

# Clustering algorithms (*Algorithmes de partitionnement*)

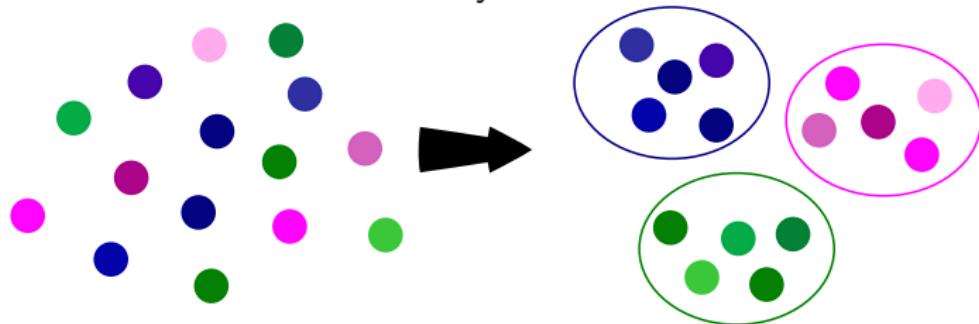
Diane Lingrand and many contributors



Master DSAI M1

2021 - 2022

- **Unsupervised learning** : no class label needed.
- Find the class labels directly from the data.



- Cluster : a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
  - Need a way to calculate object similarity/distance
- Typical applications :
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

- Marketing : Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- Land use : Identification of areas of similar land use in an earth observation database
- Insurance : Identifying groups of motor insurance policy holders with a high average claim cost
- City-planning : Identifying groups of houses according to their house type, value, and geographical location
- web : Cluster Web log data to discover groups of similar access patterns
- Earth-quake studies : Observed earth quake epicenters should be clustered along continent faults

# What Is Good Clustering ?

- A good clustering method will produce high quality clusters with
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering result depends on both the similarity measure used by the method and its implementation
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns

# Outline

- images from six classes :
  - muffin,
  - bagel,
  - fried chicken,
  - chihuahua,
  - labradoodle and
  - puppy
- How would you do a binary clustering of these images ?

- images from six classes :
  - muffin,
  - bagel,
  - fried chicken,
  - chihuahua,
  - labradoodle and
  - puppy
- How would you do a binary clustering of these images ?
  - food : muffin, bagel and fried chicken
  - pet : chihuahua, labradoodle and puppy

# Similarity : an example

- food : muffin, bagel and fried chicken
- pet : chihuahua, labradoodle and puppy



The distance measures the similarity between data.

- Types of Data in Clustering :
  - Interval-scaled variables
  - Binary variables
  - Nominal, ordinal, and ratio variables
  - Variables of mixed types

- Standardize data

- Calculate the mean absolute deviation :  $s_j = \frac{1}{n} \sum_{i=1}^n |x_{ij} - \mu_j|$  where

$$\mu_j = \frac{1}{n} \sum_{i=1}^n x_{ij}$$

- Calculate the standardized measurement (z-score) :  $z_{ij} = \frac{x_{ij} - \mu_j}{s_j}$

- Using mean absolute deviation is more robust than using standard deviation
- A classical example ( $\mu_{age} = 60$  ;  $\mu_{salary} = 11074$  ;  $s_{age} = 5$  ;  $s_{salary} = 37$ ) :

	age	salary
alice	50	11000
bob	70	11100
charly	60	11122
dany	60	11074

⇒

	age	salary
alice	-2	-2
bob	2	0.7
charly	0	1.3
dany	0	0

# Why standardizing data ?

	age	salary
alice	50	11000
bob	70	11100
charly	60	11122
dany	60	11074

Manhattan distance	alice	bob	charly	dany
alice	0	120	132	84
bob	120	0	32	36
charly	132	32	0	48
dany	84	36	48	0

# Why standardizing data ?

	age	salary
alice	-2	-2
bob	2	0.7
charly	0	1.3
dany	0	0

Manhattan distance	alice	bob	charly	dany
alice	0	6.7	5.3	4
bob	6.7	0	2.6	2.7
charly	5.3	2.6	0	1.3
dany	4	2.7	1.3	0

- Minkowski distance

$$d_q(i, j) = \sqrt[q]{\sum_{k=1}^m |x_{ik} - x_{jk}|^q}$$

- if  $q = 1$  : Manhattan distance

$$d_1(i, j) = \sum_{k=1}^m |x_{ik} - x_{jk}|$$

- if  $q = 2$  : Euclidean distance

$$d_2(i, j) = \sqrt{\sum_{k=1}^m |x_{ik} - x_{jk}|^2}$$

- if  $q = \infty$  : Max. distance

# binary variables

- A contingency table for binary data :

		Obj. $j$	
		0	1
Obj.	0	a	b
	1	c	d

- Example :  $x_i = (0, 1, 1, 0, 0, 1)$  and  $x_j = (0, 0, 1, 1, 1, 0)$ .  
 $a = 1, b = 2, c = 2, d = 1$
- distance for symmetric binary variables (e.g. male/female) :

$$d(i, j) = \frac{b + c}{a + b + c + d}$$

- distance for asymmetric binary variables (1 assigned to less frequent class, e.g. Nobel prize laureate) :

$$d(i, j) = \frac{b + c}{a + b + c}$$

# binary variables : an example

Name	Gender	Fever	Cough	Test 1	Test 2	Test 3	Test 4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	Y	N	N	N	N

- Gender is a symmetric attribute
- For assymetric attributes, let Y and P be set to 1, N set to 0
- Lets compute the distance on asymmetric values :
  - $d(Jack, Mary) = \frac{0+1}{2+0+1} = 0.33$
  - $d(Jack, Jim) = \frac{1+1}{1+1+1} = 0.67$
  - $d(Jim, Mary) = \frac{1+2}{1+1+2} = 0.75$
  - Jack and Mary may have similar illness

- generalisation of binary variables (more than 2 values)
  - e.g. : flavour ∈ vanilla, strawberry, chocolate
- Two methods :
  - Simple matching :  $h$  : number of matches,  $m$  : total number of variables

$$d(i, j) = \frac{m - h}{m}$$

- Use binary variables : create a new binary variable for each of the nominal states
  - `flavour=='vanilla'`
  - `flavour=='strawberry'`
  - `flavour=='chocolate'`

- An ordinal variable can be discrete or continuous
- Order is important, e.g., rank
- Can be treated like interval-scaled
  - replace  $x_{ik}$  by their rank  $r_{ik} \in 1 \dots M_k$
  - map the range of each variable onto  $[0, 1]$  by replacing i-th object in the f-th variable by
$$z_{ik} = \frac{r_{ik} - 1}{M_k - 1}$$
  - compute the dissimilarity using methods for interval-scaled variables

# Variables of mixed types

- A database may contain all the types of variables
- One may use a weighted formula to combine their effects :

$$d(i, j) = \frac{\sum_{k=1}^m \delta_{ij}^{(k)} d_{ij}^{(k)}}{\sum_{k=1}^m \delta_{ij}^{(k)}}$$

- Column  $k$  is binary or nominal :  $d_{ij}(k) = 0$  if  $x_{ik} = x_{jk}$  ,  $d_{ij}(k) = 1$  otherwise
- Column  $k$  is interval-based : use the normalized distance
- Column  $k$  is ordinal or ratio-scaled
  - compute ranks  $r_{ik}$  and
  - and treat  $z_{ik}$  as interval-scaled

$$z_{ik} = \frac{r_{ik} - 1}{M_k - 1}$$

- Distance = effort needed for the transformation of one sample to another one



Substitution: hat, hair, mouth  
Insertion: flowers

distance = 4



Deletion: hair  
Substitution: hat  
Insertion: shoes  
distance = 3



Deletion: shoes  
Substitution: mouth  
Insertion: flowers, hair  
distance = 4

- Transformation of strings using only Substitution, Insertion and Deletion
  - Eg : Bad - Gad - Gaod - Good (2 substitutions and 1 insertion)
- Similarity between 2 strings : the cost of the cheapest transformation
  - Eg : 3 for 'Bad' and 'Good'
  - Eg : Piotr, Pyotr, Petros, Pietro, Pedro, Pierre, Piero, Peter
- How to find the cheapest ?

- Single linkage : smallest distance between an element in one cluster and an element in the other, i.e.,  $d(K_p, K_q) = \min_{x_i \in K_p, x_j \in K_q} d(x_i, x_j)$
- Complete linkage : largest distance between an element in one cluster and an element in the other, i.e.,  $d(K_p, K_q) = \max_{x_i \in K_p, x_j \in K_q} d(x_i, x_j)$
- Average linkage : average distance between an element in one cluster and an element in the other
- Centroid : distance between the centroids of two clusters, i.e.,  $d(K_p, K_q) = d(C_p, C_q)$
- Medoid : distance between the medoids of two clusters, i.e.,  $d(K_p, K_q) = d(M_p, M_q)$ 
  - Medoid : one chosen, centrally located object in the cluster

# Centroid, Radius and Diameter

**Centroid** : the “midpoint” of a cluster

$$C_p = \frac{1}{N_p} \sum_{i=1}^{N_p} x_i^{(p)}$$

**Radius** : square root of average distance from any point of the cluster to its centroid

$$R_p = \sqrt{\frac{1}{N_p} \sum_{i=1}^{N_p} (x_i^{(p)} - C_p)^2}$$

**Diameter** : square root of average mean squared distance between all pairs of points in the cluster

$$D_p = \sqrt{\frac{1}{N_p(N_p - 1)} \sum_{i=1}^{N_p} \sum_{j \neq i} (x_i^{(p)} - x_j^{(p)})^2}$$

- Partitioning (iterative construction of partitions)
  - K-Means, k-Medoids, etc.
- Hierarchical (construct a dendrogram of instances)
  - Diana, Agnes, BIRCH, ROCK, CAMELEON
- Density-Based (based on connectivity and density function)
  - DBSCAN, OPTICS, DenClue
- Grid-Based
  - STING, WaveCluster, CLIQUE
- Model-Based
  - Expectation Maximization
  - Self-organizing maps (SOM)
- Spectral clustering
- Frequent-Pattern-Based

# Outline

- Partitioning method : Construct a partition of a dataset D of n objects into a set of k clusters, minimizing the sum of squared distances :

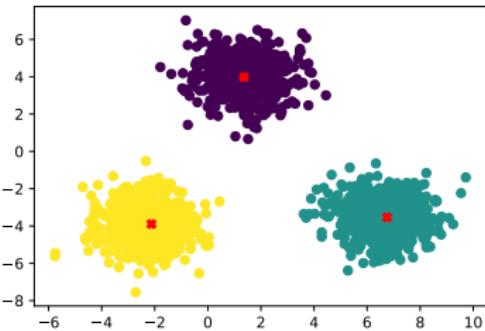
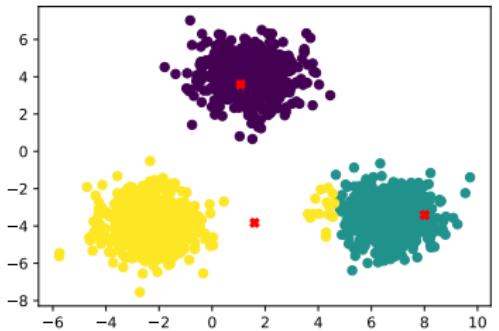
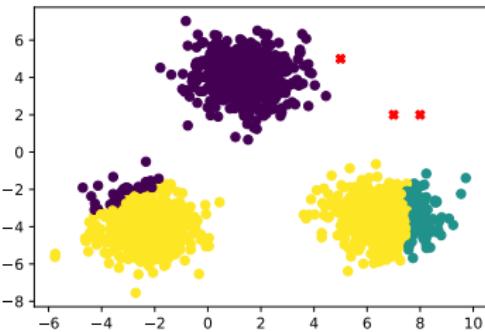
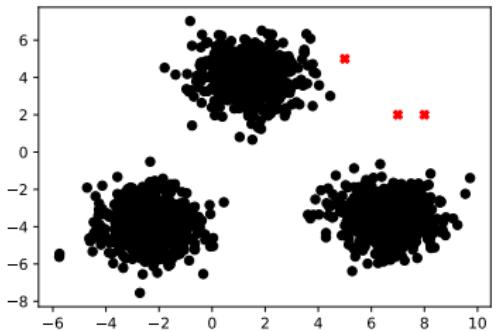
$$\sum_{p=1}^k \sum_{i=1}^{N_p} (x_i^{(p)} - C_p)^2$$

- Given a k, find a partition of k clusters that optimizes the chosen partitioning criterion
  - Global optimal : exhaustively enumerate all partitions
  - Heuristic methods : k-means and k-medoids algorithms
    - k-means (MacQueen '67) : Each cluster is represented by the centroid of the cluster
    - k-medoids or PAM (Partition around medoids) (Kaufman and Rousseeuw '87) : Each cluster is represented by one of the objects in the cluster

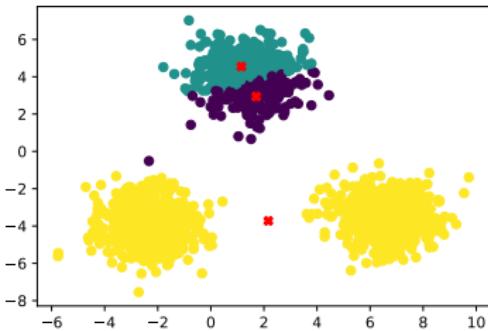
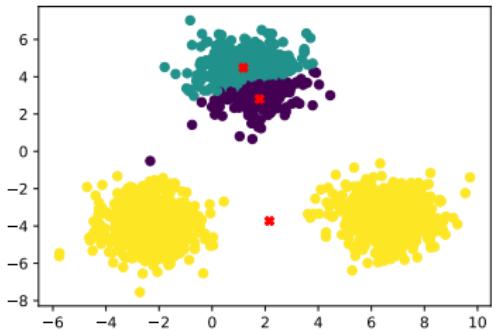
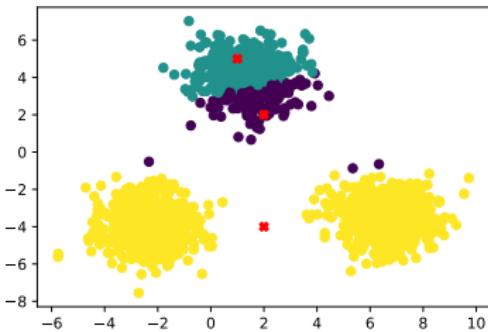
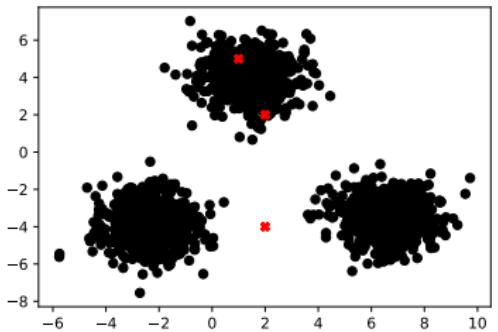
Given  $k$ , the k-means algorithm is implemented in four steps :

- ① Choose  $k$  initial centroids (could be samples), each labeled as  $k$
- ② Assign each object to the cluster having the nearest centroid point
- ③ Compute centroids of the clusters of the current partition (the centroid is the center, i.e., mean point, of the cluster)
- ④ Go back to Step 2, stop when no more change

# Example of k-means

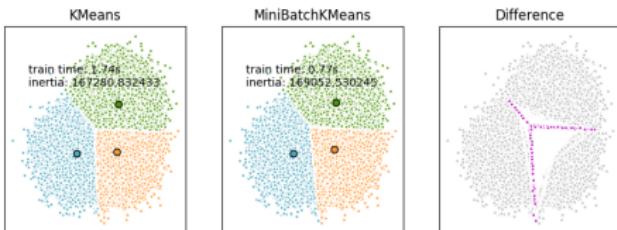


# Example of k-means : bad initialisation

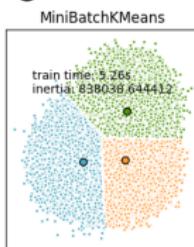


- Strength : Relatively efficient :  $O(tkn)$ , where  $n$  nb. of objects,  $k$  nb. of clusters, and  $t$  nb. of iterations. Normally,  $k, t \ll n$ .
  - For comparison : PAM :  $O(k(n-k)^2)$ , CLARA :  $O(ks^2 + k(n-k))$
- Comment : Often terminates at a local optimum.
  - The global optimum may be found using optimization methods such as : simulated annealing and evolutionary algorithms.
  - In practice : several trials using aleatory initialisations and some heuristics.
- Weaknesses :
  - Applicable only when mean is defined, then what about categorical data ?
  - Need to specify  $k$ , the number of clusters, in advance
    - or use the Elbow method
  - Unable to handle noisy data and outliers
  - Not suitable to discover clusters with non-convex shapes

- a variant of k-mean for scalability improving
- mini-batch = subset of dataset
  - randomly sampled at each iteration of the training process
- results almost as good as the standard algorithm
  - code from [https://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_mini\\_batch\\_kmeans.html#sphx-glr-auto-examples-cluster-plot-mini-batch-kmeans-py](https://scikit-learn.org/stable/auto_examples/cluster/plot_mini_batch_kmeans.html#sphx-glr-auto-examples-cluster-plot-mini-batch-kmeans-py) with modification of the number of data to 200000 (upper limit to have k-means running).

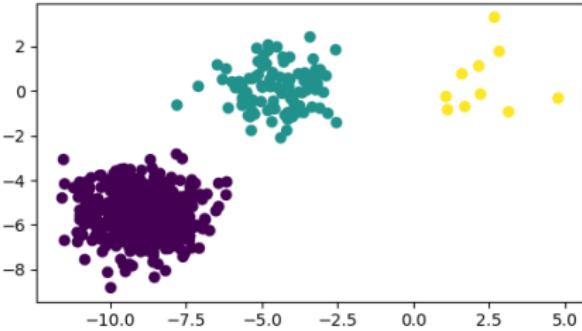
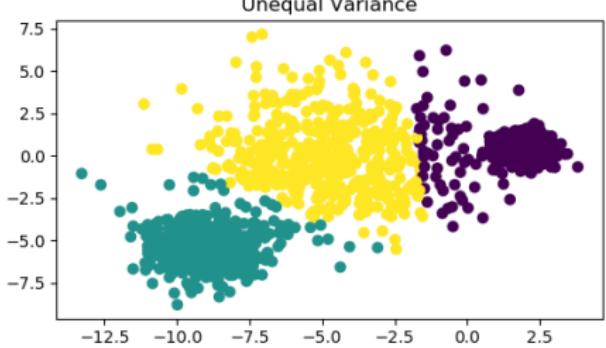
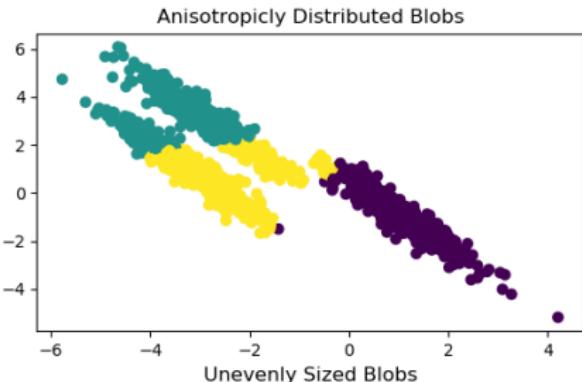
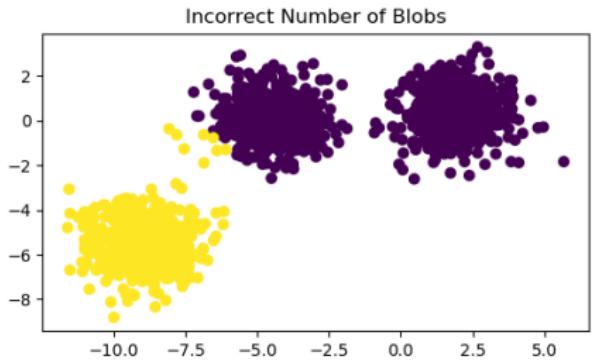


- Only with mini-batch and using 1 million of data :



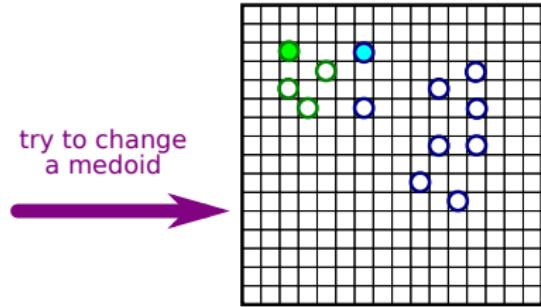
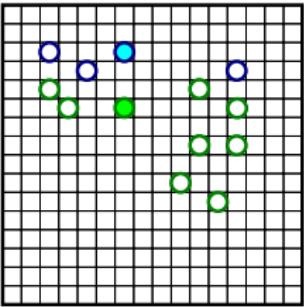
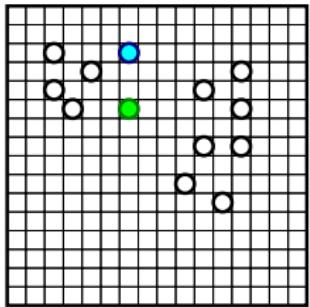
# Demonstration of k-means assumptions

[https://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_kmeans\\_assumptions.html](https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_assumptions.html)

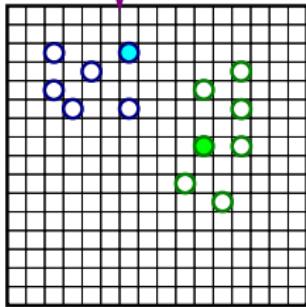


- Find representative objects, called medoids, in clusters
- PAM (Partitioning Around Medoids, 1987)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - less sensitive to outliers than k-mean
  - PAM works effectively for small data sets, but does not scale well for large data sets
- CLARA (Clustering LARge Applications, Kaufmann & Rousseeuw, 1990) : PAM on several subsets of samples (take the best clustering)
- CLARANS (Clustering Large Applications based upon RANdomized Search, Ng & Han, 1994) : Randomized sampling. Graph of solutions (=set of centroids).

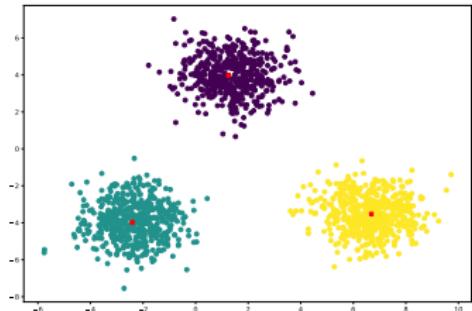
# k-medoids : an example



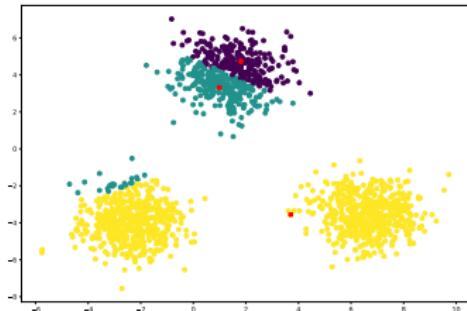
keep this change  
and try other changes  
until no possible change



# k-medoids : previous example



good initialization



bad initialization

- A fuzzy extension of the k-means algorithm
- A record can belong to more than one cluster to a degree (soft vs hard assignment) :
  - how much  $x_i$  belongs to cluster  $k$  :  $0 \leq \mu_k(x_i) \leq 1$
  - constraint ( $c$  the nb. of clusters) :  $\sum_{k=1}^c \mu_k(x_i) = 1$
  - objective function ( $v_k$  represents the cluster  $k$ ) :

$$\min \sum_{k=1}^c \sum_{i=1}^N \mu_k(x_i) d(x_i, v_k)$$

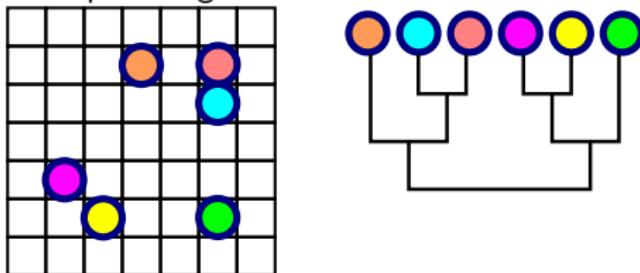
- play with simple data (slides 25 or 26)

```
from sklearn.datasets import make_blobs
n_samples = 1500
random_state = 160
nb = 3 # number of blobs
X, y = make_blobs(n_samples=n_samples, centers=nb, random_state=random_state)
```

- try different initialisations, number of points, ...
- play in higher dimension
  - choose a dataset of higher dimension : Iris (4) or MNIST/FMNIST (784)
  - test different algorithms from the family of k-means
  - visualize the results using dimension reduction to 2 (t-SNE)

# Outline

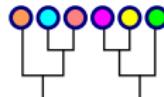
- Input : distance matrix - Output : a dendrogram (tree of clusters)
- This method does not require the number of clusters  $k$  as an input, but may need a termination condition
- Two approaches :
  - Bottom up approach = agglomerative
    - Example using Manhattan distance and complete linkage



- Top down approach (DIANA = DIvide ANALysis)
  - start with all samples in one cluster. Hierarchical division of clusters related to a dispersion criterion (divide into clusters where closest samples are the more distant ones).

# Agglomerative hierarchical clustering methods

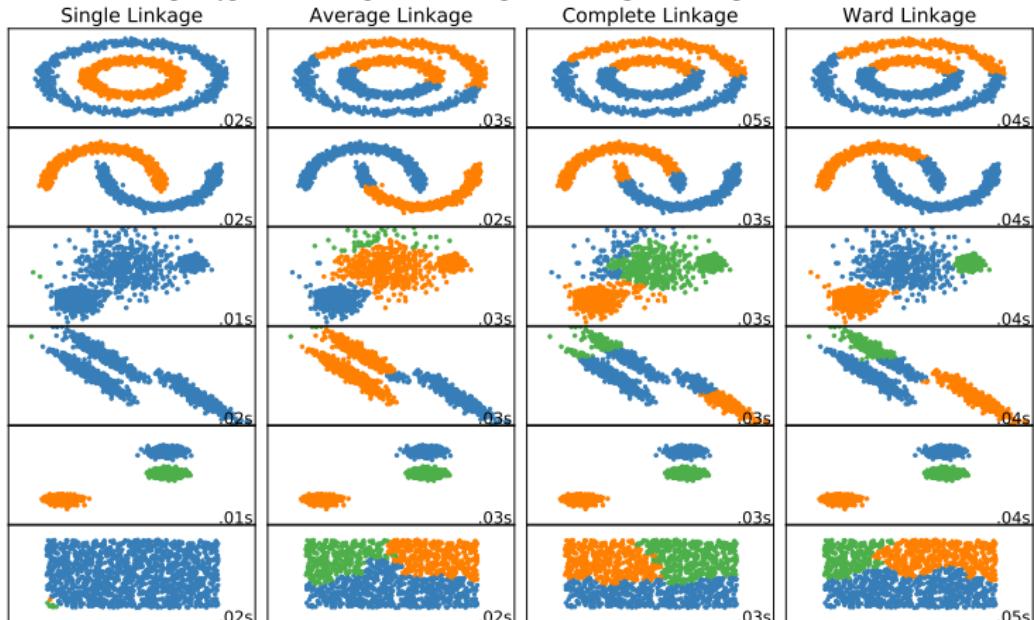
- Algorithm :
  - starts with all single data as their own cluster
  - merged clusters together according to a linkage criteria
  - ends when all data are in the same cluster
- The linkage criteria determines the metric used for the merge strategy : (from <https://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering>)
  - Ward** minimizes the sum of squared differences within all clusters. It is a variance-minimizing approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.
  - Maximum or complete linkage** minimizes the maximum distance between observations of pairs of clusters.
  - Average linkage** minimizes the average of the distances between all observations of pairs of clusters.
  - Single linkage** minimizes the distance between the closest observations of pairs of clusters.



# Impact of the linkage criteria

From <https://scikit-learn.org/stable/modules/clustering.html#>

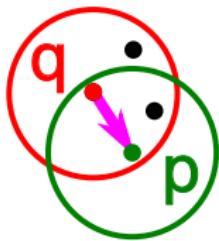
different-linkage-type-ward-complete-average-and-single-linkage.



# Outline

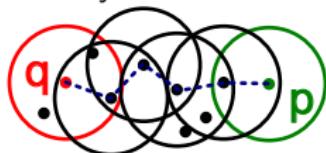
- Clusters : areas of high density separated by areas of low density
- Clustering based on density (local cluster criterion), such as density-connected points
- Major features :
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition
- Examples of density-based methods :
  - DBSCAN, OPTICS, DENCLUE, CLIQUE

- Two parameters :
  - $\epsilon$  : Maximum radius of the neighbourhood
  - MinPts : Minimum number of points in an  $\epsilon$ -neighbourhood of that point
- Neighborhood :  $N_\epsilon(p) = \{q \in D | d(p, q) \leq \epsilon\}$
- Core point : if at least MinPts points are within distance  $\epsilon$  of  $p$  (including  $p$ ).
- Directly density-reachable : A point  $p$  is directly density-reachable from a point  $q$  w.r.t.  $\epsilon$ , MinPts if :
  - $p$  belongs to  $N_\epsilon(q)$
  - core point condition :  $|N_\epsilon(q)| \geq \text{MinPts}$



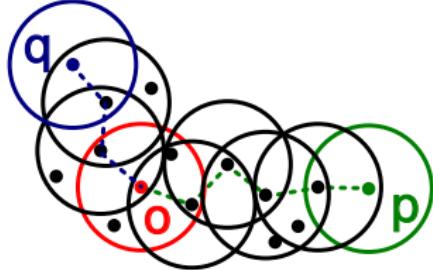
- Density-Reachable :

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

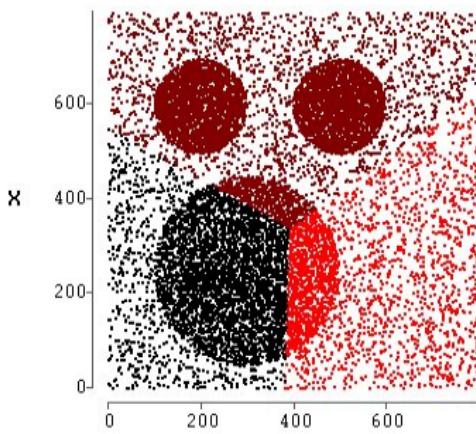


- Density-Connected :

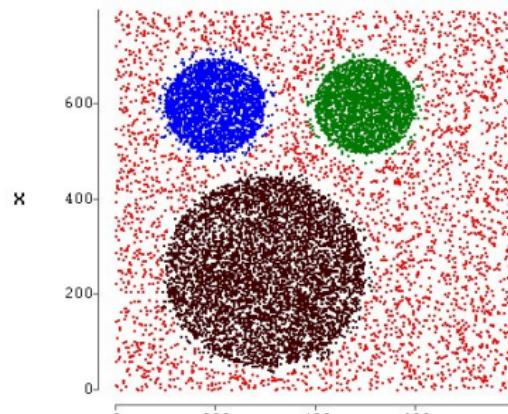
- A point  $p$  is density-connected to a point  $q$  w.r.t.  $\epsilon$ , MinPts if there is a point  $o$  such that both,  $p$  and  $q$  are density-reachable from  $o$  w.r.t.  $\epsilon$  and MinPts



- choose a core point  $p$ , not already assigned to a cluster
- add all density-reachable points from  $p$  to this cluster
- go back to step 1 until no more points without a cluster
- all points not reachable from any other point are outliers or noise points



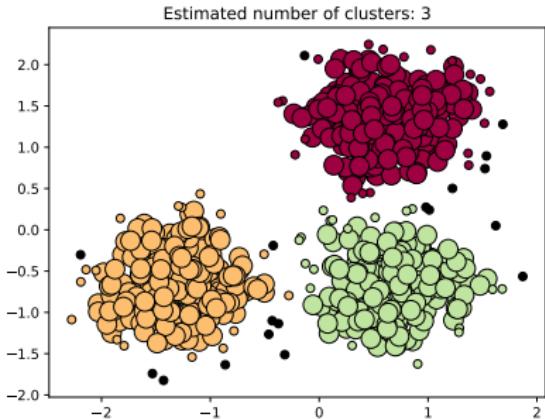
using k-means



using DBSCAN

# DBSCAN in scikit-learn

```
from https://scikit-learn.org/stable/auto_examples/cluster/plot_dbSCAN.html#  
sphx-glr-download-auto-examples-cluster-plot-dbscan-py
```



```
from sklearn.cluster import DBSCAN  
db = DBSCAN(eps=0.3, min_samples=10)
```

# Outline

- A popular probability-based iterative refinement algorithm
- Based on the assumption that the distribution of samples is a mixture of gaussians (GMM)
  - likelihood of  $x$  knowing its gaussian distribution  $(\mu, \Sigma)$  :

$$p(x|\mu, \Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{|\Sigma|}} \exp \left\{ -\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) \right\}$$

- likelihood of  $x$  knowing the GMM (the  $K$ ,  $\mu_k$ ,  $\Sigma_k$  and  $w_k$ ) with

$$\forall k w_k > 0 \text{ and } \sum_{k=1}^K w_k = 1 :$$

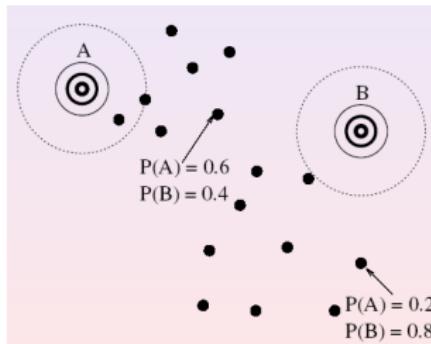
$$p(x) = \sum_{k=1}^K w_k p(x|\mu_k, \Sigma_k) = \sum_{k=1}^K p(k)p(x|k)$$

- Learning the GMM : Expectation Maximisation
- `sklearn : mixture.GaussianMixture`

- A 3 steps algorithm :
  - step 1 : Initialization
  - step 2 : Expectation (compute probability for each sample of belonging to each gaussian)
  - step 3 : Maximisation (compute parameters of the GMM using result of step 2)
- Loop steps 2 and 3 until convergence
- The larger the log likelihood, the better the model fits the data

- estimate the posterior probability of cluster  $k$  for each  $x_i$
- for each  $x_i$  estimate the probability of belonging to each gaussian :  
 $p(k|x_i)$  :

$$p(k|x_i) = \frac{p(x_i|k)p(k)}{p(x_i)}$$



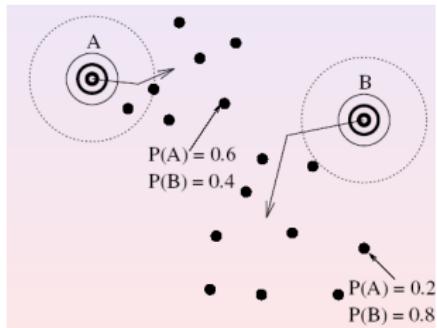
# Maximisation

- maximisation of the log of likelihood (partial derivatives equal to 0)
- update of the parameters  $\mu_k$ ,  $\Sigma_k$  and  $w_k$  :

$$\mu_k = \frac{\sum_{i=1}^n p(k|x_i)x_i}{\sum_{i=1}^n p(k|x_i)}$$

$$\Sigma_k^2 = \frac{\sum_{i=1}^n p(k|x_i)(x_i - \mu_k)^2}{\sum_{i=1}^n p(k|x_i)}$$

$$w_k = \frac{1}{n} \sum_{i=1}^n p(k|x_i)$$



# Outline

- Affinity matrix representation of samples :  $A_{ij}$  is high if samples  $x_i$  and  $x_j$  are close.
  - could be measured using truncated gaussian :
    - if  $\|x_j - x_i\| > R$  :  $A_{ij} = 0$
    - else  $A_{ij} = \exp(-\alpha \|x_j - x_i\|^2)$
- Eigenvalues and eigenvectors decomposition  $\Rightarrow$  space reduction using eigenvectors corresponding to highest eigenvalues
- Clustering of points in new dimension
- only for small number of clusters

# Outline

# Huge number of data

- use a canopy clustering as pre-clustering
  - algorithm using 2 thresholds :  $T_1$  and  $T_2$ , a set of samples  $S$

take one element  $e$  of  $S$  and starts a new canopy

**for** all element  $s$  of  $S$  **do**

**if**  $d(e, s) < T_1$  **then**

        | assign  $s$  to the current canopy

**end**

**if**  $d(e, s) < T_2$  **then**

        | remove  $s$  from  $S$

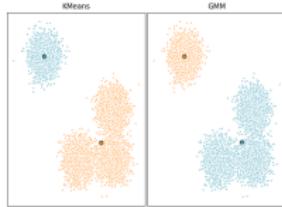
**end**

**end**

repeat until  $S$  empty

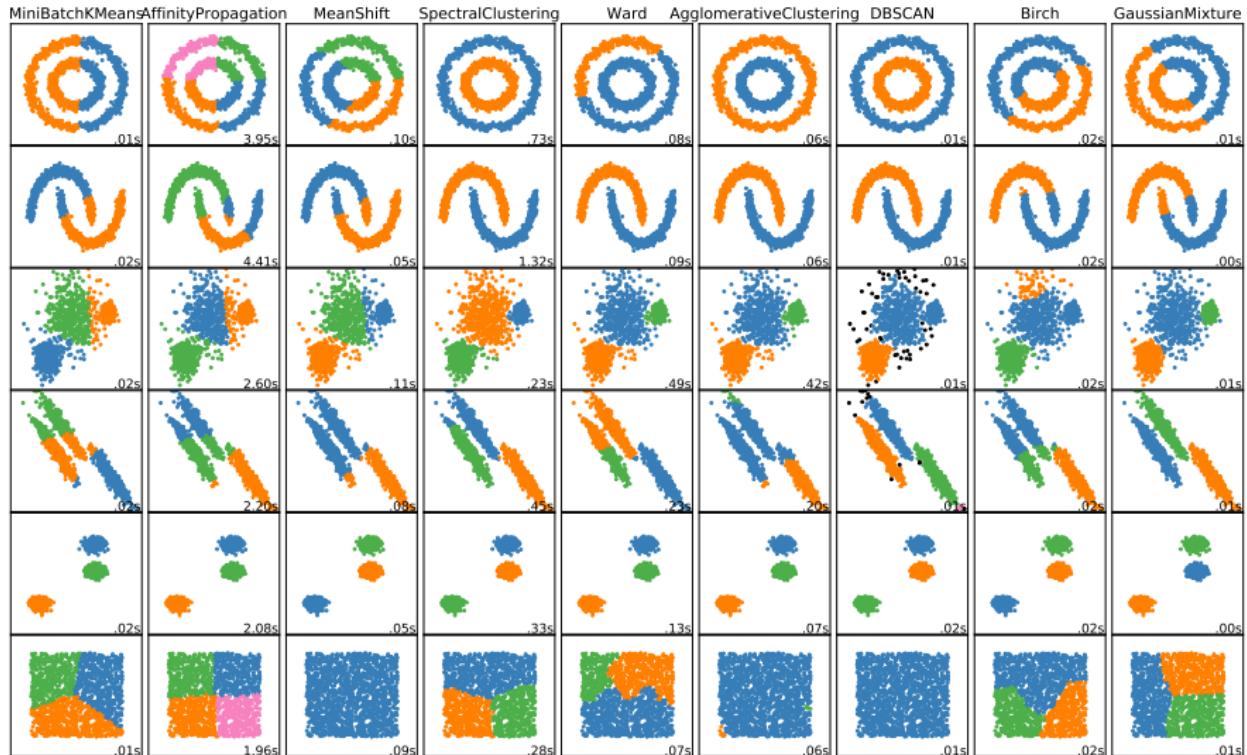
- one sample may belongs to several canopies
- refine the results using a clustering algorithm
- sample the data
- mini batch

- curse of dimensionality
  - volume of hypersphere << volume of hypercube
  - distance from center to corner of hyper cube grows without limit
- reducing the dimension
  - PCA, ...



- Example : k-means and GMM
- Comparison between k-means and GMM
  - increase the size of the dataset
  - increase the dimension of the data
  - increase the number of clusters
  - change the shape of clusters
- try initialisation with k-means and refinement with GMM

# Outline



- ground truth known
  - ARI (Adjusted Rand Index)
  - MI (Mutual Information)
  - V-measure
- ground truth not known
  - Davies-Bouldin index
  - Dunn index
  - Calinski-Harabaz
  - silhouette

# Adjusted Rand Index (ARI)

- value in  $[-1 + 1]$ 
  - good : +1
  - bad : negative or close to 0
  - random label assignments : value close to 0
- no assumption on the data structure
- Rand Index :
  - $n$  : number of data samples
  - $a$  : number of pairs of samples in the same set in C and in K
  - $b$  : number of pairs of samples in different sets in C and in K
  - $C_2^n$  : total number of pairs of samples

$$RI = \frac{a + b}{C_2^n}$$

- Adjusted Rand Index :
  - $E[RI]$  is computed from random clusterings

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

- can be used to measure similarity between 2 clustering results
- sklearn : `metrics.adjusted_rand_score`

- quality of clustering :
  - homogeneity : each cluster contains only samples of a single class
  - completeness : all members of a given class are assigned to the same cluster
- V-measure represents the (weighted) harmonic mean of these 2 quantities
- value in  $[0 + 1]$ 
  - random clustering : be careful if small dataset, high number of clusters
- no assumption on the data structure
- sklearn :
  - `metrics.homogeneity_score`
  - `metrics.completeness_score`
  - `metrics.v_measure_score`

- Internal validation : Separation ? Compactness ?
  - E.g. Dunn, Davies-Bouldin, Calinski-Harabaz, and Silhouette indexes.
  - Problems :
    - Different performance WRT existence of noise, variable densities, and non well-separated clusters.
    - Overrate the algorithm that uses the same clustering model.

number of clusters	average silhouette score	Calinski Harabaz score
2	0.71	1604
3	0.59	1810
4	0.65	2704
5	0.56	2263
6	0.45	2043

# Davies-Bouldin score

- average similarity between each cluster  $C_i$  and its most similar one  $C_j$ .  
In the context of this index, similarity is defined as a measure  $R_{ij}$  that trades off :
  - $s_i$  the average distance between each point of cluster  $i$  and the centroid of that cluster
  - $d_{ij}$  the distance between cluster centroids  $c_i$  and  $c_j$

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

- Davies-Bouldin score :

$$DB = \frac{1}{k} \sum_{i=1}^k \max_{i \neq j} R_{ij}$$

- Zero is the lowest possible score. Values closer to zero indicate a better partition.
- sklearn : `davies_bouldin_score`

For  $k$  clusters, the Calinski-Harabaz score  $s$  is given as the ratio of the between-clusters dispersion mean and the within-cluster dispersion ( $N$  is the size of the dataset) :

$$s(k) = \frac{\text{Tr}(B_k)}{\text{Tr}(W_k)} \frac{N - k}{k - 1}$$

with :

$$W_k = \sum_{q=1}^k \sum_{x \in C_q} (x - c_q)(x - c_q)^T$$

and

$$B_k = \sum_q n_q (c_q - c)(c_q - c)^T$$

$c_q$  is the centroid of cluster  $C_q$  containing  $n_q$  samples.  $c$  is the centroid of the dataset

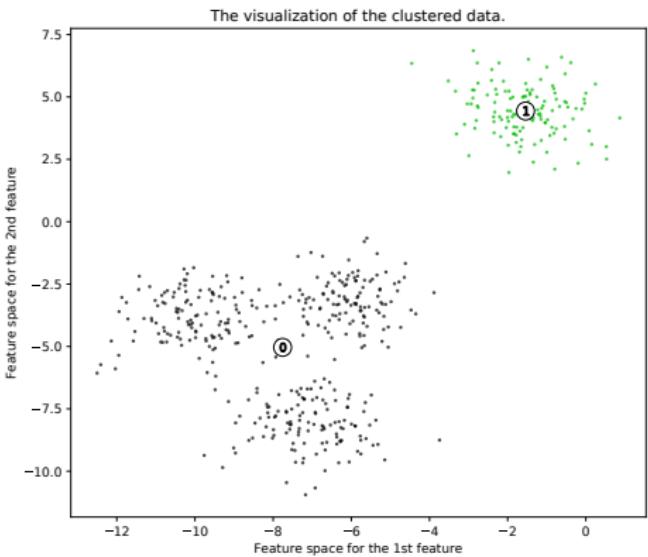
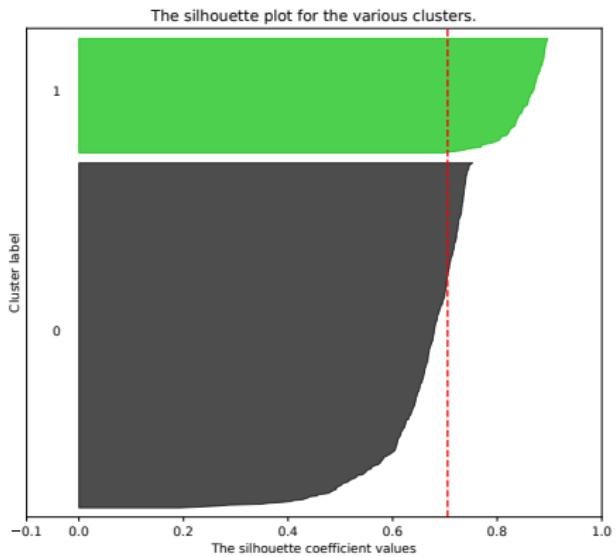
in sklearn : `metrics.calinski_harabasz_score`

- for a single sample : Silhouette coefficient  $s = \frac{b-a}{\max(a,b)}$  with
  - $a$  : mean distance between a sample and all other samples from the same cluster
  - $b$  : mean distance between a sample and all other samples in the next nearest cluster
- for a dataset : mean of Silhouette coefficients for each sample
- sklearn : `metrics.silhouette_score`

# k-means with $k = 2$ and silhouette profiles

inspired by [https://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_kmeans\\_silhouette\\_analysis.html](https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html)

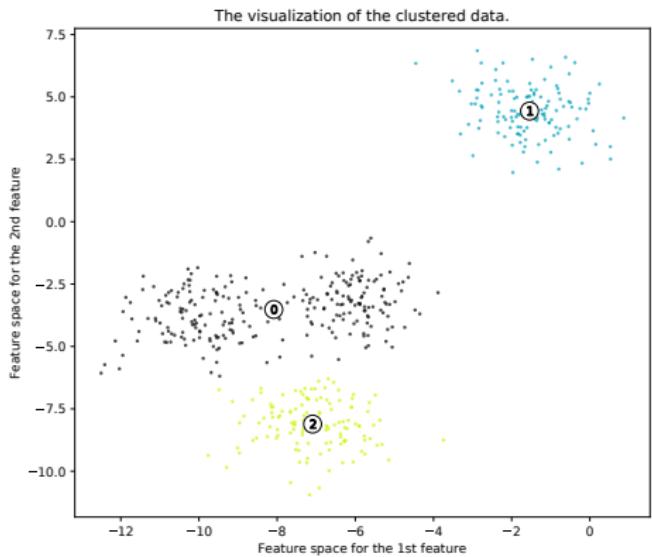
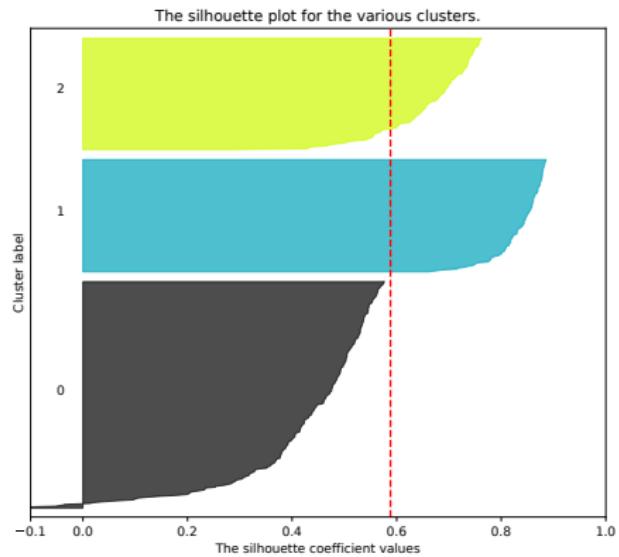
Silhouette analysis for KMeans clustering on sample data with n\_clusters = 2



# k-means with $k = 3$ and silhouette profiles

inspired by [https://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_kmeans\\_silhouette\\_analysis.html](https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html)

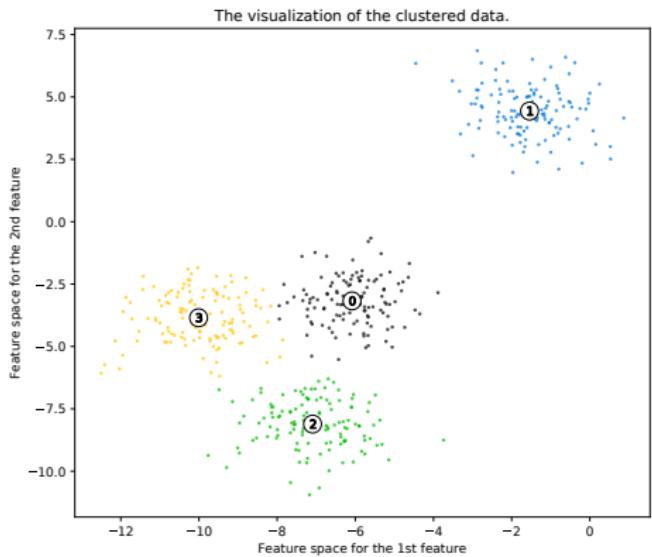
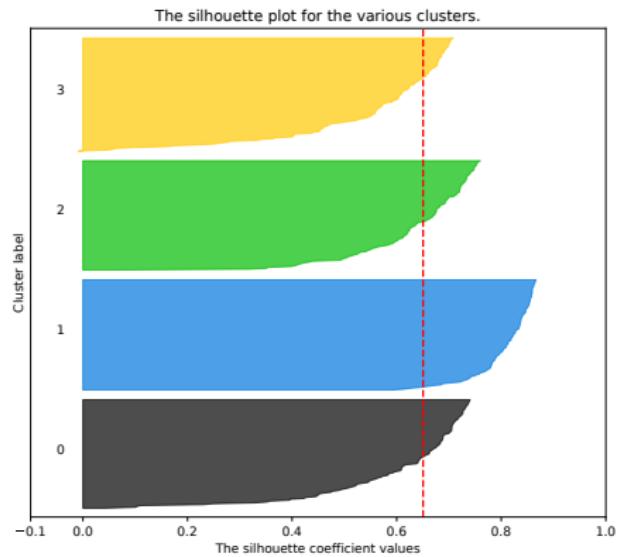
Silhouette analysis for KMeans clustering on sample data with  $n_{clusters} = 3$



# k-means with $k = 4$ and silhouette profiles

inspired by [https://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_kmeans\\_silhouette\\_analysis.html](https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html)

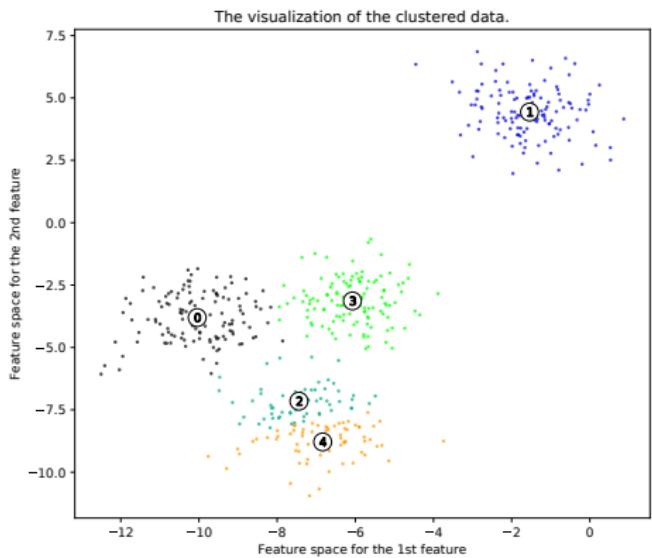
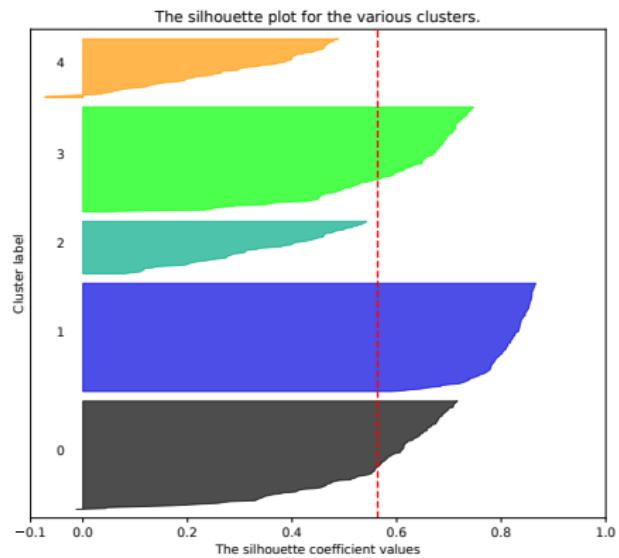
Silhouette analysis for KMeans clustering on sample data with n\_clusters = 4



# k-means with $k = 5$ and silhouette profiles

inspired by [https://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_kmeans\\_silhouette\\_analysis.html](https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html)

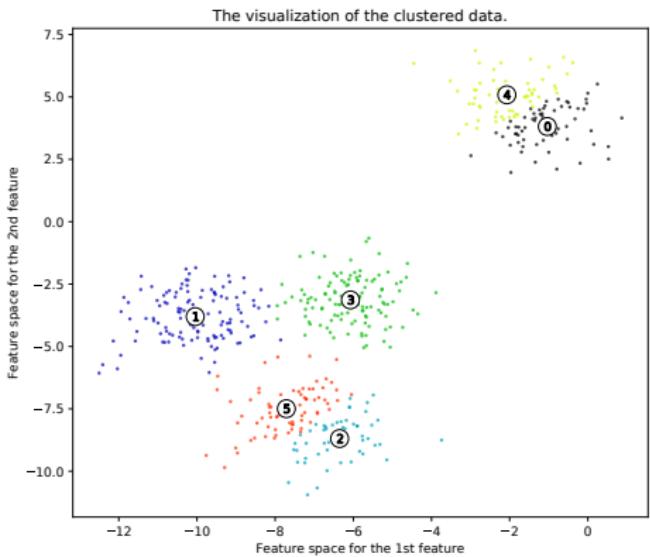
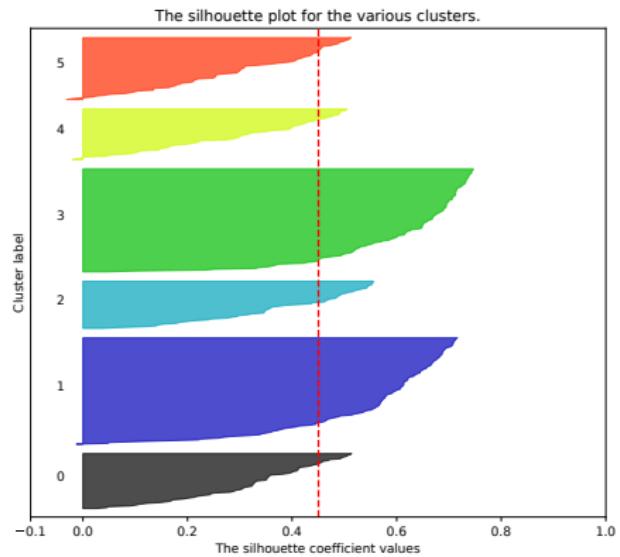
Silhouette analysis for KMeans clustering on sample data with  $n_{clusters} = 5$



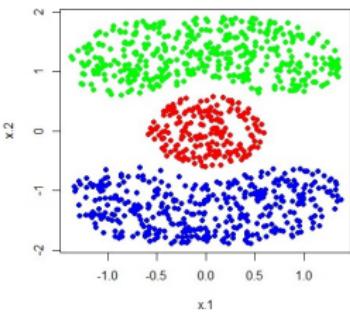
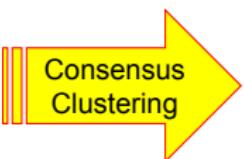
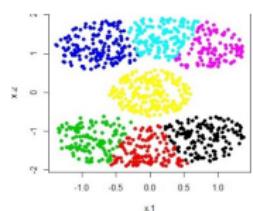
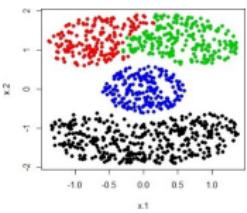
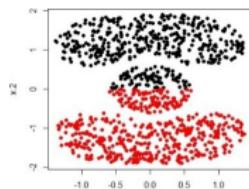
# k-means with $k = 6$ and silhouette profiles

inspired by [https://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_kmeans\\_silhouette\\_analysis.html](https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html)

Silhouette analysis for KMeans clustering on sample data with  $n_{clusters} = 6$



# Consensus clustering



Ensemble of 3 base clusterings

Consensus solution