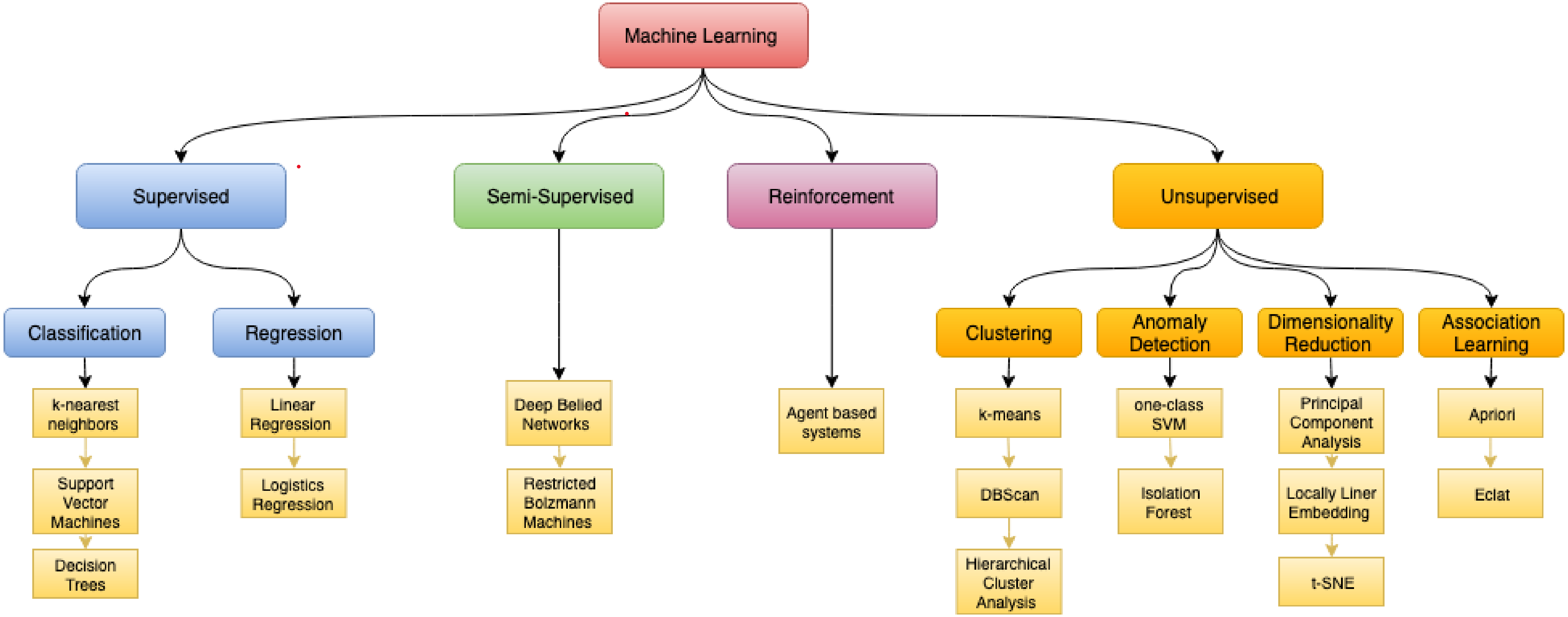
Machine Learning Short Laboratory Course

04/03/2022

About me

- Second year PhD student
- MSc in Data science at Sapienza University of Rome
- Interest in NLP and SNA
- Reach me by email : abbonato@unistra.fr
- Material : MLAdventure/ML_Short_Lab

Types of Machine Learning



Unsupervised learning

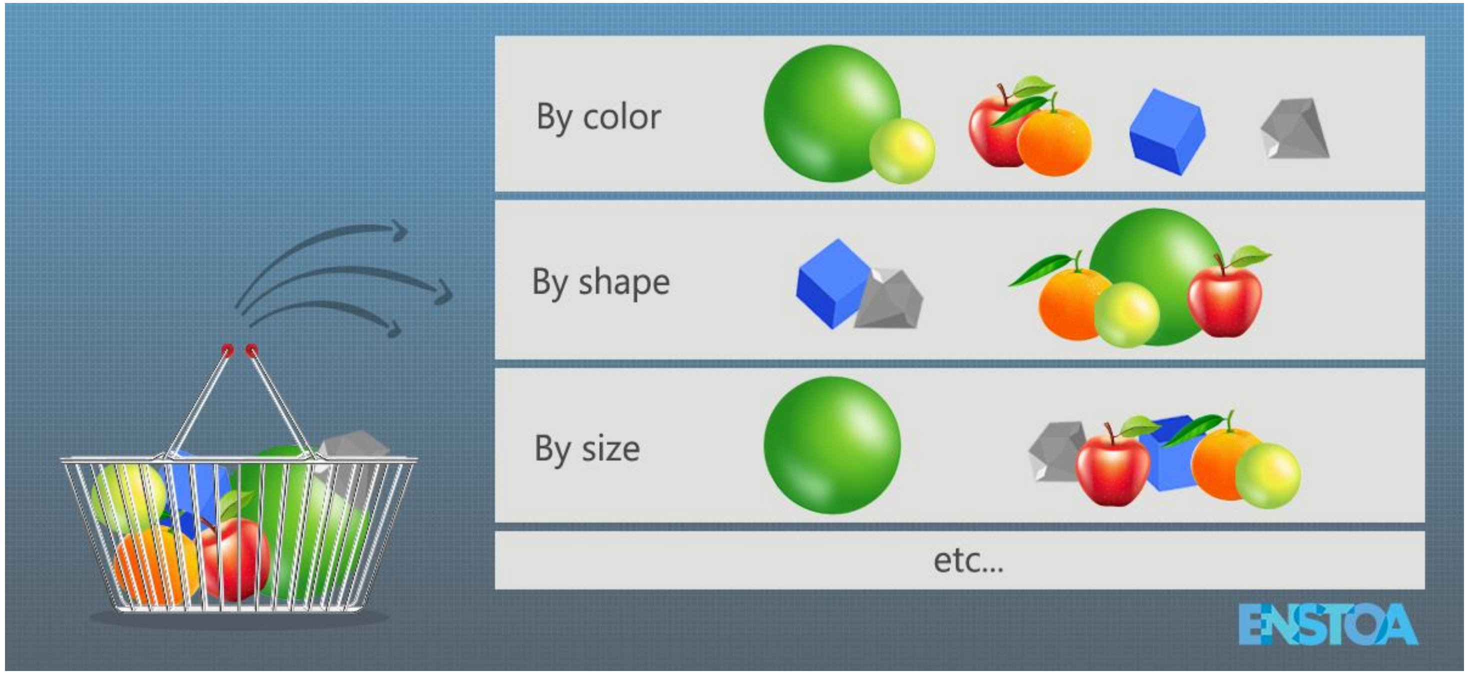
- Work with unlabeled data

- Main task :
  - Clustering
  - Dimensionality Reduction

- Applications :
  - News Section
  - Computer Vision
  - Medical imaging
  - Anomaly detection
  - Profiling

Clustering

- Goal: Discover groups



Clustering

- How: using different types of algorithms based on our task

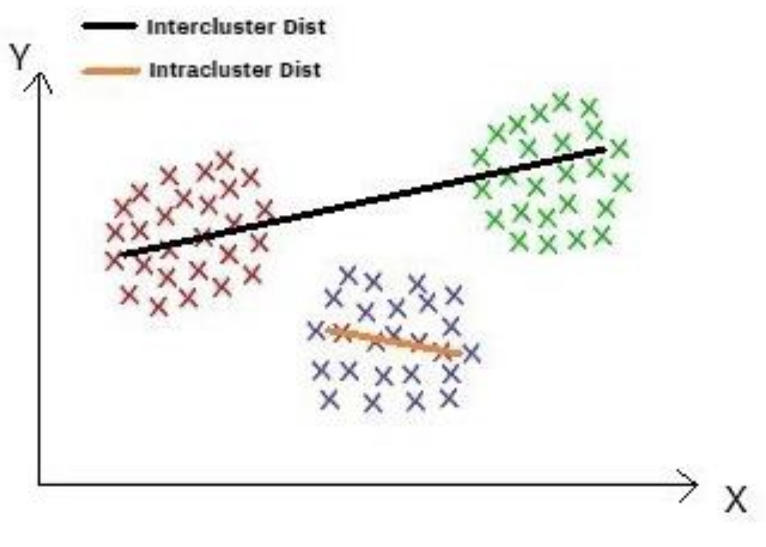
- Most Famous:
  - K – Means
  - DBScan
  - Hierarchical Cluster Analysis

Clustering: K-means

The K-means problem:

- Consider a set X = {x1,…xn} of n points in R^d
- Assume that the number k is given
- Problem:
    - Find k points c1,…ck (named centers or means) so that the cost is minimized :

$$C_1, C_2, \cdots, C_k = argmin \sum_{i=1}^{k} \sum_{x \epsilon S_i} ||x - C_i||^2$$

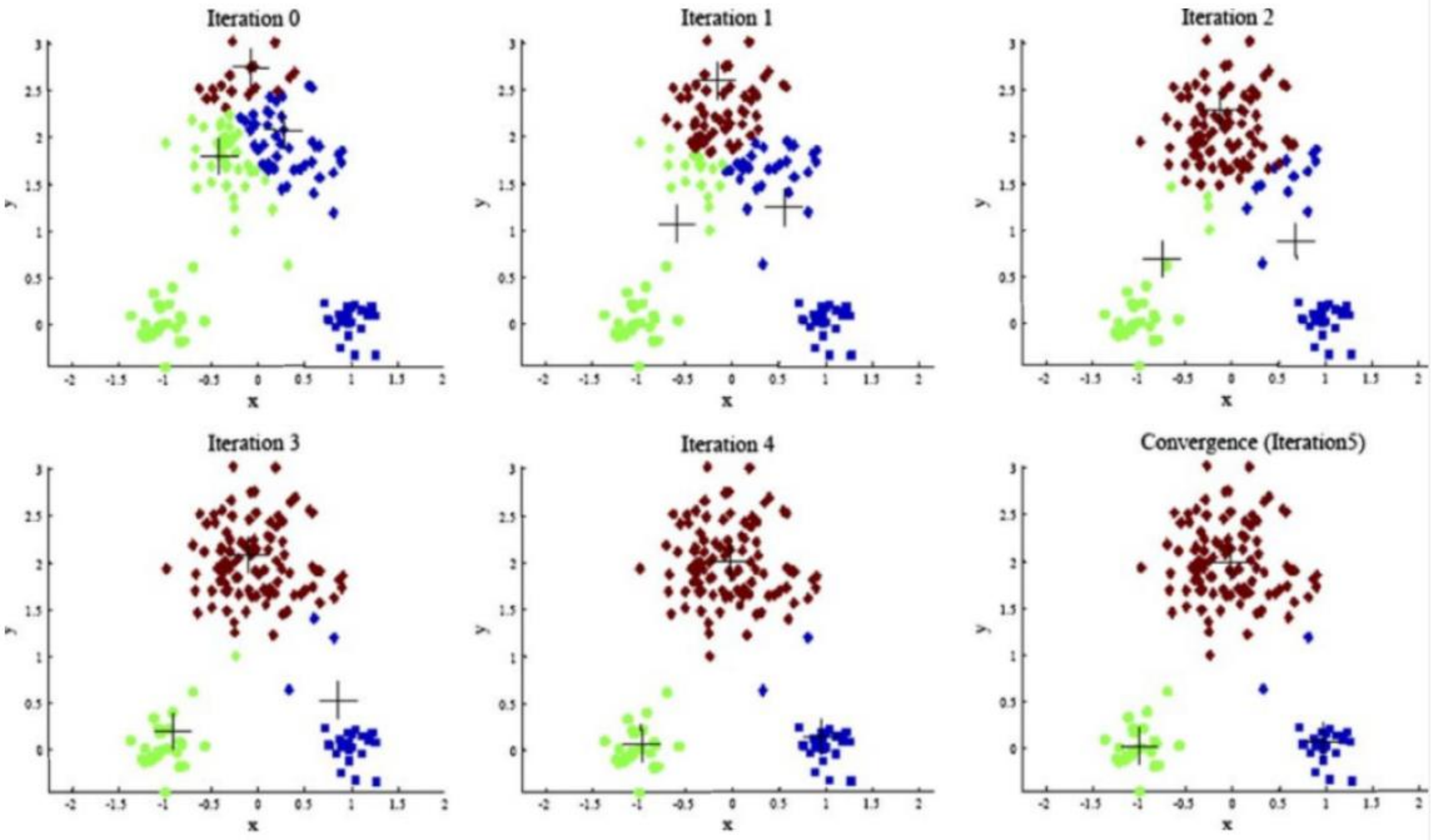


Clustering: K-means

- Algorithm:
  1. Cluster the data into k groups where k is predefined
  2. Select k points at random as cluster centers
  3. Assign objects to their closest cluster center according to the Euclidean distance function
  4. Calculate the centroid or mean of all objects in each cluster
  5. Repeat steps 2,3 and 4 until the same points are assigned to each cluster in consecutive rounds.

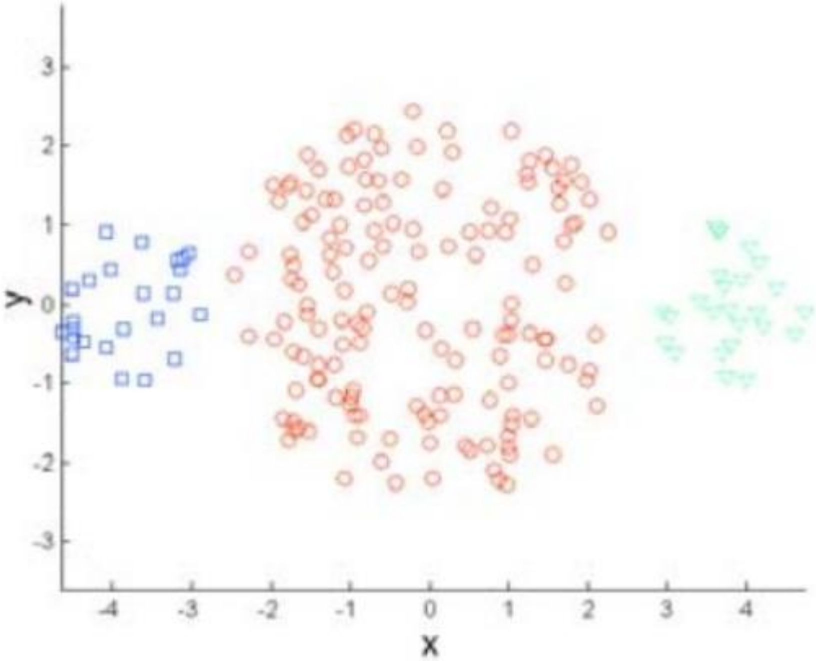
Clustering: K-means

- K = 1 and K = n are easy special case… why?
- K – means is a NP-hard problem if the dimension of the data is at least 2 (d>=2)
- Keep attention on initialization..

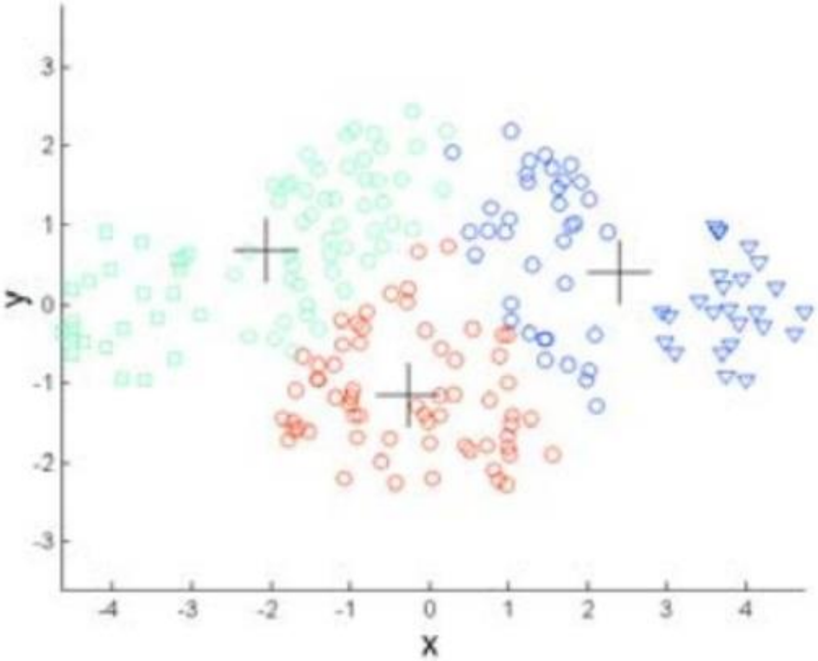


Clustering: K-means Limitation

- Different Size



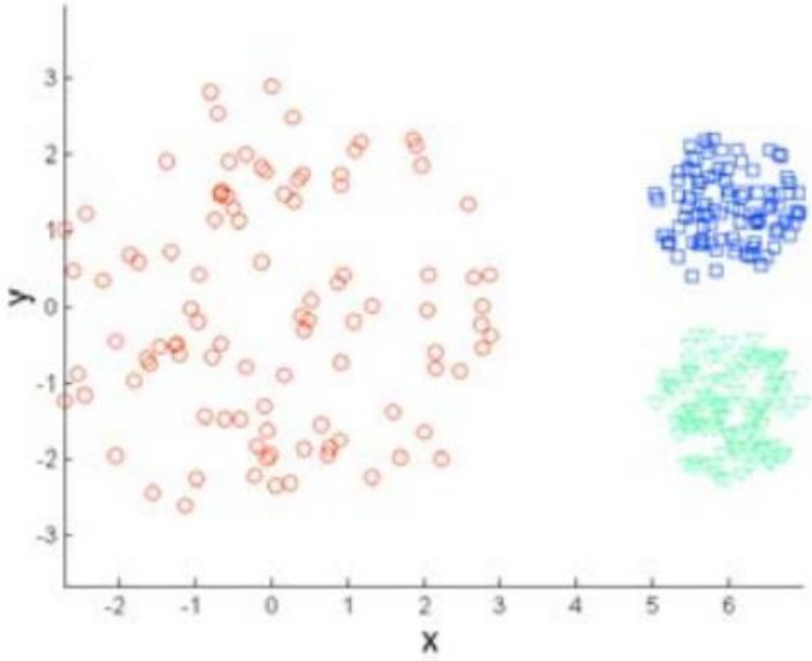
Original Points



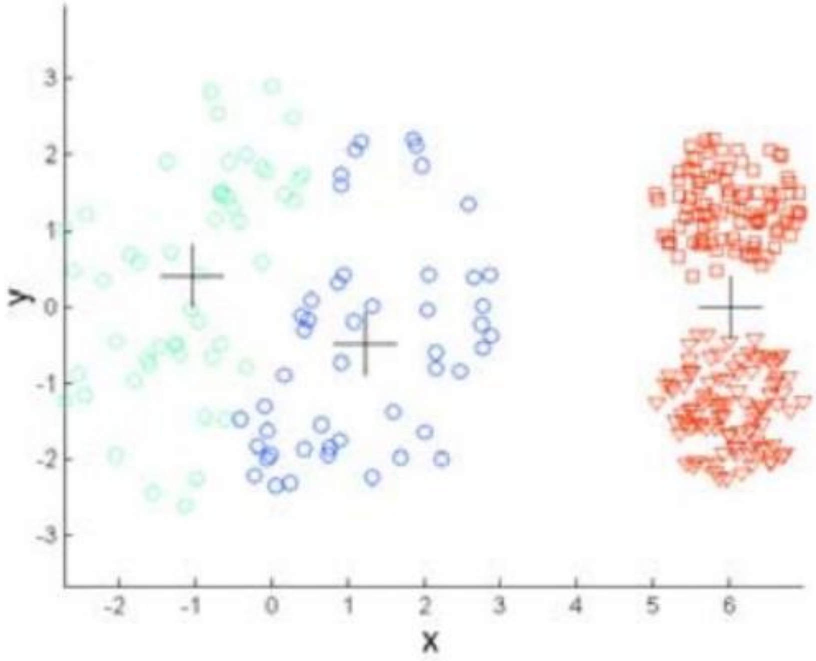
K-means (3 Clusters)

Clustering: K-means Limitation

- Different Density



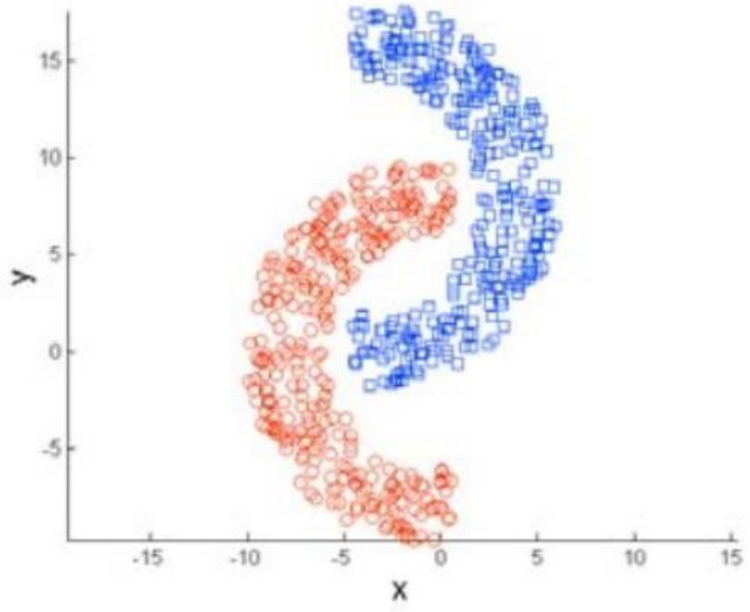
Original Points



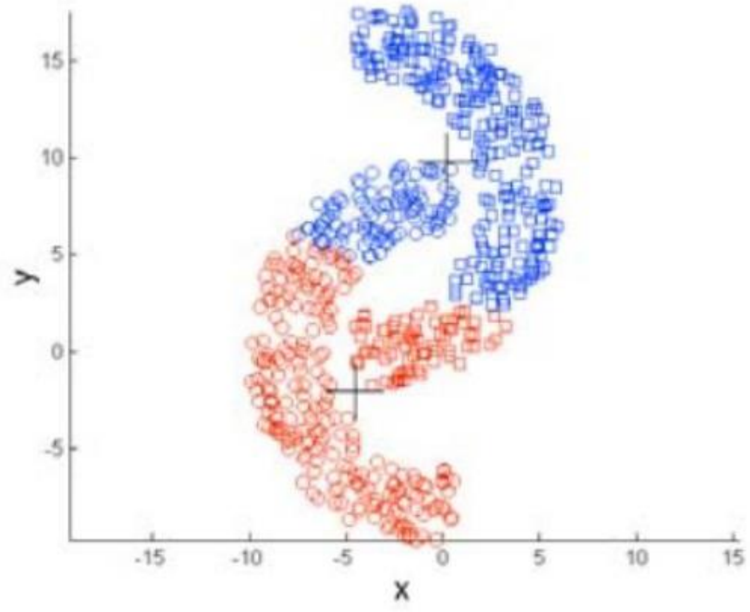
K-means (3 Clusters)

Clustering: K-means Limitation

- Non-Spherical shapes



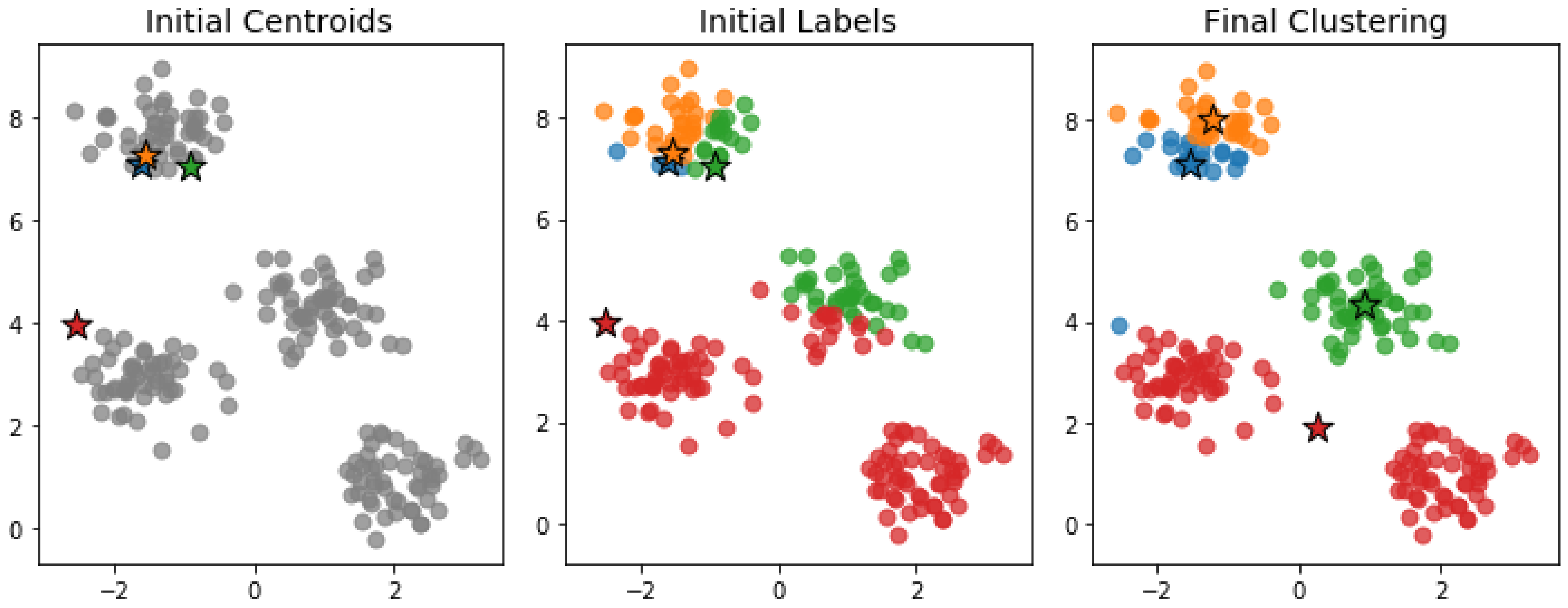
Original Points



K-means (2 Clusters)

Clustering: K-means Limitation

- Effects of bad initialization



Original points **K-means (4 cluster)**

Clustering: K-means ++

- Identical to k-means except for initialization

How:

- Pick the first centroid point (C_1) randomly.
- Compute distance of all points in the data from the selected centroid.

$$d_i = max_{(j:1 \mapsto m)} ||x_i - C_j||^2$$

- Repeat till you find k-centroids

Clustering: K-Medoids (PAM)

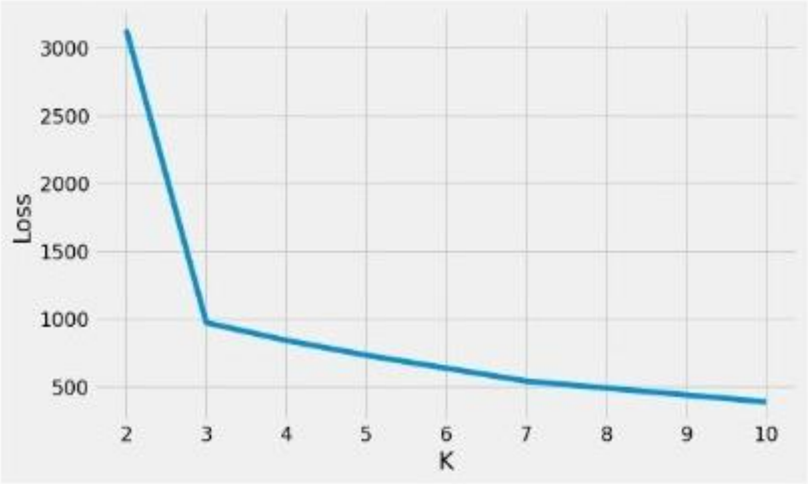
- Idea : make the final centroids as actual data-points
- How:
  - Initialization : Same as K-means++
  - Assignment: Same as K-means
  - Update centroids: If there are m-point in a cluster, swap the previous centroid with all other (m-1) points from the cluster and finalize the point as a new centroid that has a minimum loss.

$$M_1, M_2, \cdots, M_k = argmin \sum_{i=1}^{k} \sum_{x \epsilon S_i} ||x - M_i||^2$$

  - Repeat: Same as that of K-Means

Clustering: Best K value

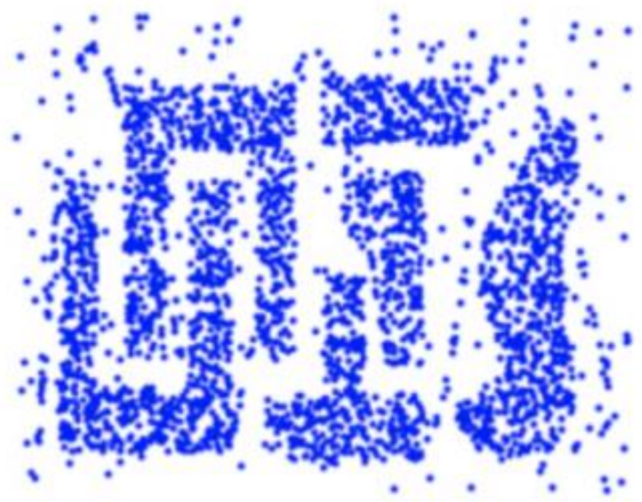
- To determine the right K, draw a plot between loss vs K using Elbow method



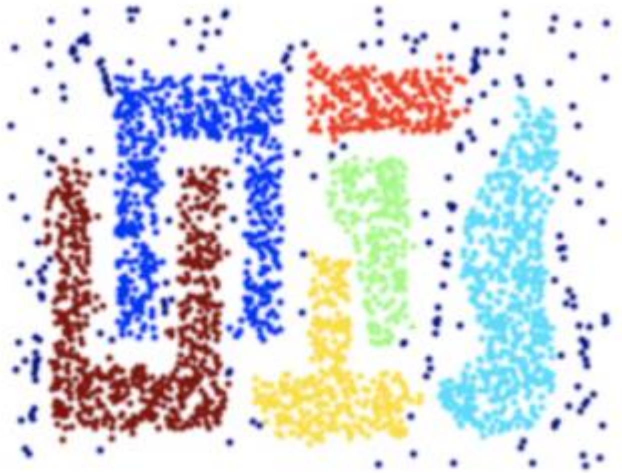
Optimal number of clusters is 3

Clustering: DBScan

- Density-Based Spatial Cluster of Application with Noise
- Aim : separate clusters of high density from clusters of low density



Original Points

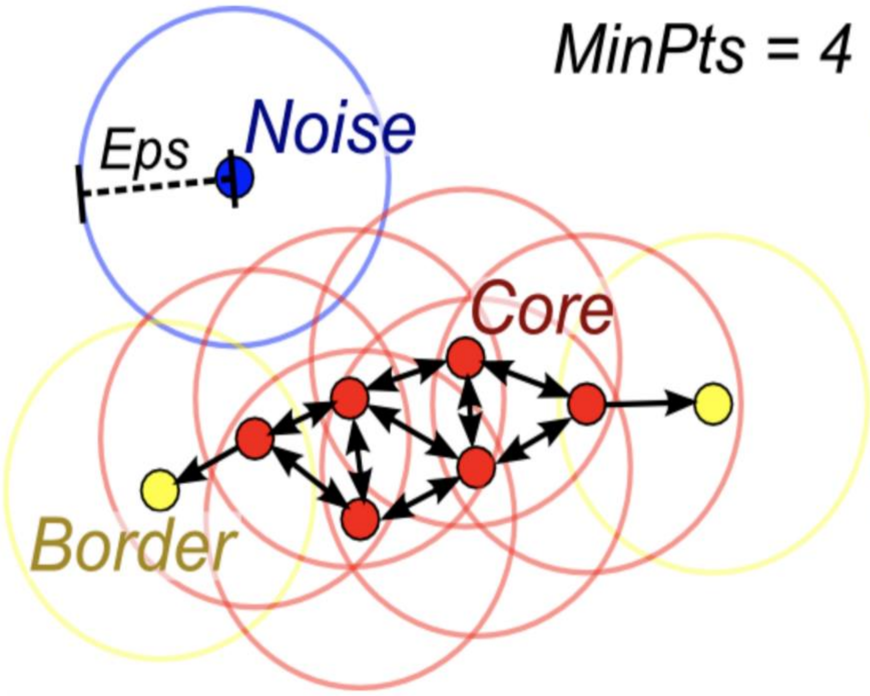


DBScan (6 cluster)

Clustering: DBScan

- How :
  - Divides the dataset into n dimension
  - For each point in the dataset, the algorithm forms an n dimensional shape around that data point, and then counts how many data points fall within that shape
  - The shape will count as a cluster
  - DBScan iteratively expands the cluster by going through each individual point within the cluster, and counting the number of other data points nearby

Clustering: DBScan



Red: Core Points

Yellow: Border points. Still part of the cluster because it's within epsilon of a core point, but not does not meet the min_points criteria

Blue: Noise point. Not assigned to a cluster

If you want to know more [here]()

Advantages :

- It is great at separating clusters of high density versus clusters of low density within a given dataset.
- It is great with handling outliers within the dataset.

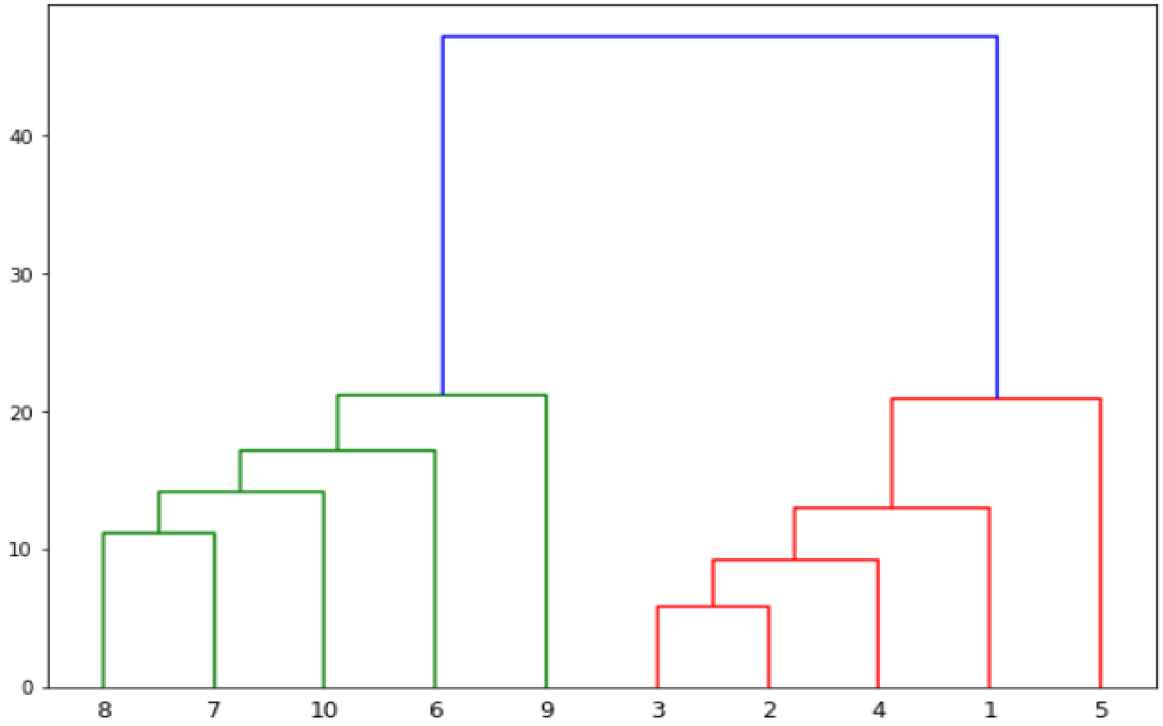
Disadvantages:

- Issues with clusters of similar density
- Issues with high dimensionality data

Clustering: Hierarchical Clustering

- Like K-means, groups togheter the data points with similar characteristics

- Two types of algorithm:
- Agglomerative : bottom-up approach
- Divisive : top-down approach



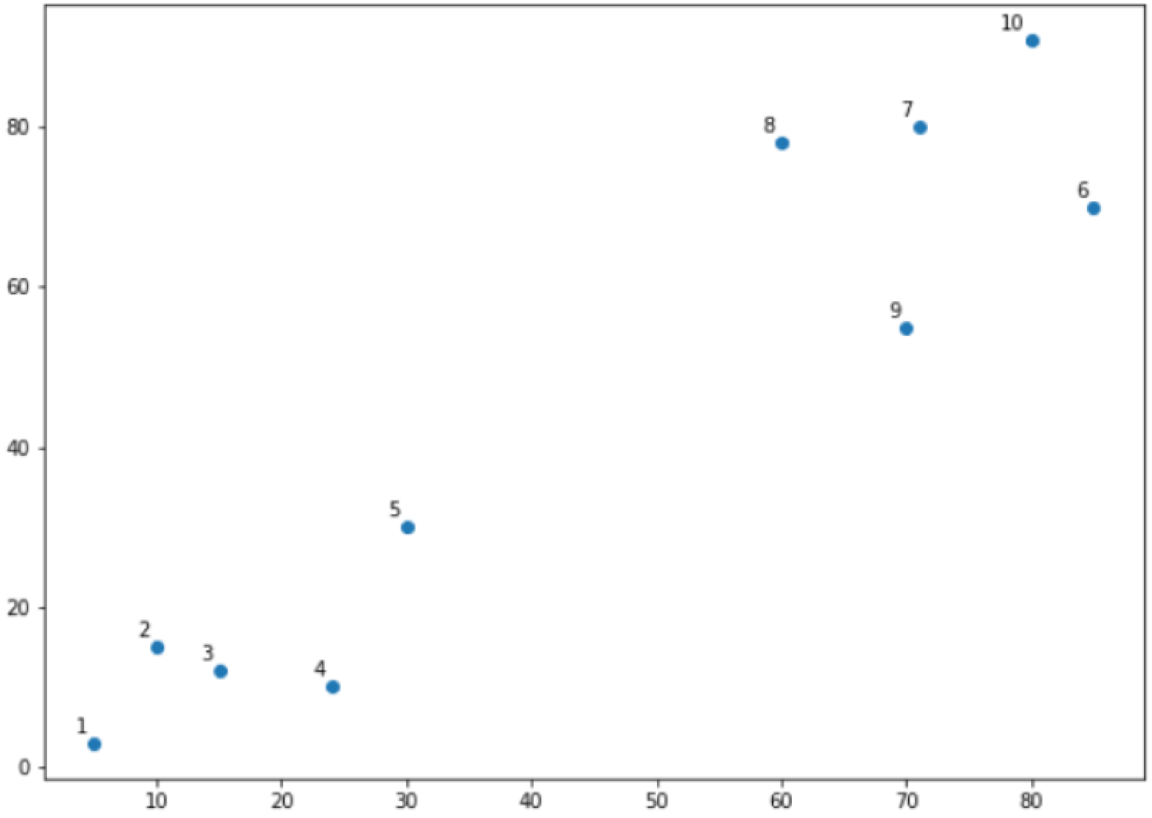
Dendogram Agglomerative Clustering

Clustering: Hierarchical Clustering

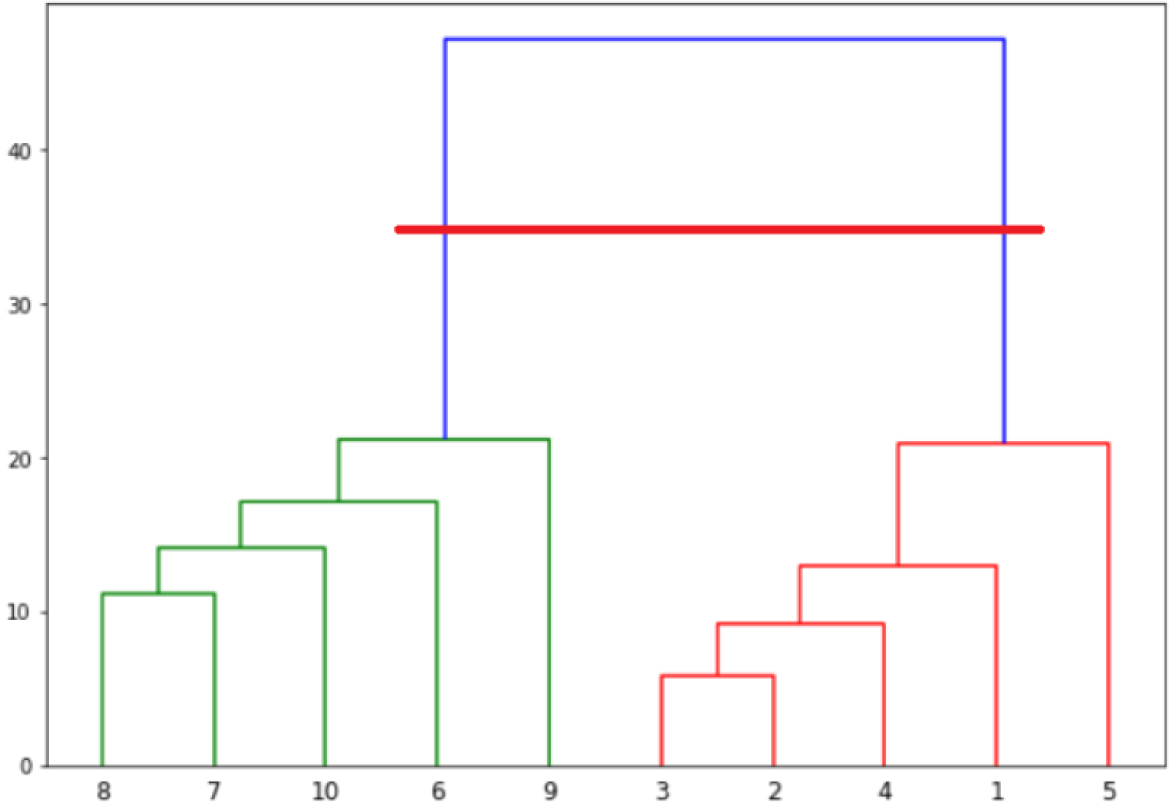
- Agglomerative
- How:

1. At the start, treat each data point as one cluster. So we will have a number of cluster equal to K
2. Form a cluster by joining the two closest data points resulting in K-1 clusters.
3. Form more clusters by joining the two closest clusters resulting in K-2 clusters.
4. Repeat the above three steps until one big cluster is formed.

Clustering: Hierarchical Clustering



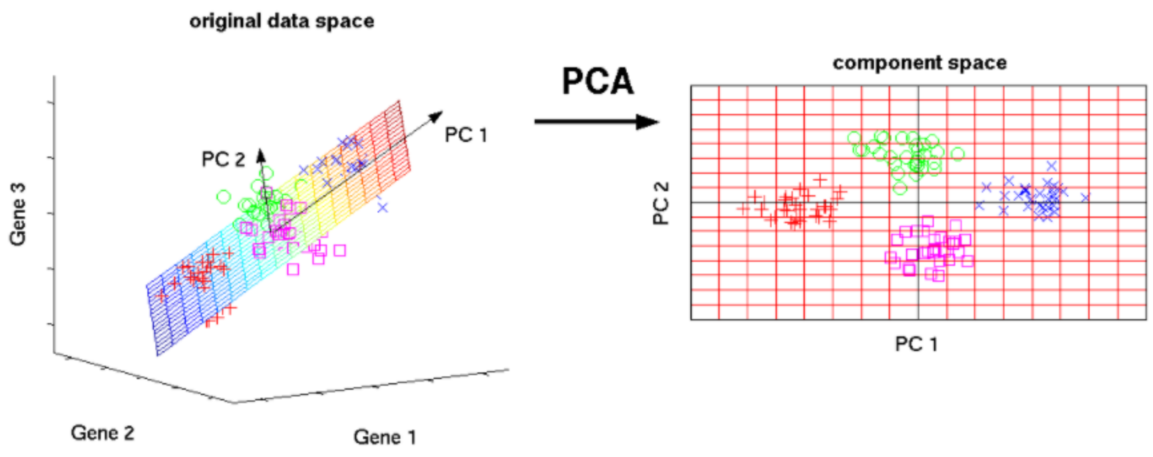
Original Points

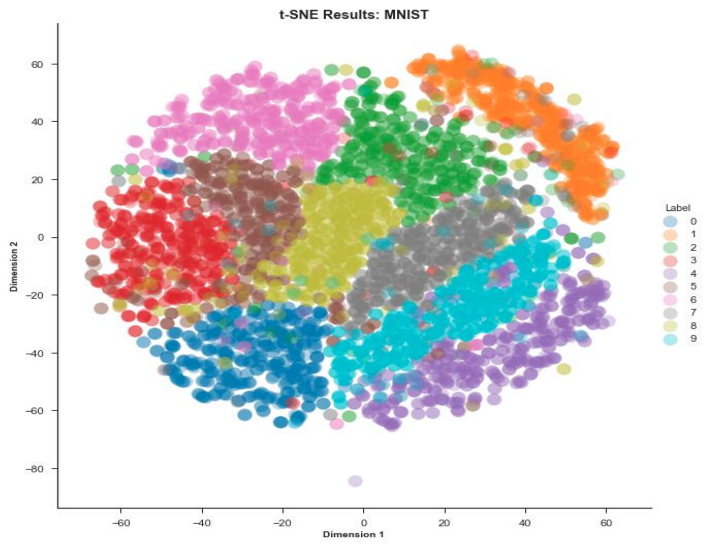


Bottom-up approach with Single link cluster distance

Dimensionality Reduction

- Goal: Reduce the number of variable in our data
- Types of algorithm:
    - Principal Component Analysis (PCA)
    - t-distributed stochastic neighbor embedding (t-SNE)





Dimensionality Reduction : PCA

- Aim: reduce the dimensionality of large datasets with creating new components that will preserve as much information as possible
- Obs: your data should be numeric
- How:
    1. Standardize your data
    2. Compute the covariance matrix to identify correlations
    3. Compute the eigenvector and eigenvalues of the covariance matrix to identify the principal components
    4. Create a feature vector with the components
    5. Based on variance decide how many components to use

Dimensionality Reduction: t-SNE

- Aim: find non-linear connections in the data in order to have a dimensionality reduction
- How:
    1. Calculating a joint probability distribution that represents the similarities between the data points
    2. Creating a dataset of points in the target dimension and then calculating the joint probability distribution for them as well
    3. Using gradient descent to change the dataset in the low-dimensional space so that the joint probability distribution representing it would be as similar as possible to the one in the high dimension

If you want to know more here