REVA UNIVERSITY
Bengaluru, India

M23DE0201 – Machine Learning

II Semester MCA D Academic Year : 2024 - 2025

School of Computer Science and Applications

Dr.P SREE LAKSHMI
Assistant Professor

www.reva.edu.in

Course content:

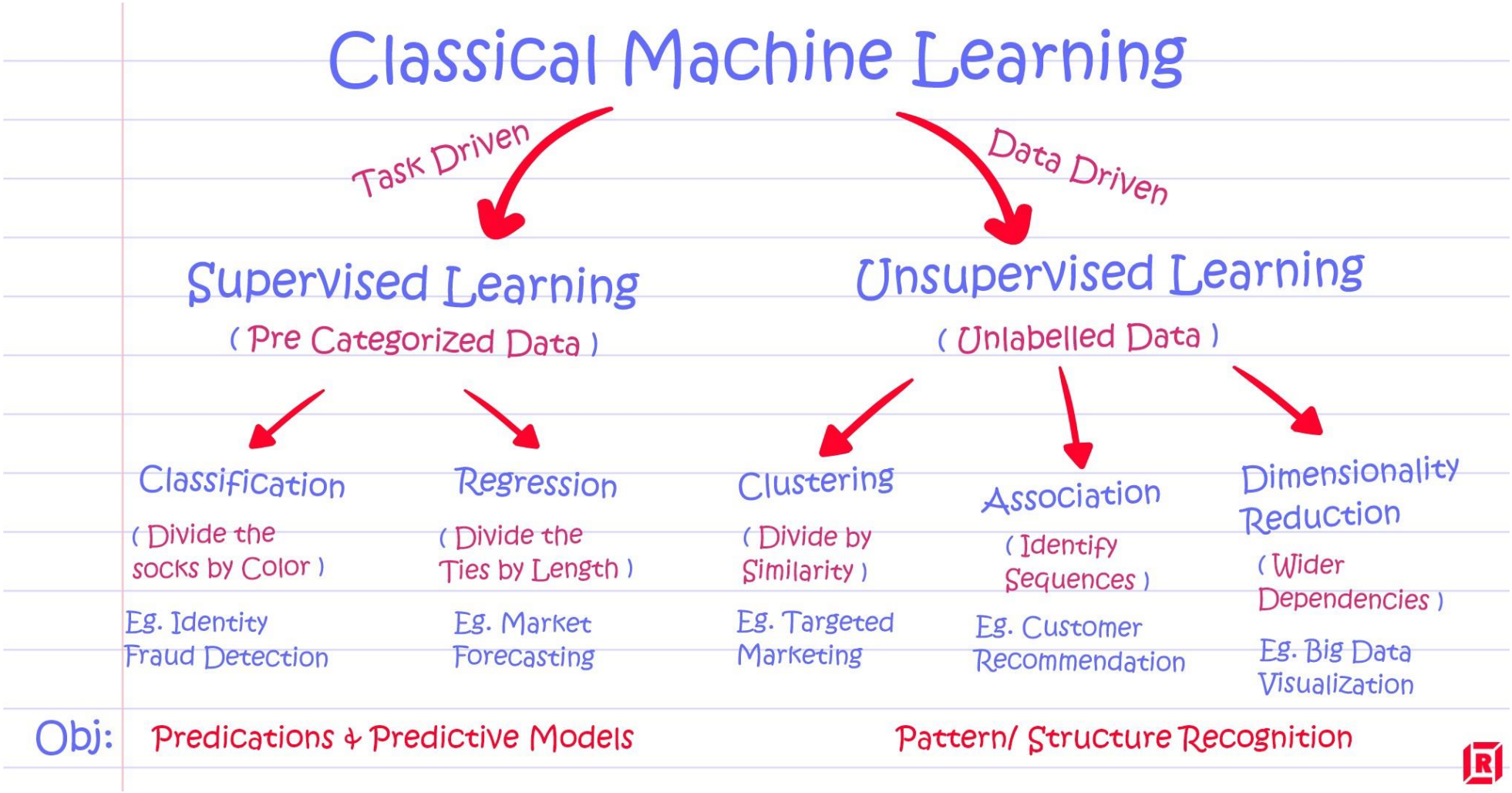
UNIT 3: **[14 Hours]**

Unsupervised Learning: Introduction, types and challenges, preprocessing and scaling of datasets, Dimensionality reduction, feature extraction. Principal Component Analysis (PCA), k-means, agglomerative and DBSCAN clustering algorithms.

UNSUPERVISED LEARNING (UL)

- learning from observation and discovery
- is a type of algorithm that learns patterns from untagged/ Unlabelled data
- The machine is forced to build a compact internal representation of its world. In contrast to supervised learning (SL) where data is tagged by a human, e.g. as "car" or "fish" etc,
- UL exhibits self-organization that captures patterns as neuronal predilection or probability densities.
- In Unsupervised Learning, only the input data is known, and no known output data is given to the algorithm.
- For Ex., Identifying topics in a set of blog posts, Segmenting Customers into groups with similar preferences

TYPES OF UNSUPERVISED LEARNING



UNSUPERVISED LEARNING TYPES

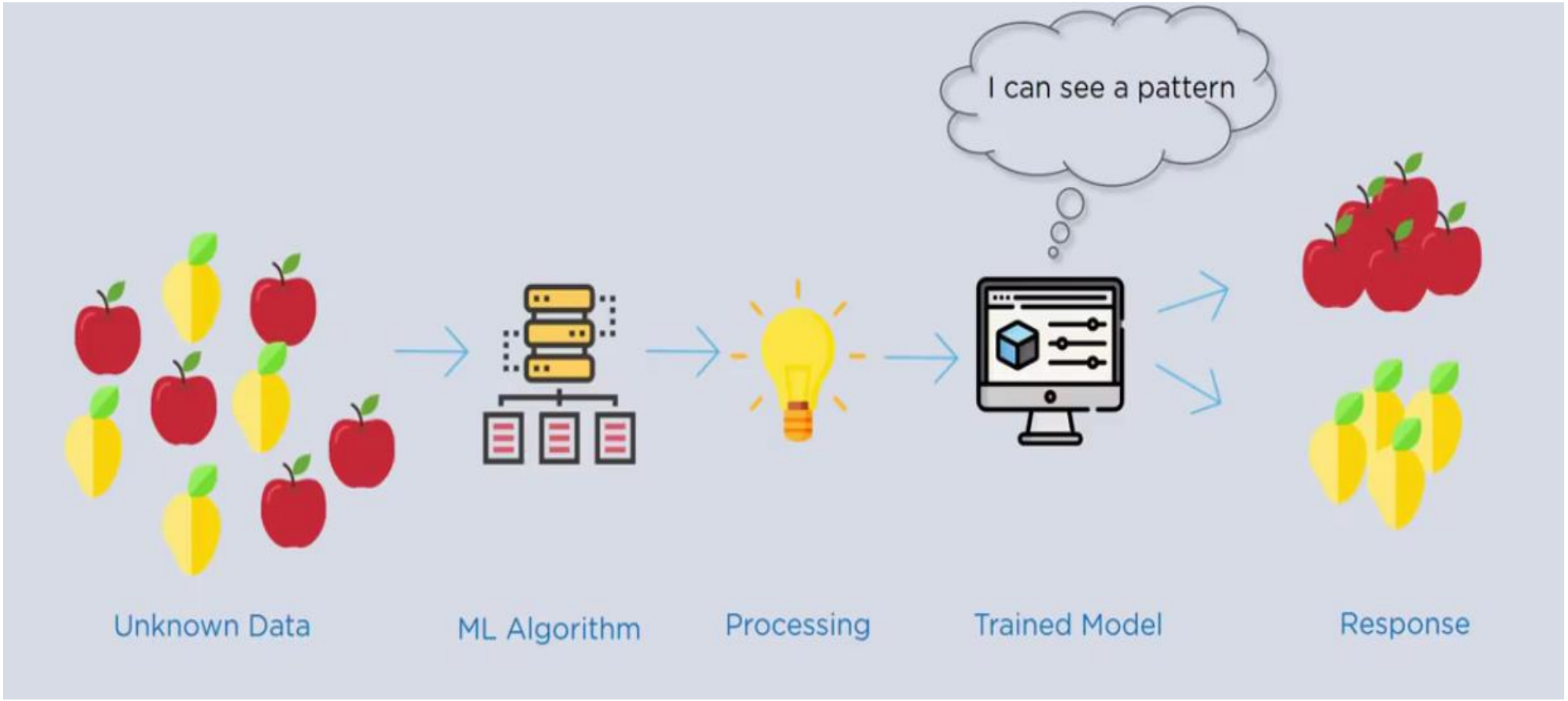
sample Cluster/group

1) Clustering

- **Cluster:** A collection of data objects
  - similar (or related) to one another within the same group
  - dissimilar (or unrelated) to the objects in other groups
- **Cluster analysis (or *clustering*, *data segmentation*, …)**
  - Finding similarities between data according to the characteristics found in the data and grouping similar data objects into clusters
- **Unsupervised learning**: no predefined classes (i.e., *learning by observations* vs. learning by examples: supervised)
- **Typical applications**
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

UNSUPERVISED LEARNING

- Learning from observation and discovery



MAJOR CLUSTERING APPROACHES

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

MAJOR CLUSTERING APPROACHES

- Model-based:
  - A model is hypothesized for each of the clusters and tries to find the best fit of that model to each other
  - Typical methods: EM, SOM, COBWEB
- Frequent pattern-based:
  - Based on the analysis of frequent patterns
  - Typical methods: p-Cluster
- User-guided or constraint-based:
  - Clustering by considering user-specified or application-specific constraints
  - Typical methods: COD (obstacles), constrained clustering
- Link-based clustering:
  - Objects are often linked together in various ways
  - Massive links can be used to cluster objects: SimRank, LinkClus

2) ASSOCIATION:

- An unsupervised learning method, used to find **the relationships between variables in the large database.**
- It determines the set of items that occur together in the dataset.
- Association rule makes marketing strategy more effective.
- Such as People who buy X items (suppose bread) also tend to purchase Y (Butter/Jam) items. A typical example of an Association rule is Market Basket Analysis.

APPLICATIONS OF ASSOCIATION RULE LEARNING

Market Basket Analysis:

used by big retailers to determine the association between items.

Medical Diagnosis:

With the help of association rules, patients can be cured easily, as it helps in identifying the probability of illness for a particular disease.

Protein Sequence:

The association rules help in determining the synthesis of artificial Proteins.

3) DIMENSIONALITY REDUCTION

- Refers to techniques for reducing the number of input variables in training data.
- When dealing with high dimensional data, it is often useful to reduce the dimensionality by projecting the data to a lower dimensional subspace which captures the “essence” of the data. This is called dimensionality reduction.
- Dimensionality reduction is a data preparation technique performed on data before modeling.
- It might be performed after data cleaning and data scaling and before training a predictive model.
- any dimensionality reduction performed on training data must also be performed on new data, such as a test dataset, validation dataset, and data when predicting with the final model.

Dimensionality Reduction Techniques: Examples

- Manifold Learning (t-SNE, UMAP)
- Principal Component Analysis (PCA)
- Independent Component Analysis (ICA)
- Sequential Non-negative Matrix Factorization (NMF)
- Linear Discriminant Analysis (LDA)
- Generalized Discriminant Analysis (GDA)
- Missing Values Ratio (MVR): Threshold Setting
- Low Variance Filter
- High Correlation Filter
- Forward Feature Construction
- Backward Feature Elimination
- Autoencoders

UNSUPERVISED LEARNING: CHALLENGES

- **Noisy Data**: Outliers and noise can distort patterns and reduce the effectiveness of algorithms.
- **Assumption Dependence**: Algorithms often rely on assumptions (e.g., cluster shapes), which may not match the actual data structure.
- **Overfitting Risk**: Overfitting can occur when models capture noise instead of meaningful patterns in the data.
- **Limited Guidance**: The absence of labels restricts the ability to guide the algorithm toward specific outcomes.
- **Cluster Interpretability**: Results, such as clusters, may lack clear meaning or alignment with real-world categories.
- **Sensitivity to Parameters**: Many algorithms require careful tuning of hyperparameters, such as the number of clusters in k-means.
- **Lack of Ground Truth**: Unsupervised learning lacks labeled data, making it difficult to evaluate the accuracy of results.

PREPROCESSING AND SCALING OF DATASETS

Refer UNIT 1 content

MAJOR TASKS IN DATA PREPROCESSING

- **Data cleaning**
  - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- **Data integration**
  - Integration of multiple databases, data cubes, or files
- **Data reduction**
  - Dimensionality reduction
  - Numerosity reduction
  - Data compression
- **Data transformation and data discretization**
  - Normalization
  - Concept hierarchy generation

SCALING OF DATASETS- NORMALIZATION

- **Min-max normalization** performs a linear transformation on the original data. Suppose that min_A and max_A are the minimum and maximum values of an attribute, A. Min-max normalization maps a value, v_i, of A to v_i' in the range $[new\_min_A, new\_max_A]$ by computing
  - Fourth level
    - Fifth level

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

$$\frac{73{,}600 - 12{,}000}{98{,}000 - 12{,}000}(1.0 - 0) + 0 = 0.716$$

NORMALIZATION

- Edit Master text styles

In **z-score normalization** (or *zero-mean normalization*), the values for an attribute, A, are normalized based on the mean (i.e., average) and standard deviation of A. A value, v_i, of A is normalized to v_i' by computing

- Fifth level

$$v' = \frac{v - \mu_A}{\sigma_A}$$

where $\bar{A}$ and σ_A are the mean and standard deviation, respectively, of attribute A.

$$\frac{73{,}600 - 54{,}000}{16{,}000} = 1.225$$

3) Normalization by decimal scaling

Normalization by decimal scaling normalizes by moving the decimal point of values of attribute A. The number of decimal points moved depends on the maximum absolute value of A.

A value, v_i, of A is normalized to v_i' by computing $v' = \frac{v}{10^j}$

Example: Suppose that the recorded values of A range from −986 to 917.

The maximum absolute value of A is 986. To normalize by decimal scaling, we therefore divide each value by 1000 (i.e., j = 3)

so that −986 normalizes to −0.986 and 917 normalizes to 0.917.

DIMENSIONALITY REDUCTION

What is Dimensionality Reduction?

- The number of input features, variables, or columns present in a given dataset is known as **dimensionality,** and the process to reduce these features is called dimensionality reduction.
- ***"It is a way of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information."***
- These techniques are widely used in machine learning for obtaining a better fit predictive model while solving the classification and regression problems.
- commonly used in the fields that deal with high-dimensional data, such as **speech recognition, signal processing, neuroinformatics**, **bioinformatics, data visualization, noise reduction, cluster analysis**, etc.

Feature Extraction:

- Feature extraction is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing.
- A characteristic of these large data sets is a large number of variables that require a lot of computing resources to process.
- Feature extraction is the name for methods that select and /or combine variables into features, effectively reducing the amount of data that must be processed, while still accurately and completely describing the original data set.

Why is this Useful?

- The process of feature extraction is useful when you need to reduce the number of resources needed for processing without losing important or relevant information.
- Feature extraction can also reduce the amount of redundant data for a given analysis.
- Also, the reduction of the data and the machine's efforts in building variable combinations (features) facilitate the speed of learning and generalization steps in the machine learning process.

PRACTICAL USES OF FEATURE EXTRACTION

- **Autoencoders**– The purpose of autoencoders is unsupervised learning of efficient data coding. Feature extraction is used here **to identify key features in the data for coding by learning from the coding of the original data set to derive new ones**.
- **Bag-of-Words**– A technique for natural language processing that extracts the words (features) used in a sentence, document, website, etc. and classifies them by frequency of use. This technique can also be applied to image processing.
- **Image Processing** – Algorithms are used to detect features such as shaped, edges, or motion in a digital image or video.

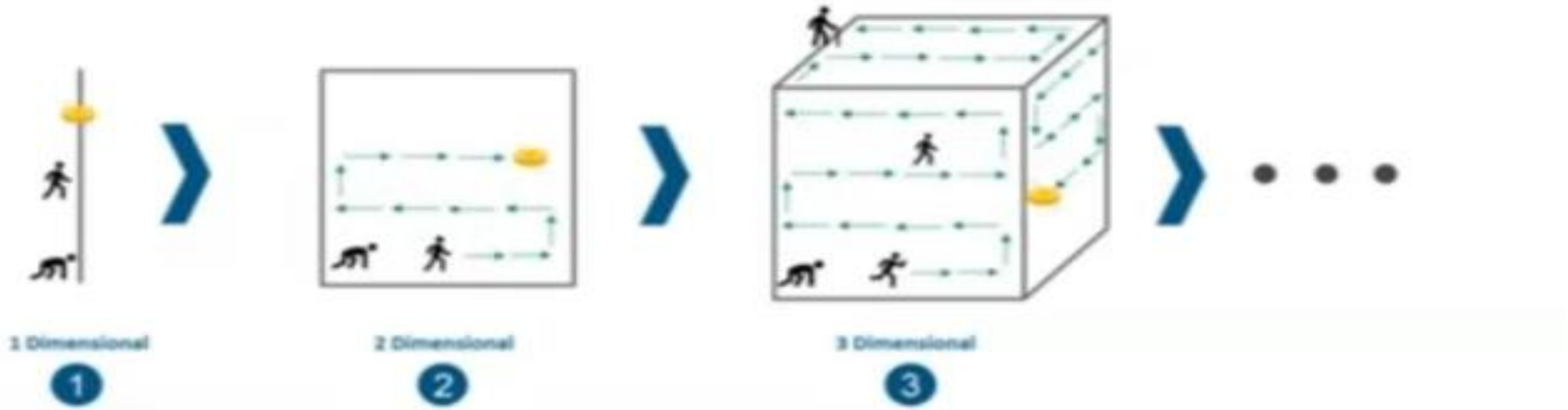
Some common feature extraction techniques are:

1. Principal Component Analysis
2. Linear Discriminant Analysis
3. Kernel PCA
4. Quadratic Discriminant Analysis

PRINCIPAL COMPONENT ANALYSIS - PCA

Need of PCA

High dimensional data is extremely complex to process due to inconsistencies in the feature which increase the computation time and make data processing and EDA more convoluted.



Curse of dimensionality

WHAT IS PCA?

PCA is a technique used to:

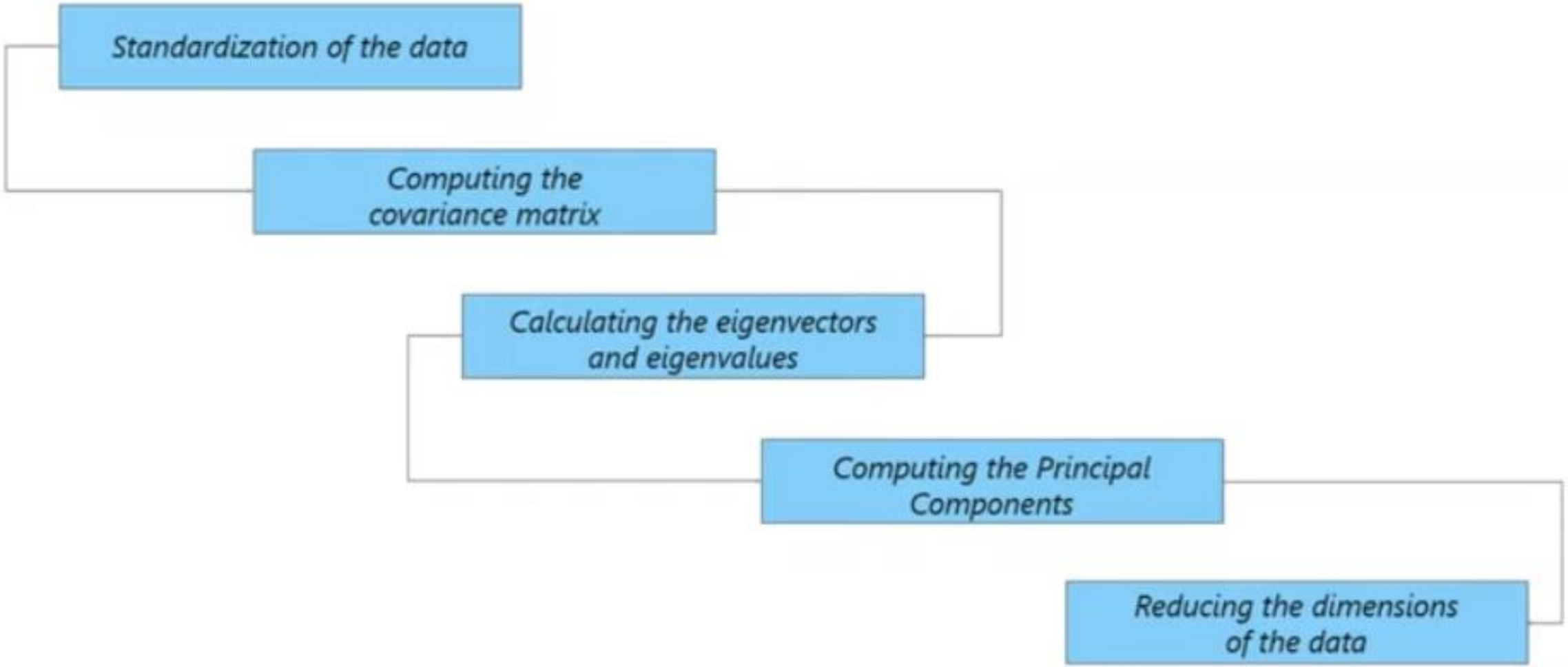
- **Reduce the number of features (dimensions)** in the dataset
- While keeping **as much useful information (variance)** as possible
- It finds new axes (directions) — called **principal components** — where your data varies the most, and **projects your data onto them**.

Real-Life Analogy:

Imagine you are looking at a 3D object (say, a football). You want to draw it on paper — a 2D space — but still make it look as realistic as possible.

PCA helps you **choose the best angle** to flatten the object **without losing details**.

PCA - STEPS


Standardization of the data
Computing the covariance matrix
Calculating the eigenvectors and eigenvalues
Computing the Principal Components
Reducing the dimensions of the data

Step 1:Standardization of the data

- Standardization is all about scaling your data in such a way that all the variables and their values lie within a similar range.
- If you don't do standardization output will be biased.

Rating	# of downloads
5	1383
3	668
2	763
5	839
1	342

Rating feature ranges between 0-5

# of downloads ranges between 100-5000

$$Z = \frac{\text{Variable value - mean}}{\text{Standard deviation}}$$

Step 2: Computing the covariance matrix

- Covariance matrix expresses the correlation between the different variables in the dataset.
- It is essential to identify heavily dependent variables because they contain biased and redundant information, which reduces the overall performance of the model.
- The covariance matrix is a $p \times p$ symmetric matrix (where p is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables. For example, for a 2-dimensional data set with 2 variables a, b, the covariance matrix is a 2×2matrix of this from:

$$\mathrm{Cov}(X,Y) = \frac{1}{n-1}\sum(X_i - \bar{X})(Y_i - \bar{Y})$$

$$\begin{bmatrix} Cov(a,a) & Cov(a,b) \\ Cov(b,a) & Cov(b,b) \end{bmatrix}$$

The covariance value denotes:

- *How co-dependent two variables are*
- *Negative covariance - indirectly proportional*
- *positive covariance - directly proportional to*

What do the covariances that we have as entries of the matrix tell us about the correlations between the variables?

- if positive then : the two variables increase or decrease together (correlated)
- if negative then : One increases when the other decreases (Inversely correlated)

the covariance matrix is not more than a table that summaries the correlations between all the possible pairs of variables,

Step 3: Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components

Eigen vectors and eigen values are mathematical constructs that must be computed from the covariance matrix in order to determine the ***principal components*** of the data set.

- **Eigenvectors** indicate the **direction** (axes) in which the data varies the most.
- **Eigenvalues** tell us the **magnitude** of variance in the direction of the corresponding eigenvector.
- They are computed from the **covariance matrix** of the dataset.

Why are They Important in PCA?

- PCA aims to **reduce the dimensionality** of data while retaining most of the variance.
- The **principal components** are the **eigenvectors** of the covariance matrix.
- The **largest eigenvalues** correspond to the **principal components** that capture the **most variance** in the data.

- Geometrically speaking, **principal components represent the directions of the data that explain a maximal amount of variance,** that is to say, the lines that capture most information of the data.

Formula:

$$|C - \lambda I| = 0 \quad \text{(to find eigenvalues)}$$

$$(C - \lambda I)v = 0 \quad \text{(to find eigenvectors)}$$

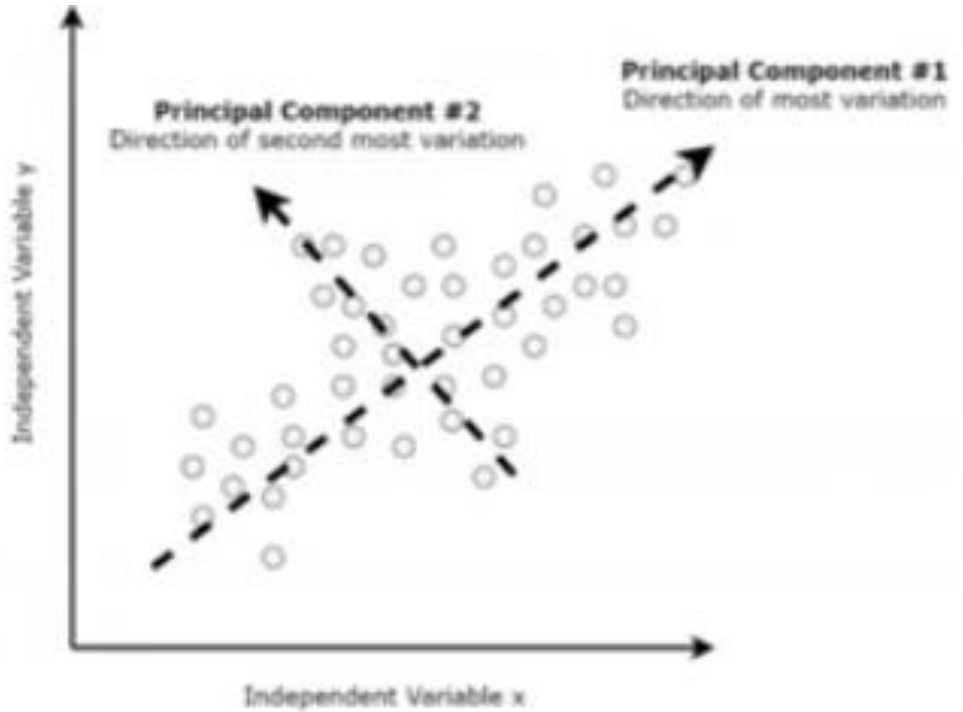
Where:

- λ is an eigenvalue.
- v is the corresponding eigenvector.
- I is the identity matrix.

Step 4: Computing the Principal Components

Once we have computed Eigen vectors and Eigen values, order them in the descending order, where the eigen vector with highest value is the most significant and thus forms first principal component

- *PC1 is the most significant and stores the maximum possible information.*
- *PC2 is the second most significant PC and stores remaining maximum info and so on*



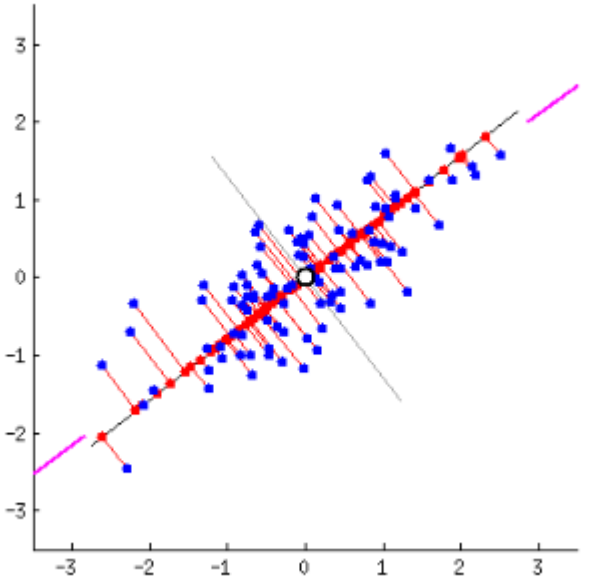
As there are as many principal components as there are variables in the data,

principal components are constructed in such a manner that the first principal component accounts for the **largest possible variance** in the data set.

For example, let's assume that the scatter plot of our data set is as shown below, can we guess the first principal component ?

Yes, it's approximately the line that matches the purple marks because it goes through the origin and it's the line in which the projection of the points (red dots) is the most spread out.

Or mathematically speaking, it's the line that maximizes the variance (the average of the squared distances from the projected points (red dots) to the origin).



- The second principal component is calculated in the same way, with the condition that it is uncorrelated with (i.e., perpendicular to) the first principal component and that it accounts for the next highest variance.
- This continues until a total of p principal components have been calculated, equal to the original number of variables.
- eigenvector has an eigenvalues number is equal to the number of dimensions of the data. For example, for a 3-dimensional data set, there are 3 variables, therefore there are 3 eigenvectors with 3 corresponding eigenvalues.
- eigenvectors of the Covariance matrix are actually *the directions of the axes where there is the most variance*(most information) and that we call Principal Components. And eigenvalues are simply the coefficients attached to eigenvectors, which give the *amount of variance carried in each Principal Component*.

Feature Vector

- The final step in computing the Principal Components is to form a matrix known as the feature matrix that contains all the significant data variables that possess maximum information about the data.
- feature vector is simply a matrix that has as columns the eigenvectors of the components that we decide to keep.
- This makes it the first step towards dimensionality reduction, because if we choose to keep only ***p*** eigenvectors (components) out of ***n***, the final data set will have only ***p*** dimensions.

Step 5: Reducing the dimensions of the dataset

- Rearrange the original data with the final principal components which represent the maximum and the most significant information of the dataset.
- In order to replace the original data axis with the newly formed Principal Components, you simply multiply the transpose of the original data set by the transpose of the obtained feature vector.

$$FinalDataSet = FeatureVector^T * StandardizedOriginalDataSet^T$$

Original data

	sepal length	sepal width	petal length	petal width
0	-0.900681	1.032057	-1.341272	-1.312977
1	-1.143017	-0.124958	-1.341272	-1.312977
2	-1.385353	0.337848	-1.398138	-1.312977
3	-1.506521	0.106445	-1.284407	-1.312977
4	-1.021849	1.263460	-1.341272	-1.312977

Reduced data using PCA

	principal component 1	princial component 2
0	-2.264542	0.505704
1	-2.086426	-0.655405
2	-2.367950	-0.318477
3	-2.304197	-0.575368
4	-2.388777	0.674767

PRINCIPAL COMPONENT ANALYSIS(PCA)

Given N data vectors from n-dimensions, find k ≤ n orthogonal vectors (principal components) that can be best used to represent data.

- **Steps**

– Normalize input data: Each attribute falls within the same range

– Compute k orthonormal (unit) vectors, i.e., principal components

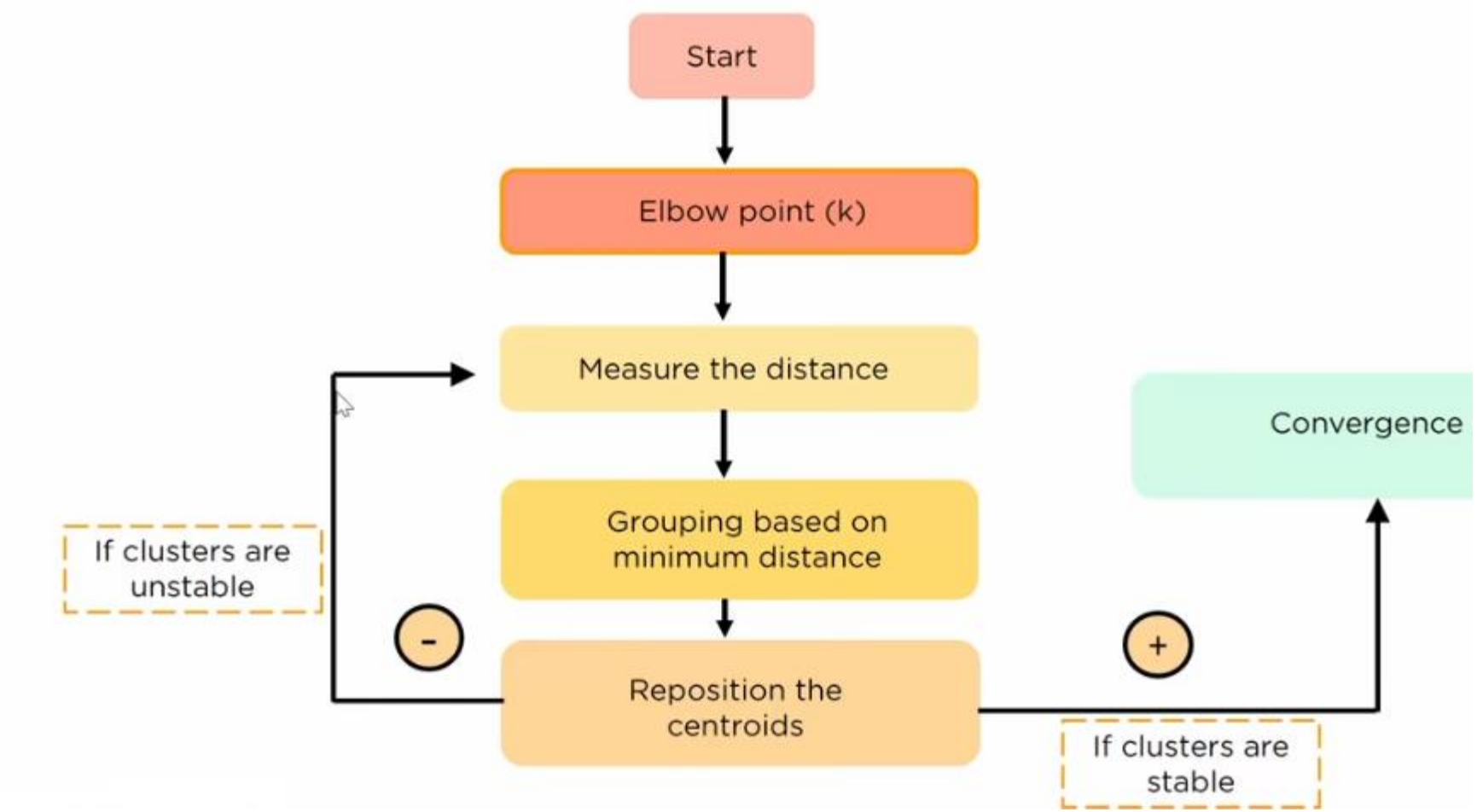
– Each input data (vector) is a linear combination of the k principal component vectors

– The principal components are sorted in order of decreasing “significance” or strength

– Since the components are sorted, the size of the data can be reduced by eliminating the weak components, i.e., those with low variance. (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data

- Works for numeric data only
- Used when the number of dimensions is large

HOW DOES K MEANS CLUSTERING WORKS



How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence

- Let's say, you have a dataset for a **Grocery shop**

- Now, the important question is, "*how would you choose the optimum number of clusters?*"

How does K-Means clustering work?

Elbow point

Measure the distance

Grouping

Reposition the centroids

Convergence

- The best way to do this is by **Elbow method**
- The idea of the elbow method is to run K-Means clustering on the dataset where 'k' is referred as number of clusters
- Within sum of squares (WSS) is defined as the sum of the squared distance between each member of the cluster and its centroid

$$WSS = \sum_{i=1}^{m} (x_i - c_i)^2$$

Where x_i = data point and c_i = closest point to centroid

How does K-Means clustering work?

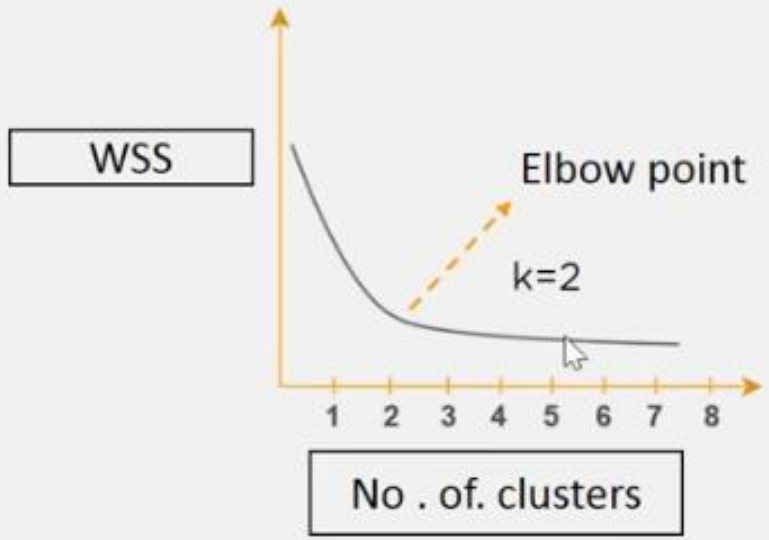
Elbow point

Measure the distance

Grouping

Reposition the centroids

Convergence



- Now, we draw a curve between *WSS* (within sum of squares) and the *number of clusters*
- Here, we can see a very slow change in the value of WSS after k=2, so you should take that elbow point value as the final number of clusters

How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence

Step 2: We can randomly initialize two points called the cluster centroids,

Euclidean distance is a distance measure used to find out which data point is closest to our centroids

How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence

Step 3: Based upon the distance from c1 and c2 centroids, the data points will group itself into clusters

How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence

Step 4: Compute the centroid of data points inside the blue cluster

Step 5: Reposition the centroid of the blue cluster to the new centroid

How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence

Step 6: Now, compute the centroid of data points inside the orange cluster

Step 7: Reposition the centroid of the orange cluster to the new centroid

How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence

Step 8: Once the clusters become static, K-Means clustering algorithm is said to be converged

K MEANS CLUSTERING ALGORITHM

Assuming we have inputs x1,x2,x3,..., and value of k,

Step 1 : Pick k random points as cluster centers called centroids

Step 2 : Assign each xi to nearest cluster by calculating its distance to each centroid

Step 3 : Find new cluster center by taking the average of the assigned points

Step 4 : Repeat Step 2 and 3 until none of the cluster assignments change

K-Means Clustering Algorithm

Step 1 :

We randomly pick K cluster centers (centroids). Let's assume these are c1,c2,...,ckc1,c2,...,ck, and we can say that;

$$C = c_1, c_2, \ldots . c_k$$

C is the set of all centroids.

Step 2:

In this step, we assign each data point to closest center, this is done by calculating Euclidean distance

$$\arg\min_{c_i \in C} dist\,(c_i, x)^2$$

Where **dist()** is the Euclidean distance.

K-Means Clustering Algorithm

Step 3:

In this step, we find the new centroid by taking the average of all the points assigned to that cluster.

$$c_i = \frac{1}{|Si|} \sum_{xi \in Si} x_i$$

S_i is the set of all points assigned to the i th cluster

Step 4:

In this step, we repeat step 2 and 3 until none of the cluster assignments change

That means until our clusters remain stable, we repeat the algorithm

PARTITIONAL ALGORITHMS – K - MEANS CLUSTERING

- {2,4,10,12,3,20,30,11,25}, k=2
- Randomly assign means : m_1=2, m_2=4

m_1	m_2	K_1	K_2
2	4	{2,3}	{4,10,12,20,30,11,25}
2.5	16	{2,3,4}	{10,12,20,30,11,25}
3	18	{2,3,4,10}	{12,20,30,11,25}
4.75	19.6	{2,3,4,10,11,12}	{20,30,25}
7	25	{2,3,4,10,11,12}	{20,30,25}

- Stop as the clusters with these means are the same
- Our answer
  - K_1 = {2,3,4,10,11,12}, K_2 = {20,30,25}

EXAMPLE 2

Identify the clusters by applying the K- Means algorithm, with k=2.

(assume initial mean m1=16,m2=22) for the following data set of a group of visitors using a website whose age is provided in a one-dimensional space.

15,15,16,19,19,20,20,21,22,28,35,40,41,42,43,44,60,61,65

Solution: Cluster1={15,15,16,19,19,20,20,21,22,28}

Cluster2= {35,40,41,42,43,44,60,61,65}

PARTITIONAL ALGORITHMS – K - MEANS CLUSTERING

- **Weakness**
  - Does not work on categorical data
  - Only convex-shaped cluster are found
  - Need to specify k, the number of clusters, in advance
  - Unable to handle noisy data and outliers
  - always tries to create the clusters of the same size.

  To solve these two challenges, we can opt for the hierarchical clustering algorithm because, in this algorithm, we don't need to have knowledge about the predefined number of clusters.

HIERARCHICAL CLUSTERING

- Hierarchical clustering is another unsupervised machine learning algorithm, which is used to group the unlabeled datasets into a cluster and also known as **hierarchical cluster analysis** or HCA.
- In this algorithm, we develop the hierarchy of clusters in the form of a tree, and this tree-shaped structure is known as the **dendrogram**

The hierarchical clustering technique has two approaches:

- Agglomerative clustering
- Divisive clustering

HIERARCHICAL ALGORITHMS

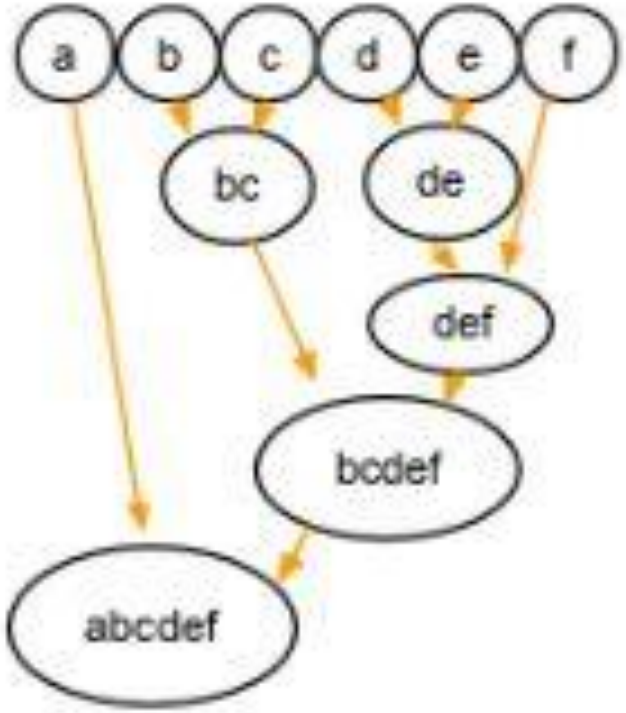
- Clusters are created in levels, creating sets of clusters at each level.
- Well-suited for many clustering applications that naturally exhibit a nesting relationship between clusters.
- Types of Hierarchical Algorithms
  - Agglomerative Algorithms
  - Divisive Algorithms

1) Agglomerative clustering

- is a bottom-up approach.
- We begin with each element as a separate cluster and merge them into successively more massive clusters, as shown below:

1. start with 1 point (singleton)
2. recursively add two or more appropriate clusters
3. Stop when k number of clusters is achieved.
4. Links:

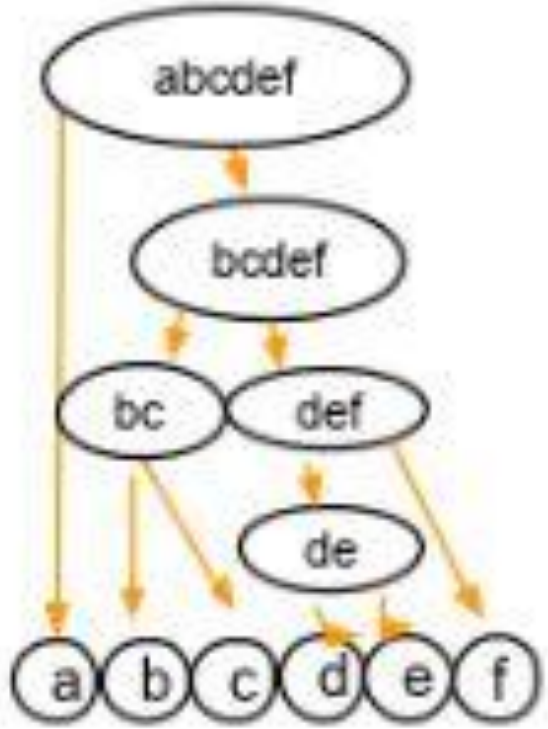
- Single Link
- MST Single Link
- Complete Link
- Average Link



2) Divisive clustering

- is a top-down approach.
- We begin with the whole set and proceed to divide it into successively smaller clusters, as you can see below:

1. Start with a big cluster
2. Recursively divide into smaller clusters
3. Stop when k number of clusters is achieved.



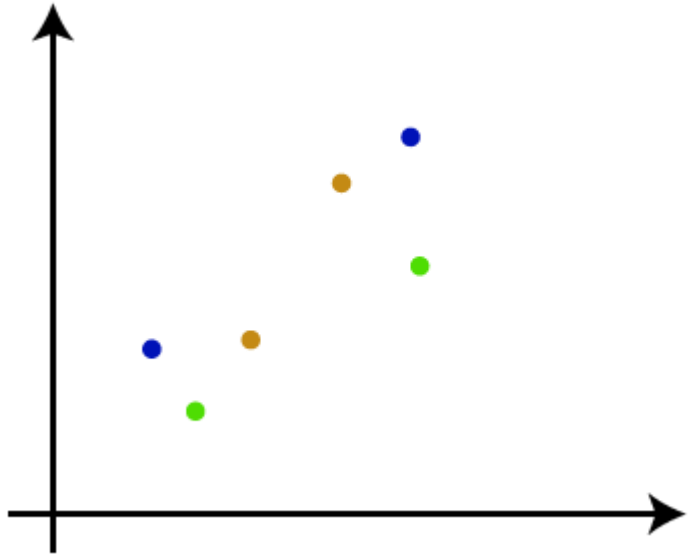
AGGLOMERATIVE HIERARCHICAL CLUSTERING

- To group the datasets into clusters, it follows the **bottom-up approach**.
- It means, this algorithm considers each dataset as a single cluster at the beginning, and then start combining the closest pair of clusters together.
- It does this until all the clusters are merged into a single cluster that contains all the datasets.
- This hierarchy of clusters is represented in the form of the dendrogram.

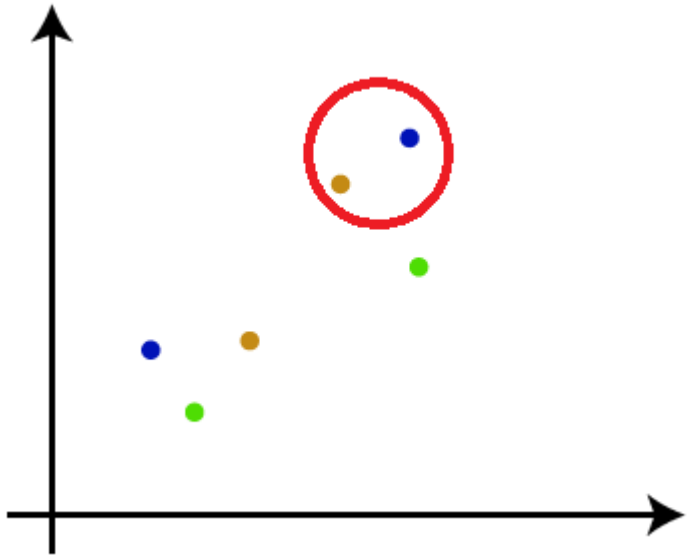
HOW THE AGGLOMERATIVE HIERARCHICAL CLUSTERING WORK?

The working of the AHC algorithm can be explained using the below steps:

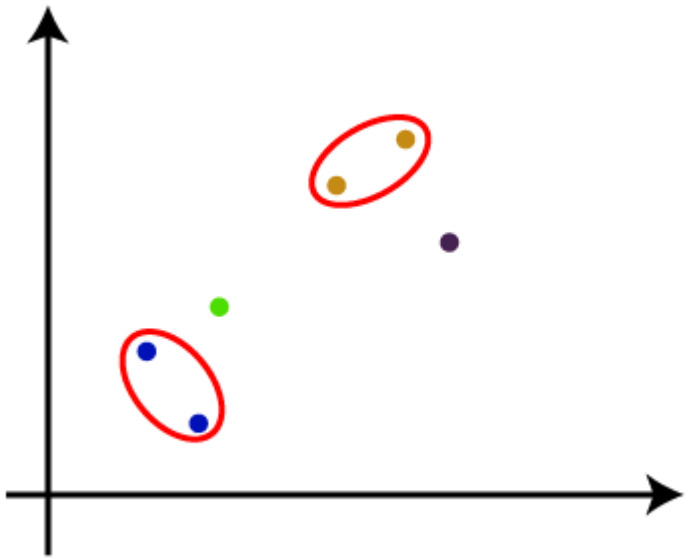
- **Step-1:** Create each data point as a single cluster. Let's say there are N data points, so the number of clusters will also be N.



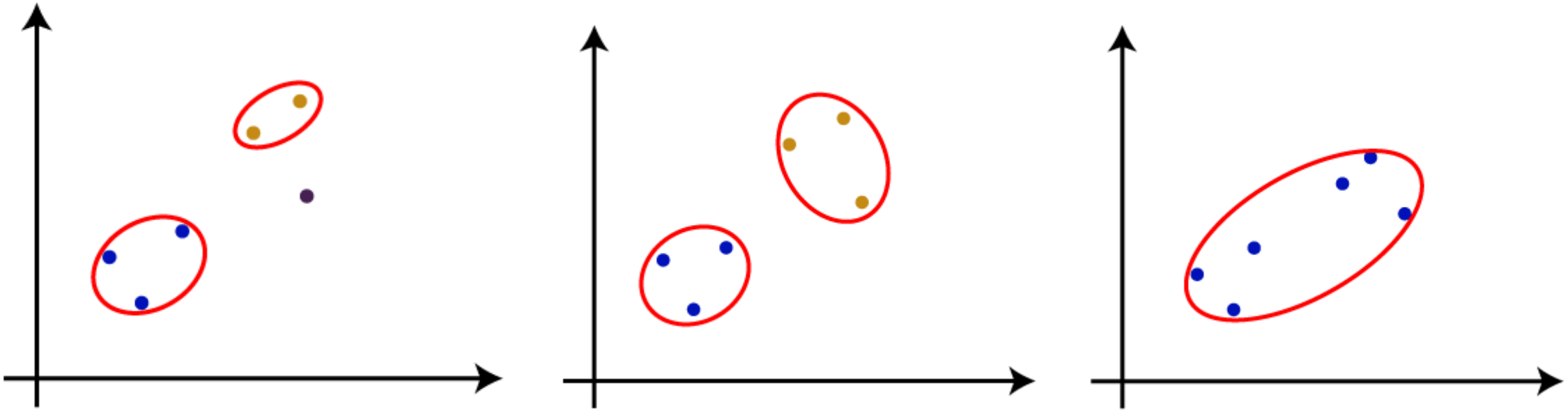
Step-2: Take two closest data points or clusters and merge them to form one cluster. So, there will now be N-1 clusters.



Step-3: Again, take the two closest clusters and merge them together to form one cluster. There will be N-2 cluster



Step-4: Repeat Step 3 until only one cluster left. So, we will get the following clusters. Consider the below images:

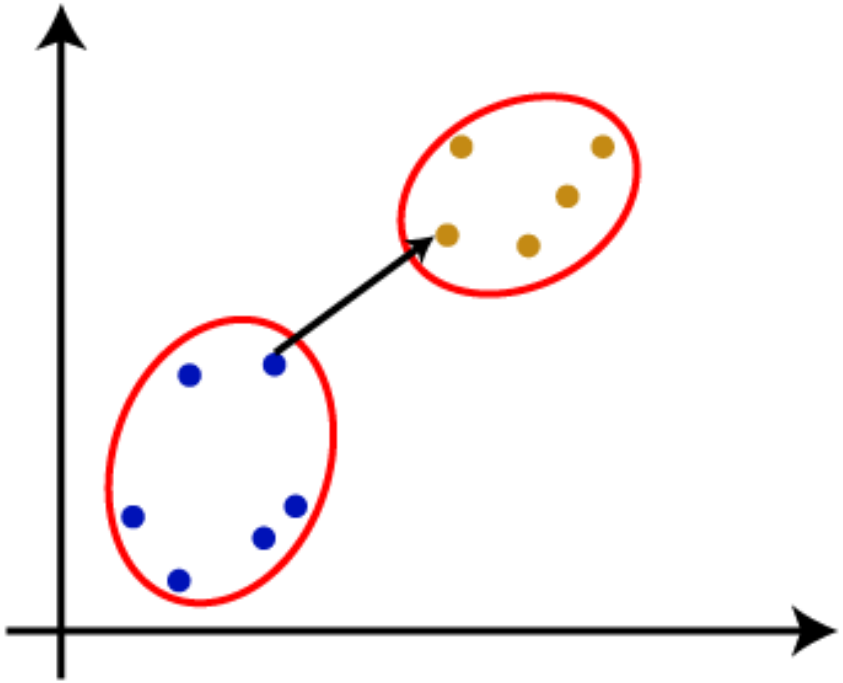


Step-5: Once all the clusters are combined into one big cluster, develop the dendrogram to divide the clusters as per the problem.

MEASURE FOR THE DISTANCE BETWEEN TWO CLUSTERS

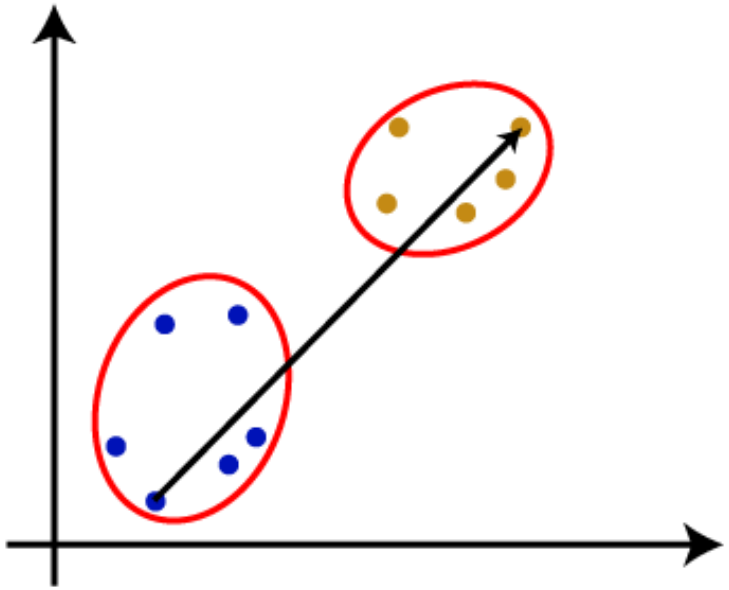
- The **closest distance** between the two clusters is crucial for the hierarchical clustering.
- There are various ways to calculate the distance between two clusters, and these ways decide the rule for clustering.
- These measures are called **Linkage methods**.
- Some of the popular linkage methods are given below:
  - **Single Linkage**
  - **Complete Linkage**
  - **Average Linkage**
  - **Centroid Linkage**

1. **Single Linkage:** It is the Shortest Distance between the closest points of the clusters



2) Complete Linkage: It is the farthest distance between the two points of two different clusters.

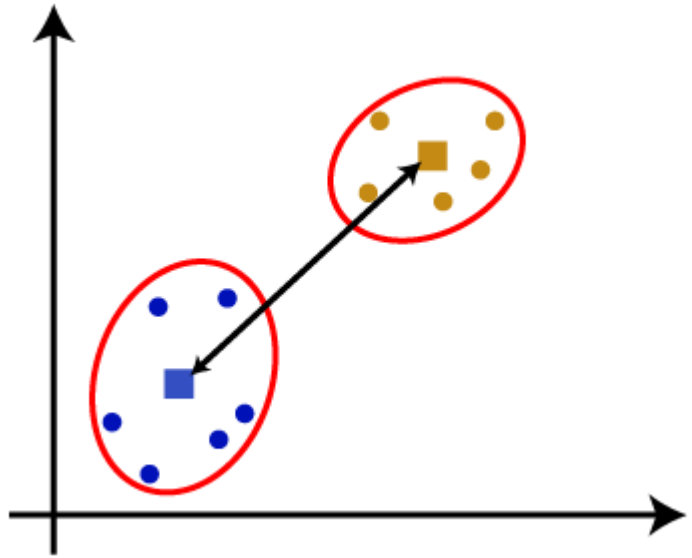
It is one of the popular linkage methods as it forms tighter clusters than single-linkage.



3) Average Linkage:

It is the linkage method in which the distance between each pair of datasets is added up and then divided by the total number of datasets to calculate the average distance between two clusters.

4) Centroid Linkage: It is the linkage method in which the distance between the centroid of the clusters is calculated. Consider the below image:



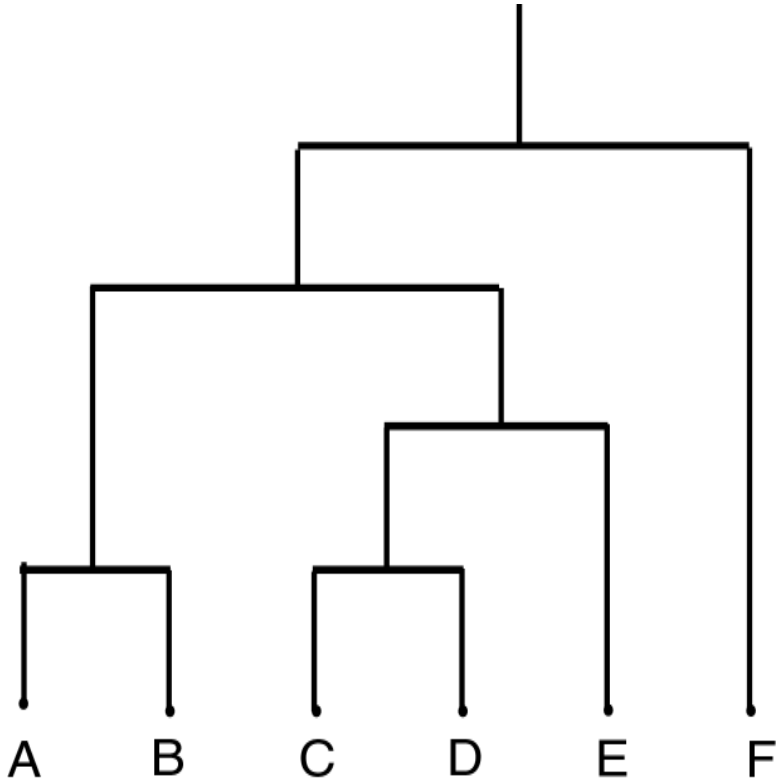
WORKING OF DENDROGRAM IN HIERARCHICAL CLUSTERING

- The dendrogram is a tree-like structure that is mainly used to store each step as a memory that the HC algorithm performs.
- In the dendrogram plot

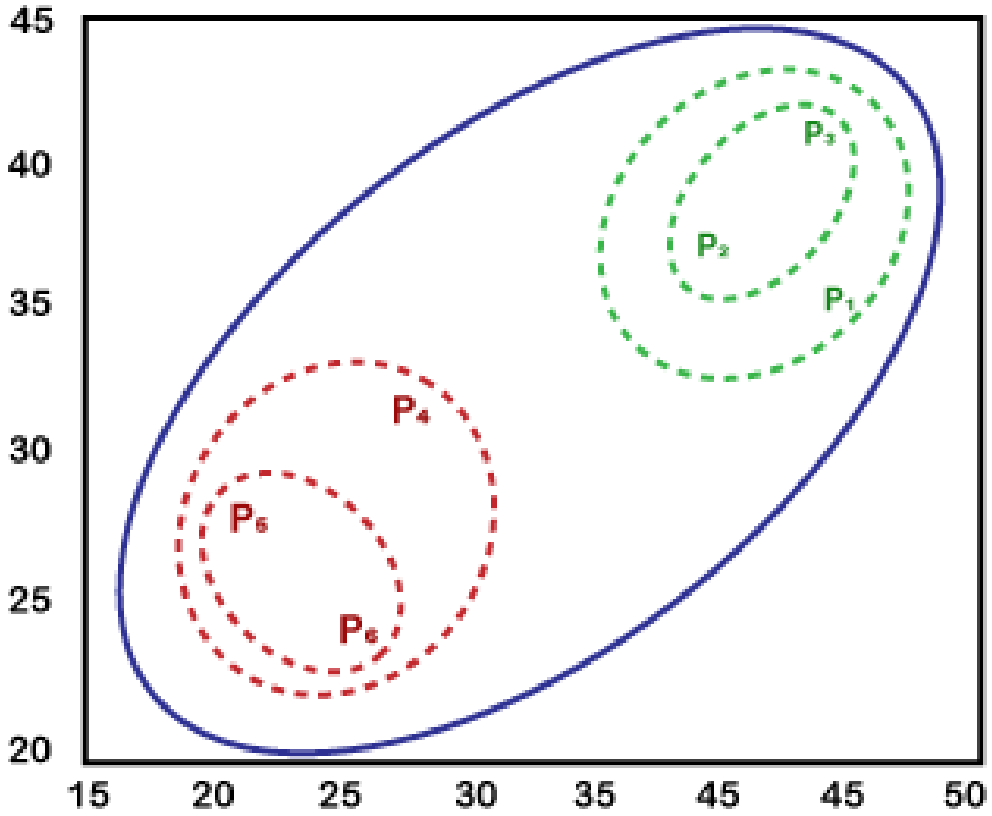
  the Y-axis shows the Euclidean distances between the data points, and the x-axis shows all the data points of the given dataset.

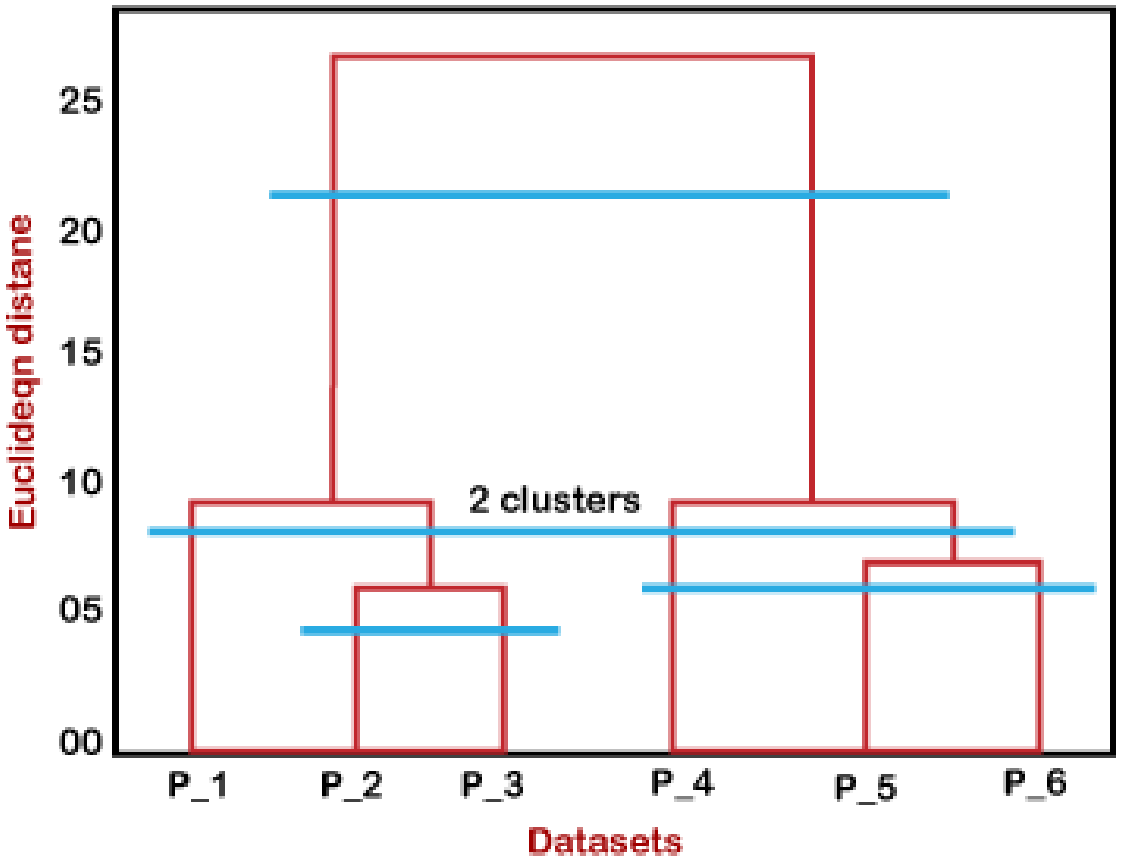
DENDROGRAM

- ***Dendrogram:*** a tree data structure which illustrates hierarchical clustering techniques.
- Each level shows clusters for that level.
  - Leaf – individual clusters
  - Root – one cluster
- A cluster at level i is the union of its children clusters at level i+1.



working of the dendrogram can be explained using the below diagram:

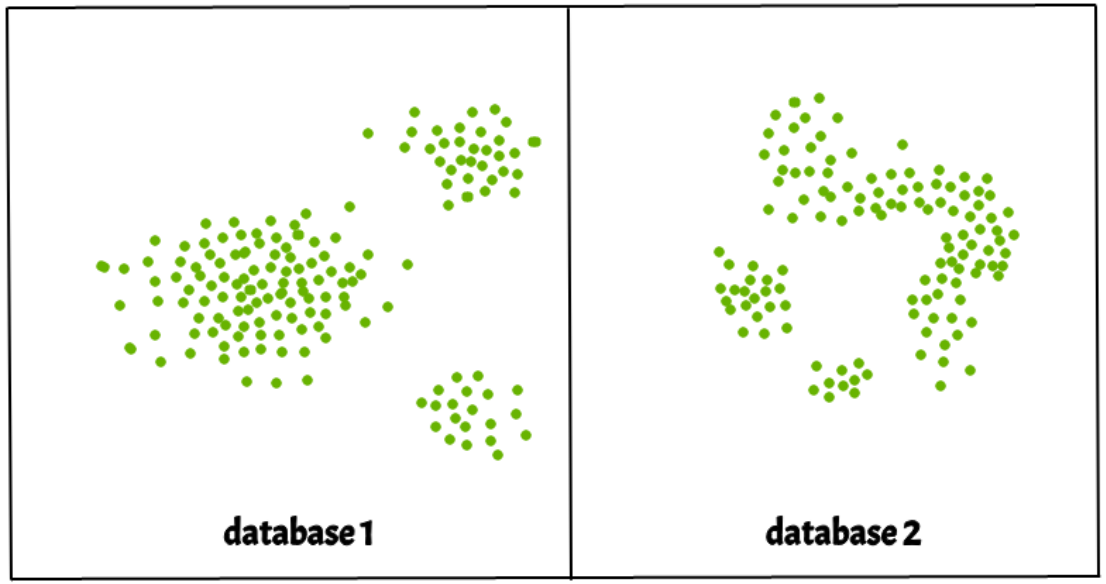




In the above diagram, the left part is showing how clusters are created in agglomerative clustering, and the right part is showing the corresponding dendrogram.

DBSCAN (DENSITY-BASED SPATIAL CLUSTERING OF APPLICATIONS WITH NOISE) ALGORITHM

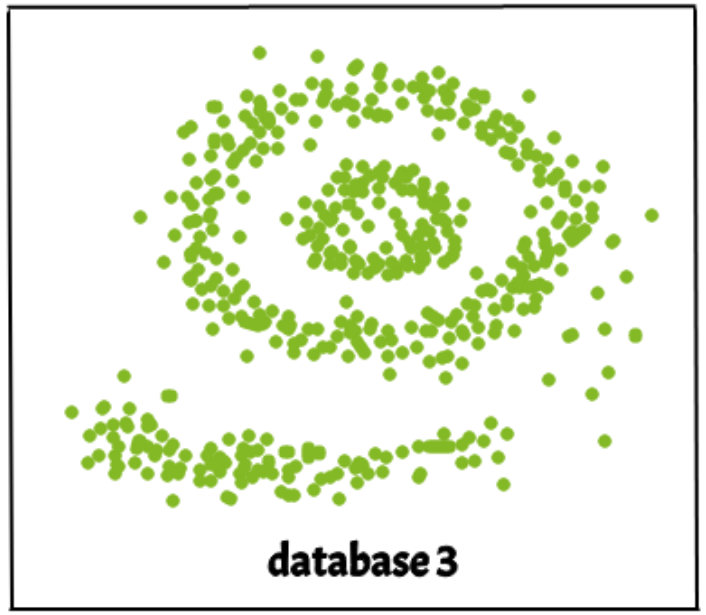
- Clusters are dense regions in the data space, separated by regions of the lower density of points.
- The ***DBSCAN algorithm*** is based on this intuitive notion of “clusters” and “noise”.
- The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.



Why DBSCAN ?

- Partitioning methods (K-means, PAM clustering) and hierarchical clustering work for finding spherical-shaped clusters or convex clusters.
- In other words, they are suitable only for compact and well-separated clusters.
- they are also severely affected by the presence of noise and outliers in the data.
- Real life data may contain irregularities, like –

i) Clusters can be of arbitrary shape
ii) Data may contain noise.



DBSCAN: DENSITY-BASED SPATIAL CLUSTERING OF APPLICATIONS WITH NOISE

"How can we find dense regions in density-based clustering?"

The density of an object O can be measured by the number of objects close to O.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) finds core objects, i.e. objects that have dense neighborhoods.

It connects core objects and their neighborhoods to form dense regions as clusters

"How does DBSCAN quantify the neighborhood of an object?"

A user-specified parameter epsilon ε- > 0 is used to specify the radius of a neighborhood we consider for every object.

Two hyperparameters: *epsilon* and *minPoints* to arrive at clusters.

1) Eps: Maximum radius of the neighborhood we consider for every object.

The ε- neighborhood of an object o is the space within a radius centered at o.

2) The density of a neighborhood can be measured simply by the number of objects in the neighborhood.

To determine whether a neighborhood is dense or not, DBSCAN uses another user-specify parameter, MinPts,

- *MinPts*: which specifies the density threshold of dense regions.
- An object is a core object if the ε - neighborhood of the object contains at least MinPts objects. Core objects are the pillars of dense regions

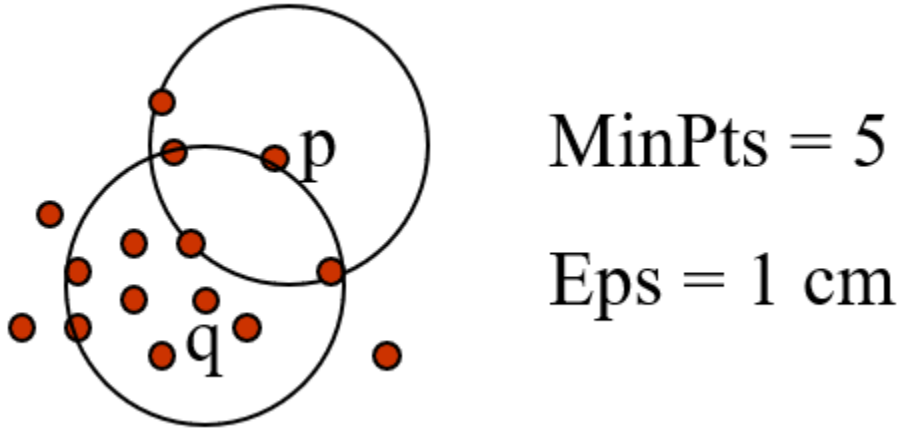
- $N_{Eps}(p)$: {q belongs to D | dist(p,q) ≤ Eps}

DENSITY-BASED CLUSTERING: BASIC CONCEPTS

Given a set, D, of objects, we can identify all core objects with respect to the given parameters, ε- and MinPts.

The clustering task is therein reduced to using core objects and their neighborhoods to form dense regions, where the dense regions are clusters.

- Directly density-reachable: A point *p* is directly density-reachable from a core point *q* w.r.t. *Eps*, *MinPts* if
  - *p* belongs to $N_{Eps}(q)$
  - core point condition: $|N_{Eps}(q)| \geq MinPts$



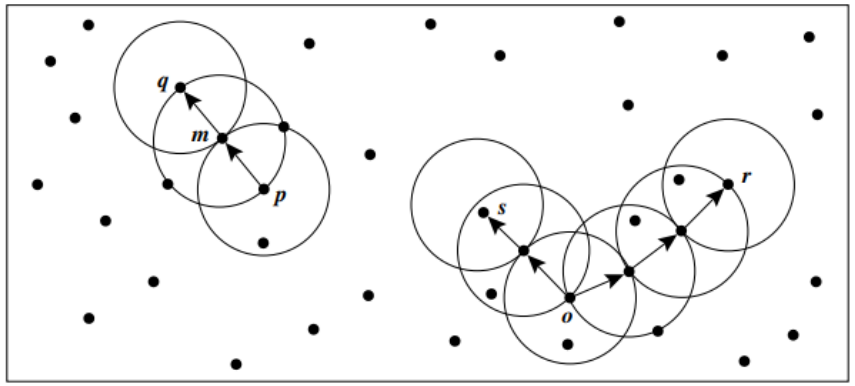
How can we assemble a large dense region using small dense regions centered by core objects?”

Density-reachable:

A point *p* is density-reachable from a point *q* w.r.t. *Eps*, *MinPts* if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i

Density-connected

A point *p* is density-connected to a point *q* w.r.t. *Eps*, *MinPts* if there is a point *o* such that both, *r* and *s* are density-reachable from *o* w.r.t. *Eps* and *MinPts*



Algorithm: DBSCAN: a density-based clustering algorithm.

Input:

- D: a data set containing n objects,
- ϵ: the radius parameter, and
- *MinPts*: the neighborhood density threshold.

Output: A set of density-based clusters.

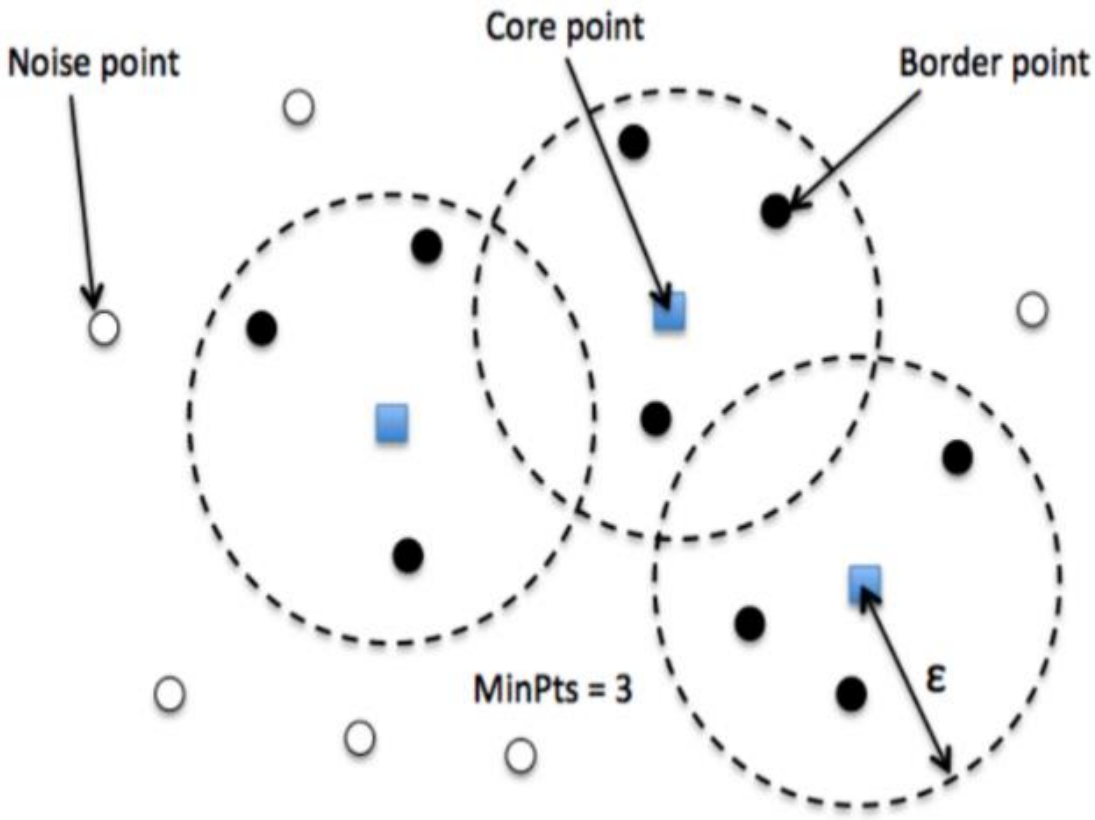
Method:

```
(1)  mark all objects as unvisited;
(2)  do
(3)      randomly select an unvisited object p;
(4)      mark p as visited;
(5)      if the ε-neighborhood of p has at least MinPts objects
(6)          create a new cluster C, and add p to C;
(7)          let N be the set of objects in the ε-neighborhood of p;
(8)          for each point p' in N
(9)              if p' is unvisited
(10)                 mark p' as visited;
(11)                 if the ε-neighborhood of p' has at least MinPts points,
                     add those points to N;
(12)             if p' is not yet a member of any cluster, add p' to C;
(13)         end for
(14)         output C;
(15)     else mark p as noise;
(16) until no object is unvisited;
```

DBSCAN algorithm.

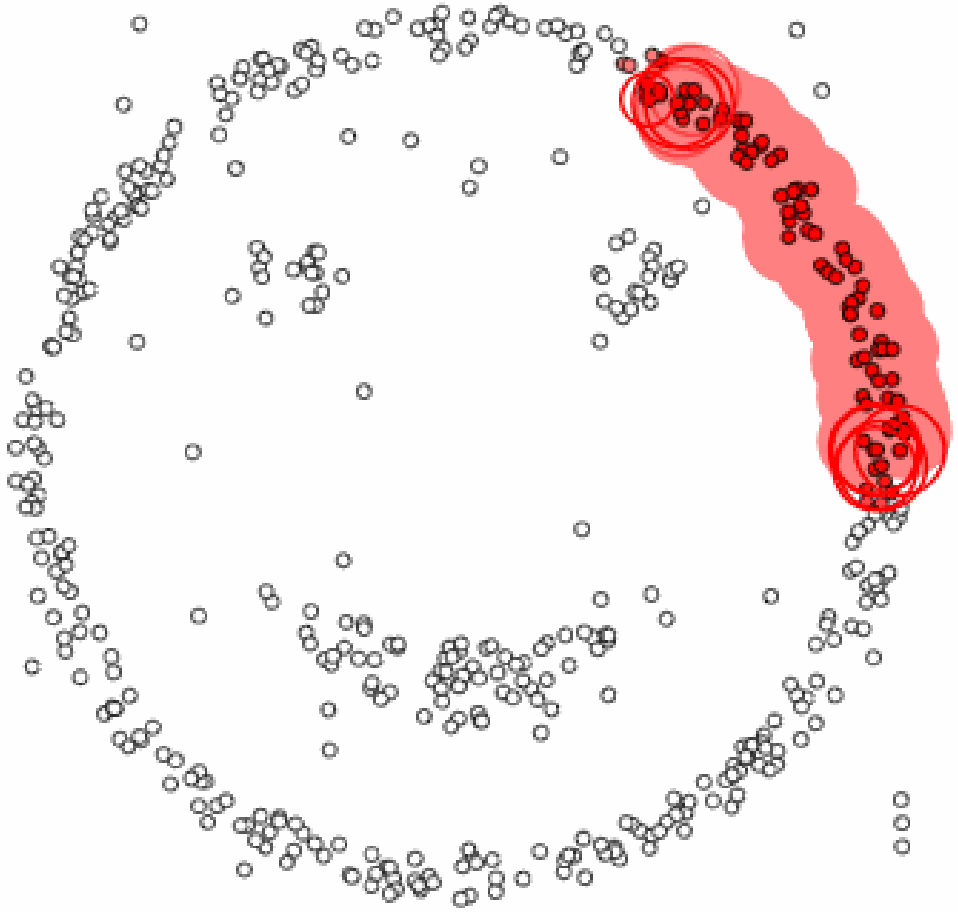
There are three types of points after the DBSCAN clustering is complete:

- **Core** — This is a point that has at least *m* points within distance *n* from itself.
- **Border** — This is a point that has at least one Core point at a distance *n*.
- **Noise** — This is a point that is neither a Core nor a Border. And it has less than *m* points within distance *n* from itself.



HOW DOES THE DBSCAN ALGORITHM CREATE CLUSTERS?

- Algorithms start by picking a point(one record) **x** from your dataset at random and assign it to a cluster 1.
- Then it counts how many points are located within the ε (epsilon) distance from **x**.
- If this quantity is greater than or equal to minPoints (n), then considers it as core point, then it will pull out all these ε-neighbours to the same cluster 1.
- It will then examine each member of cluster 1 and find their respective ε -neighbours.
- If some member of cluster 1 has *n* or more ε-neighbours, it will expand cluster 1 by putting those ε-neighbours to the cluster.
- It will continue expanding cluster 1 until there are no more examples to put in it.


epsilon = 1.00
minPoints = 4
Restart
Pause

DBSCAN Parameter Selection

DBSCAN is very sensitive to the values of *epsilon* and *minPoints*. Therefore, it is important to understand how to select the values of *epsilon* and *minPoints*.

- **minPoints(n):**

As a starting point, a minimum n can be derived from the number of dimensions D in the data set, as $n \geq D + 1$.

- **Epsilon(ε):**

If a small epsilon is chosen, a large part of the data will not be clustered. Whereas, for a too high value of ε, clusters will merge and the majority of objects will be in the same cluster.

Hence, the **value for ε can then be chosen by using a k-graph**, plotting the distance to the k = minPoints-1 nearest neighbour ordered from the largest to the smallest value.

- **Distance Function:**

By default, DBSCAN uses Euclidean distance, although other methods can also be used (like great circle distance for geographical data).

DBSCAN Vs K-means Clustering

K-means Clustering	DBSCAN
Distance based clustering	Density based clustering
Every observation becomes a part of some cluster eventually	Clearly separates outliers and clusters observations in high density areas
Build clusters that have a shape of a hypersphere	Build clusters that have an arbitrary shape or clusters within clusters.
Sensitive to outliers	Robust to outliers
Require no. of clusters as input	Doesn't require no. of clusters as input