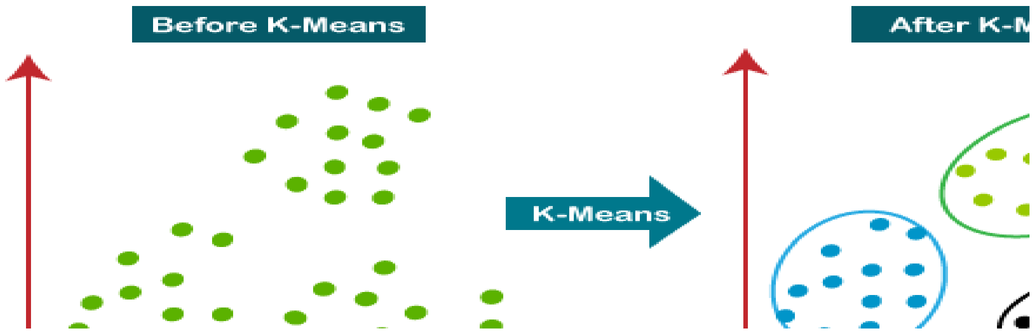
MACHINE LEARNING
21CSC305P
Unit-3
Mixture Models and EM

- **K-Means Clustering**
- **Mixtures of Gaussians**
- **An Alternative View of EM**
- **Factor Analysis**
- **PCA**
- **Choosing the Number of Latent Dimensions**
- **Clustering**
- **Measuring Dissimilarity**
- **Evaluating the Output of Clustering Methods**
- **Hierarchical Clustering**

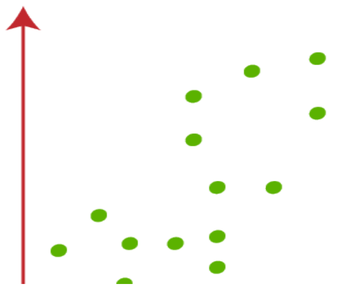
K-MEANS CLUSTERING

K-Means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems. K defines the number of pre-defined clusters that need to be created in the process, as if K=2, there will be two clusters, and for K=3, there will be three clusters, and so on.



How does the K-Means Work?

Step-1: Select the number K to decide the number of clusters.
Step-2: Select random K points or centroid. (It can be other from the input dataset).
Step-3: Assign each data point to their closest centroid, which will form the predefined K clusters.
Step-4: Calculate the variance and place a new centroid of each cluster.
Step-5: Repeat the third steps, which mean reassign each data point to the new closest centroid of each cluster.
Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.
Step-7: The model is ready.
Suppose we have two variables M1 and M2. The x-y axis scatter plot of these two variables is given below:



Let's take number k of clusters, i.e., K=2, to identify the dataset and to put them into different clusters.

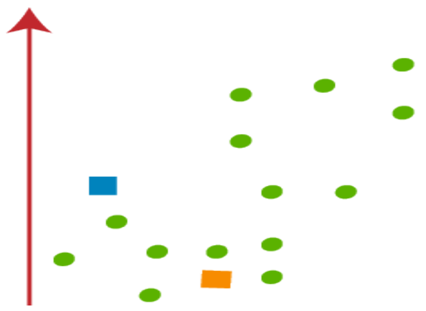


Figure: 1

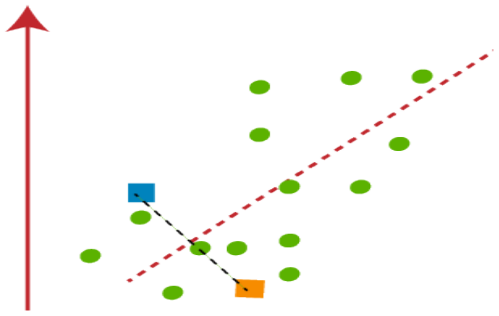


Figure: 2

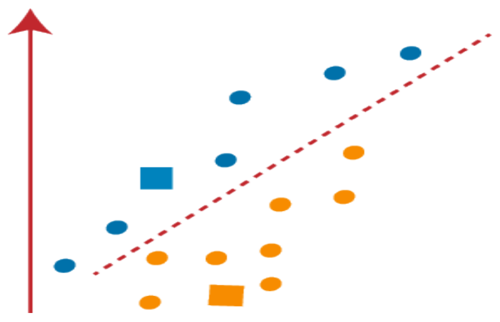


Figure: 3

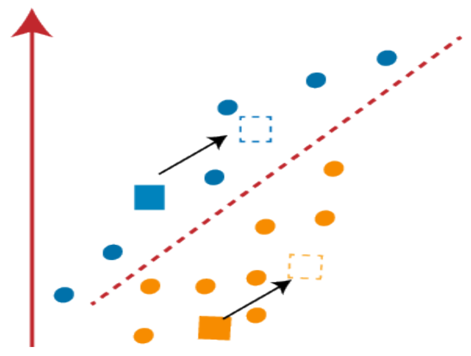


Figure: 4

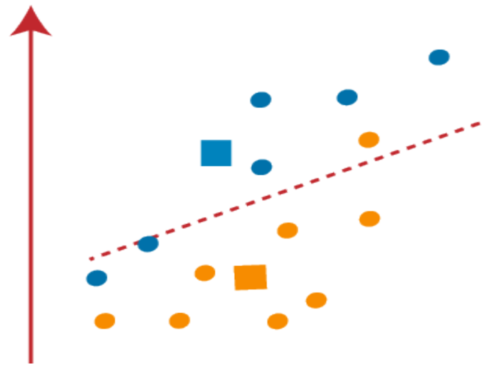


Figure: 5

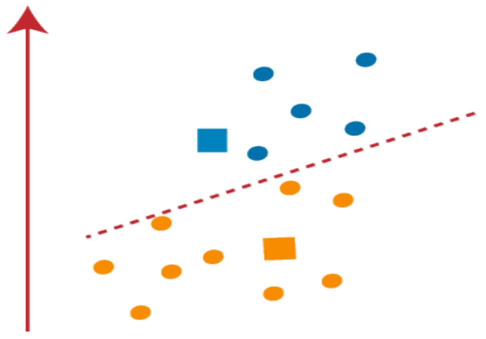


Figure: 6

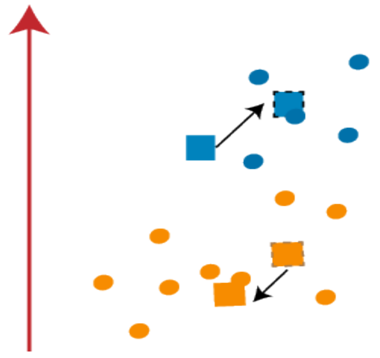


Figure: 7

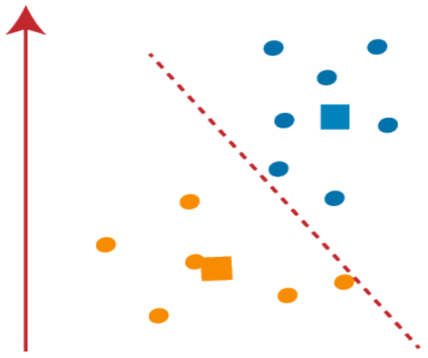


Figure: 8

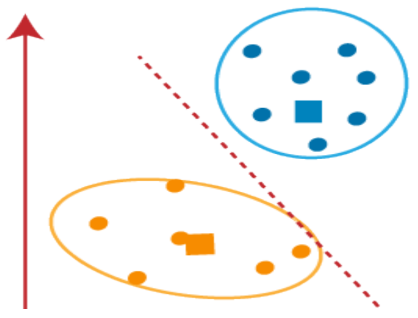


Figure: 9

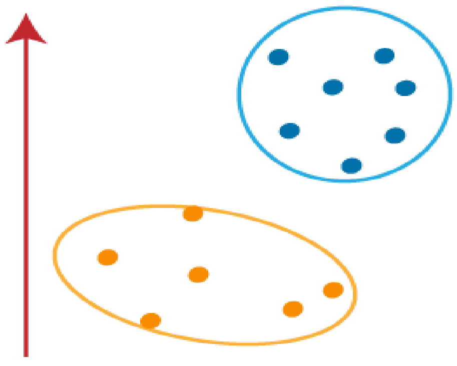


Figure: 10

How to choose the value of "K number of clusters" in K-means Clustering?

1. **Elbow Method**: It is one of the most popular ways to find the optimal number of clusters. WCSS stands for Within Cluster Sum of Squares, which defines the total variations within a cluster. The formula to calculate the value of WCSS (for 3 clusters) is given below:

$$\text{WCSS} = \sum_{P_i \text{ in Cluster1}} \text{distance}(P_i\, C_1)^2 + \sum_{P_i \text{ in Cluster2}} \text{distance}(P_i\, C_2)^2 + \sum_{P_i \text{ in CLuster3}} \text{distance}(P_i\, C_3)^2$$

2. **Euclidean distance**

$$\Delta_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2$$

3. **Manhattan distance**

$$\Delta_j(x_{ij}, x_{i'j}) = |x_{ij} - x_{i'j}|$$

Algorithm 1 k-means algorithm

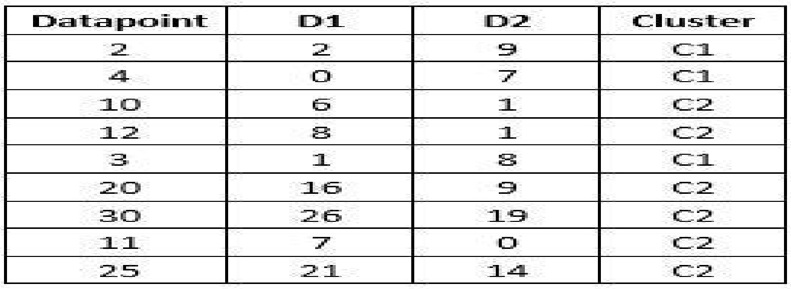
```
1: Specify the number $k$ of clusters to assign.
2: Randomly initialize $k$ centroids.
3: repeat
4:     expectation: Assign each point to its closest centroid
5:     maximization: Compute the new centroid (mean) of
```

Question: We will apply k-means on the following 1 dimensional data set for K=2. Data set {2, 4, 10, 12, 3, 20, 30, 11, 25}.

Solution:

- **Iteration 1**

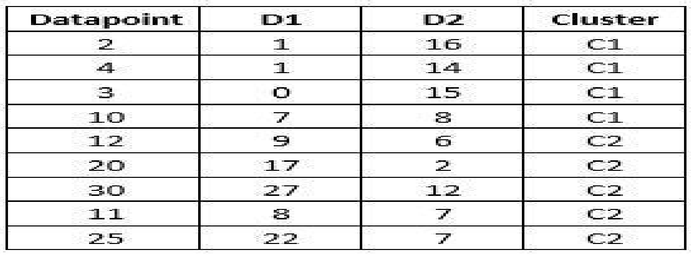
- M1, M2 are the two randomly selected centroids/means where: M1= 4, M2=11
- The initial clusters are: C1= {4}, C2= {11}
- Calculate the Euclidean distance as: $D=[x,a]=\sqrt{(x-a)^2}$
- D1 is the distance from M1
- D2 is the distance from M2



Datapoint	D1	D2	Cluster
2	2	9	C1
4	0	7	C1
10	6	1	C2
12	8	1	C2
3	1	8	C1
20	16	9	C2
30	26	19	C2
11	7	0	C2
25	21	14	C2

➢ Iteration 2

- Calculate new mean of datapoints in C1 and C2.
  M1= (2+3+4)/3= 3
  M2= (10+12+20+30+11+25)/6= 18

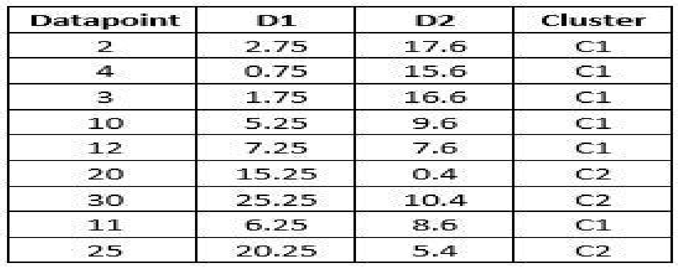


Datapoint	D1	D2	Cluster
2	1	16	C1
4	1	14	C1
3	0	15	C1
10	7	8	C1
12	9	6	C2
20	17	2	C2
30	27	12	C2
11	8	7	C2
25	22	7	C2

New Clusters
C1= {2, 3, 4, 10}
C2= {12, 20, 30, 11, 25}

➢ Iteration 3

Calculate new mean of datapoints in C1 and C2.
M1= (2+3+4+10)/4= 4.75
M2= (12+20+30+11+25)/5= 19.6

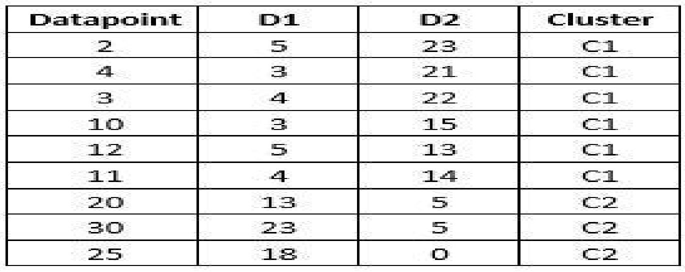


Datapoint	D1	D2	Cluster
2	2.75	17.6	C1
4	0.75	15.6	C1
3	1.75	16.6	C1
10	5.25	9.6	C1
12	7.25	7.6	C1
20	15.25	0.4	C2
30	25.25	10.4	C2
11	6.25	8.6	C1
25	20.25	5.4	C2

New Clusters
C1= {2, 3, 4, 10, 12, 11}
C2= {20, 30, 25}

➢ Iteration 4

Calculate new mean of datapoints in C1 and C2.
M1= (2+3+4+10+12+11)/6=7
M2= (20+30+25)/3= 25



Datapoint	D1	D2	Cluster
2	5	23	C1
4	3	21	C1
3	4	22	C1
10	3	15	C1
12	5	13	C1
11	4	14	C1
20	13	5	C2
30	23	5	C2
25	18	0	C2

As we can see that the data points in the cluster C1 and C2 in iteration 3 are same as the data points of the cluster C1 and C2 of iteration 2.
Answer: C1= {2, 3, 4, 10, 12, 11}
C2= {20, 30, 25}

Advantages

- It is very easy to understand and implement.
- If we have large number of variables then, K-means would be faster than Hierarchical clustering.
- On re-computation of centroids, an instance can change the cluster.
- Tighter clusters are formed with K-means as compared to Hierarchical clustering.

Disadvantages

- It is a bit difficult to predict the number of clusters i.e. the value of k.
- Output is strongly impacted by initial inputs like number of clusters (value of k).
- Order of data will have strong impact on the final output.
- It is very sensitive to rescaling. If we will rescale our data by means of normalization or standardization, then the output will completely change final output.
- It is not good in doing clustering job if the clusters have a complicated geometric shape.

Applications of K-Means Clustering Algorithm

- Market segmentation
- Document Clustering
- Image segmentation
- Image compression
- Customer segmentation
- Analyzing the trend on dynamic data

MIXTURES OF GAUSSIANS

The Gaussian Mixture Distribution can be written as a linear superposition of Gaussians in the form:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

Let k-dimensional binary random variable z having of 1-of-K representation in which a particular element $z_k = 1$ and all other elements are equal to 0.

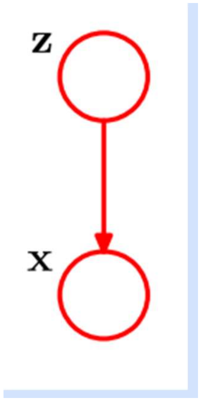


Figure: Graphical representation of a mixture model, in which the joint distribution is expressed in the form p(x, z) = p(z) p(x| z)

The marginal distribution over z in terms of mixing coefficients π_k, such that:

$$P(z_k = 1) = \pi_k$$

Where the parameters $\{\pi_k\}$ must satisfy:

$$0 <= \pi_k <= 1$$

Together with

$$\sum_{k=1}^{K} \pi_k = 1$$

in order to be valid probabilities. Because $\mathbf{z}$ uses a 1-of-K representation, we can also write this distribution in the form

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}. \tag{9.10}$$

Similarly, the conditional distribution of $\mathbf{x}$ given a particular value for $\mathbf{z}$ is a Gaussian

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

which can also be written in the form

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}. \tag{9.11}$$

The joint distribution is given by $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$, and the marginal distribution of $\mathbf{x}$ is then obtained by summing the joint distribution over all possible states of $\mathbf{z}$ to give

$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{z})p(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{9.12}$$

Another quantity that will play an important role is the conditional probability of $\mathbf{z}$ given $\mathbf{x}$. We shall use $\gamma(z_k)$ to denote $p(z_k = 1|\mathbf{x})$, whose value can be found using Bayes' theorem

$$\begin{aligned}\gamma(z_k) \equiv p(z_k = 1|\mathbf{x}) &= \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \end{aligned} \tag{9.13}$$

We shall view π_k as the prior probability of $z_k = 1$, and the quantity $\gamma(z_k)$ as the corresponding posterior probability once we have observed $\mathbf{x}$. As we shall see later, $\gamma(z_k)$ can also be viewed as the *responsibility* that component k takes for 'explaining' the observation $\mathbf{x}$.

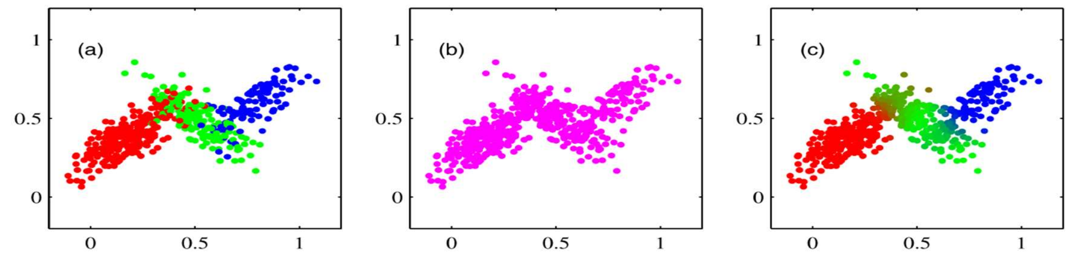


Figure 9.5 Example of 500 points drawn from the mixture of 3 Gaussians shown in Figure 2.23. (a) Samples from the joint distribution $p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$ in which the three states of $\mathbf{z}$, corresponding to the three components of the mixture, are depicted in red, green, and blue, and (b) the corresponding samples from the marginal distribution $p(\mathbf{x})$, which is obtained by simply ignoring the values of $\mathbf{z}$ and just plotting the $\mathbf{x}$ values. The data set in (a) is said to be *complete*, whereas that in (b) is *incomplete*. (c) The same samples in which the colours represent the value of the responsibilities $\gamma(z_{nk})$ associated with data point $\mathbf{x}_n$, obtained by plotting the corresponding point using proportions of red, blue, and green ink given by $\gamma(z_{nk})$ for $k = 1, 2, 3$, respectively

AN ALTERNATIVE VIEW OF EM

The goal of the EM algorithm is to find maximum likelihood solutions for models having latent variables. We denote the set of all observed data by $\mathbf{X}$, in which the n^{th} row represents $\mathbf{x}_n^{\mathrm{T}}$, and similarly we denote the set of all latent variables by $\mathbf{Z}$, with a corresponding row $\mathbf{z}_n^{\mathrm{T}}$. The set of all model parameters is denoted by $\boldsymbol{\theta}$, and so the log likelihood function is given by

$$\ln p(\mathbf{X}|\boldsymbol{\theta}) = \ln \left\{ \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}) \right\}. \tag{9.29}$$

Now suppose that, for each observation in $\mathbf{X}$, we were told the corresponding value of the latent variable $\mathbf{Z}$. We shall call $\{\mathbf{X}, \mathbf{Z}\}$ the *complete* data set, and we shall refer to the actual observed data $\mathbf{X}$ as *incomplete*, as illustrated in Figure 9.5. The likelihood function for the complete data set simply takes the form $\ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$,

The General EM Algorithm

Given a joint distribution $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ over observed variables $\mathbf{X}$ and latent variables $\mathbf{Z}$, governed by parameters $\boldsymbol{\theta}$, the goal is to maximize the likelihood function $p(\mathbf{X}|\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$.

1. Choose an initial setting for the parameters $\boldsymbol{\theta}^{\text{old}}$.

2. **E step** Evaluate $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$.

3. **M step** Evaluate $\boldsymbol{\theta}^{\text{new}}$ given by

$$\boldsymbol{\theta}^{\text{new}} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) \tag{9.32}$$

where

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \tag{9.33}$$

4. Check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then let

$$\boldsymbol{\theta}^{\text{old}} \leftarrow \boldsymbol{\theta}^{\text{new}} \tag{9.34}$$

and return to step 2.

FACTOR ANALYSIS

- Factor Analysis is an **unsupervised**, **probabilistic** machine learning algorithm used for dimensionality reduction.
- Factor analysis is a **statistical approach** that seeks to elucidate the variations present in observed data by uncovering latent variables, commonly known as **factors**.
- By revealing hidden patterns and relationships that may not be readily discernible, factor analysis offers valuable insights into the intricate structure of complex datasets.
- It is a component of the **general linear model**. It can also lead to several assumptions like **linear relationships** and no **multicollinearity**.

Objectives

- Determining the number of factors required to explain common themes within a particular variable set.
- Determining the extent to which every variable in the dataset is connected to a common factor or theme.
- Interpreting the common factors in a dataset.
- Understanding the degree to which each observed data point is representative of a theme or factor.

Terminology

1. Factor: The factor is a latent (hidden or unobserved) variable representing the correlated variables that share a common variance. The maximum number of factors is equal to the number of variables.

2. Eigenvalues (Characteristic Roots): Variance cannot be negative, so negative eigenvalues imply an incorrect model. In contrast, eigenvalues close to zero indicate multicollinearity as the first component can take up all the variance.

3. Factor Loadings: Factor loading is the correlation coefficient for the variable and factor. High factor loading score means that the variables better consider the dimensions of the factors.

4. Communalities: Communalities are the sum of the squared loadings for each variable. If the communalities for a particular variable are low, say between 0–0.5, then this suggests the variable will not load significantly on any factor.

Types of Factor Analysis

1. **Exploratory Factor Analysis**: EFA does not impose any predetermined structure or assume pre-existing relationships among variables.
2. **Confirmatory Factor Analysis**: Confirmatory Factor Analysis is a technique used to evaluate predetermined hypotheses regarding the relationships between variables and factors. Its purpose is to determine whether the observed data align with the proposed factor structure.

Pre-requisites for factor analysis

- Apply Factor Analysis on the **large datasets** to diminish the error in the final results.
- There should be **high factor loading scores (>0.80)** to use the factor analysis on a small dataset.
- The **correlation** between the factors and variables should be **at least 0.30**, as anything lower than this would imply a weak relationship between the variables.
- The **SMC (Squared Multiple Correlation)** of the dataset needs to be checked, and variables that have issues with a singularity, i.e., **SMC close to 0 and multicollinearity, i.e., SMC close to 1.0, should be removed from your dataset.**
- We should also **remove outliers** from the dataset.
- The dataset should be **standard scaled**, and we should convert the categorical features to numerical features.

Factor Analysis Steps

1. Determine the Suitability of Data for Factor Analysis

- **Bartlett's Test:** Check the significance level to determine if the correlation matrix is suitable for factor analysis.
- **Kaiser-Meyer-Olkin (KMO) Measure:** Verify the sampling adequacy. A value greater than 0.6 is generally considered acceptable.

2. Choose the Extraction Method

- **Principal Component Analysis (PCA**): Used when the main goal is data reduction.
- **Principal Axis Factoring (PAF):** Used when the main goal is to identify underlying factors.

3. Factor Extraction

- Use the chosen extraction method to identify the initial factors.
- Extract eigenvalues to determine the number of factors to retain. Factors with eigenvalues greater than 1 are typically retained in the analysis.
- Compute the initial factor loadings.

4. Determine the Number of Factors to Retain

- **Scree Plot**: Plot the eigenvalues in descending order to visualize the point where the plot levels off (the "elbow") to determine the number of factors to retain.
- **Eigenvalues**: Retain factors with eigenvalues greater than 1.

5. Factor Rotation

- **Orthogonal Rotation (Varimax, Quartimax):** Assumes that the factors are uncorrelated.
- **Oblique Rotation (Promax, Oblimin):** Allows the factors to be correlated.
- Rotate the factors to achieve a simpler and more interpretable factor structure.
- Examine the rotated factor loadings.

6. Interpret and Label the Factors

- Analyze the rotated factor loadings to interpret the underlying meaning of each factor.
- Assign meaningful labels to each factor based on the variables with high loadings on that factor.

7. Compute Factor Scores (if needed)

- Calculate the factor scores for each individual to represent their value on each factor.

8. Report and Validate the Results

- Report the final factor structure, including factor loadings and communalities.
- Validate the results using additional data or by conducting a confirmatory factor analysis if necessary.

PCA

- Principal Component Analysis (PCA) is a popular dimensionality reduction technique.
- It is often used to visualize datasets by projecting features onto 2 or 3 dimensional space.

- Transforming the variables to a new set of variables, which are known as the principal components (or simply, the PCs) and are orthogonal, ordered such that the retention of variation present in the original variables decreases as we move down in the order.
- The principal components are the eigenvectors of a covariance matrix, and hence they are orthogonal.
- Dimensionality: It is the number of random variables in a dataset or simply the number of features, or rather more simply, the number of columns present in your dataset.
- Correlation: It shows how strongly two variable are related to each other. The value of the same ranges for -1 to +1. Positive indicates that when one variable increases, the other increases as well, while negative indicates the other decreases on increasing the former. And the modulus value of indicates the strength of relation.
- Orthogonal: Uncorrelated to each other, i.e., correlation between any pair of variables is 0.
- Eigenvectors: Eigenvectors and Eigenvalues are in itself a big domain, let's restrict ourselves to the knowledge of the same which we would require here. So, consider a non-zero vector ***v***. It is an eigenvector of a square matrix ***A***, if ***Av*** is a scalar multiple of **v**. Or simply:

$$Av = \lambda v$$

Here, ***v*** is the eigenvector and λ is the eigenvalue associated with it.
- Covariance Matrix: This matrix consists of the covariances between the pairs of variables. The *(i,j)*th element is the covariance between *i*-th and *j*-th variable.

Properties of Principal Component

1. The PCs are essentially the linear combinations of the original variables, the weights vector in this combination is actually the eigenvector found which in turn satisfies the principle of least squares.
2. The PCs are orthogonal.
3. The variation present in the PCs decrease as we move from the 1st PC to the last one, hence the importance.

Geometric Rationale of PCA

- Degree to which the variables are linearly correlated is represented by their covariances.

$$C_{ij} = \frac{1}{n-1}\sum_{m=1}^{n}(X_{im} - \bar{X}_i)(X_{jm} - \bar{X}_j)$$

Where, C_{ij} = Covariance of variables *i* and *j*
m = Sum over all *n* objects
X_{im} Value of variable *i* in object *m*
X_i Mean of variable *i*
Xjm Value of variable *j* in object *m*
Xj Mean of variable *j*

Method

Step 1: Get some data

2 dimensions, and the reason why I have chosen this is so that I can provide plots of the data to show what the PCA analysis is doing at each step.

sStep 2: Subtract the mean

Subtract the mean from each of the data dimensions. The mean subtracted is the average across each dimension. So, all the x values have x̄ (the mean of the x values of all the data points) subtracted, and all the y values have $\bar{Y}$ subtracted from them. This produces a data set whose mean is zero.

Data =

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

DataAdjust =

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

Fig: PCA example data, original data on the left, data with the means subtracted on the right, and a plot of the data.

Step 3: Calculate the covariance matrix

The data is 2 dimensional, the covariance matrix will be 2 X 2.

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

Since the non-diagonal elements in this covariance matrix are positive, we should expect that both the x and y y variable increase together.

Step 4: Calculate the eigen vectors and eigen values of the Covariance matrix

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

Step 5: Choosing components and forming a feature vector

The eigen vector with the highest eigen value is the principle component of the data set.

$$FeatureVector = (eig_1\ eig_2\ eig_3\ ....\ eig_n)$$

We can either form a feature vector with both of the eigen vectors:

$$\begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

Choose to leave out the smaller, less significant component and only have a single column:

$$\begin{pmatrix} -.677873399 \\ -.735178656 \end{pmatrix}$$

Step 6: Deriving the new data set

Take the transpose of the vector and multiply it on the left of the original data set, transposed.

$$FinalData = RowFeatureVector \times RowDataAdjust,$$

where, *RowFeatureVector* is the matrix with the eigen vectors in the columns transposed so that the eigen vectors are now in the rows, with the most significant eigen vector at the top, and *RowDataAdjust* is the mean-adjusted data transposed, ie. the data items are in each column, with each row holding a separate dimension.

CHOOSING THE NUMBER OF LATENT DIMENSIONS

- **A latent space**, also known as a **latent feature space or embedding space**, is an embedding of a set of items within a manifold in which items resembling each other are positioned closer to one another. Position within the latent space can be viewed as being defined by a set of latent variables that emerge from the resemblances from the objects.
- In most cases, **the dimensionality of the latent space** is chosen to be lower than the dimensionality of the feature space from which the data points are drawn, making the construction of a latent space an example of dimensionality reduction, which can also be viewed as a form of data compression. Latent spaces are usually fit via machine learning, and they can then be used as feature spaces in machine learning models, including classifiers and other supervised predictors.

Model selection for probabilistic models

The optimal Bayesian approach is to pick the model with the largest marginal likelihood,

$$K^* = \mathrm{argmax}_k\ p(D|K)$$

There are two problems with this.

- First, evaluating the marginal likelihood for LVMs is quite difficult.
- The second issue is the need to search over a potentially large number of models.

Model selection for non-probabilistic methods

Define the squared reconstruction error of a data set D, using model complexity K, as follows:

$$E(\mathcal{D}, K) = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2$$

In supervised learning, we can always use cross validation to select between non-probabilistic models of different complexity, but this is not the case with unsupervised learning.

CLUSTERING

- Clustering is an unsupervised machine-learning approach that is used to group comparable data points based on specific traits or attributes.
- Clustering is the process of grouping similar objects together.

MEASURING (DIS) SIMILARITY

- In **similarity-based clustering**, the input to the algorithm is an N ×N dissimilarity matrix or distance matrix D. Similarity-based clustering has the advantage that it allows for easy inclusion of domain-specific similarity or kernel functions.
- In **feature-based clustering**, the input to the algorithm is an N × D feature matrix or design matrix X. Feature- based clustering has the advantage that it is applicable to "raw", potentially noisy data.

In addition to the two types of input, there are two possible types of output:

- **Flat clustering**, also called partitional clustering, where we partition the objects into disjoint sets; flat clustering are usually faster to create (O(ND) for flat
- **Hierarchical clustering**, where we create a nested tree of partitions. vs O(N^2 log N) for hierarchical)

MEASURING (DIS)SIMILARITY MATRIX

A dissimilarity matrix D is a matrix where $d_{i,i} = 0$ and $d_{i,j} \geq 0$ is a measure of "distance" between objects i and j. the triangle inequality, $d_{i,j} \leq d_{i,k} + d_{j,k}$, often does not hold. Some algorithms require **D** to be a true distance matrix, but many do not. If we have a similarity matrix **S**, we can convert it to a dissimilarity matrix by applying any monotonically decreasing function, e.g.,

$$\mathbf{D} = \max(\mathbf{S}) - \mathbf{S}.$$

The most common way to define dissimilarity between objects is in terms of the dissimilarity of their attributes:

$$\Delta(\mathbf{x}_i, \mathbf{x}_{i'}) = \sum_{j=1}^{D} \Delta_j(x_{ij}, x_{i'j})$$

Some common attribute dissimilarity functions are as follows:

- **Squared (Euclidean) distance**:

$$\Delta_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2$$

- **City Block Distance**: In 2D, the distance can be computed by counting how many rows and columns we have to move horizontally and vertically to get from x_i to x_i'

$$\Delta_j(x_{ij}, x_{i'j}) = |x_{ij} - x_{i'j}|$$

- **Correlation:** If $\mathbf{x}_i$ is a vector (e.g., a time-series of real-valued data), it is common to use the correlation coefficient. If the data is standardized**,**

$$\text{corr}[\mathbf{x}_i, \mathbf{x}_{i'}] = \sum_j x_{ij} x_{i'j},$$

$$\sum_j (x_{ij} - x_{i'j})^2 = 2(1 - \text{corr}[\mathbf{x}_i, \mathbf{x}_{i'}]).$$

- **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 possible values.
- **For categorical variables**: variables, such as {red, green, blue}, we usually assign a distance of 1 if the features are different, and a distance of 0 otherwise. Summing up over all the categorical features gives:

$$\Delta(\mathbf{x}_i, \mathbf{x}_i) = \sum_{j=1}^{D} \mathbb{I}(x_{ij} \neq x_{i'j})$$

  This is called the **hamming distance**.

EVALUATING THE OUTPUT OF CLUSTERING METHODS

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.

a) **Purity:** Let N_{ij} be the number of objects in cluster i that belong to class j, and let

$$N_i = \sum_{j=1}^{C} N_{ij}$$

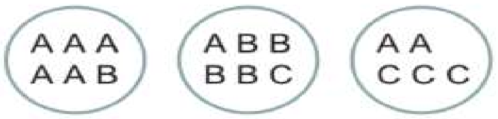
be the total number of objects in cluster *i*. Define **p_{ij} = N*ij*/N$_i$**; this is the empirical distribution over class labels for cluster i. We define the **purity** of a cluster as

$$p_i \triangleq \max_j p_{ij},$$

and the overall purity of a clustering as:

$$\text{purity} \triangleq \sum_i \frac{N_i}{N} p_i$$

Example: Three clusters with labelled objects inside.



The purity is:

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

The purity ranges between 0 (bad) and 1 (good). However, we can trivially achieve a purity of 1 by putting each object into its own cluster, so this measure does not penalize for the number of clusters.

b) **Rand index:** Let **U = {u$_1$,...,u$_R$}** and **V = {v$_1$,...,V$_C$ }** be two different partitions of the N data points, i.e., two different (flat) clustering's. For example, U might be the estimated clustering and V is reference clustering derived from the class labels. Now define a 2 × 2 contingency table, containing the following numbers:

- **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).

A common summary statistic is the Rand index:

$$R \triangleq \frac{TP + TN}{TP + FP + FN + TN}$$

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

The 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 label) is:

$$TP + FP = \binom{6}{2} + \binom{6}{2} + \binom{5}{2} = 40$$

Of these, the number of true positives is given by

$$TP = \binom{5}{2} + \binom{5}{2} + \binom{3}{2} + \binom{2}{2} = 20$$

where the last two terms come from cluster 3: there are $\binom{3}{2}$ pairs labeled C and $\binom{2}{2}$ pairs labelled A. So FP = 40 − 20 = 20. Similarly, one can show FN = 24 and TN = 72. So the Rand index is (20 + 72)/(20 + 20 + 24 + 72) = 0.68.

The Rand index only achieves its lower bound of 0 if TP = TN = 0, which is a rare event.

One can define an **adjusted Rand index** (Hubert and Arabie 1985) as follows:

$$AR \triangleq \frac{\text{index} - \text{expected index}}{\text{max index} - \text{expected index}}$$

c) **Mutual Information:** To measure cluster quality is to compute the mutual information between U and V:

$$p_{UV}(i,j) = \frac{|u_i \cap v_j|}{N}$$

be the probability that a randomly chosen object belongs to cluster u_i in U and v_j in V. Also, let **P_U (i) = |u_i|/N** be the be the probability that a randomly chosen object belongs to cluster u_i in U; define**: P_V (j) = |v_j |/N** similarly. Then we have:

$$\mathbb{I}(U,V) = \sum_{i=1}^{R}\sum_{j=1}^{C} p_{UV}(i,j) \log \frac{p_{UV}(i,j)}{p_U(i)p_V(j)}$$

This lies between **0 and min {H (U), H (V)}.** Unfortunately, the maximum value can be achieved by using lots of small clusters, which have low entropy. To compensate for this, we can use the normalized mutual information,

$$NMI(U,V) \triangleq \frac{\mathbb{I}(U,V)}{(\mathbb{H}(U) + \mathbb{H}(V))/2}$$

This lies between 0 and 1.

HIERARCHICAL CLUSTERING

Mixture models, whether finite or infinite, produce a “flat” clustering. We want to learn a **hierarchical clustering**, where clusters can be nested inside each other.

There are **two main approaches** to hierarchical clustering:

1. **Bottom-up or Agglomerative clustering**: In the bottom-up approach, the most similar groups are merged at each step.
2. **Top-down or Divisive clustering**: In the top-down approach, groups are split using various different criteria.

Both methods take as input a dissimilarity matrix between the objects.

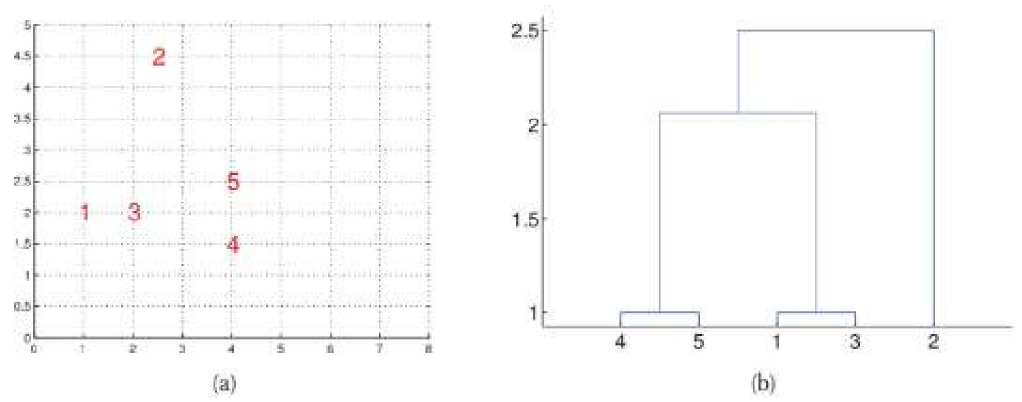


Figure 1: (a) An example of single link clustering using city block distance. Pairs (1,3) and (4,5) are both distance 1 apart, so get merged first. (b) The resulting dendrogram.

a) AGGLOMERATIVE CLUSTERING

Agglomerative clustering starts with N groups, each initially containing one object, and then at each step it merges the two most similar groups until there is a single group, containing all the data. Picking the two most similar clusters to merge takes O (N^2) time, and there are O (N) steps in the algorithm, the total running time is O (N^3). By using a priority queue, this can be reduced to O (N^2 log N).

For large N, a common heuristic is to first run K-means, which takes O(KND) time, and then apply hierarchical clustering to the estimated cluster centres. The merging process can be represented by a binary tree, called a **dendrogram**.

The initial groups (objects) are at the leaves (at the bottom of the figure), and every time two groups are merged, we join them in the tree.

- The **height** of the branches represents the dissimilarity between the groups that are being joined.
- The **root** of the tree (which is at the top) represents a group containing all the data.
- If we cut the tree at any given height, we induce a clustering of a given size.
- For example, if we cut the tree in Figure 1 (b) at height 2, we get the clustering {{{4, 5}, {1, 3}}, {2}}. We discuss the issue of how to choose the height/ number of clusters below.

Algorithm 25.2: Agglomerative clustering

```
1  initialize clusters as singletons: for $i \leftarrow 1$ to $n$ do $C_i \leftarrow \{i\}$;
2  initialize set of clusters available for merging: $S \leftarrow \{1, \dots, n\}$;
3  repeat
4      Pick 2 most similar clusters to merge: $(j, k) \leftarrow \arg\min_{j,k \in S} d_{j,k}$;
5      Create new cluster $C_\ell \leftarrow C_j \cup C_k$;
6      Mark $j$ and $k$ as unavailable: $S \leftarrow S \setminus \{j, k\}$;
7      if $C_\ell \neq \{1, \dots, n\}$ then
8          Mark $\ell$ as available, $S \leftarrow S \cup \{\ell\}$;
9      foreach $i \in S$ do
10         Update dissimilarity matrix $d(i, \ell)$;
11 until no more clusters are available for merging;
```

There are actually three variants of agglomerative clustering, depending on how we define the dissimilarity between groups of objects:

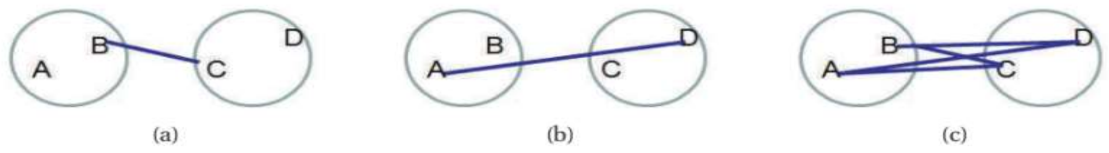
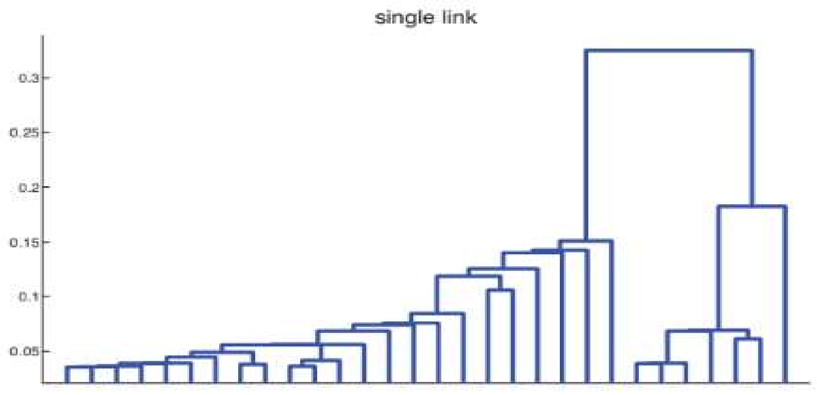


Figure: Illustration of (a) Single linkage. (b) Complete linkage. (c) Average linkage.

a) **Single link:** In single link clustering, also called nearest neighbor clustering, the distance between two groups G and H is defined as the distance between the two closest members of each group:

$$d_{SL}(G,H) = \min_{i \in G, i' \in H} d_{i,i'}$$

The tree built using single link clustering is a minimum spanning tree of the data, which is a tree that connects all the objects in a way that minimizes the sum of the edge weights (distances). We can actually implement single link clustering in O(N^2) time, whereas the other variants take O(N^3) time.

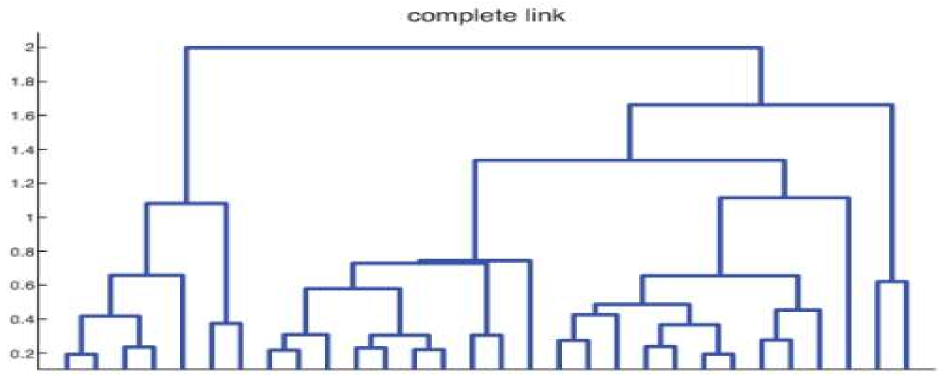


b) **Complete link:** In complete link clustering, also called furthest neighbour clustering, the distance between two groups is defined as the distance between the two most distant pairs:

$$d_{CL}(G,H) = \max_{i \in G, i' \in H} d_{i,i'}$$

Thus, clusters can be formed that violate the **compactness property**, which says that all the observations within a group should be similar to each other. In particular if we define the diameter of a group as the largest dissimilarity of its members;

$$d_G = \max_{i \in G, i' \in G} d_{i,i'}$$

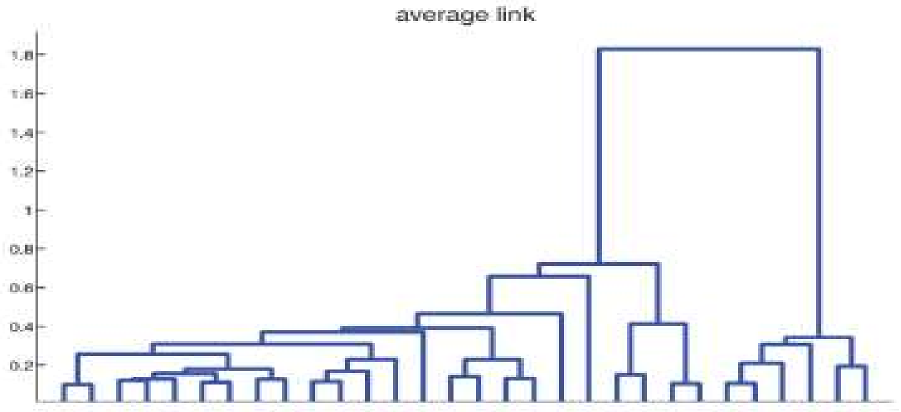


c) Average link:

The preferred method is average link clustering, which measures the average distance between all pairs:

$$d_{avg}(G,H) = \frac{1}{n_G n_H} \sum_{i \in G} \sum_{i' \in H} d_{i,i'}$$

where n_G and n_H are the number of elements in groups G and H.
Average link clustering represents a compromise between single and complete link clustering. It tends to produce relatively compact clusters that are relatively far apart.



b) DIVISIVE CLUSTERING

Divisive clustering starts with all the data in a single cluster, and then recursively divides each cluster into two daughter clusters, in a top-down fashion. Since there are $2^{N-1} - 1$ ways to split a group of N items into 2 groups, it is hard to compute the optimal split, so various heuristics are used.

a) **Bisecting K-Means Algorithm**: One approach is picking the cluster with the largest diameter, and split it in two using the K-means or K-medoids algorithm with K = 2. This is called the **bisecting K-means algorithm**. We can repeat this until we have any desired number of clusters. This can be used as an alternative to regular K-means, but it also induces a hierarchical clustering.

b) **Minimum Spanning Tree**: Another method is to build a minimum spanning tree from the dissimilarity graph, and then to make new clusters by breaking the link corresponding to the largest dissimilarity.

c) **Dissimilarity Analysis:** We start with a single cluster containing all the data, G = {1,...,N}. We then measure the average dissimilarity of i ∈ G to all the other i ′ ∈ G:

$$d_i^G = \frac{1}{n_G} \sum_{i' \in G} d_{i,i'}$$

We remove the most dissimilar object and put it in its own cluster H:

$$i^* = \arg\max_{i \in G} d_i^G, \quad G = G \setminus \{i^*\}, \quad H = \{i^*\}$$

We now continue to move objects from G to H until some stopping criterion is met. Specifically, we pick a point i^* to move that maximizes the average dissimilarity to each i′ ∈ G but minimizes the average dissimilarity to each i′ ∈ H:

$$d_i^H = \frac{1}{n_H} \sum_{i' \in H} d_{i,i'}, \quad i^* = \arg\max_{i \in G} d_i^G - d_i^H$$

We continue to do this until $d^G i - d^H i$ is negative. The final result is that we have split G into two daughter clusters, G and H. We can then recursively call the algorithm on G and/or H, or on any other node in the tree.

d) **Bayesian Hierarchical Clustering:** It uses Bayesian hypothesis tests to decide which clusters to merge (if any), rather than computing the similarity between groups of points in some ad-hoc way.

Algorithm 25.3: Bayesian hierarchical clustering

```
1 Initialize $\mathcal{D}_i = \{\mathbf{x}_i\}$, $i = 1 : N$ ;
2 Compute $p(\mathcal{D}_i|T_i)$, $i = 1 : N$ ;
3 repeat
4     for each pair of clusters $i, j$ do
5         Compute $p(\mathcal{D}_{ij}|T_{ij})$
6     Find the pair $\mathcal{D}_i$ and $\mathcal{D}_j$ with highest merge probability $r_{ij}$;
7     Merge $\mathcal{D}_k := \mathcal{D}_i \cup \mathcal{D}_j$;
8     Delete $\mathcal{D}_i$, $\mathcal{D}_j$ ;
9 until all clusters merged;
```

Advantages:

- It can be faster; if we only split for a constant number of levels, it takes just O(n) time.
- The splitting decision are made in the context of seeing all the data; whereas bottom up methods make myopic merge decisions.

Disadvantages:

Divisive clustering is less popular than agglomerative clustering, but it has two advantages.

- First, it can be faster, since if we only split for a constant number of levels, it takes just O(N) time.
- Second, the splitting decisions are made in the context of seeing all the data, whereas bottom-up methods make myopic merge decisions.