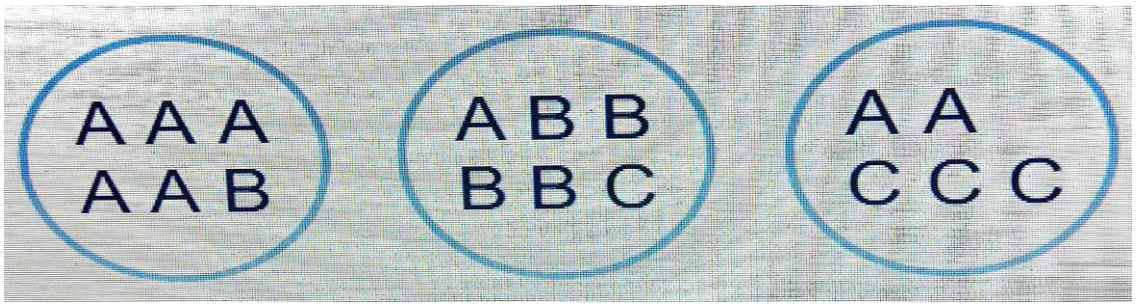
TN is the number of pairs that are in the different clusters in both U and V (True Negatives)

FN is the number of pairs that are in the different clusters in U but the same cluster in V (False Negatives)

FP is the number of pairs that are in the same cluster in U but different clusters in V (False Positives)

$$\boldsymbol{Rand\ Index\ (R)} \triangleq \frac{\boldsymbol{TP + TN}}{\boldsymbol{TP + FP + FN + TN}}$$

This can be interpreted as the fraction of clustering decisions that are correct. Clearly $0 \leq R \leq 1$.




These three clusters contain 6, 6 and 5 points, so the number of "POSITIVES" (i.e. pairs of objects put in the same cluster, regardless of the label) is

$TP = 5 + 4 + 3 = 12.$

$FP = 3 + 1 + 1 = 5.$

$TN = 0.$

$FN = 1 + 2 + 2 = 5$

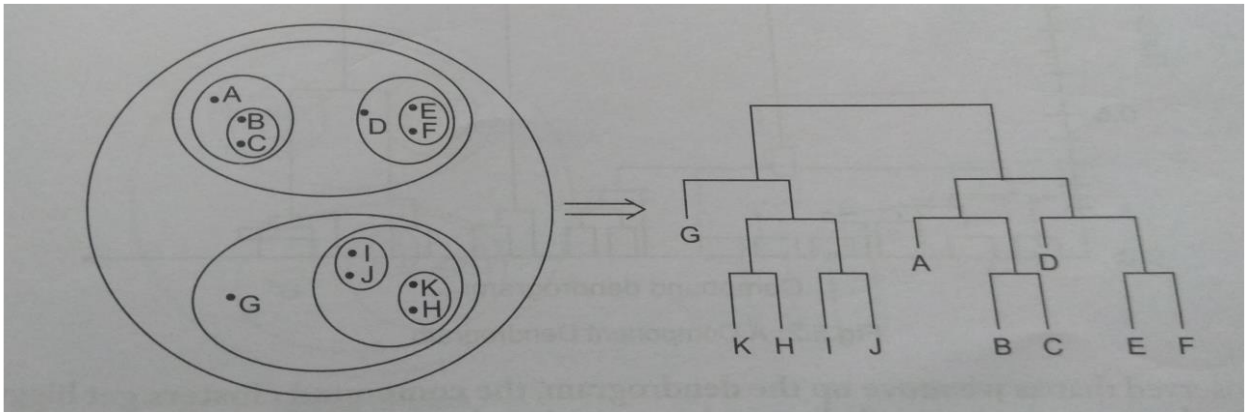
So,

$$\textbf{\textit{Rand Index}}\ (\boldsymbol{R}) = \frac{\mathbf{12+0}}{\mathbf{12+5+0+5}} = \mathbf{0.56}$$

Rand Index only achieves its LOWER BOUND of "0" if TP = TN = 0, which is rare event.

## DENDROGRAM

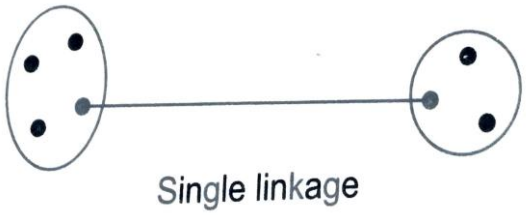
1. The visual representation of the compound correlation data is known as a Dendrogram.
2. It is a commonly used tree structure of hierarchical clustering. It shows how the clusters are **merged or split** iteratively to arrive at an **optimal clustering solution**.
3. The individual compounds are arranged along the bottom of the Dendrogram. These are called as **leaf nodes**.
4. In the picture, Dendrogram (on right) shows NESTED CLUSTERS (on left).




5. Compound Clusters are formed by joining individual compounds or existing compound clusters with the joint point named as a **node**.
6. **Vertical axis** is labelled distance and it refers to a **distance measure** between compounds or compound clusters.
7. **Horizontal axis** on the Dendrogram represents the distance or dissimilarity between clusters.
8. The **height** of the node can be thought of as the distance value between the right and left sub-branch clusters.
9. The **distance measure** between two clusters is calculated by D = 1 – C, where D = Distance and C = correlation between compound clusters.
10. If compounds are correlated, then they will have a **correlation value close to 1**. So, D = 1 – C will have a value close to **ZERO**.
11. If compounds are not correlated, then they will have a **correlation value close to zero**. So, D = 1 – C will have a value close to **1**.
12. Highly correlated clusters **are nearer** to the bottom of the Dendrogram.
13. Compounds, that **are negatively correlated**, will have a correlation value of (-1) and D = 1 – (-1) = 1 + 1 = 2.
14. It is observed that, when we move up the Dendrogram, **the compound clusters get bigger and the distance between compound clusters increases in value**.
15. **A Dendrogram shows how the clusters are "merged" iteratively (AGGLOMERATIVE CLUSTERING) or "split" iteratively (DIVISIVE CLUSTERING) to the OPTIMAL CLUSTERING** solution.

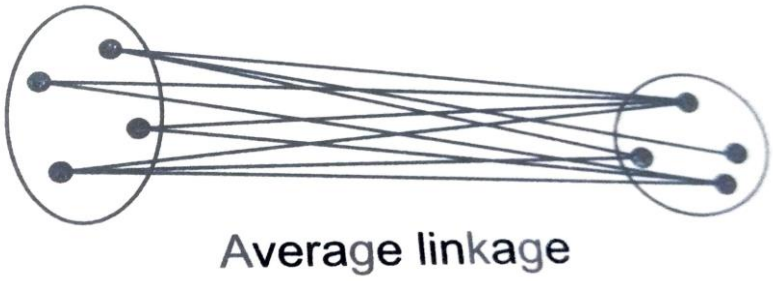
# HIERARCHICAL CLUSTERING

- It makes hierarchy of clusters
- It cannot handle huge amount of data
- Different types of **distance metrics** are used to compute distance between points in hierarchical clustering, like Euclidean Distance, Manhattan Distance, Maximum Distance etc.
- Using **linkage between two clusters**, cluster assignment can be done:
    - **Single Linkage**: It is defined as the distance between two clusters as the **minimum distance** between any points in both the clusters. One point belongs to one cluster and second point belongs to another cluster.

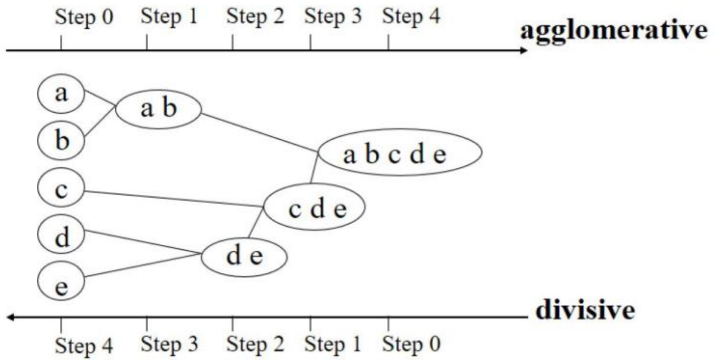



    - **Complete Linkage**: It is defined as the distance between two clusters as the **maximum distance** between any points in both the clusters. One point belongs to one cluster and second point belongs to another cluster.

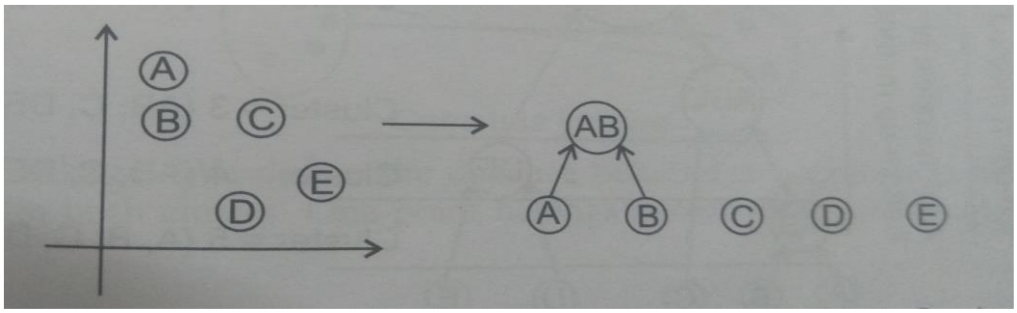

    - **Average Linkage**: It is defined as the distance between two clusters as the **mean distance** between all points in both the cluster. One point belongs to one cluster and second point belongs to another cluster.

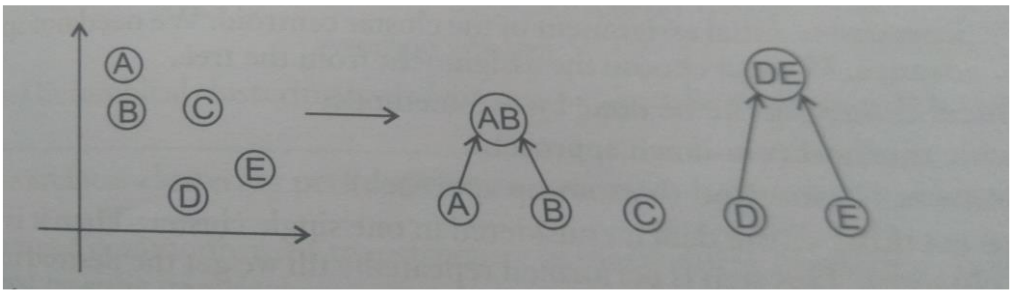


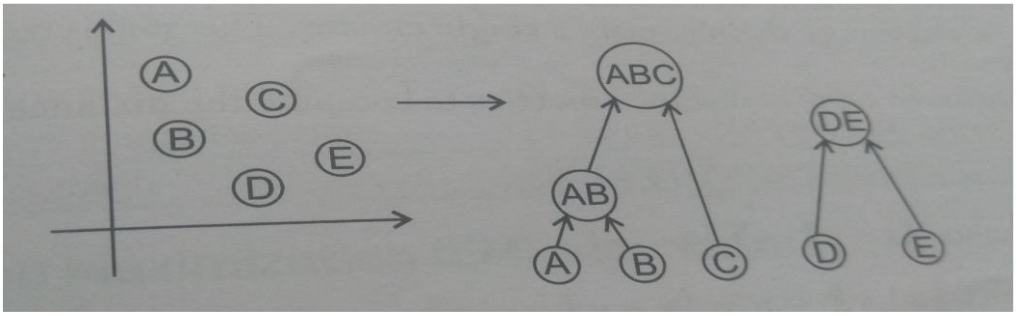

- It can be done in two methods:
    - **Divisive Method** (Top-Down approach) – entire data is considered in one single cluster. Then that is **portioned** into **two separate clusters**. This step is repeated till desired number of clusters are achieved or **one cluster for each point** is created.
    - **Agglomerative Method** (Bottom-Up approach) – each point is considered as a single cluster. In each step, **two closest point** is selected and **merged** to make a **single cluster**. This step is repeated till **one cluster left only**.

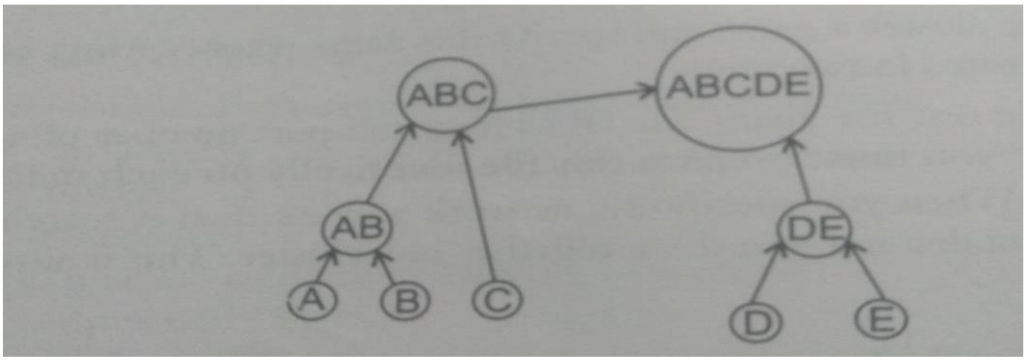


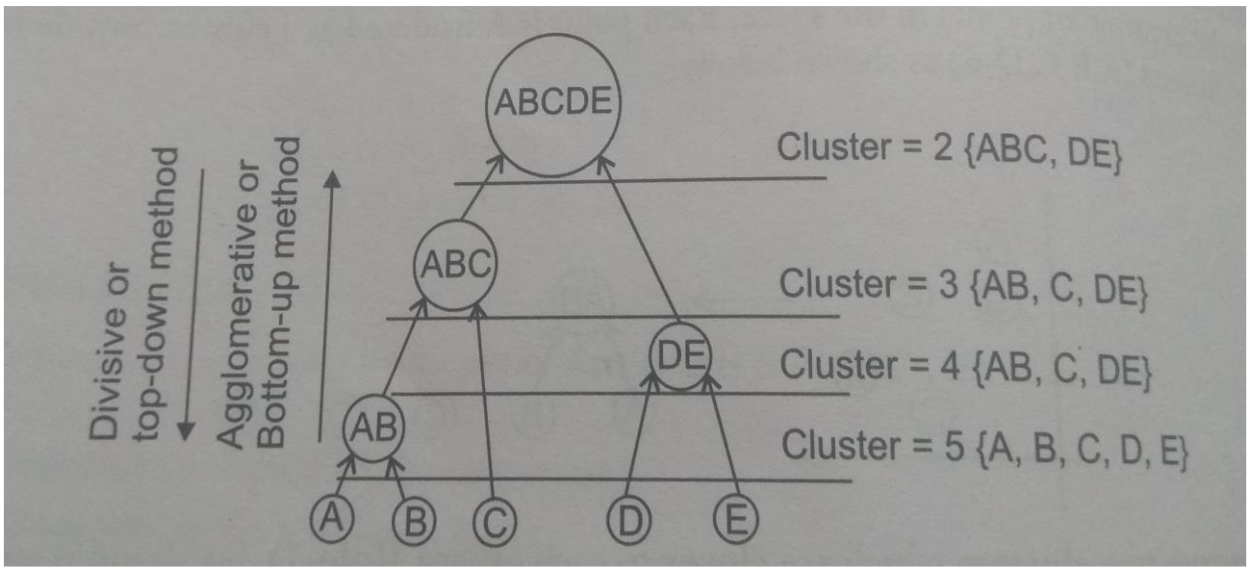

- **RULES**
  - **Rule 1**: Merge two clusters which are closer to each other.
  - **Rule 2**: Find two nearest clusters and also find the **middle** of both the clusters.
  - **Rule 3**: After building the complete hierarchy, CUT the tree at different length and get the clusters. So, number of clusters can be computed. **General rule of Thumb** is used to cut the tree at the middle of the highest branch of the tree.




ABC
ABCDE
AB
DE
A
B
C
D
E


ABCDE
Cluster = 2 {ABC, DE}
ABC
Cluster = 3 {AB, C, DE}
DE
Cluster = 4 {AB, C, DE}
AB
Cluster = 5 {A, B, C, D, E}
A
B
C
D
E
Divisive or top-down method
Agglomerative or Bottom-up method