# 11 Clustering

| | |
|---|---|
| Date | @November 26, 2024 |
| Topic | Theory |

A cluster is a group of several discrete items that are close to each other.

Unsupervised learning with clustering is about detecting groups of similar data points and it is unsupervised because there are no labelled data points, no ground truth to compare with and no y.

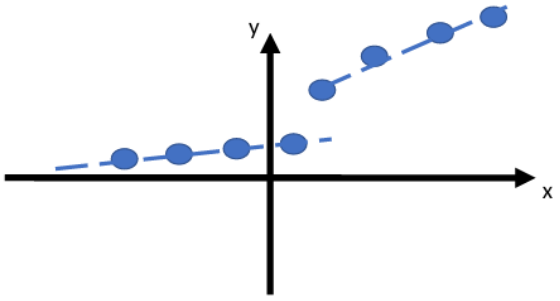
A cluster corresponds to a subset of data points that are in some sense homogeneous or similar. At the same time the elements in a cluster must be different from elements in another cluster.

## Clustering for image segmentation

We can use clustering for image segmentation. We can take a full picture, split it into subparts and all this parts represent the data points. We can use three features: average red, green and blue component. OR we can use two features (red and green).

## Clustering as pre-processing

For example, if we want to perform regression on a dataset with linear regression.




We can divide the dataset into two different groups (clusters) so that it appears to be a line for each cluster and we can apply linear regression separately.

# Hard clustering

In hard clustering data points are characterized by n features

$$\boldsymbol{x}^{(i)} = \left(x_1^{(i)}, \ldots, x_n^{(i)}\right)$$

the i-th data point belongs to one of k clusters and the cluster index of i-th datapoint is

$$y^{(i)} \epsilon \{1, \dots, k\}$$

Hard clustering methods compute predicted cluster indices based solely on features:

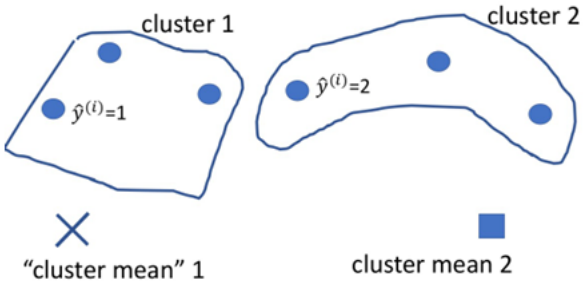
$$\boldsymbol{x}^{(i)} = \left(x_1^{(i)}, \dots, x_n^{(i)}\right)$$

This method does not require true cluster index $y^{(i)}$ of any data point.


# Hard clustering with k-means

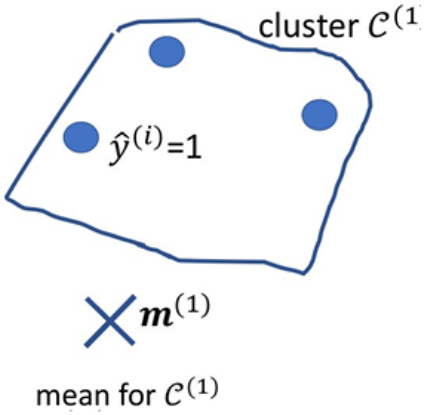
## K-Means clustering algorithm

K-Means has an hyperparameter which is the number of cluster k and each cluster is associated with a relevant point called mean or centroid.




We can have representative point for each cluster, and we call them mean.

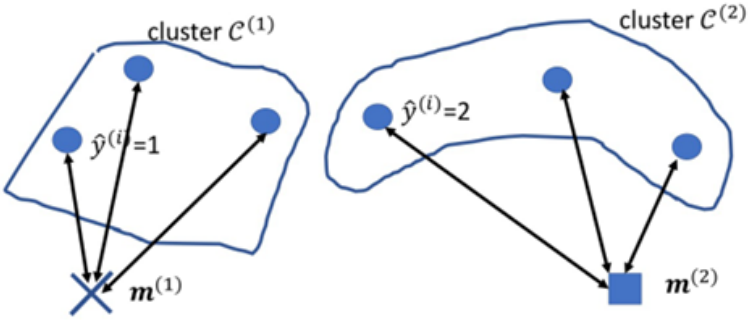
### Cluster spread




Considering the euclidean distance between points and mean of cluster, we compute

$$(1/|\ \mathcal{C}^{(1)}\ |)\sum_{i\in\mathcal{C}^{(1)}}\left\|\boldsymbol{m}^{(1)}-\boldsymbol{x}^{(i)}\right\|^2$$

## Clustering error




$$\frac{1}{m}\sum_{c=1}^{2}\sum_{i\in\mathcal{C}^{(c)}}||\mathbf{m}^{(c)}-\mathbf{x}^{(i)}||^2$$

To minimize this function we can assign, for given cluster means, the i-th data point to the cluster with the nearest cluster mean.

$$\hat{y}^{(i)} \coloneqq c$$

$$\text{with}\ \ \left\|\boldsymbol{m}^{(c)}-\boldsymbol{x}^{(i)}\right\|^2=\min_{c'=1,\ldots,k}\left\|\boldsymbol{m}^{(c')}-\boldsymbol{x}^{(i)}\right\|^2$$

For given cluster assignments, clustering error is minimized by representing the c-th cluster by the cluster mean of feature vectors of its data points

$$m^{(c)} := \frac{1}{|\mathcal{C}^{(c)}|} \sum_{i \in \mathcal{C}^{(c)}} \boldsymbol{x}^{(i)}$$

$$\text{with cluster } \mathcal{C}^{(c)} = \{i : \hat{y}^{(i)} = c\}$$

## Hard clustering error

The hard clustering error, which has to be minimized, is

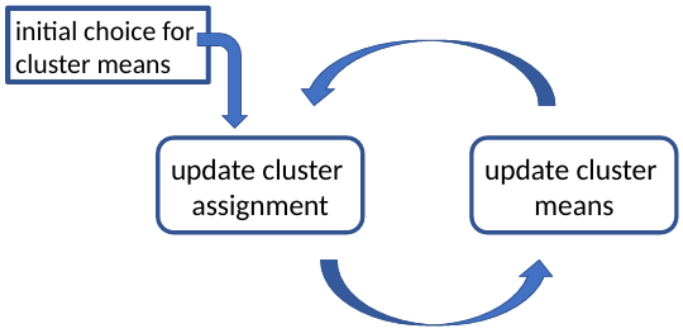
$$\mathcal{E}(\{m^{(c)}\}, \{\hat{y}^{(i)}\}) := \frac{1}{m} \sum_{i=1}^{m} \left\| \boldsymbol{m}^{(\hat{y}^{(i)})} - \boldsymbol{x}^{(i)} \right\|^2$$

and this is basically an ERM problem

$$\widehat{L}(h|\mathcal{D}) = (1/m) \sum_{i=1}^{m} \left\| \mathbf{x}^{(i)} - \frac{\sum_{i' \in \mathcal{D}^{(i)}} \mathbf{x}^{(i')}}{|\mathcal{D}^{(i)}|} \right\|^2 \qquad \text{with } \mathcal{D}^{(i)} := \{i' : h(\mathbf{x}^{(i)}) = h(\mathbf{x}^{(i')})\}$$

It means to simultaneously find cluster means $m^{(c)}$ and assignments $y^{(i)}$ that minimize the clustering error, but it is difficult.

So we can for given assignments, find cluster means that minimize the clustering error, and for given cluster means, we can find assignments that minimize the clustering error.

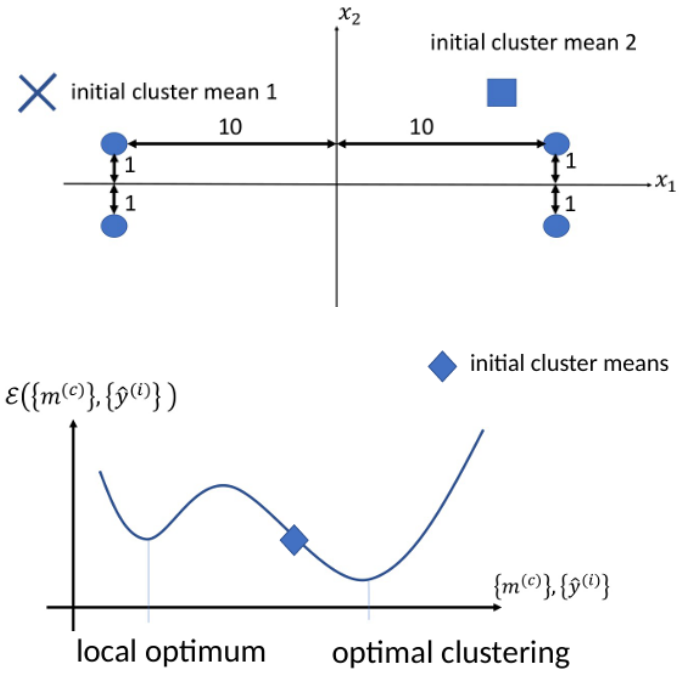



So, given the number k of cluster and the initial cluster means, we have to update the cluster assignments (step 1), update the cluster means (step 2), go back to step 1 until it's finished and then the output would be the final cluster means.

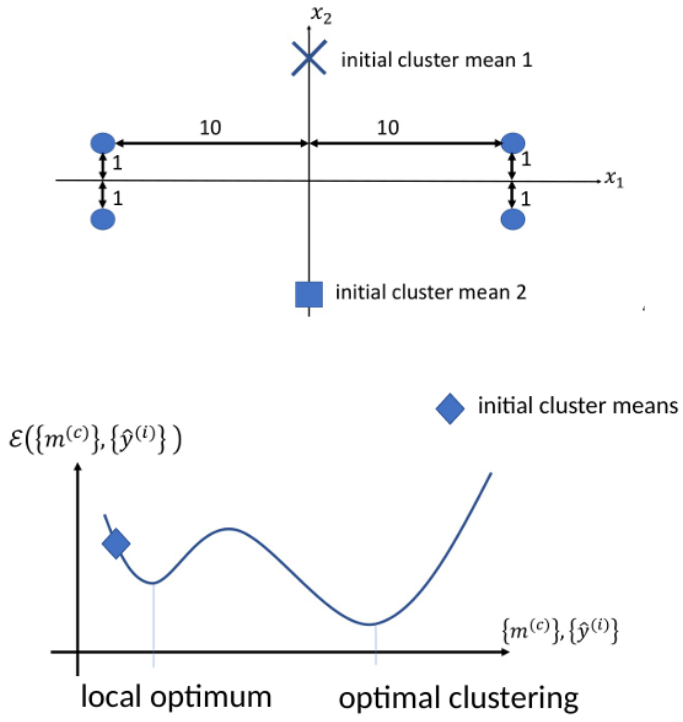
This means that k-Means never increases clustering error and K-Means is an iterative optimization method-

The initialization is crucial. k-Means requires initial cluster means as inputs and its result depends crucially on initial means. So we should repeat the k-Means several times with different initializations.

A good one can be



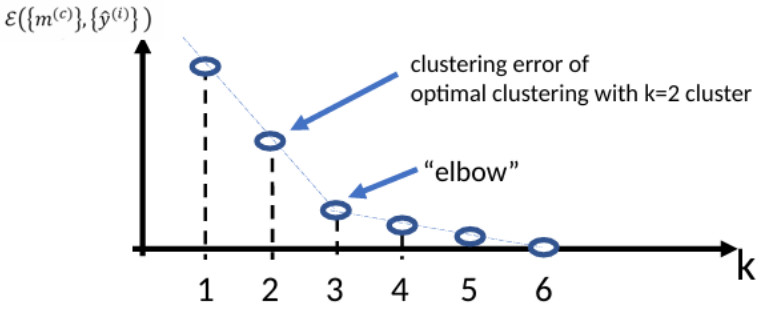

A bad one is



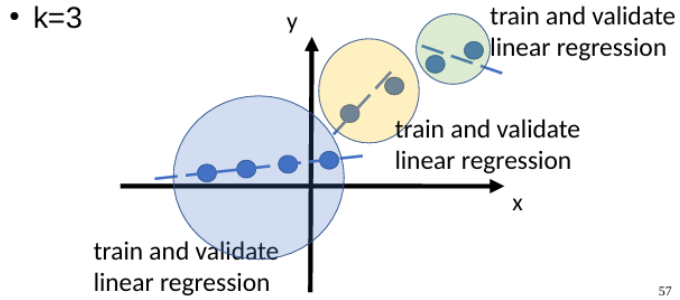

To choose the number of cluster k we can minimize the clustering error, and we can do it in different ways:

- defined by application

- desired compression rate
- elbow (or knee) method




- validation error: clustering can be used as pre-processing for follow-up supervised problem and then we can try different values of k and pick the one resulting in smallest validation error in the supervised problem




In python

**sklearn.cluster.KMeans**

*class* sklearn.cluster.**KMeans**(*n_clusters=8, *, init='k-means++', n_init='warn', max_iter=300, tol=0.0001, verbose=0, random_state=None, copy_x=True, algorithm='lloyd'*) [source]

# Soft clustering

In soft clustering we have data points characterized by n features.

$$\mathcal{D} = \left\{ \left( \mathbf{x}^{(1)}, y^{(1)} \right), \ldots, \left( \mathbf{x}^{(m)}, y^{(m)} \right) \right\}.$$

The features of the i-th data are

$$\boldsymbol{x}^{(i)} = \left( x_1^{(i)}, \ldots, x_n^{(i)} \right)$$

and the i-th data is characterized by k label values, that are

$$\boldsymbol{y}^{(i)} = \left(y_1^{(i)}, \dots, y_k^{(i)}\right)$$

so y (for a single data) is no longer a scalar number.

The k labels represent the degree of belonging of the object to the different clusters:

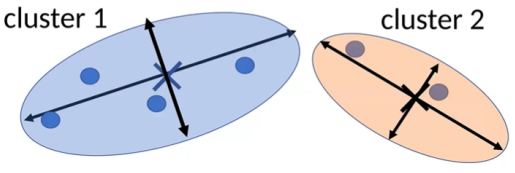
- $y_1^{(i)}$ is the degree of i-th data point belonging to cluster 1
- $y_2^{(i)}$ is the degree of i-th data point belonging to cluster 2
- ...
- $y_k^{(i)}$ is the degree of i-th data point belonging to cluster k

In hard clustering what we had were all zeros except only one, so that each object belonged to a single cluster. Here each object belongs to all clusters but with a different degree of belonging (it is a soft assignment, which gives the name to this type of clustering).

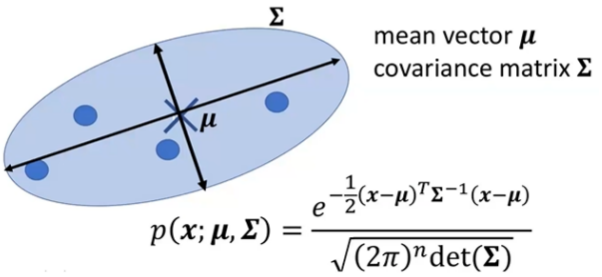
We can see these degrees of belonging with a probabilistic interpretation and see them as the probability of belonging to a cluster. It means that $y_c^{(i)}$ is the degree of the i-th datapoint belonging to cluster c. Obviously, since it is a probability, $y_c^{(i)}$ can be any number between 0 and 1 (included) and the sum of the degree of the data points belonging to a specific cluster must be equal to 1, that is $\sum_{c=1}^{k} = 1$. On the contrary, hard clustering required $y_c^{(i)}$ to be either 0 or 1.


## GMM - Gaussian Mixture Model

In this soft clustering algorithm we assume that data are generated by random rows from some Gaussian distribution (multi-dimensional since we have n features). The idea is that if you are at the edge of the distribution it means that the probability is low and you're not likely to belong to the specific cluster. Each cluster is a different Gaussian distribution and so each cluster has its own mean and standard deviation.



Each cluster will be represented with a Gaussian distribution, by its mean vector $\mu$ and its covariance matrix $\Sigma$.




$$p(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}}{\sqrt{(2\pi)^n \det(\boldsymbol{\Sigma})}}$$

## Cluster spread

We can define the cluster spread for GMM as well, that tells how good our Gaussian is with respect to our points. In this case is the distance of the points from the mean.

$$\frac{1}{m^{(c)}}\sum_{i=1}^{m}\hat{y}_c^{(i)}\left(\boldsymbol{x}^{(i)}-\boldsymbol{\mu}^{(c)}\right)^T\left(\boldsymbol{\Sigma}^{(1)}\right)^{-1}\left(\boldsymbol{x}^{(i)}-\boldsymbol{\mu}^{(c)}\right)$$

The effective cluster size in this case is

$$m^{(c)} := \sum_{i=1}^{m}\hat{y}_c^{(i)}$$

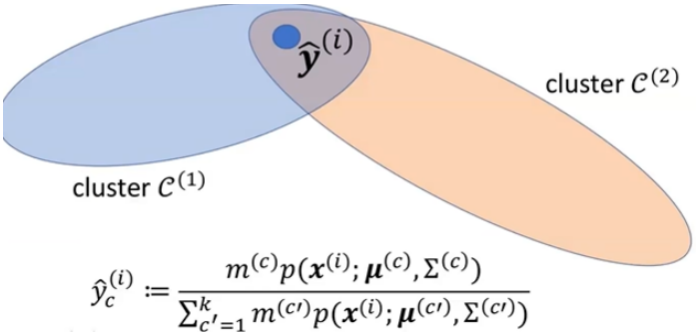
If we have a given soft cluster assignment, which means that we are given all the different labels, then we can improve our Gaussian, similarly to the case of k-means. In this case we move both the mean and the covariance matrix.

$$\boldsymbol{\mu}^{(c)} := \frac{1}{m^{(c)}}\sum_{i=1}^{m}\hat{y}_c^{(i)}\,\boldsymbol{x}^{(i)} \qquad \text{for all c =1,...,k}$$

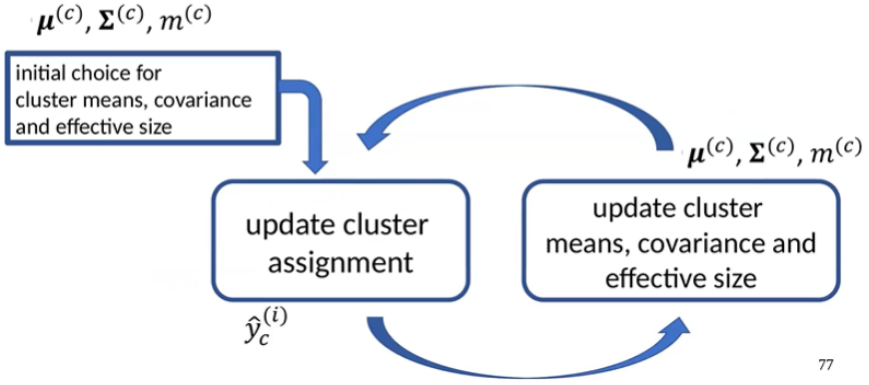
$$\boldsymbol{\Sigma}^{(c)} := \frac{1}{m^{(c)}}\sum_{i=1}^{m}\hat{y}_c^{(i)}\left(\boldsymbol{x}^{(i)}-\boldsymbol{\mu}^{(c)}\right)\left(\boldsymbol{x}^{(i)}-\boldsymbol{\mu}^{(c)}\right)^T$$

So basically we are computing the weighted sample mean and the weighted sample covariance matrix.

If we have the Gaussian, what we can do is to improve the assignment.




So we can perform an initial random choice for the mean, the covariance matrix and the effective size of clusters, then we can update the cluster assignment, updating the means, the covariance and the effective size, and continue in this loop until we reach a convergence.




## Soft clustering error

We can define a soft clustering error by using the sum over all the samples of the logarithm of the sum over each cluster of the probability of observing the features given the mean and the covariance matrix.

$$\mathcal{E}\left(\{\boldsymbol{\mu}^{(c)}\}, \{\boldsymbol{\Sigma}^{(c)}\}, \{m^{(c)}\}\right) :=$$
$$-\sum_{i=1}^{m} \log \sum_{c=1}^{k} \frac{m^{(c)}}{m} p\left(\boldsymbol{x}^{(i)}; \boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}\right)$$

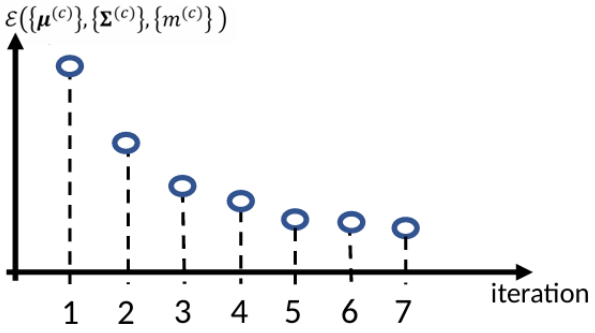
This is a way to say how likely is that we observe those features if they belong to one cluster or the other.

We put a - in front of the computation because it is an error and since the result of the logarithm will be negative, if we put a - in front of it (getting a positive value) we can speak about minimizing the error. Otherwise we can speak about maximizing the log likelihood.

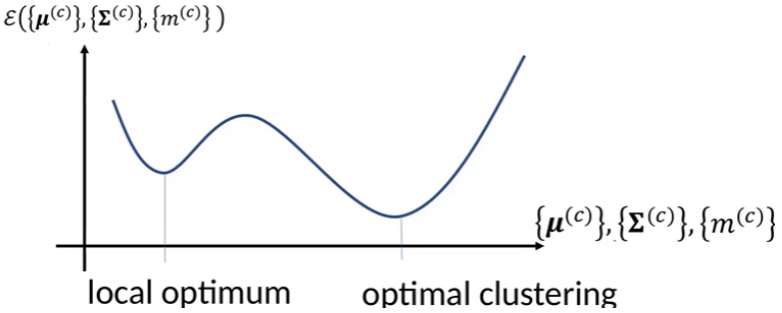
As for k-means, this is an ERM problem where the loss is the logarithm:

$$\widehat{L}(\boldsymbol{\theta} \mid \mathcal{D}) := -\log p(\mathcal{D}; \boldsymbol{\theta}) \text{ with GMM parameters } \boldsymbol{\theta} := \{\boldsymbol{\mu}^{(c)}, \boldsymbol{\Sigma}^{(c)}, p_c\}_{c=1}^{k}$$

We can decide when to stop the iterations of the algorithm or when the decrease is too small or after a fixed number of iterations.




This error is non-convex with respect to the Gaussian parameters, so we can stop in a local optimum or in a global optimum.




Also for soft clustering the initialization is crucial and that's why you need to repeat the algorithm multiple times with different initalizations.

The number k of clusters can be defined:

- by the application
- depending on the desired compression rate
- using the elbow (knee) method
- based on the validation error

In python we can use

### sklearn.mixture.GaussianMixture

*class* sklearn.mixture.**GaussianMixture**(*n_components=1, *, covariance_type='full', tol=0.001, reg_covar=1e-06, max_iter=100, n_init=1, init_params='kmeans', weights_init=None, means_init=None, precisions_init=None, random_state=None, warm_start=False, verbose=0, verbose_interval=10*) [source]

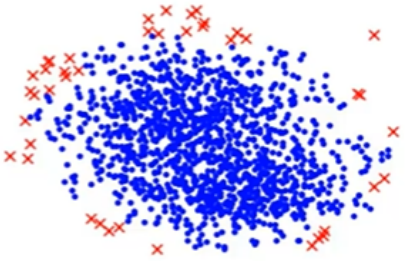
# Other clustering approaches

## Connectivity based clustering

If you have a dataset like this

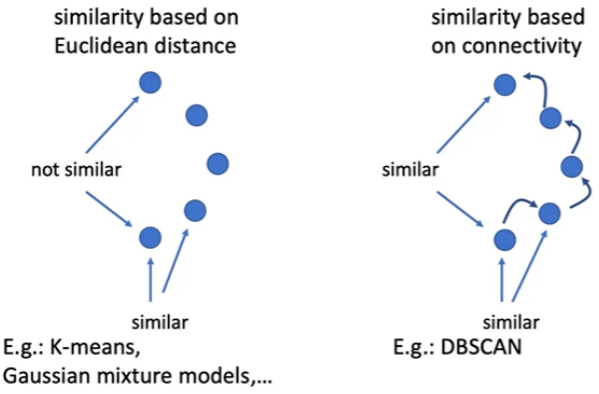
k-means and GMM cannot see what we can se with our eyes, which are these circles and groups of objects, because they fails on non-Euclidean cluster structures.

As well if you have noise or outliers, k-means and GMM cannot recognize them



As a solution, we can use the connectivity based clustering, in which we use the connectivity between samples so that if we have close objects then we can construct an empirical graph by connecting the points that are close to each other. The resulting cluster will be a connected graph component.

The difference between algorithms which use the concept of similarity based on the Euclidean distance and the ones based on the connectivity is



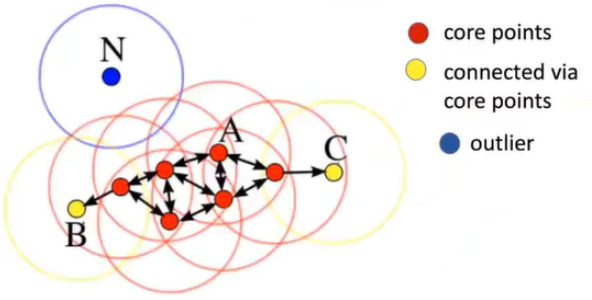

We have different algorithms which are used for connectivity based clustering. For hard clustering we have:

- spectral clustering, which creates a Laplacian matrix of the graph extracting the eigenvectors, in order to measure connectivity between nodes
- DBSCAN, which is density based spatial clustering with noise

## DBSCAN

In this clustering algorithm we define three types of samples:

- the core points, which are the ones connected with a minimum number of neighbors
- the border points are points connected via core points, which have not “enough” neighbors
- the outliers, which are not connected to any other object




With this algorithm we don’t have to specify the number of clusters we want to identify, because clusters are automatically found.

There are many parameters, but the two most important one are:

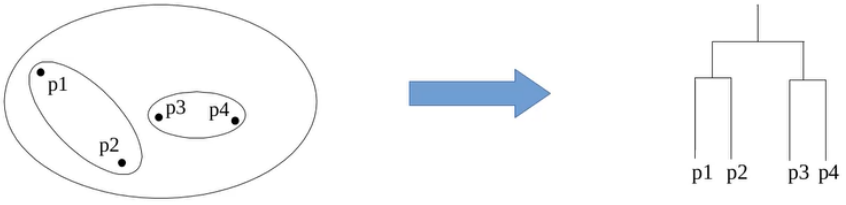
- epsilon, which is the maximum distance that two points can have in order to be considered connected

- minpts (min points), which is the minimum number of neighbors that a point must have in order to be considered a core point

Changing these parameters results in different number of clusters.


## Hierarchical clustering

Hierarchical clustering creates a set of nested clusters organized as a hierarchical tree; this means that a cluster can be connected with other clusters. We can visualize the clusters tree through a dendogram.




This is useful because any desired number of clusters can be obtained by “cutting” the dendogram at the proper level. This can be useful when you have some taxonomies, since different levels may correspond to meaningful taxonomies. The key operation of this algorithm is the computation of the proximity of two clusters.

# Evaluating/comparing clustering

## Measures of cluster validity

We can measure the validity of a cluster using:

- internal indexes, which express the goodness of the structure of a cluster without external information (silhouette index, sum of squared error,...)
- external or relative indexes, which compare the assignments of soft and hard clustering algorithms with some assignments of other clustering algorithms of with some labels (entropy, purity, rand-index,...)

## Silhouette

Silhouette measures consistency within clusters of data and tells how similar a data point is to its own cluster (this is cohesion) compared to other clusters (this is separation). The Silhouette score is defined for each sample and it is composed of two scores:

- the mean distance a between a sample and all other points in the same cluster
- the mean distance b between a sample and all other points in the next nearest cluster

$$s = \frac{b - a}{max(a, b)}$$

The Silhouette for a set of sample is given as the mean of the Silhouette for each sample.

Its value can go from -1 to +1, where a high value means that the object is well matched to its own cluster and poorly matched to neighboring clusters, while scores around 0 indicate overlapping clusters. If most of the objects have high values for the silhouette scores, it means that the clustering configuration is good.

You can compute the silhouette with any distance metric, and the average silhouette over all data of a cluster measures how tightly grouped all the data in the cluster are, while the average silhouette over all data of the dataset measures how appropriately the data has been clustered.

## Rand index

Rand index (RI) measures the similarity of two assignments. Given a dataset D and two partitions (groups of clusters) S and R, we can compute the RI as

$$RI(S, R) = \frac{a + b}{\binom{m}{2}}$$

where a is the number of pairs of elements in D that are in the same subset in S and in the same subset in R, and b is the number of pairs of elements in D that are in a different subset in S and in a different subset in R.

All the possible pairs of element, if D is the binomial coefficient, is

$$\binom{m}{2} = \frac{m(m-1)}{2}$$

## Adjusted Rand Index

Since the RI can be interpreted as accuracy and does not ensure to obtain a values close to 0.0 for a random labelling, the Adjusted Rand Index was introduced to correct for chance and give such a baseline

$$ARI(S,R) = \frac{RI(S,R) - E[RI]}{\max(RI) - E[RI]}$$

The maximum RI and the expected value are computed considering the number and size of clusters ad in S and R, and all random clusterings are generated by shuffling the elements.

# Final considerations

With clustering we want to assign a cluster y to a data point starting from its features x. Creating a model by learning an hypothesis such that h(x) minimizes an empirical risk over data points, and using a loss function which tries to quantify how good is h(x) with respect to the whole data D, which is the clustering error.

Even if not common, it is still possible to use a validation/test set for assessing how data generalize, stability of the clusters and tune hyperparmeters.