Clustering Techniques

Version 2 (2023)

Cluster Analysis

- Unsupervised(unlabeled data) machine learning technique
- Aims to find patterns(e.g., many sub-groups, size of each group, common characteristics, data cohesion…) while gathering data samples
- Group them into similar records using predefined distance measures like the Euclidean distance and such

Usage

- can be considered the initial step when dealing with a new dataset
  - to extract insights and understand the data distribution.
- can also be used to perform dimensionality reduction (e.g. encoding)
- might also serve as a preprocessing or intermediate step for others algorithms like classification, prediction, and other data mining applications.

Types of Clustering : based on the area of overlap

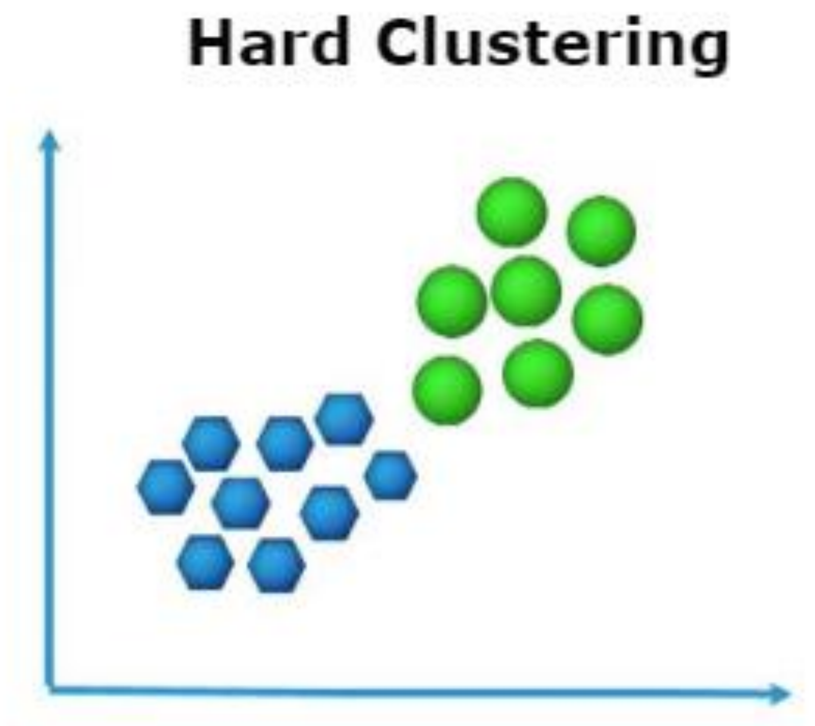
Hard clustering:

- Clusters do not overlap: k-means, k-means++.
- A data point belongs to one cluster only.

Soft clustering:

- Clusters can overlap: Fuzzy c-means, EM (Expectation Maximization).
- A data object can exist in more than one cluster with a certain probability or degree of membership.

Hard vs Soft Clustering



Type of Clustering : based on the purpose

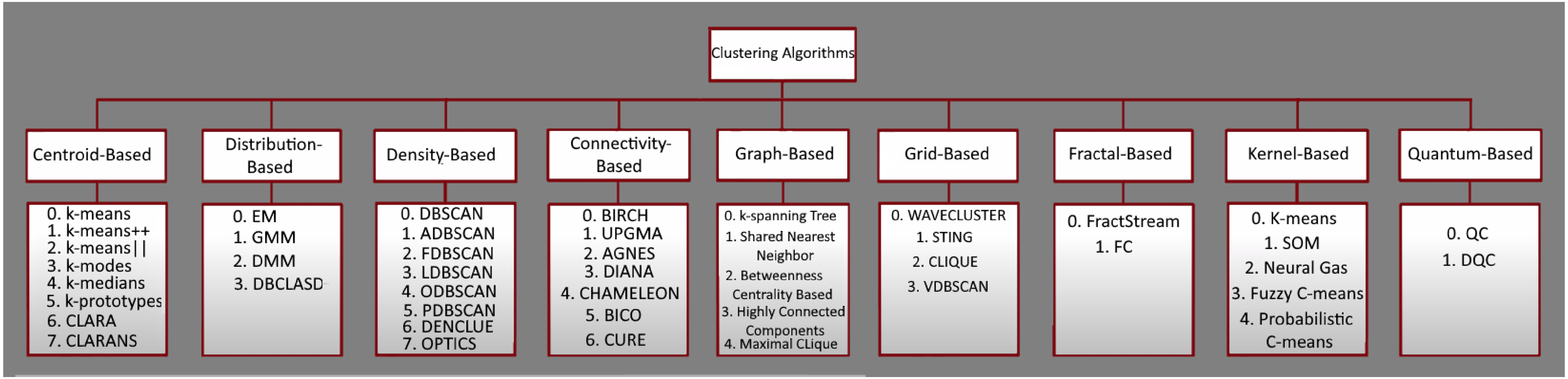
Monothetic:

- Exists some common properties between cluster members(e.g., 25% of patients show side effects due to vaccine A)
- the data are divided on values generated by a single feature.

Polythetic:

- Exists some degree of similarity between cluster members without having a common property (e.g., dissimilarity measure):
- the data are divided on values generated by all features.

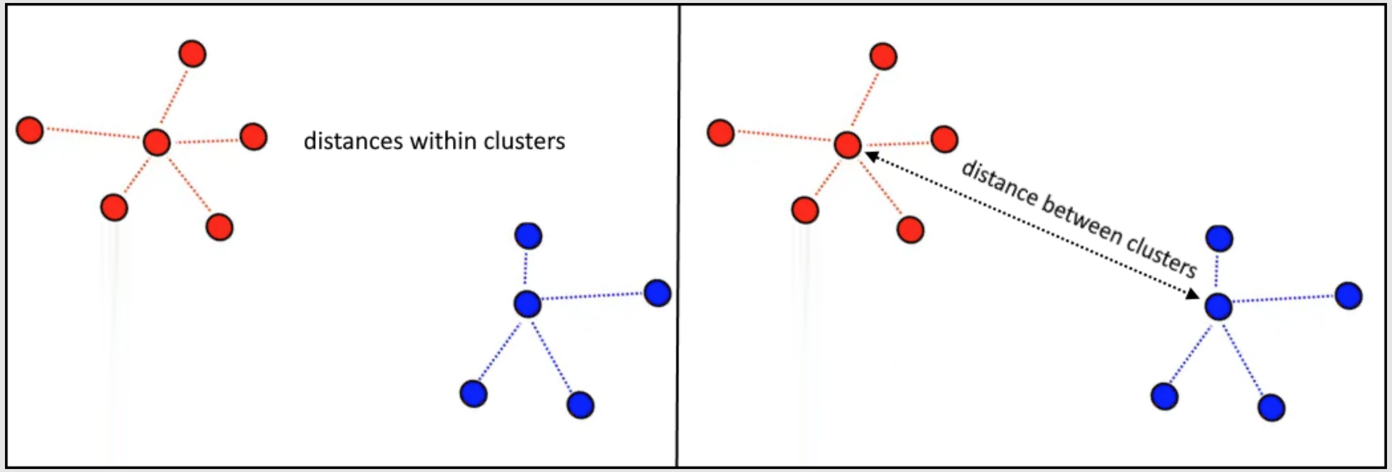
Some common categories of clustering alg's



k-means or Lloyd’s Algorithm

- A centroid-based algorithm
- One of the most popular partitioning algorithms
- n data objects are split into k partitions (k << n) where each partition represents a cluster.
- Each data object must belong to one group only.
- maximize the distance between each pair of clusters’ centers
- minimize the distance between observations within each cluster (SSE: sum of squared errors)

k-means or Lloyd's Algorithm



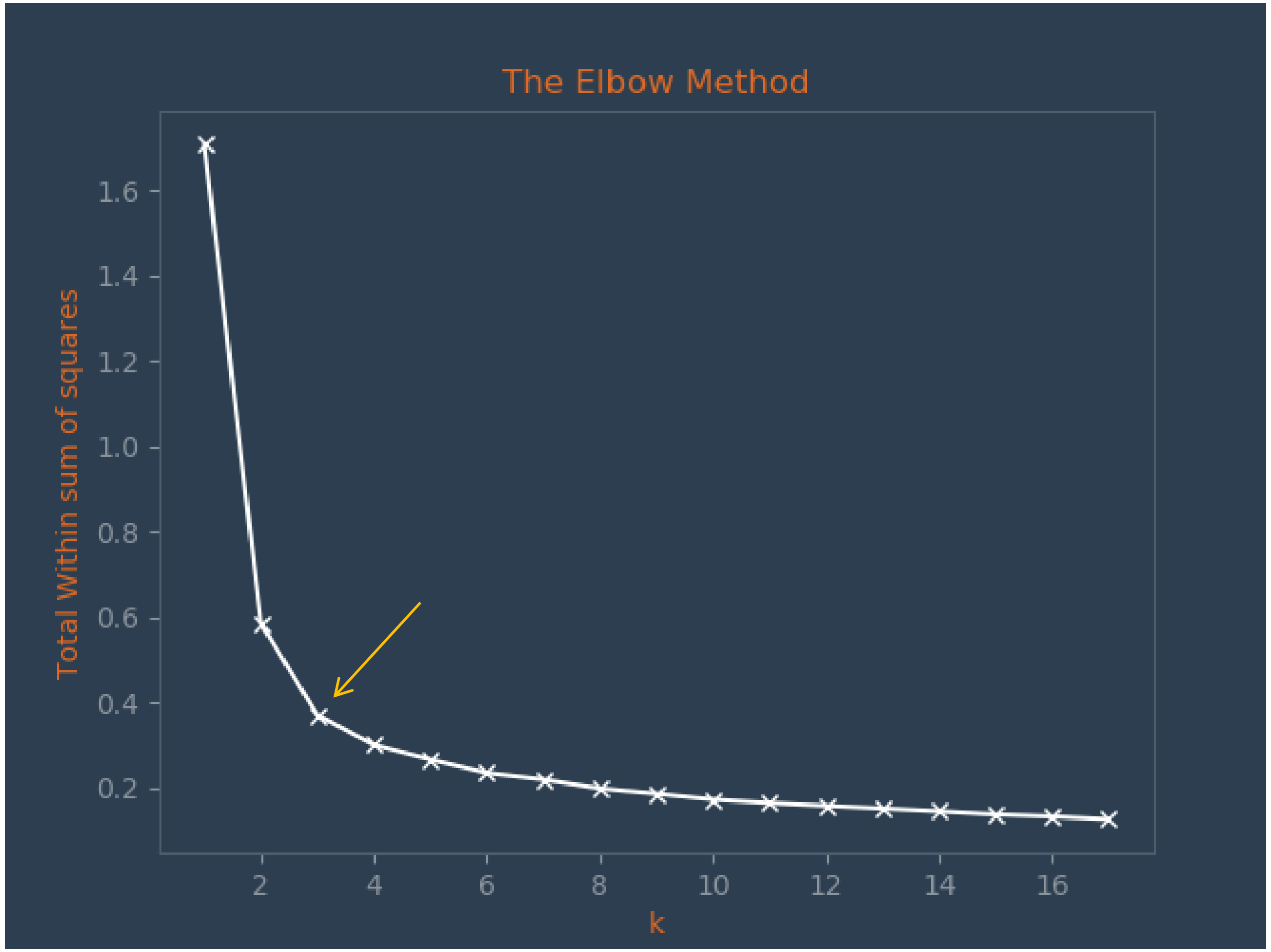
works well if the following conditions are met:

- The distribution's variance of each attribute is spherical.
- Clusters are linearly separable.
- Clusters have similar numbers of observations.
- Variables present the same variance.

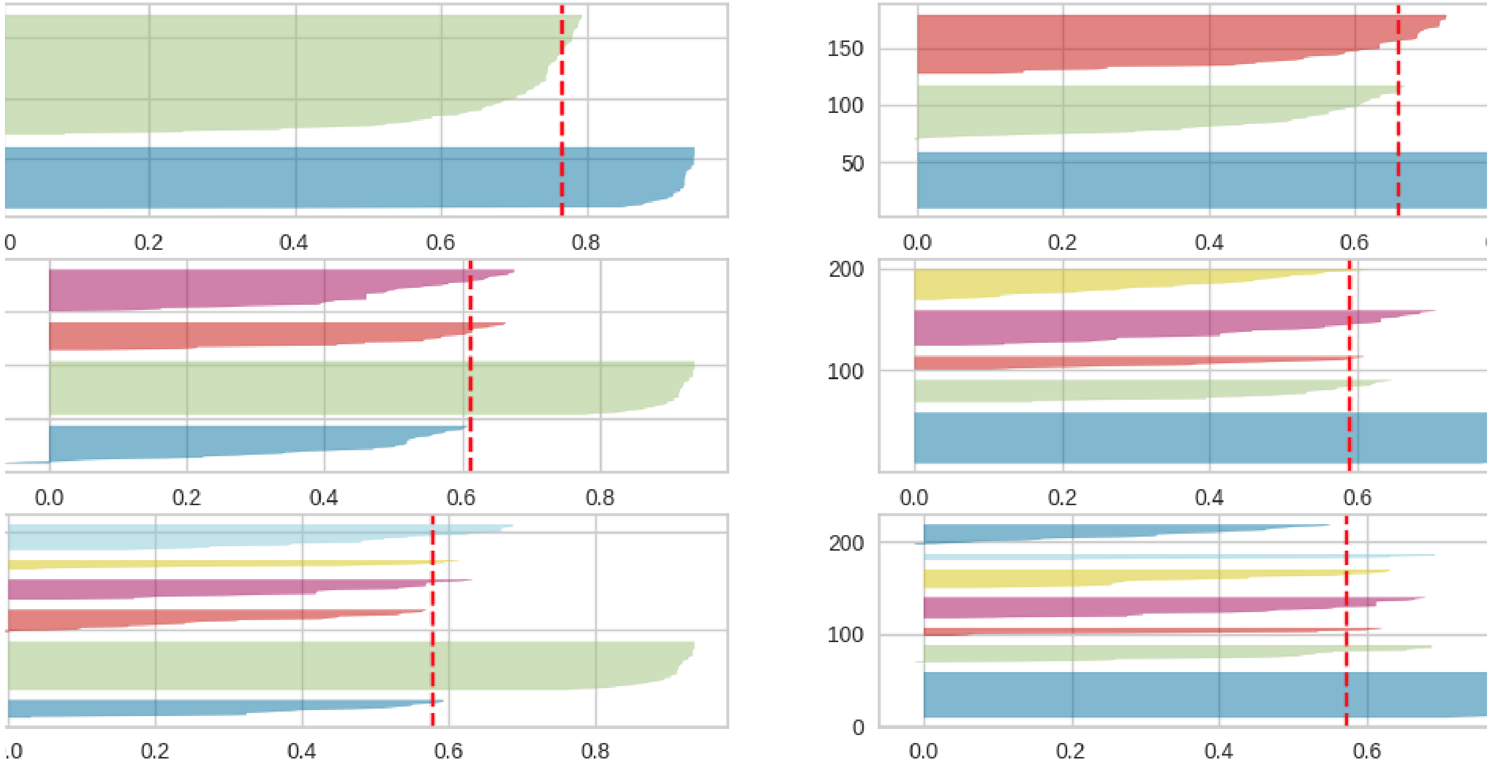
Elbow Method

The standard approach to determine k :

- The algorithm ran for different values of k (e.g., k= 1, 2, 3, 4…).
- Calculates the sum of distances between each cluster member and its centroid : WCSS - Within-Cluster Sum of Square.
    - Ex. optimum k = 3 in this graph



Silhouette Analysis



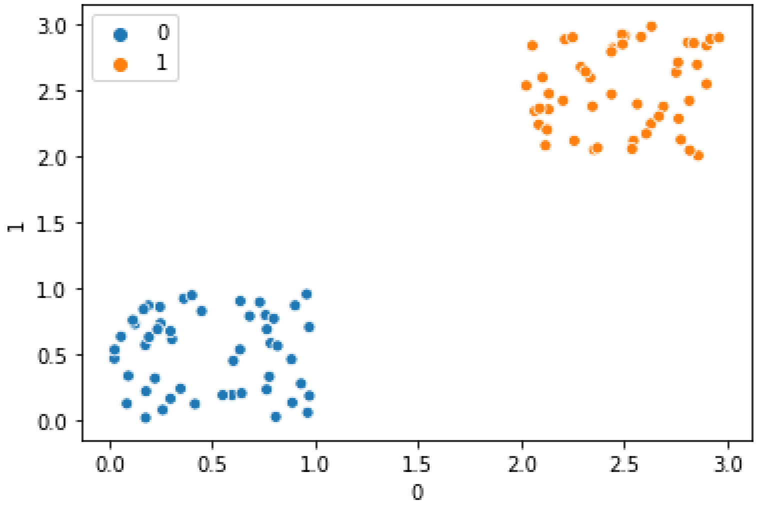
another method for choosing the right value of k by computing the Silhouette coefficient for each cluster

from sklearn.metrics import silhouette_samples

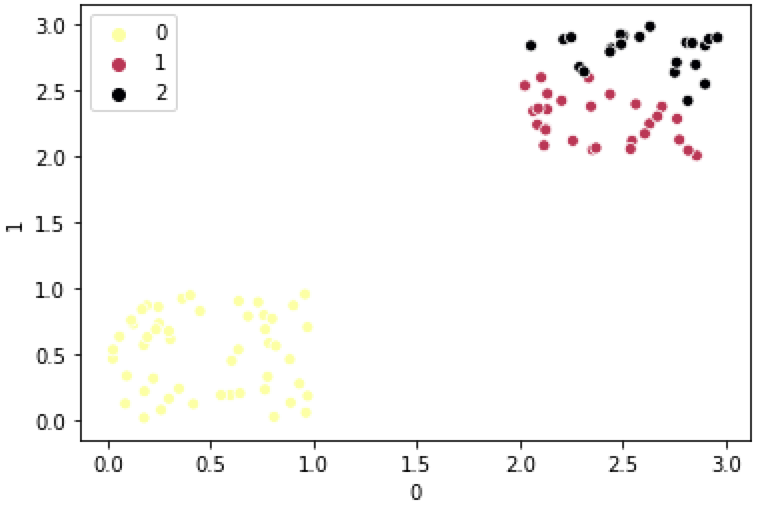
sample_silhouette_values = silhouette_samples(X, cluster_labels)

Silhouette scores for k = [2,3,4,5,6,7]

Silhouette Score



Silhouette Score(n=2):
0.8062146115881652



Silhouette Score(n=3):
0.5969732708311737

- 1: Means clusters are well apart from each other and clearly distinguished.
- 0: Means clusters are indifferent, or we can say that the distance between clusters is not significant.
- -1: Means clusters are assigned in the wrong way.

Silhouette Score = (b-a)/max(a,b)

- a = average intra-cluster distance i.e. the average distance between each point within a cluster.
- b = average inter-cluster distance i.e. the average distance between all clusters.

k-means or Lloyd's Algorithm

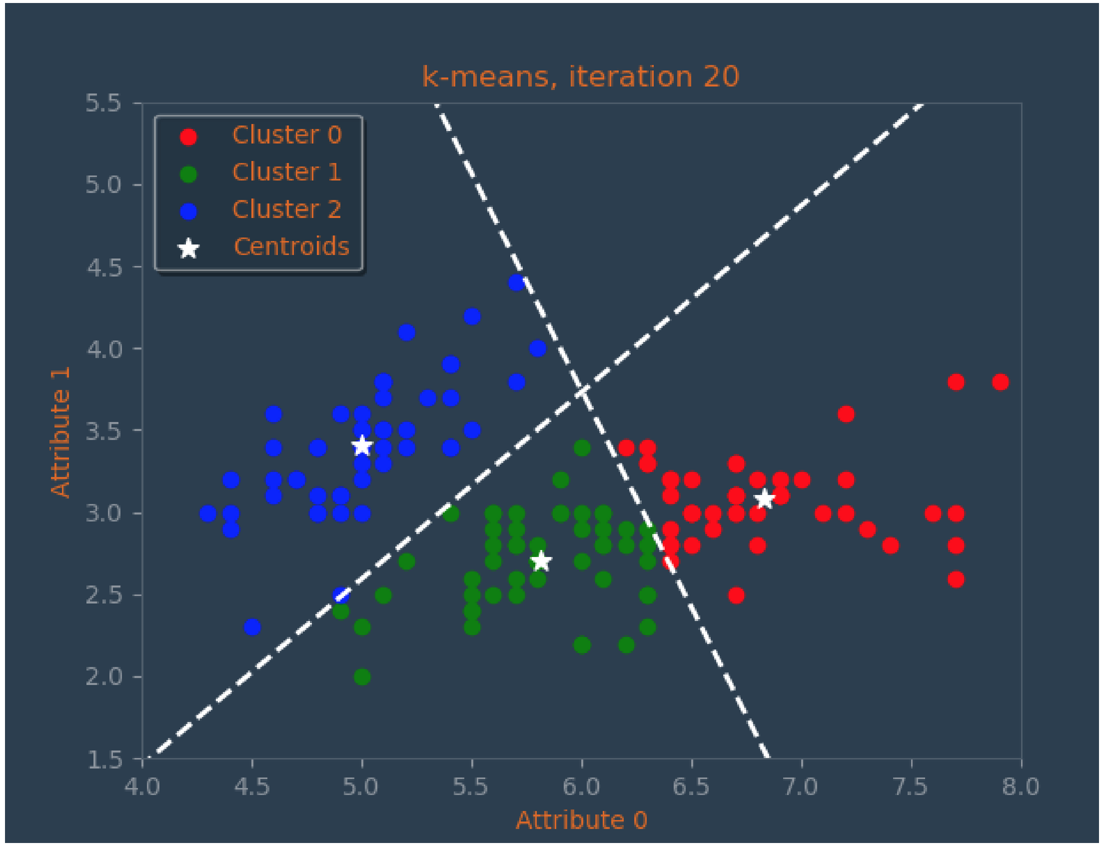
1. Pick k random centroids from the dataset.
2. Compute the distances between each data point w.r.t clusters' centroids
   - using a proper dissimilarity measure(e.g., Euclidean distance).
3. Assign each data point to the nearest cluster
4. Reposition the centroids by computing the mean of the data points.
5. Repeat until clusters become stable or the WCCS reaches its minimum.

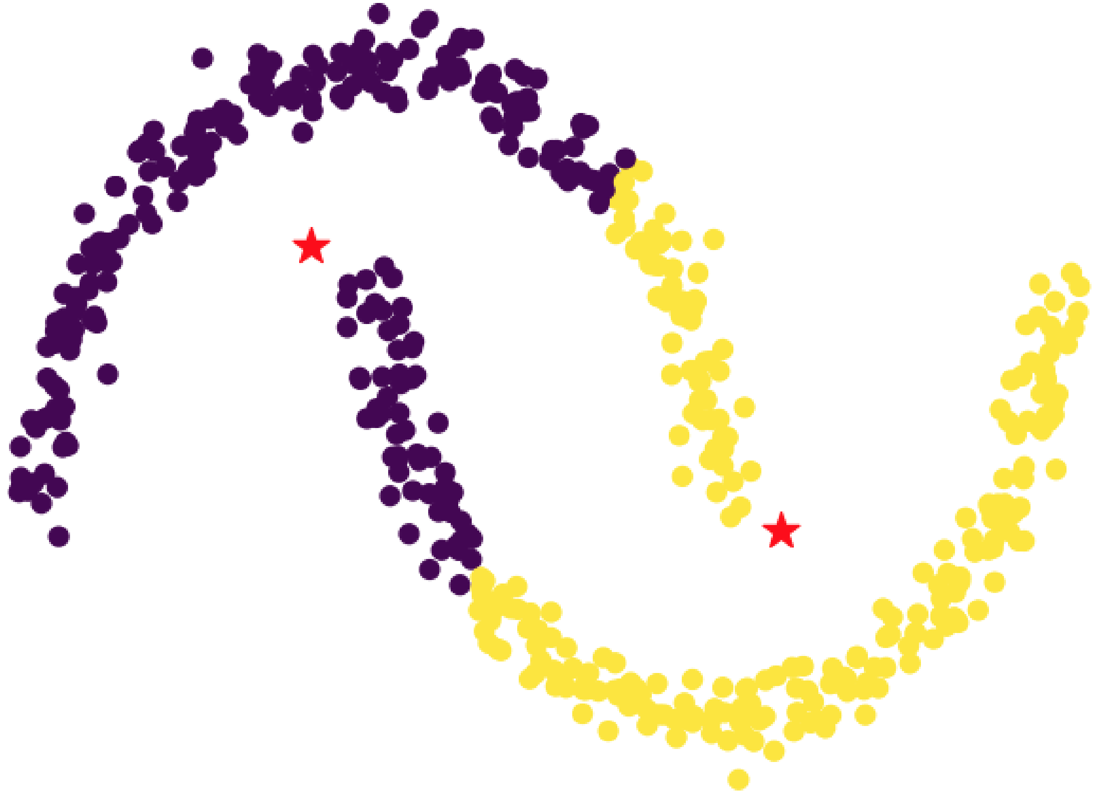
☝ Therefore k-means works only on numerical data!

k-means or Llyod’s Algorithm

Advantages	Drawbacks
• The learning curve is relatively steep • Widely implemented by a variety of packages • Fast convergence for clustering small datasets. • Easy to implement.	• Computationally expensive for large datasets (k becomes large.). • Sometimes difficult to choose an initial value for k. • Sensitive to the centroids’ initialization. • Strong sensitivity to outliers. • Works only on numerical data. • Fails to give good quality of clustering for non-convex shaped groups.

Example results





k-means++

The idea is that to try to spread out the centers while allocating a new center per iteration.

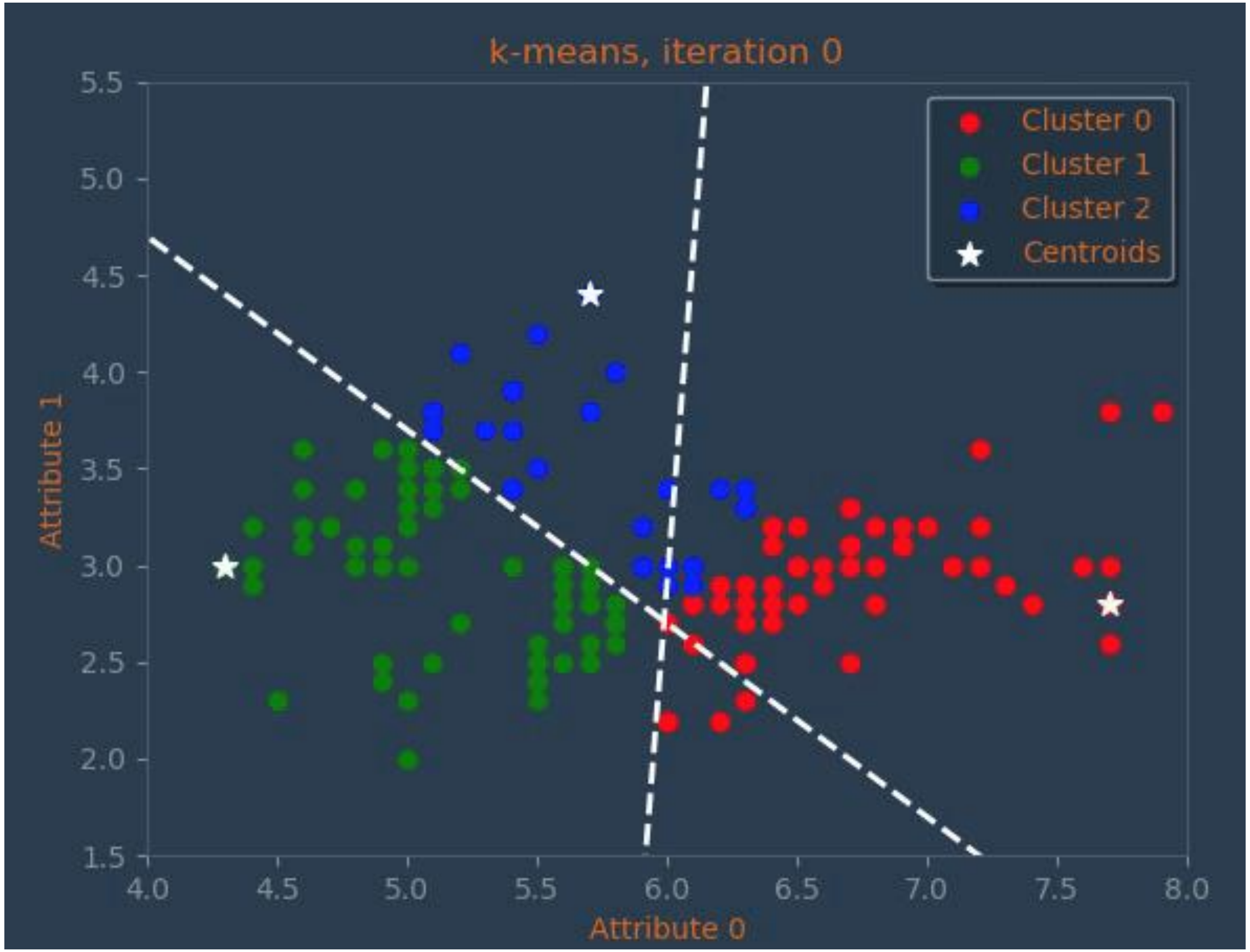
compute the probability for each data point by dividing its distance by the total distance.

assign a new cluster center to the point that has the highest probability or the highest distance.

the likelihood of a data object being the center of a new cluster is proportional to the distance squared.

converge much faster since the centroids have been chosen carefully and far away from each other.

k-means++



k-means, iteration 0
Cluster 0
Cluster 1
Cluster 2
Centroids
Attribute 1
Attribute 0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

Fuzzy C-means (FCM)

- Various shades of clusters (e.g., disjoint, non disjoints…) are allowed to form
  - A data point can exist in one or more clusters.
- A membership degree function is used to measure the degree of belonging of a data point to each cluster.
  - It describes the probability that a data point belongs to a certain cluster.
- The FCM aims to minimize an objective function:

$$\arg\min_{C} \sum_{i=1}^{n} \sum_{j=1}^{c} w_{ij}^{m} \left\| \mathbf{x}_i - \mathbf{c}_j \right\|^2$$

FCM Algorithm

1. Select a number of initial fuzzy pseudo centroids and hyper-parameter m. *The higher m is, the fuzzier the cluster.*
2. Update the cluster centers (c_k for cluster k) using a fuzzy partition.

$$c_k = \frac{\sum_x w_k(x)^m x}{\sum_x w_k(x)^m}$$

3. Updates the weights

$$w_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|\mathbf{x}_i - \mathbf{c}_j\|}{\|\mathbf{x}_i - \mathbf{c}_k\|} \right)^{\frac{2}{m-1}}}$$

4. Compute the objective function J.
5. Repeat until stabilizing the centroids ($\Delta J < \varepsilon$)

Fuzzy C-means (FCM)

Advantages	Drawbacks
• Better results for overlapped data in contrast to k-means. • Low time complexity. • Convergence is guaranteed.	• Sensitive to the initial values of number of clusters and hyper-parameter m. • Sensitive to outliers.

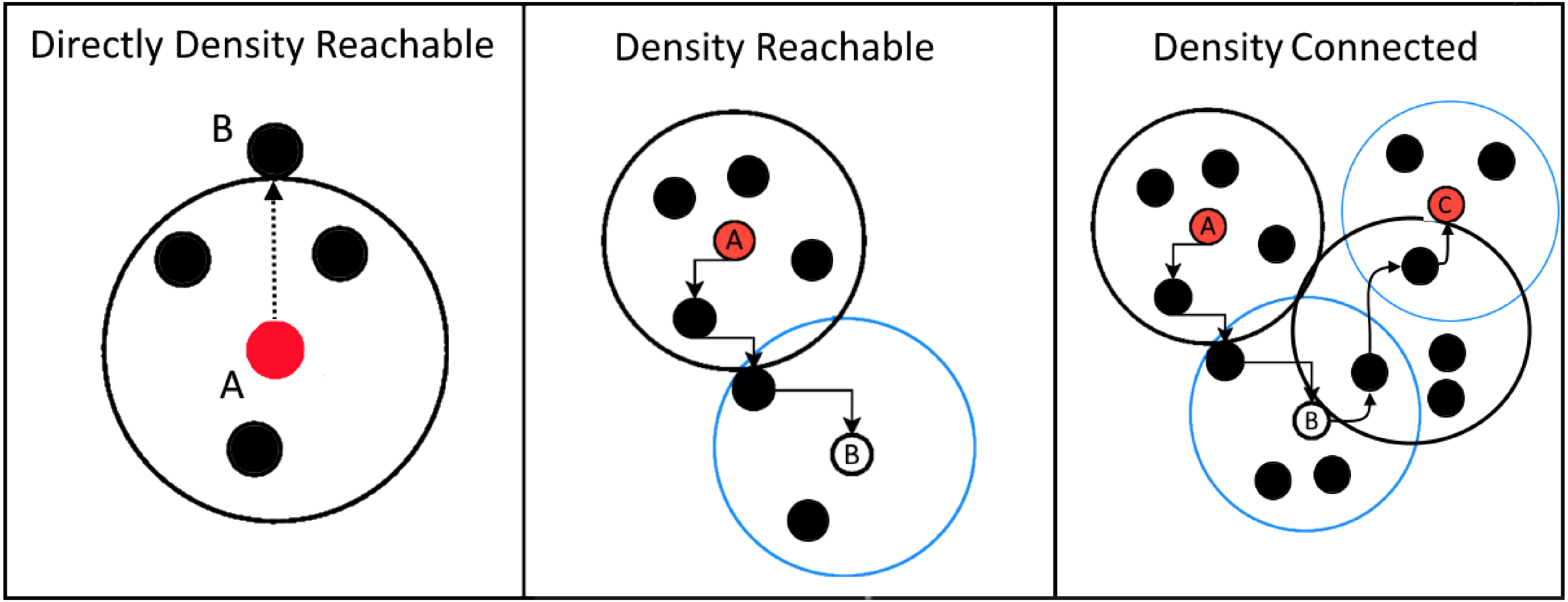
Density-based Clustering - DBSCAN

- Dense regions in the data space are separated from those with lower density.
- Observations are assigned to a given cluster if its density in a certain location is larger than a predefined threshold.
- The local density is defined by two parameters:
  1. the radius ε of the circle that contains a certain number of neighbors around a given point
  2. a minimum number of points around that radius : minPts

DBSCAN

- *Core Points*: A data point p is a *core point* if $|\mathbf{Nbhd}(p,\varepsilon)|$ >= *minPts.*
- *Border Points: A data point q* is a *border point* if $\mathbf{Nbhd}(q, \varepsilon)$ contains less than *minPts* data points,
  but q is *reachable* from some *core point* p.
- *Outlier*: A data point o is an *outlier* if it is neither a core point nor a border point. Essentially, this is the “other” class.

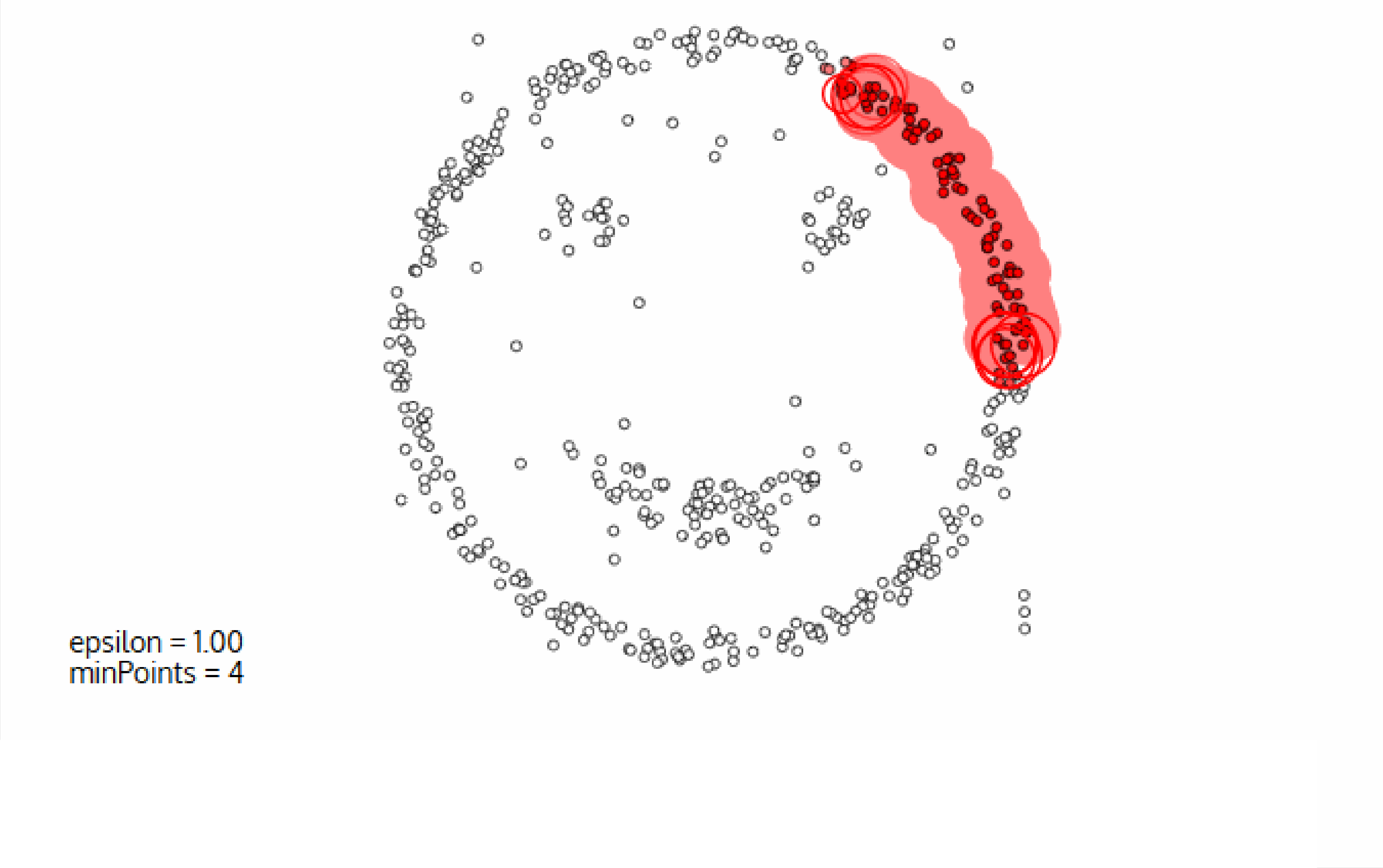
Density-reachable Concepts

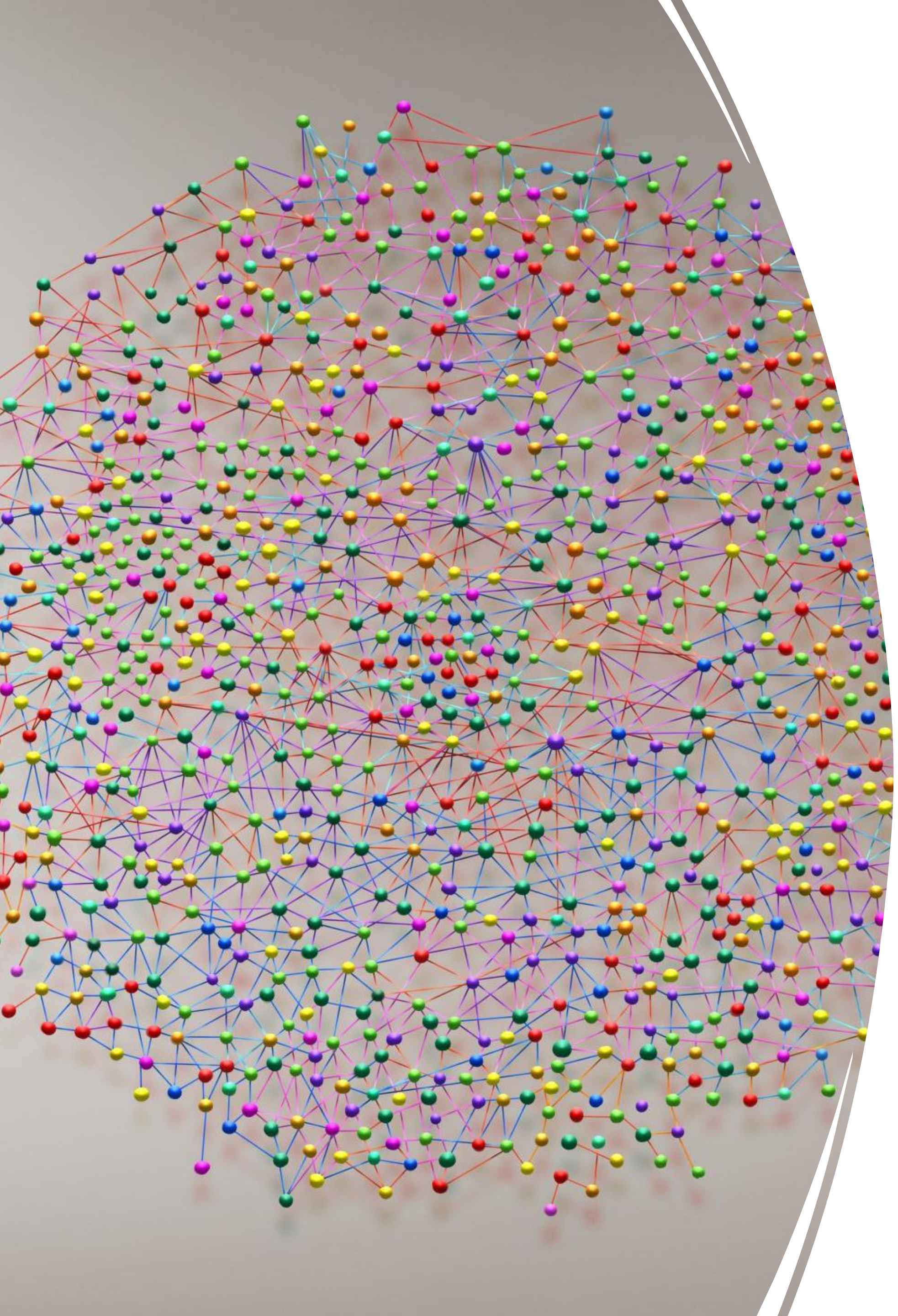


DBSCAN Algorithm

1. Pick a point at random that has not been assigned to a cluster or been designated as an *outlier*.
   - Compute its neighborhood to determine if it's a *core point*.
   - If yes, start a cluster around this point.
   - If no, label the point as an *outlier*.
2. For a *core point* (and thus a cluster):
   - expand the cluster by adding all *directly-reachable* points to the cluster.
   - Perform "neighborhood jumps" to find all *density-reachable* points
     - add them to the cluster (reachable = same cluster).
     - If an *outlier* is added, change that point's status from *outlier* to *border point*.
3. Repeat until all points are either assigned to a cluster or designated as an *outlier*.

DBSCAN Algorithm



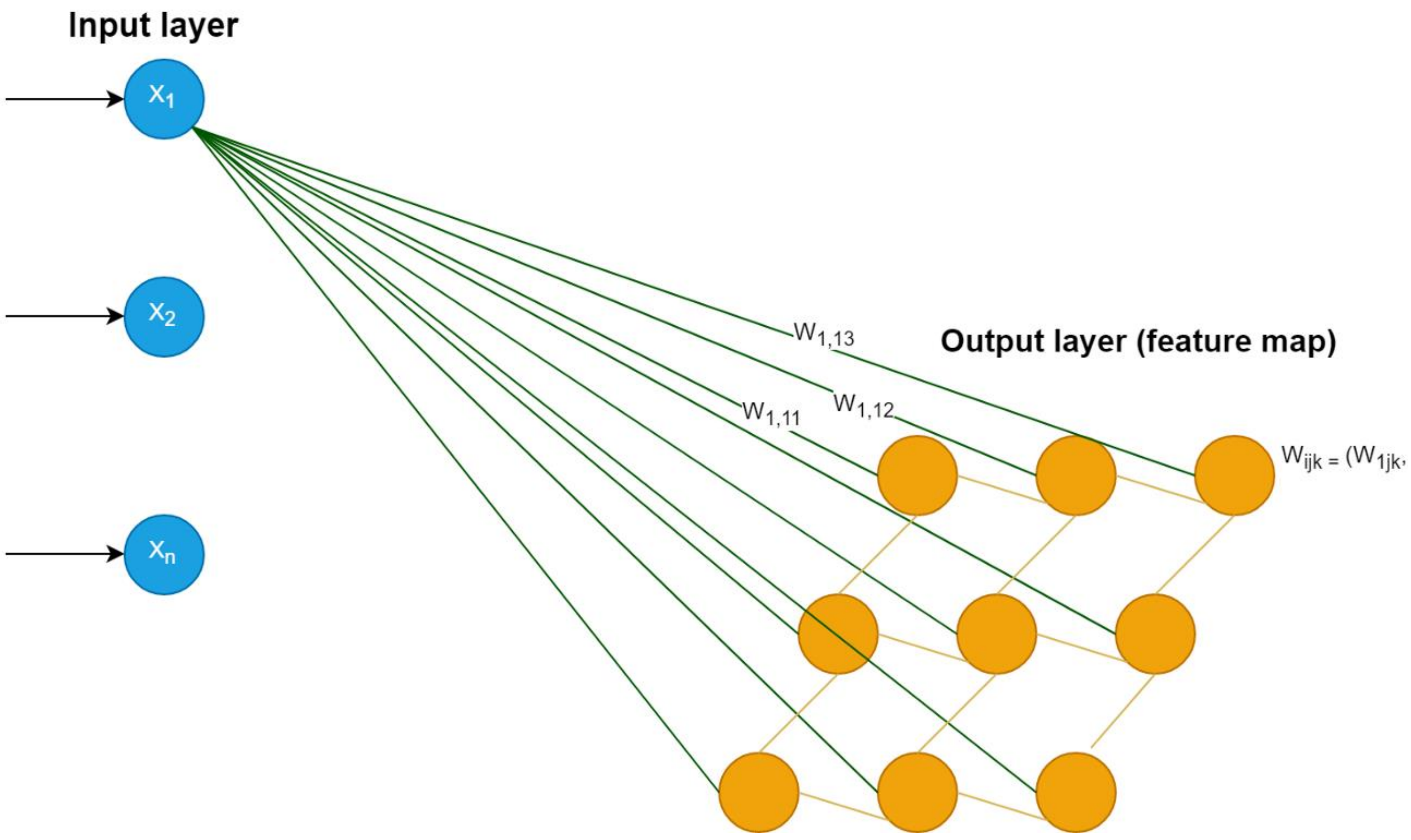


Self-Organizing Map (SOM)

- an unsupervised neural network
- produces a low dimensional, discretized representation from the input space of the training samples, known as a map
- utilizes competitive learning techniques
  - unlike others using error-correction learning methods
- maintains the structural information from the training data.

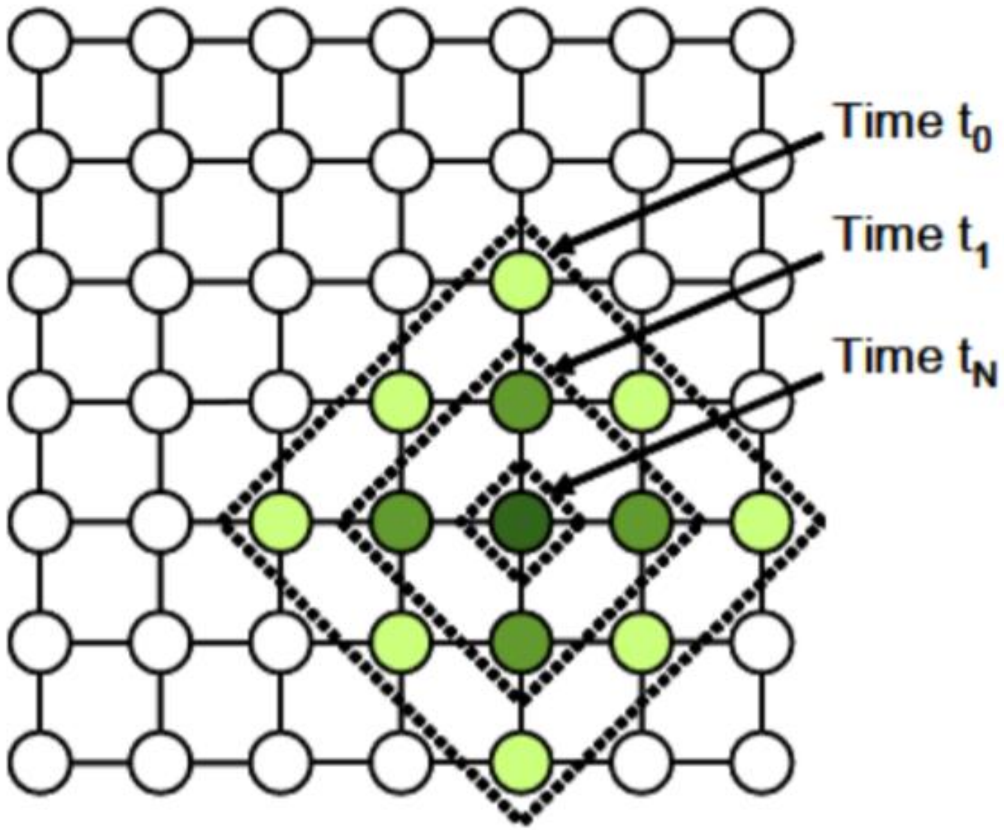
Feature Mapping

- The neuron with the lowest distance will be the winner of the competition.
- The Euclidean metric is commonly used to measure distance.

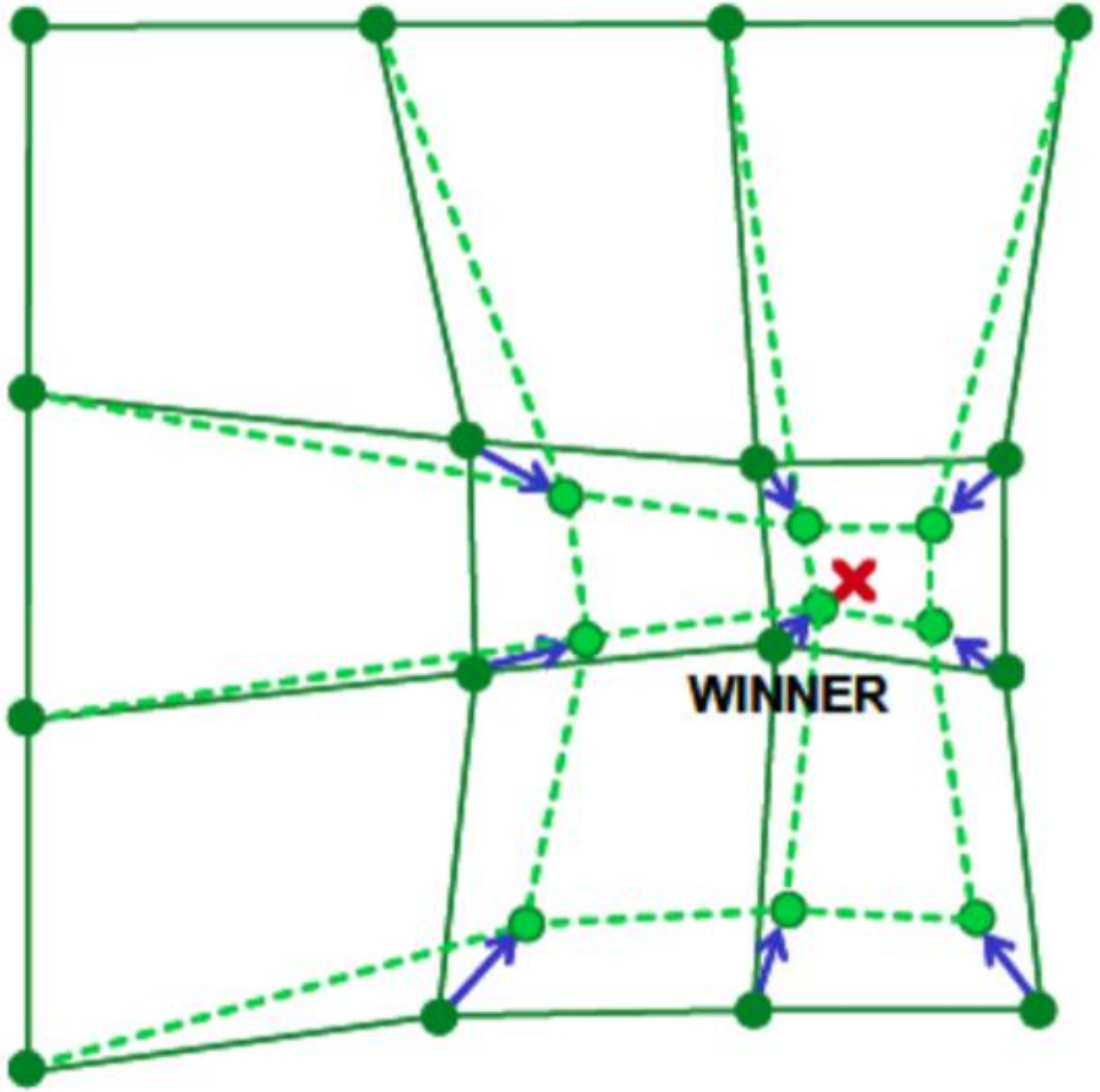


Neighborhood

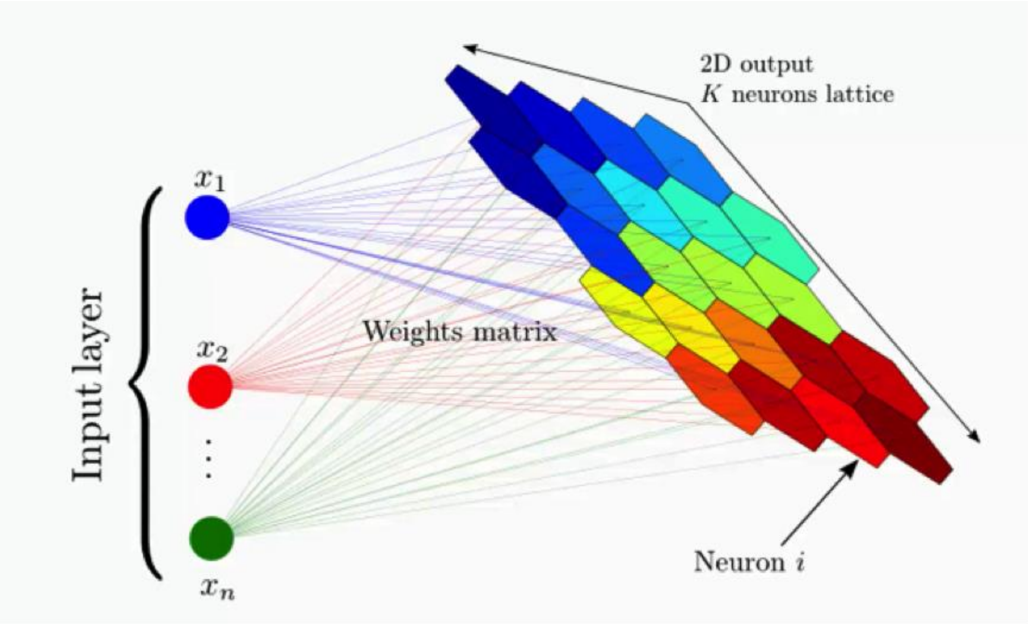
- Winning neuron's weights is to be updated.
  - So are its neighbors
- Neighborhood is dependent on
  - time (time incremented each new input data)
  - distance between the winner neuron and the other neuron (How far is the neuron from the winner neuron).

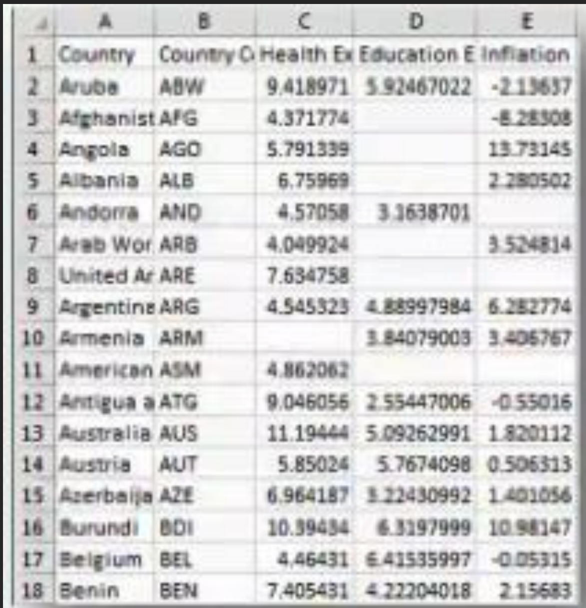


Adaptation

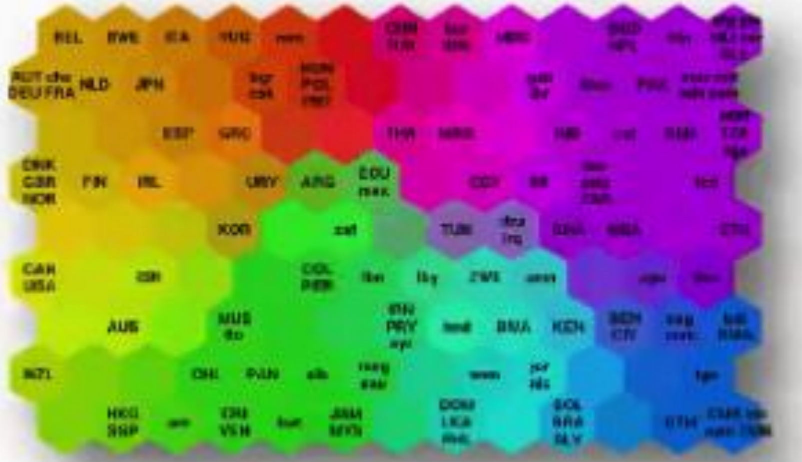


SOM Example: Economic wellbeing





	A	B	C	D	E
1	Country	Country C	Health Ex	Education E	Inflation
2	Aruba	ABW	9.418971	5.92467022	-2.13637
3	Afghanist	AFG	4.371774		-8.28308
4	Angola	AGO	5.791339		13.73145
5	Albania	ALB	6.75969		2.280502
6	Andorra	AND	4.57058	3.1638701	
7	Arab Wor	ARB	4.049924		3.524814
8	United Ar	ARE	7.634758		
9	Argentine	ARG	4.545323	4.88997984	6.282774
10	Armenia	ARM		3.84079003	3.406767
11	American	ASM	4.862062		
12	Antigua a	ATG	9.046056	2.55447006	-0.55016
13	Australia	AUS	11.19444	5.09262991	1.820112
14	Austria	AUT	5.85024	5.7674098	0.506313
15	Azerbaija	AZE	6.964187	3.22430992	1.401056
16	Burundi	BDI	10.39434	6.3197999	10.98147
17	Belgium	BEL	4.46431	6.41535997	-0.05315
18	Benin	BEN	7.405431	4.22204018	2.15683



SOM Pros and Cons

Pros

- Data can be easily interpreted and understood
  - with the help of techniques like reduction of dimensionality and grid clustering.
- Self-Organizing Maps are capable of handling several types of classification problems
  - providing a useful, and intelligent summary from the data at the same time.

Cons

- It does not create a generative model for the data
  - therefore the model does not understand how data is being created.
- Self-Organizing Maps do not perform well while working with categorical data and even worse for mixed types of data.
- The model preparation time is comparatively very slow and hard to train against the slowly evolving data.