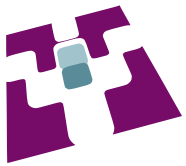
Unsupervised Machine Learning Approaches : Clustering

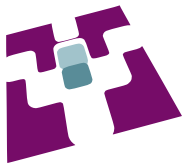
Samir Loudni

TASC (LS2N – CNRS) – DAPI
IMT Atlantique

FIL A3 – cours
10/02/2021



Outline



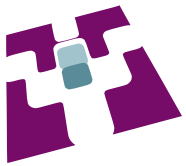
Clustering : An optimization problem

Clustering

Partitioning a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).

Input:

	a_1	a_2	$\dots$	a_n
o_1	$d_{1,1}$	$d_{1,2}$	$\dots$	$d_{1,n}$
o_2	$d_{2,1}$	$d_{2,2}$	$\dots$	$d_{2,n}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
o_m	$d_{m,1}$	$d_{m,2}$	$\dots$	$d_{m,n}$



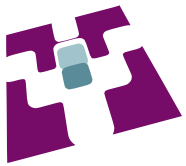
Clustering : An optimization problem

Clustering

Partitioning a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).

Output:

	a_1	a_2	$\dots$	a_n	*cluster*
o_1	$d_{1,1}$	$d_{1,2}$	$\dots$	$d_{1,n}$	c_1
o_2	$d_{2,1}$	$d_{2,2}$	$\dots$	$d_{2,n}$	c_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
o_m	$d_{m,1}$	$d_{m,2}$	$\dots$	$d_{m,n}$	c_1



Clustering: A main task of exploratory data mining

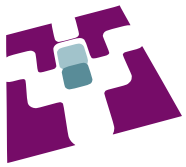
Some of the practical applications of Clustering :

1. **Customer Segmentation**: Finding a group of customers with similar behavior given a large database of customers (a practical example is given using banking customer segmentation).

2. **Classifying network traffic**: Grouping together characteristics of the traffic sources. Traffic types can be easily classified using clusters.

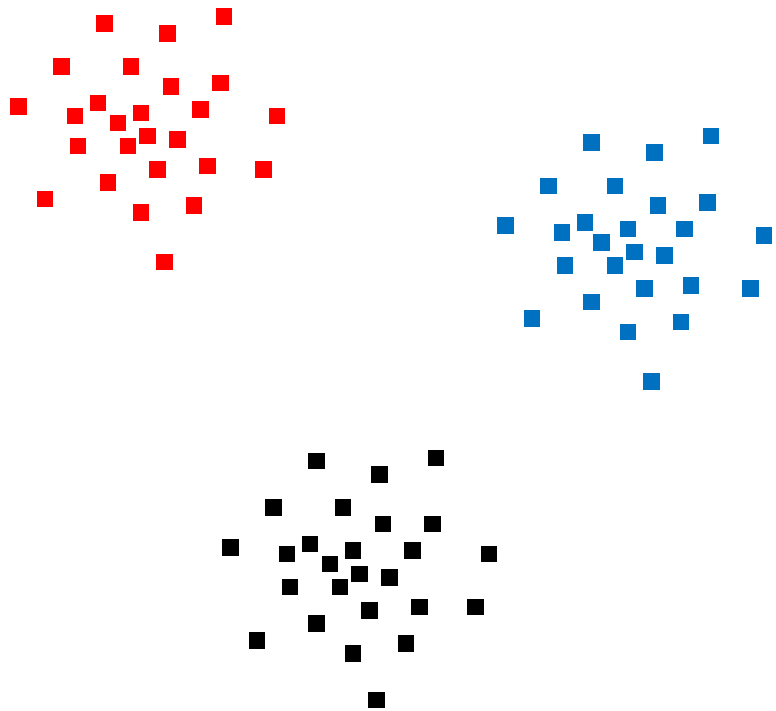
3. **Email Spam filter**: The data is grouped looking at different sections (header, sender, and content) and then can help classify which of them are spam

4. **City-Planning**: Grouping of houses according to their geo-location, value, and house type.



Different types of Clustering Algorithms

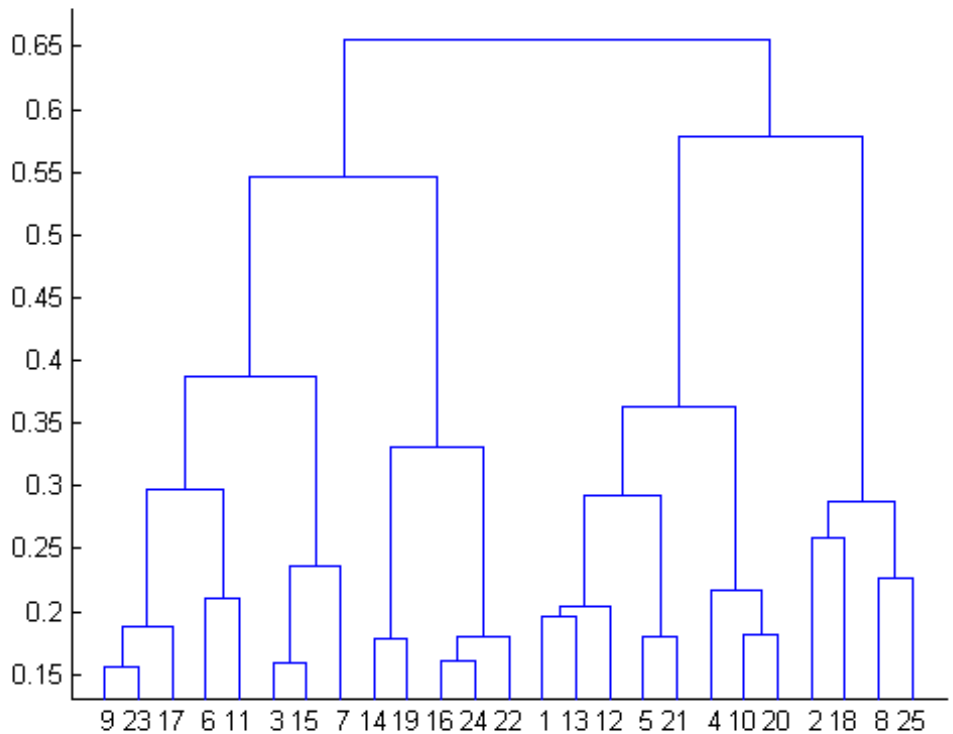
- Representative-based clustering : Given a set of points, partitioning is performed through a similarity measure to some chosen representative points.



➠ Suitable only for compact and well-separated clusters.

Different types of Clustering Algorithms

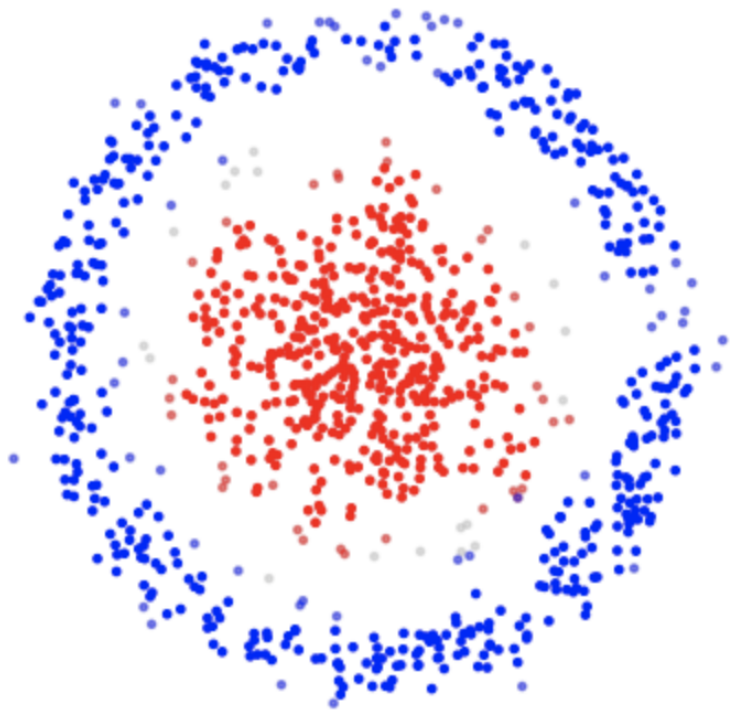
- Hierarchical Clustering : follows an agglomerative approach to group similar points to obtain a hierarchy of clusters, until fused into a single cluster.



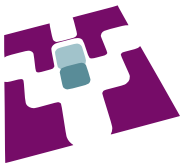
➠ provide richer information about the similarity structure of the data.

Different types of Clustering Algorithms

- Density-based Spatial Clustering : Clusters are seen as dense regions separated by regions that are much less denser (noise).



Useful in the application areas where we require concave cluster structures, purely based on neighborhood density.



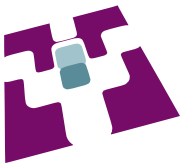
Representative-based Clustering

Given a dataset with n points in a d-dimensional space, $\mathcal{E} = \{\boldsymbol{x}_i\}_{i=1}^n$, and given the number of desired clusters k, the goal of representative-based clustering is to partition the dataset into k groups or clusters, which is called a clustering and is denoted as $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$.

For each cluster C_i there exists a representative point that summarizes the cluster, a common choice being the mean (also called the centroid) $\boldsymbol{\mu}_i$ of all points in the cluster, that is,

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{\boldsymbol{x}_j \in C_i} \boldsymbol{x}_j$$

where $n_i = |C_i|$ is the number of points in cluster C_i.



k-means: Objective

Consider a set $\mathcal{E}$ of N data points described by p variables with values in $\mathbb{R}$ and let d a distance measure over $\mathbb{R}^p$. The sum of squared Euclidean distances[1] is defined as

$$\phi_{\mathcal{E}}(\mathcal{C}) = \sum_{j=1}^{k} \sum_{\boldsymbol{x}_i \in C_j} d^2(\boldsymbol{x}_i, \boldsymbol{\mu}_j)$$

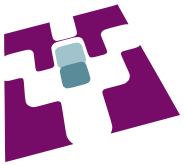
The goal is to find the clustering that minimizes the ϕ function:

$$\mathcal{C}^* = \arg\min_{\mathcal{C}} \{\phi_{\mathcal{E}}(\mathcal{C})\}$$

➠ The lower the value of $\phi_{\mathcal{E}}(\mathcal{C})$, the more "compact" the groups are around their centroids and therefore the better the quality of the partitioning obtained.

➠ Finding a global minimum of the function $\phi_{\mathcal{E}}(\mathcal{C})$ is NP-hard.

[1] intra-class inertias



k-means : Basic steps

k-means is a greedy iterative approach that always converse to a local optima instead of a globally optimal clustering.

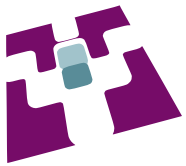
An iteration consists in two steps:

- cluster assignment : each data point is assigned to the cluster whose center is the most similar. That is, each point $\boldsymbol{x}_i$ is assigned to cluster C_{j^*}, where

$$j^* = \arg\min_{j=1}^{k} \left\{ d^2(\boldsymbol{x}_i, \boldsymbol{\mu}_j) \right\}$$

- centroid update : the center of each cluster is updated to the mean of the points assigned to it.

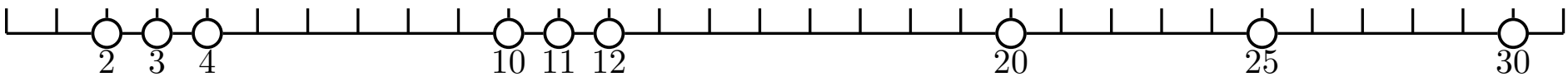
Initially, the centers of the clusters are randomly drawn. The procedure stops when, from an iteration to the next one, the centers of the clusters have not changed much (or at all).

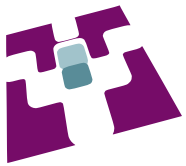


2-means in One Dimension : illustration

2-means clustering of the data points in a one-dimensional space using the Euclidean distance.

Dataset:

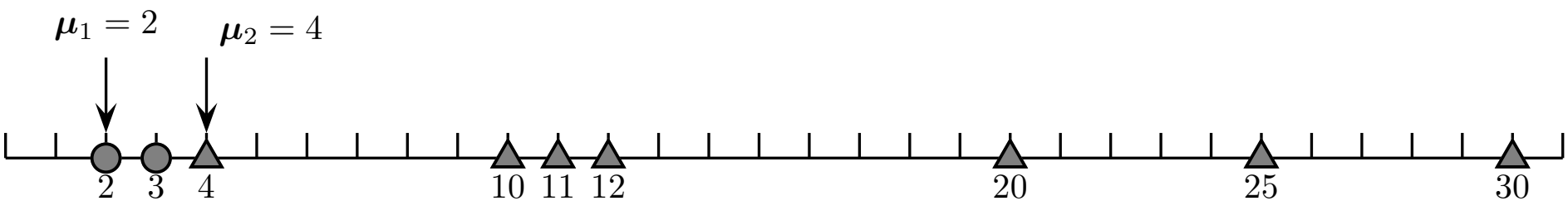


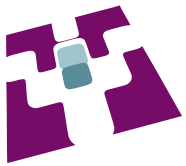


2-means in One Dimension : illustration

2-means clustering of the data points in a one-dimensional space using the Euclidean distance.

Iteration 1:

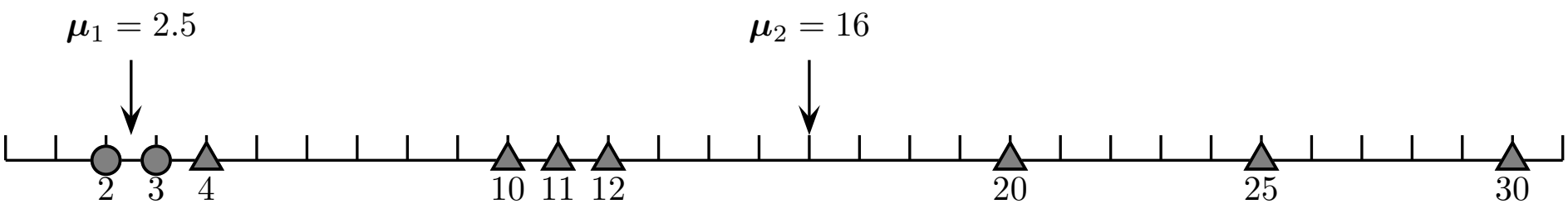


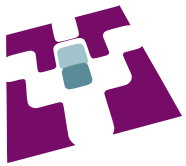


2-means in One Dimension : illustration

2-means clustering of the data points in a one-dimensional space using the Euclidean distance.

Iteration 2:

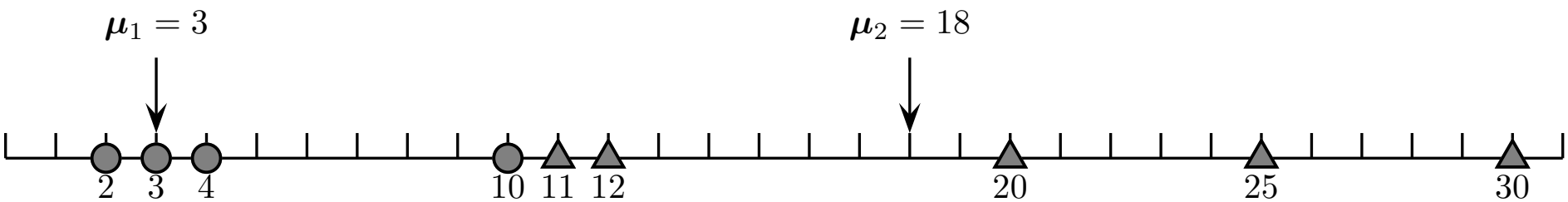


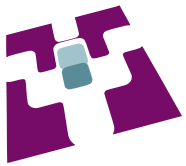


2-means in One Dimension : illustration

2-means clustering of the data points in a one-dimensional space using the Euclidean distance.

Iteration 3:

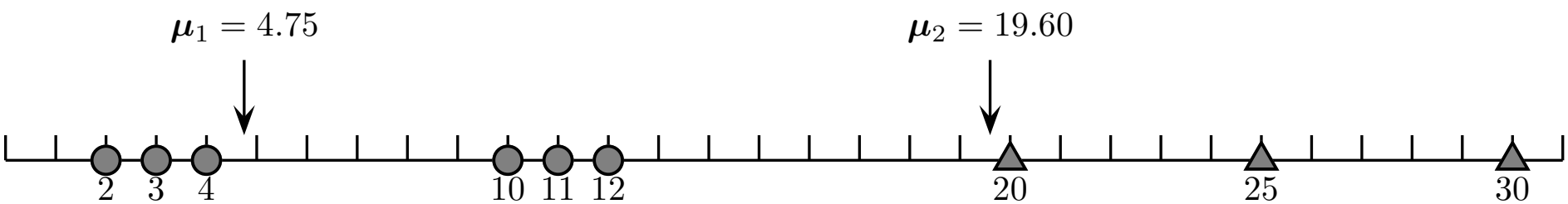


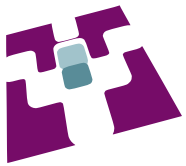


2-means in One Dimension : illustration

2-means clustering of the data points in a one-dimensional space using the Euclidean distance.

Iteration 4:

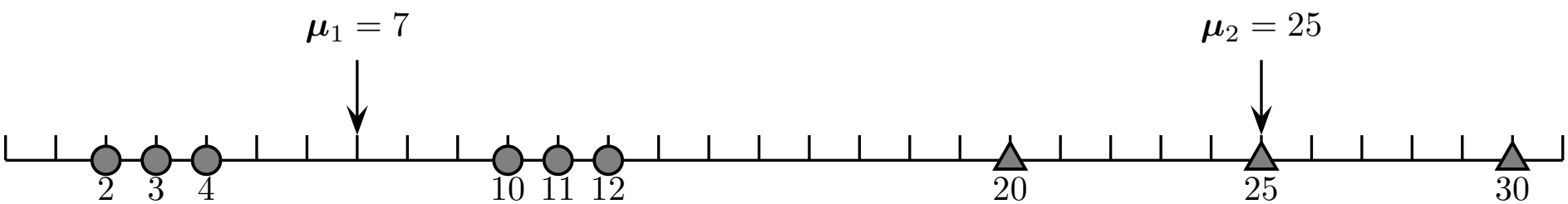




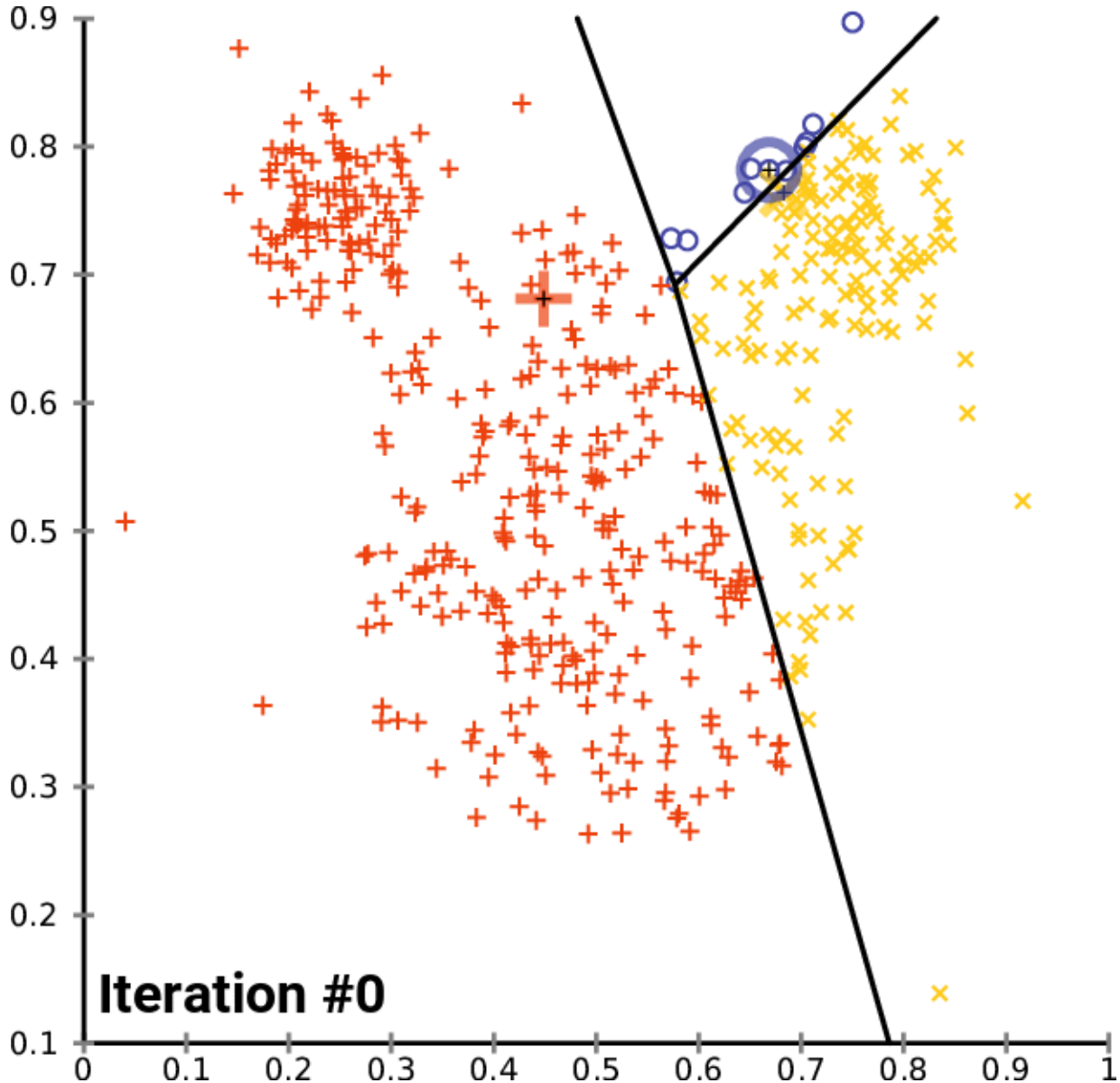
2-means in One Dimension : illustration

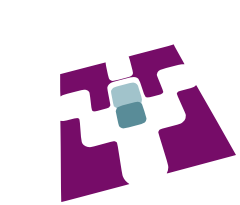
2-means clustering of the data points in a one-dimensional space using the Euclidean distance.

Iteration 5:

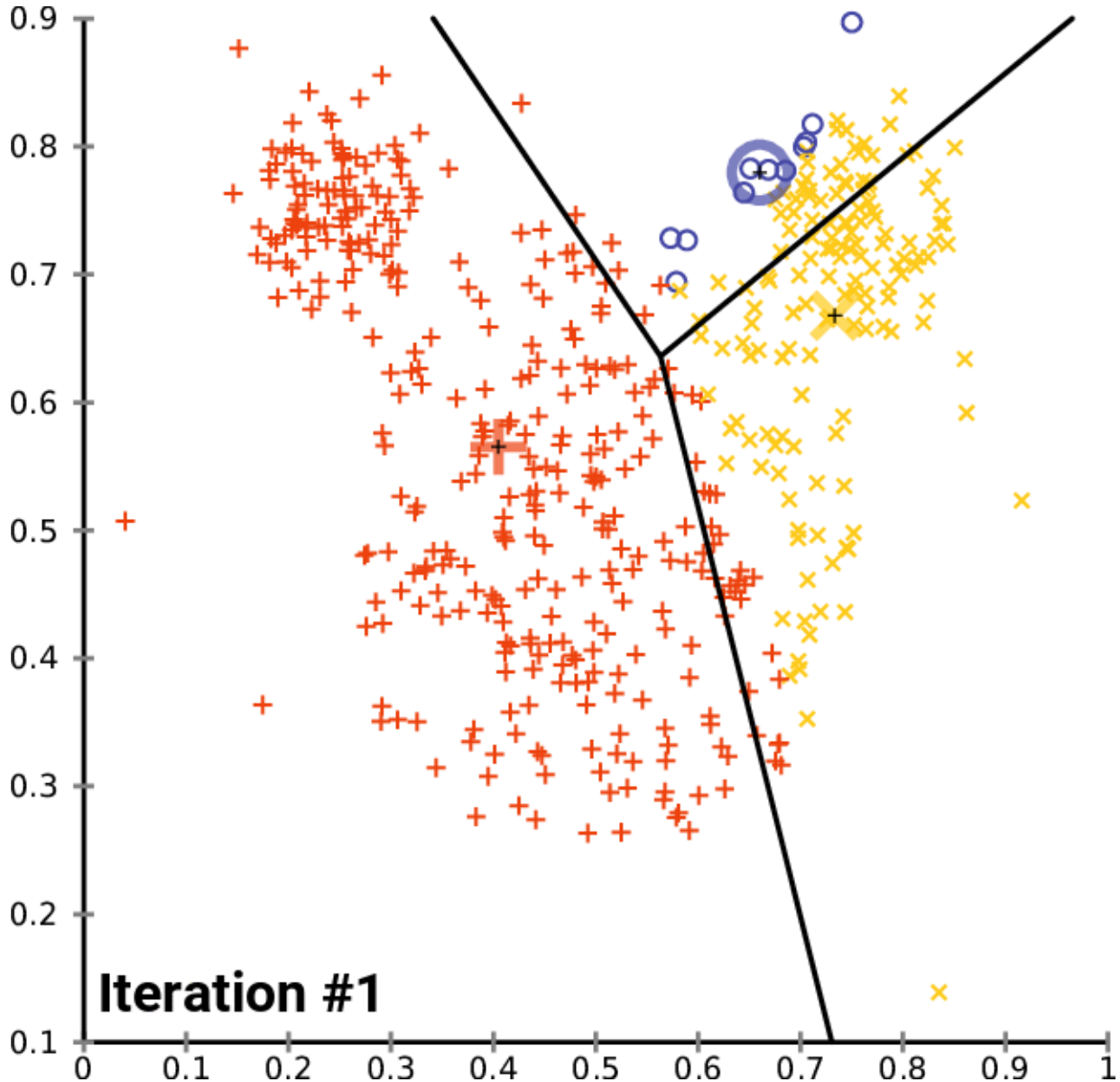


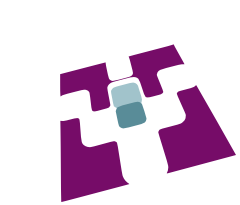
3-means in 2 Dimensions : illustration



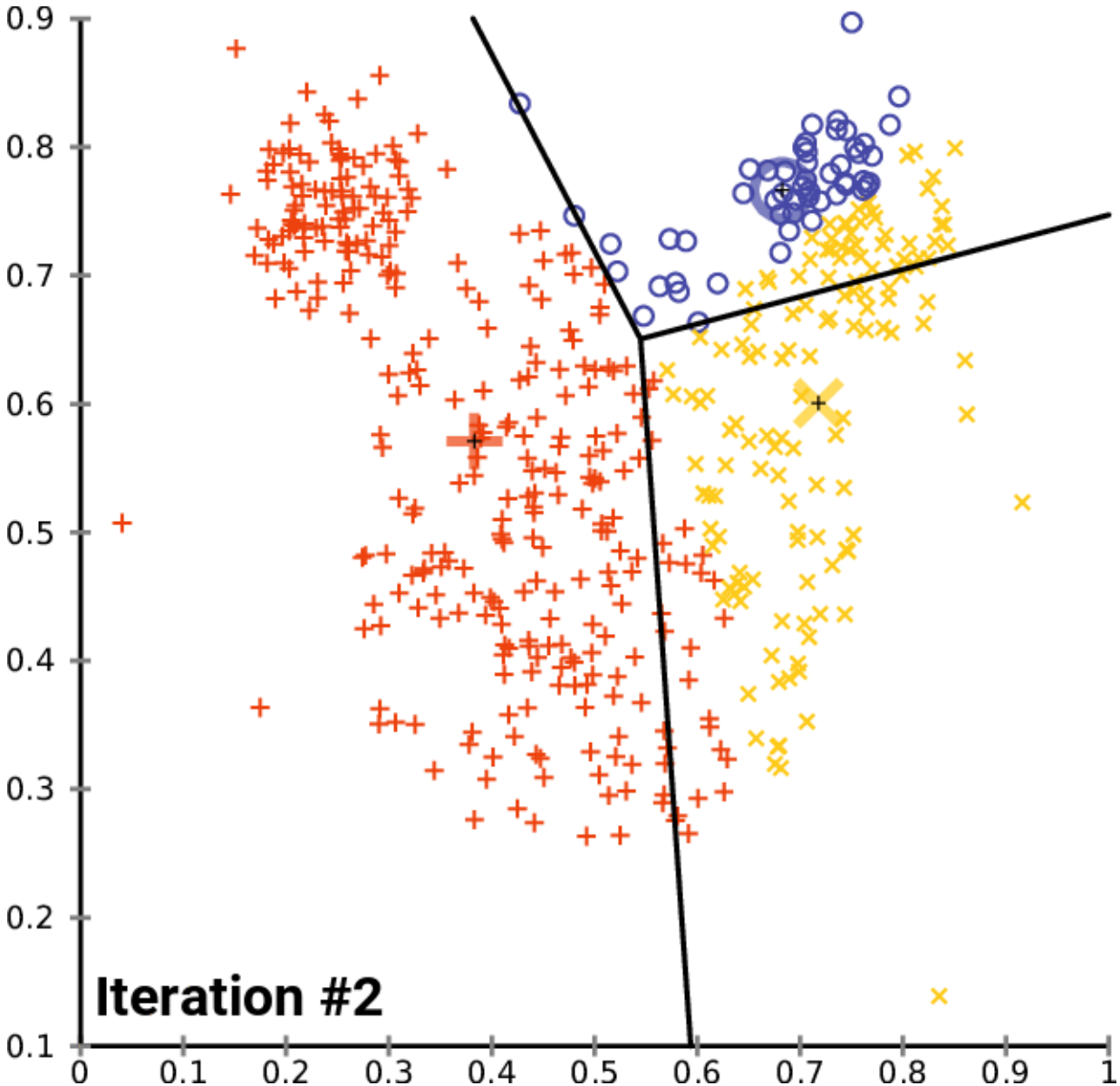


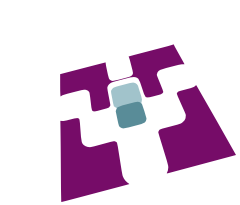
3-means in 2 Dimensions : illustration



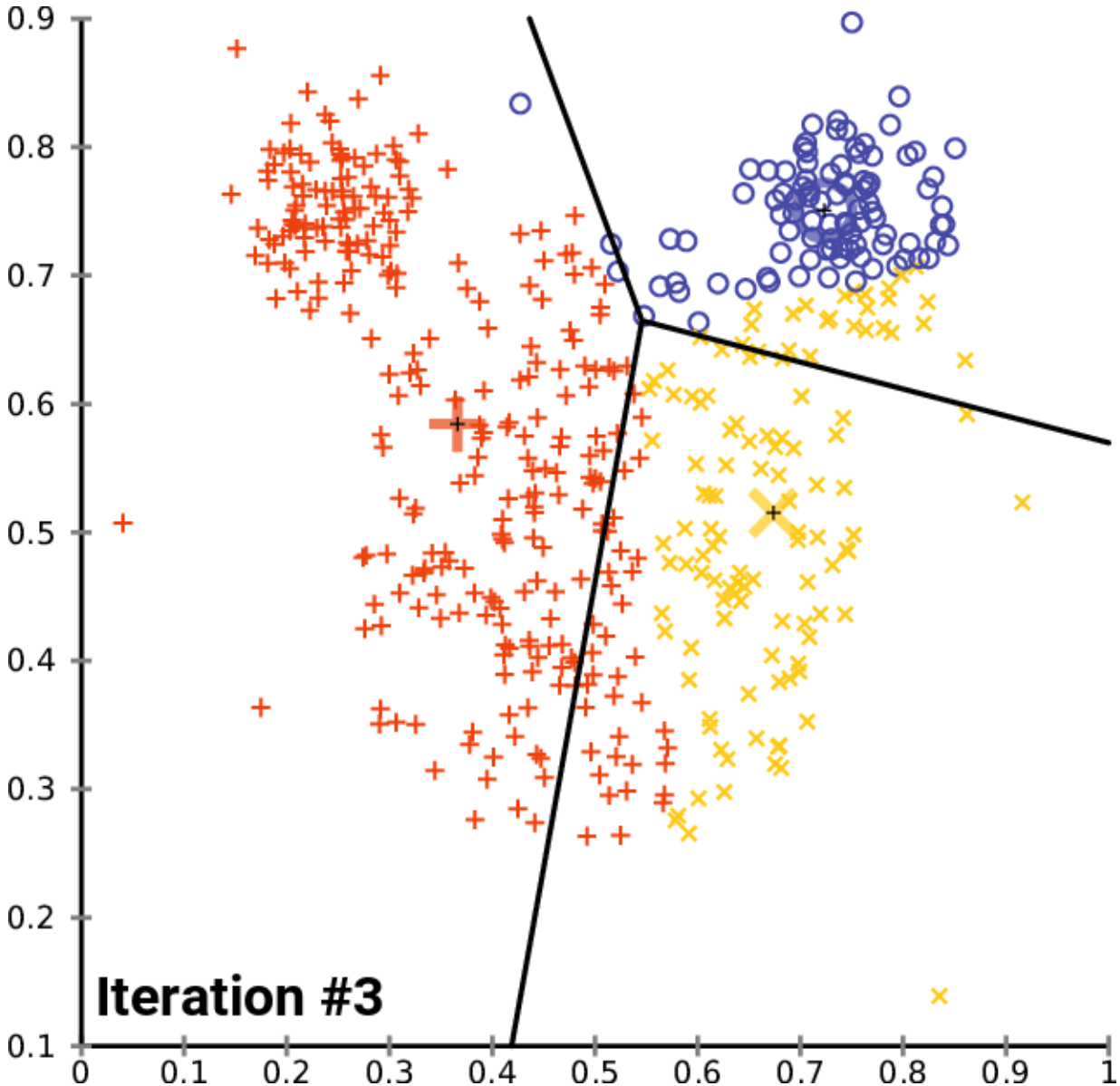


3-means in 2 Dimensions : illustration

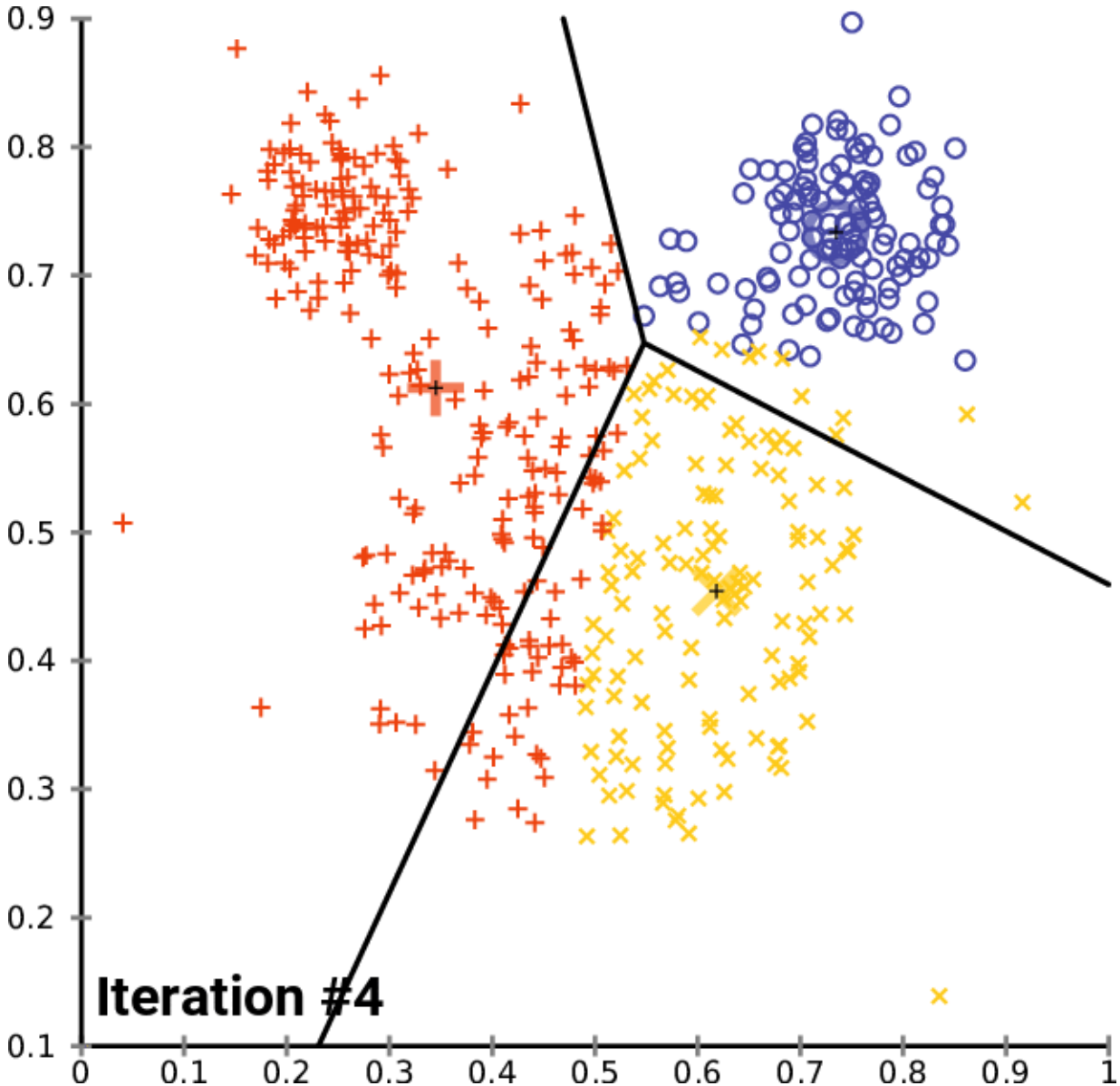


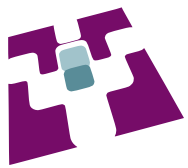


3-means in 2 Dimensions : illustration

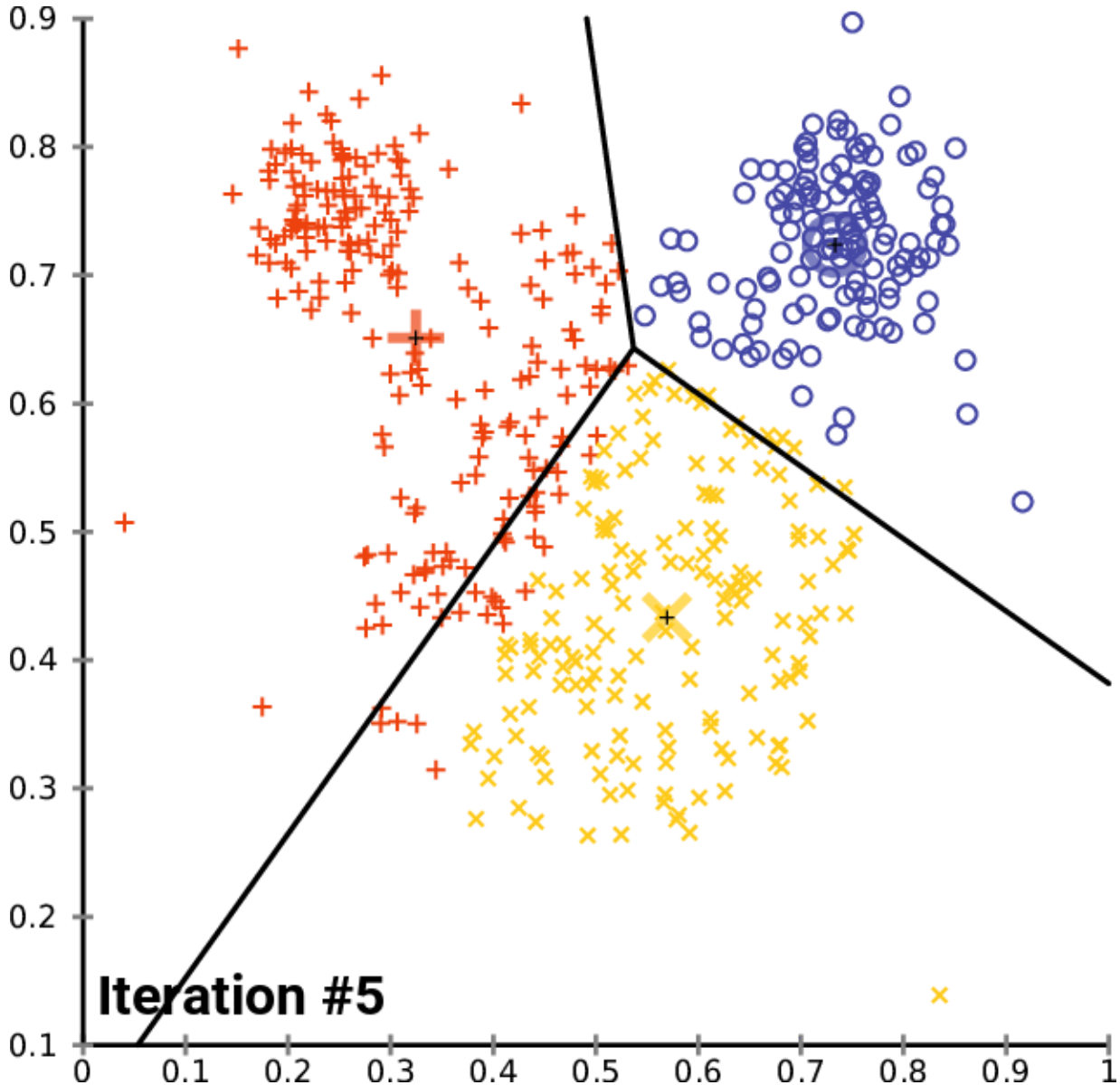


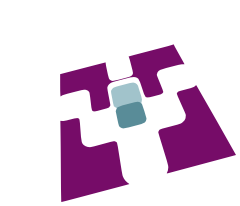
3-means in 2 Dimensions : illustration



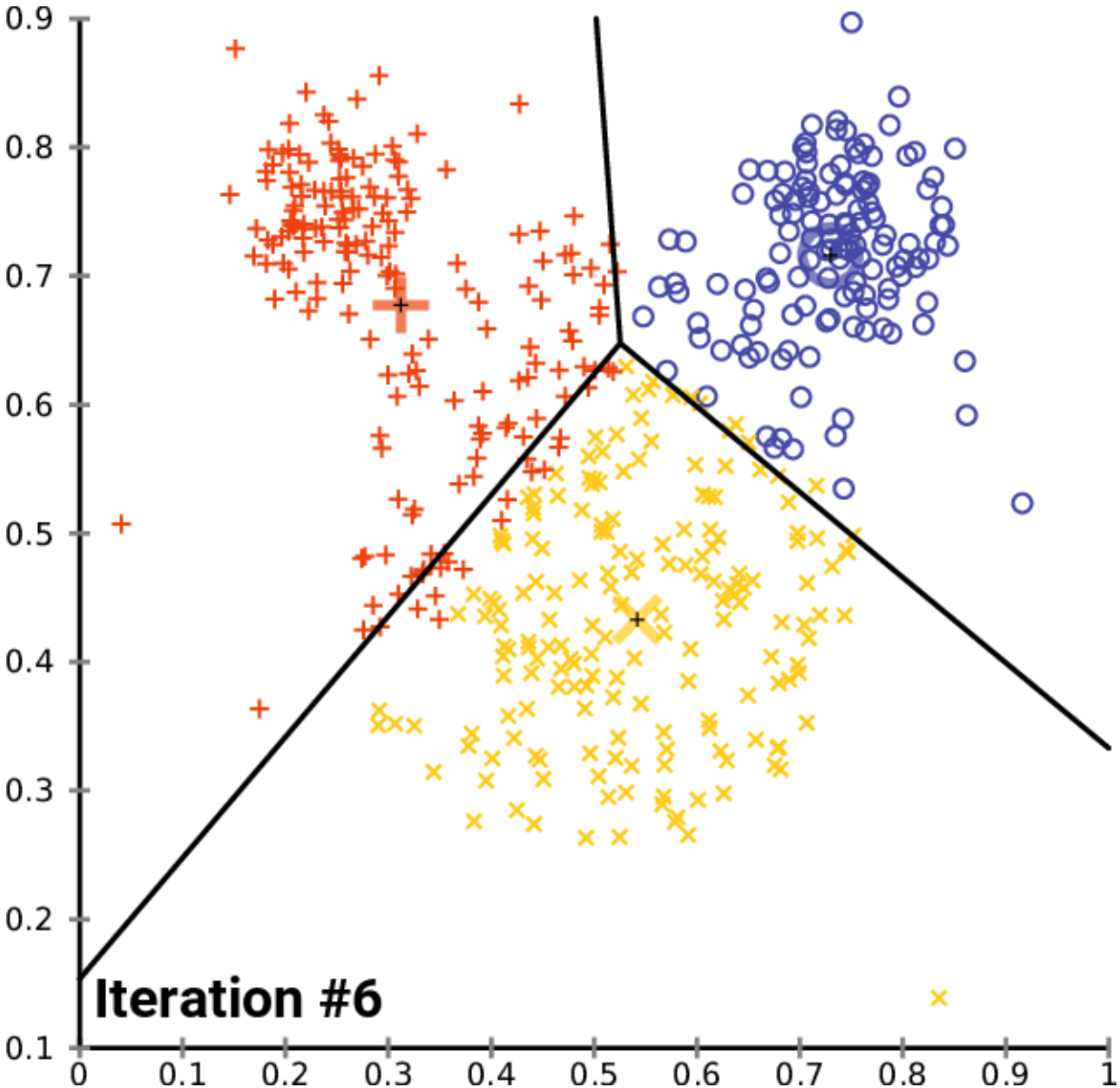


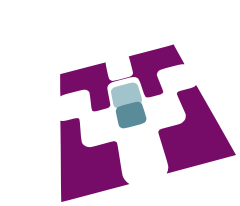
3-means in 2 Dimensions : illustration



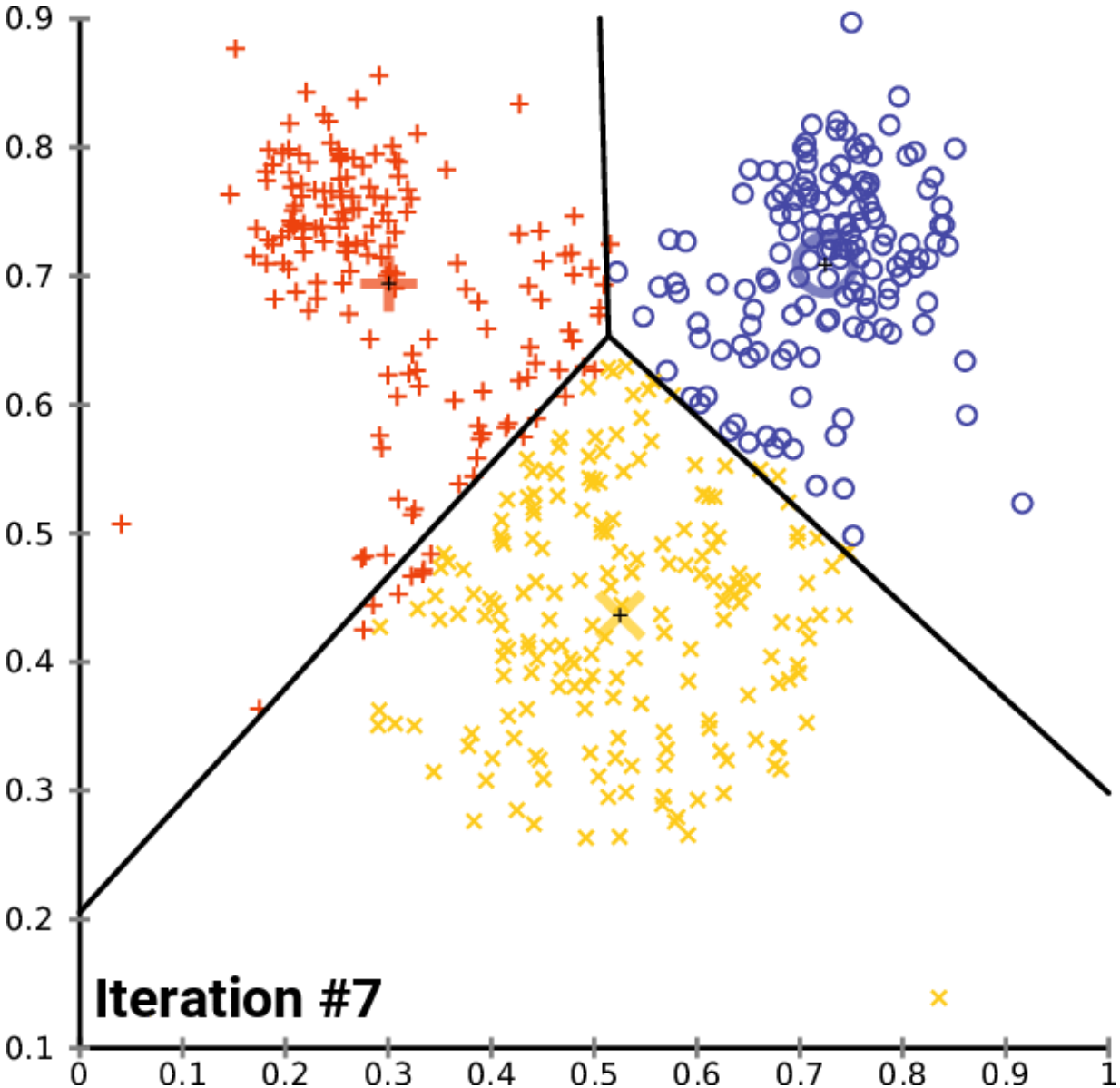


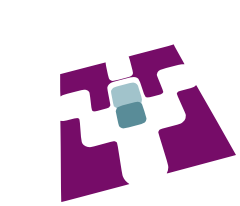
3-means in 2 Dimensions : illustration



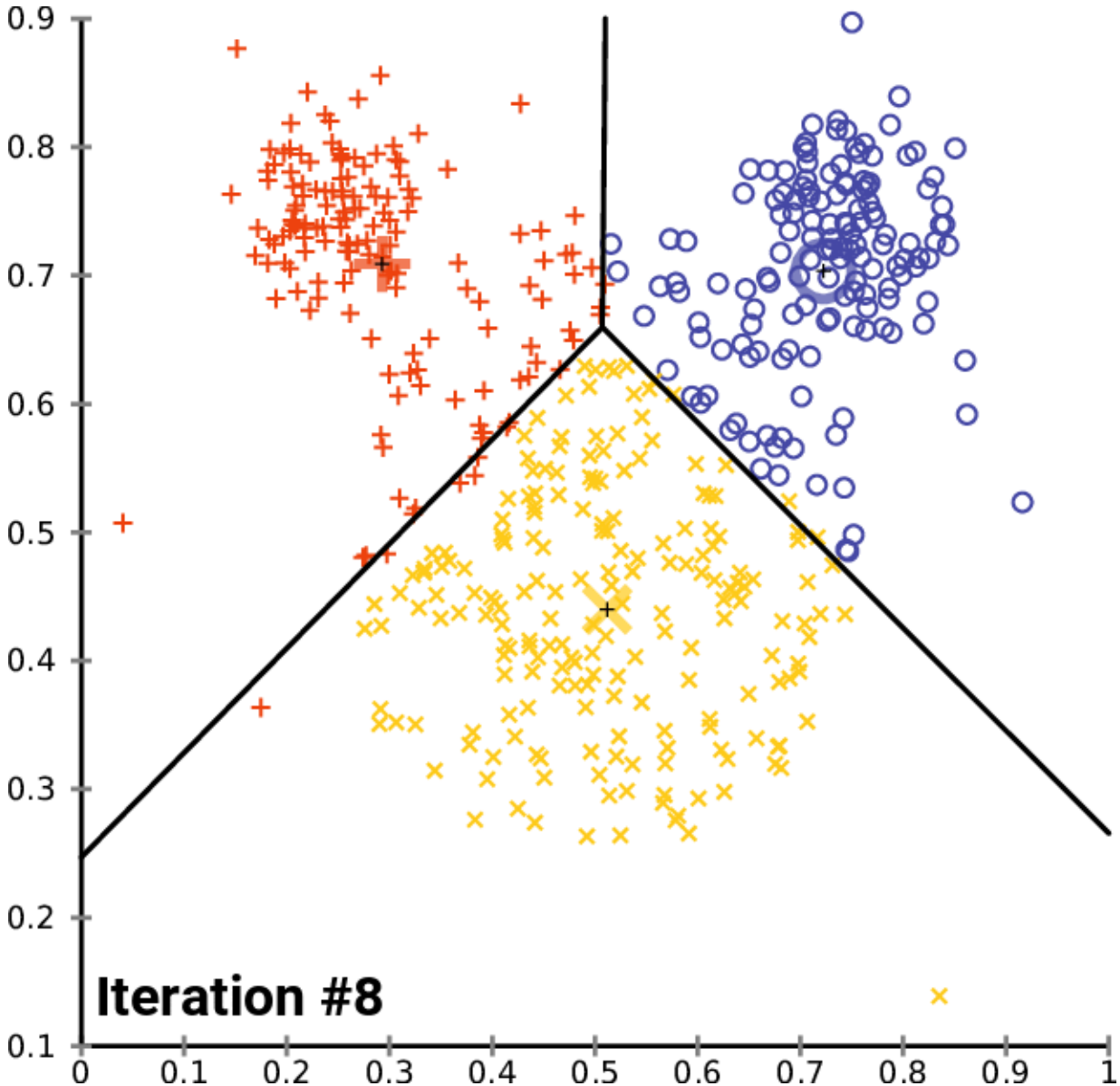


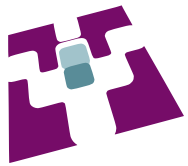
3-means in 2 Dimensions : illustration



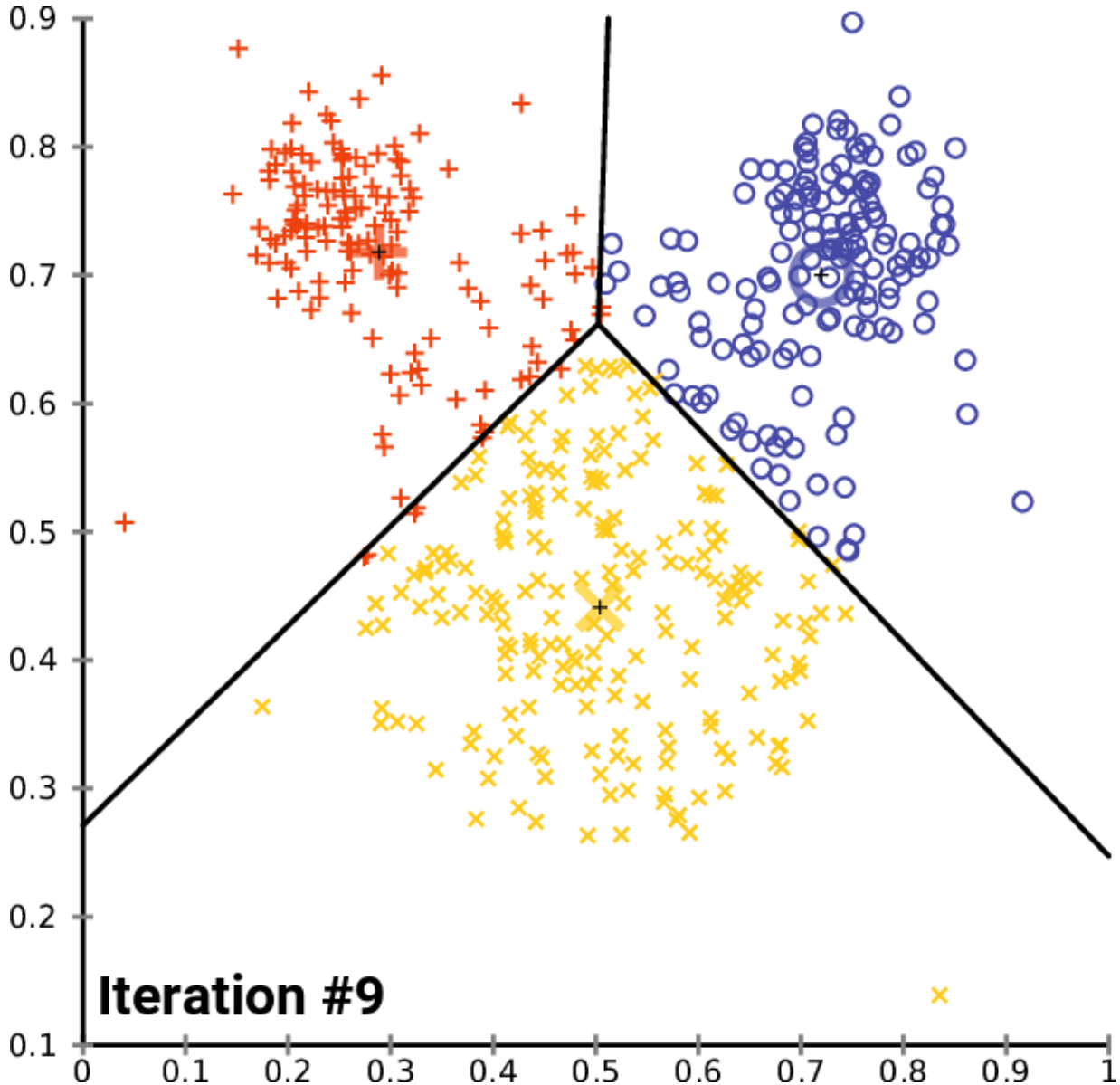


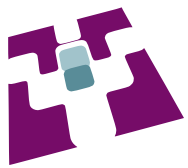
3-means in 2 Dimensions : illustration



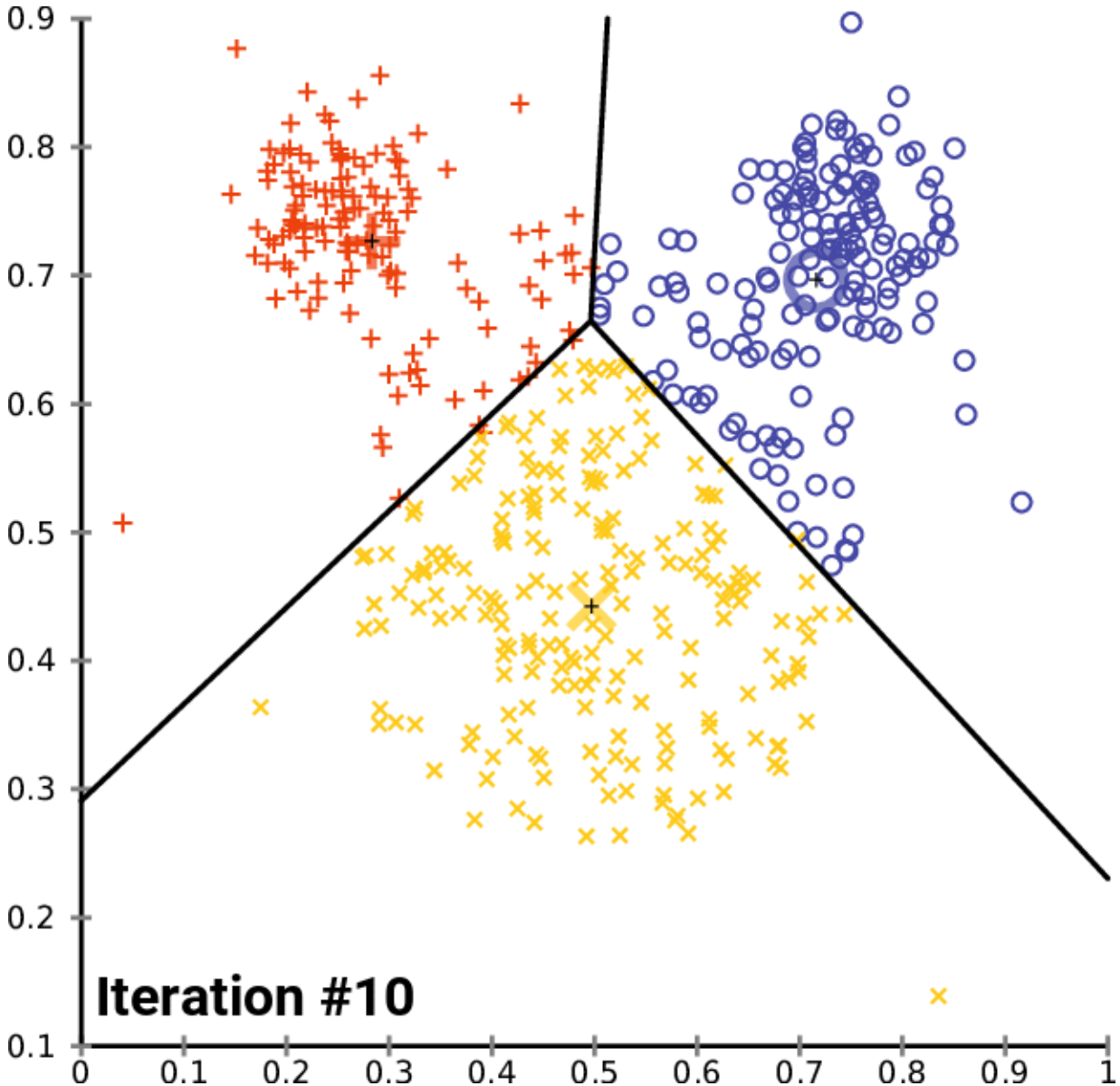


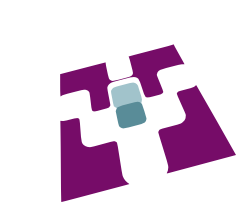
3-means in 2 Dimensions : illustration



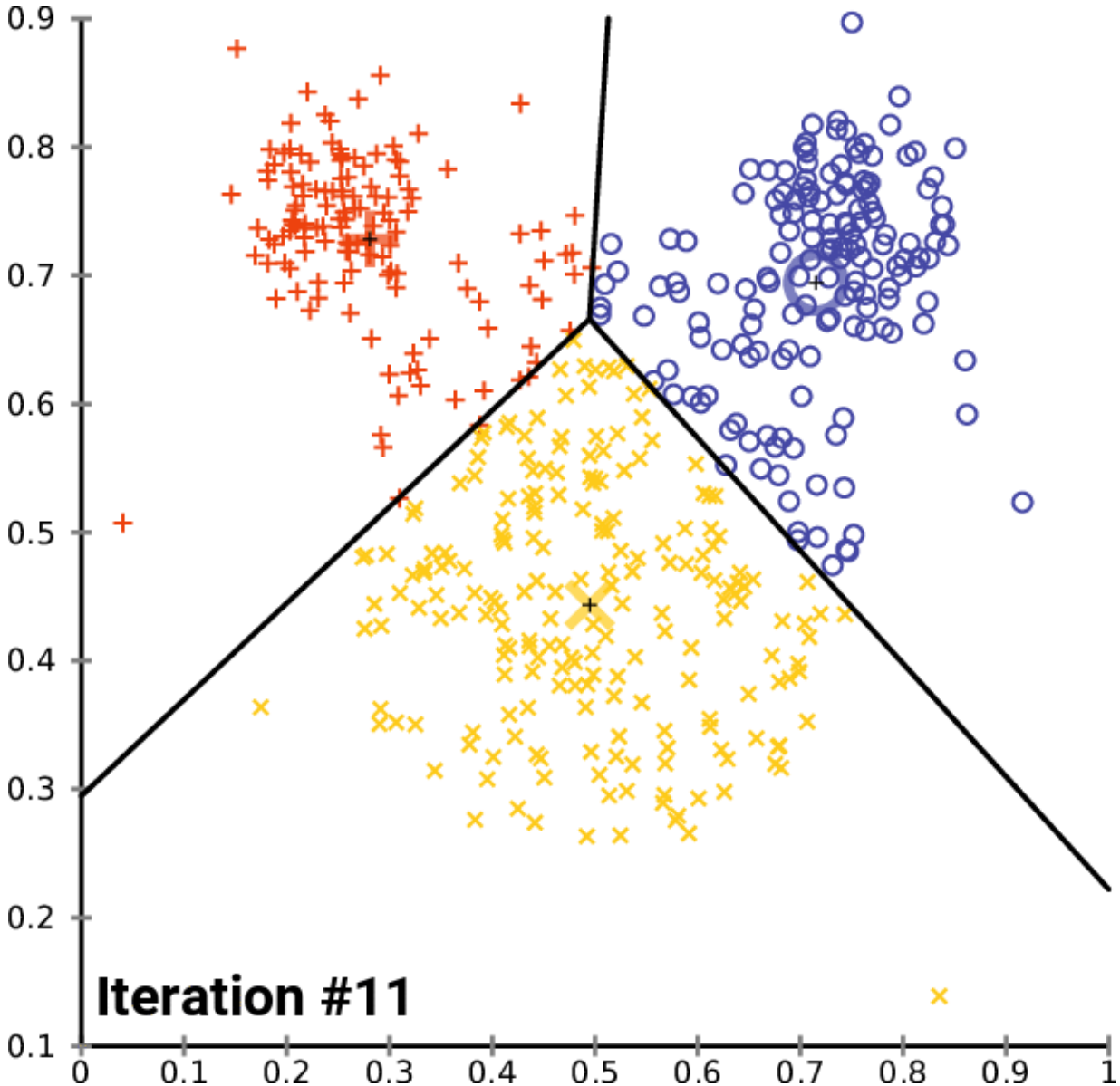


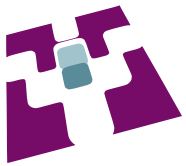
3-means in 2 Dimensions : illustration



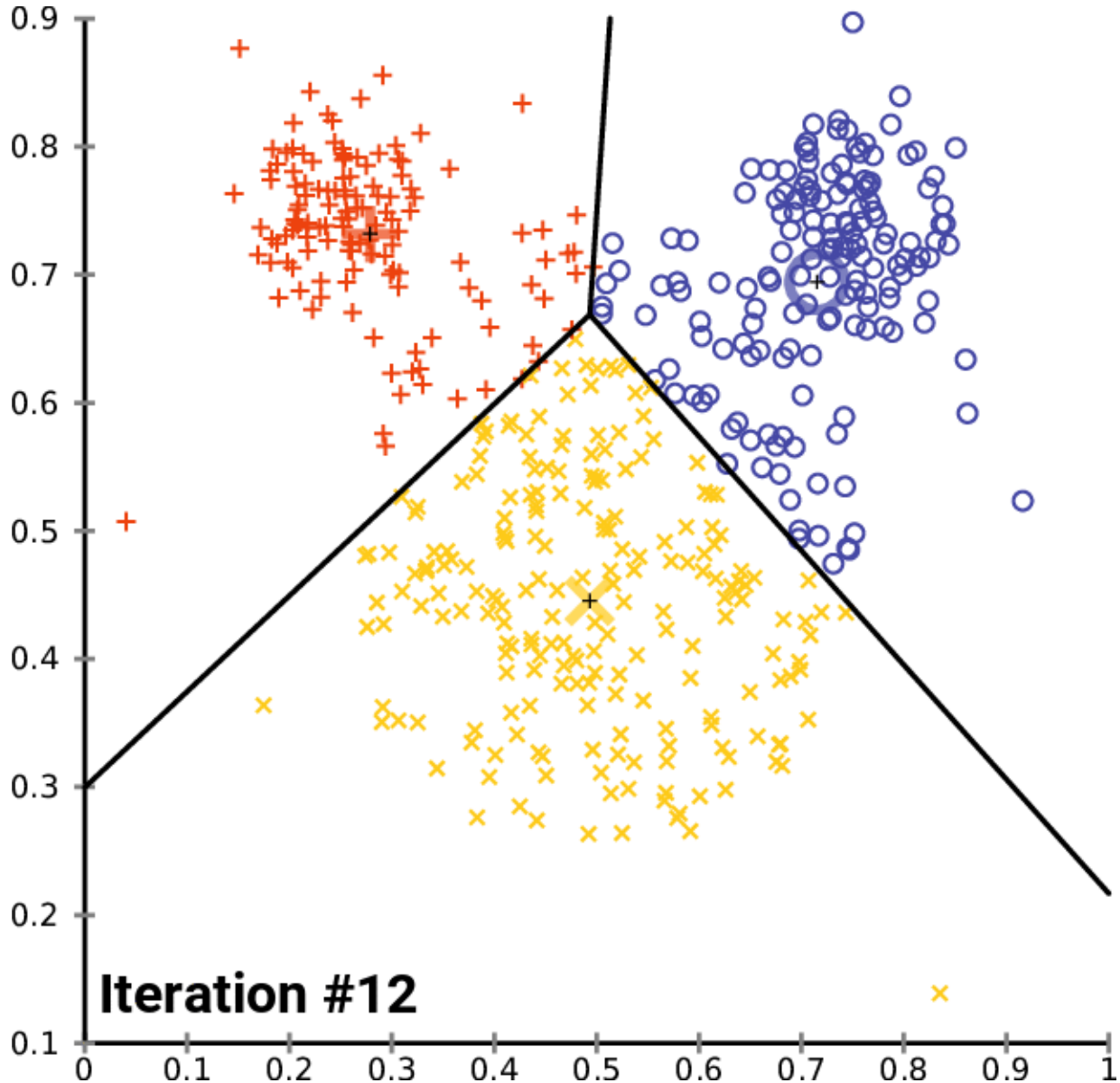


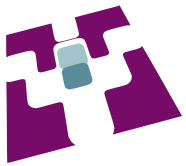
3-means in 2 Dimensions : illustration



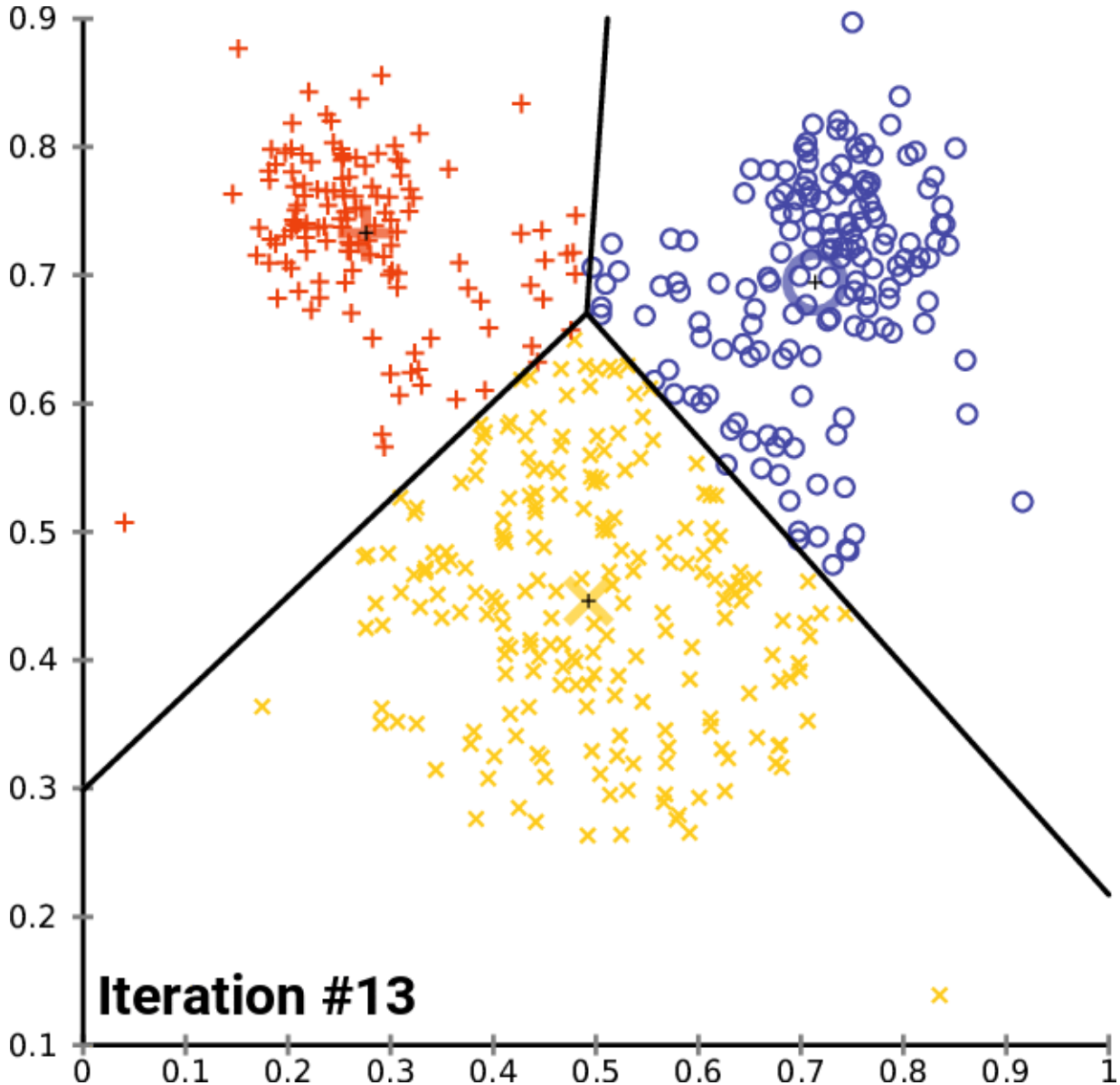


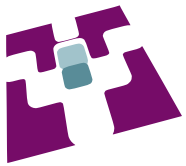
3-means in 2 Dimensions : illustration



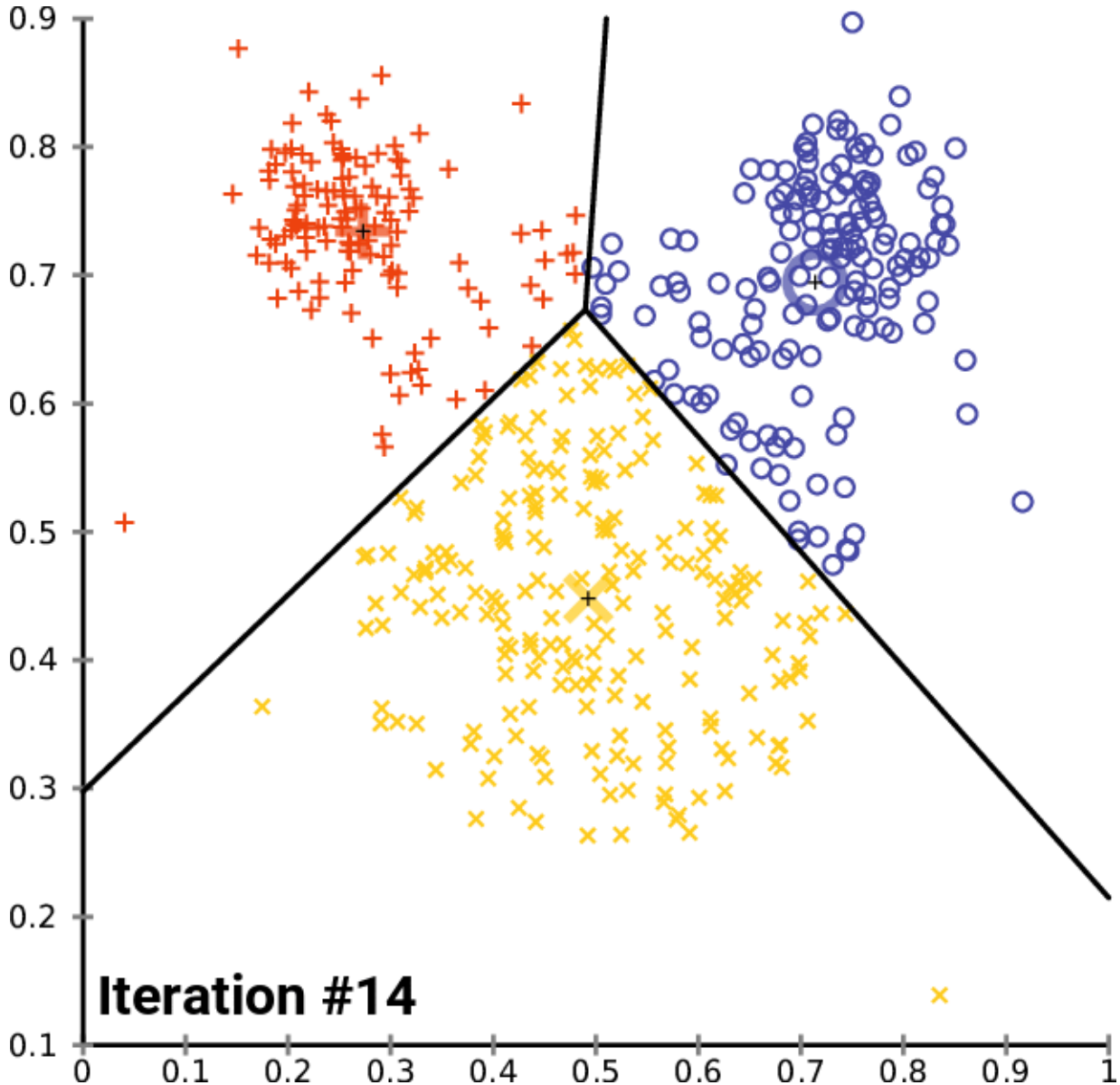


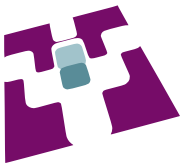
3-means in 2 Dimensions : illustration





3-means in 2 Dimensions : illustration





k-means algorithm

k-means $(\mathcal{E}, k, \epsilon)$:

$t = 0$

Randomly initialize k centroids: $\boldsymbol{\mu}_1^t, \boldsymbol{\mu}_2^t, \ldots, \boldsymbol{\mu}_k^t \in \mathbb{R}^d$

repeat

  $t \leftarrow t + 1$

  $C_j \leftarrow \emptyset$ for all $j = 1, \cdots, k$

  `// Cluster Assignment Step`

  foreach $\boldsymbol{x}_i \in \mathcal{E}$ **do**

    $j^* \leftarrow \arg\min_j \left\{ d^2(\boldsymbol{x}_i, \boldsymbol{\mu}_j^t) \right\}$ `// Assign` $\boldsymbol{x}_i$ `to closest centroid`

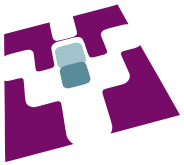
    $C_{j^*} \leftarrow C_{j^*} \cup \{\boldsymbol{x}_i\}$

  `// Centroid Update Step`

  foreach $j = 1$ to k **do**

    $\boldsymbol{\mu}_j^t \leftarrow \frac{1}{|C_j|} \sum_{\boldsymbol{x}_i \in C_j} \boldsymbol{x}_i$

until $\sum_{j=1}^{k} d^2(\boldsymbol{\mu}_j^t - \boldsymbol{\mu}_j^{t-1}) \leq \epsilon$

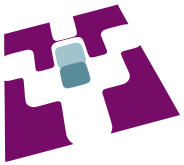


Complexity of k-means

Question
Assuming the computation of a similarity is linear in the number of attributes d, what is the complexity of **assign cluster**?

Question
What is the complexity of **update centers** ?

Question
What is the complexity of k-means if $t \in \mathbb{N}$ iterations are necessary to converge?

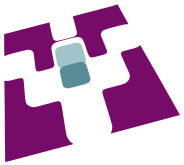


Complexity of k-means

Question
Assuming the computation of a similarity is linear in the number of attributes d, what is the complexity of **assign cluster**? $O(|\mathcal{E}| \times k \times d)$.

Question
What is the complexity of **update centers** ?

Question
What is the complexity of k-means if $t \in \mathbb{N}$ iterations are necessary to converge?

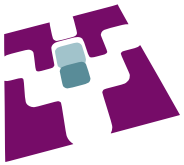


Complexity of k-means

Question
Assuming the computation of a similarity is linear in the number of attributes d, what is the complexity of **assign cluster**? $O(|\mathcal{E}| \times k \times d)$.

Question
What is the complexity of **update centers** ?
Support computation takes time $O(|\mathcal{E}| \times d)$.

Question
What is the complexity of k-means if $t \in \mathbb{N}$ iterations are necessary to converge?

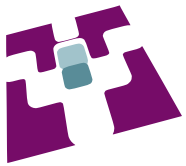


Complexity of k-means

Question
Assuming the computation of a similarity is linear in the number of attributes d, what is the complexity of **assign cluster**? $O(|\mathcal{E}| \times k \times d)$.

Question
What is the complexity of **update centers** ?
Support computation takes time $O(|\mathcal{E}| \times d)$.

Question
What is the complexity of k-means if $t \in \mathbb{N}$ iterations are necessary to converge?
$O(t \times k \times d \times |\mathcal{E}|)$.

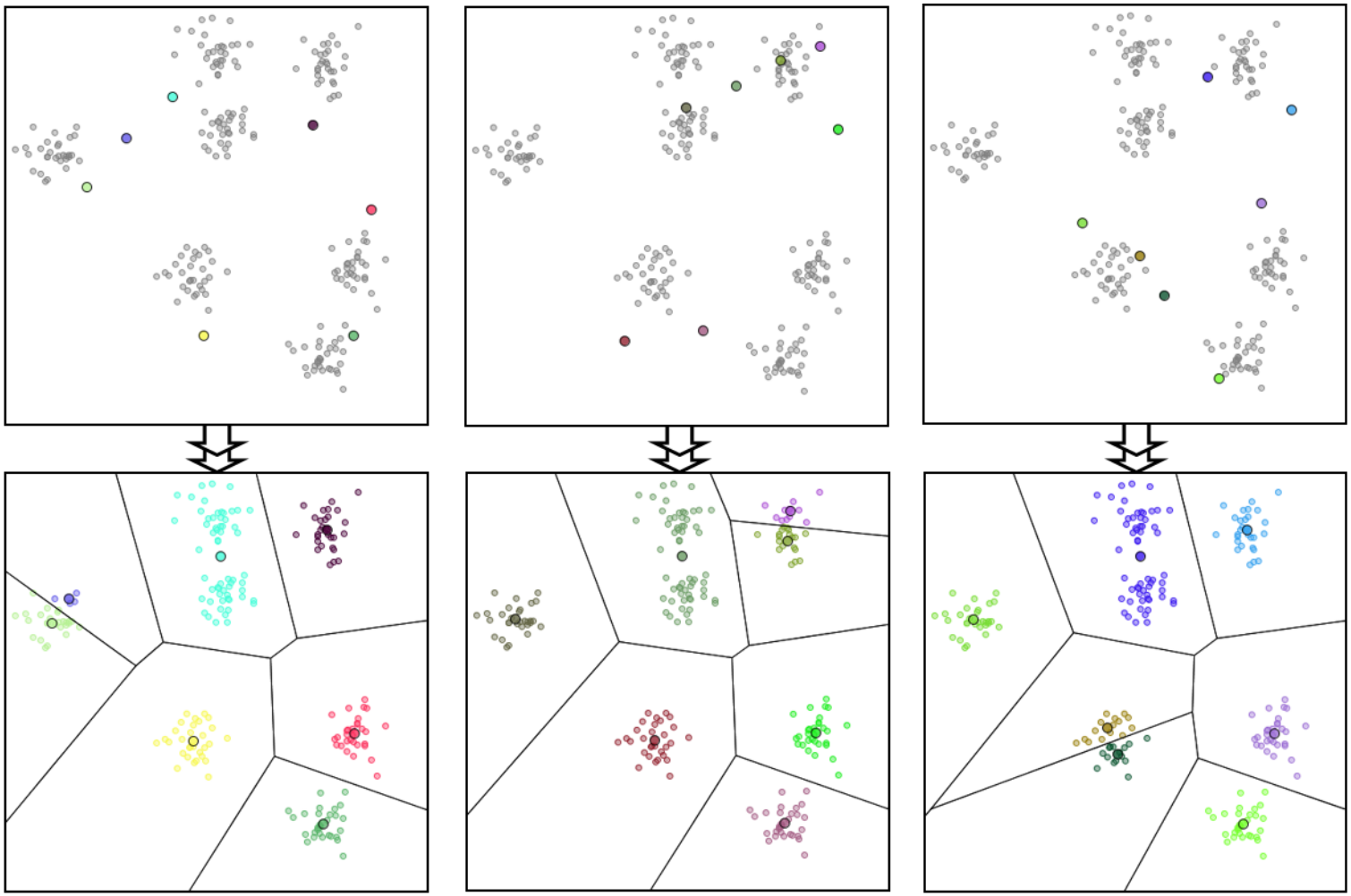


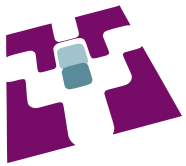
k-means : limitations

- Converge towards a local optima of the sum, over all data points, of the similarities to the centers of the assigned clusters;
- Require euclidean distance for computing the centroids;
- Sensitive to outliers and the initial seeds have a strong impact on the final results;
- The number of clusters must be known beforehand.

Initialization of k-means

A good initialization of k-means allows to obtain a solution of better quality and a faster convergence (with fewer iterations) to this solution.



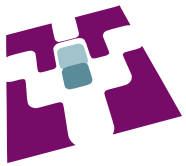


Initialization of k-means : k-means++

Idea : Select the successive means according to a non-uniform probability, which favors points further away from the already selected means.

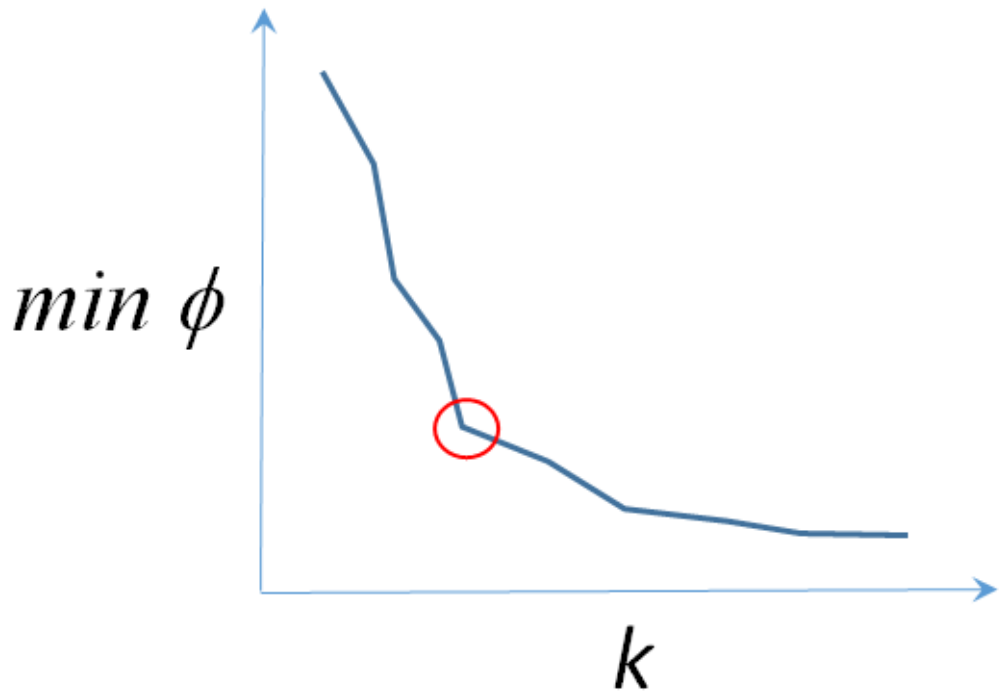
Algorithm :

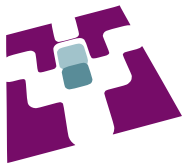
- Randomly select a point as the first-mean $\boldsymbol{c}_1$
- For all $\boldsymbol{x}_i$, compute the distance from all the existing means
$$\delta_j^i = d^2(\boldsymbol{x}_i, \boldsymbol{c}_j)$$
- Find the distance of each point from its closest centroid
$$\delta^i = \min_{j=1,\dots,k'} \left\{ d^2(\boldsymbol{x}_i, \boldsymbol{c}_j) \right\}$$
- Select $\boldsymbol{x}_i$ as the next centroids with probability proportional to δ^i
$$P(\boldsymbol{x}_i) = \frac{\delta^i}{\sum_{j=1}^{k'} \delta_j}$$
- Repeating the process until we have k centroids



Elbow method to find k number of clusters

Plot a measure of the quality of the k clusters (e. g. $\phi(\mathcal{C})$) when k increases.
Choose the sharp point of bend as the best value of k.

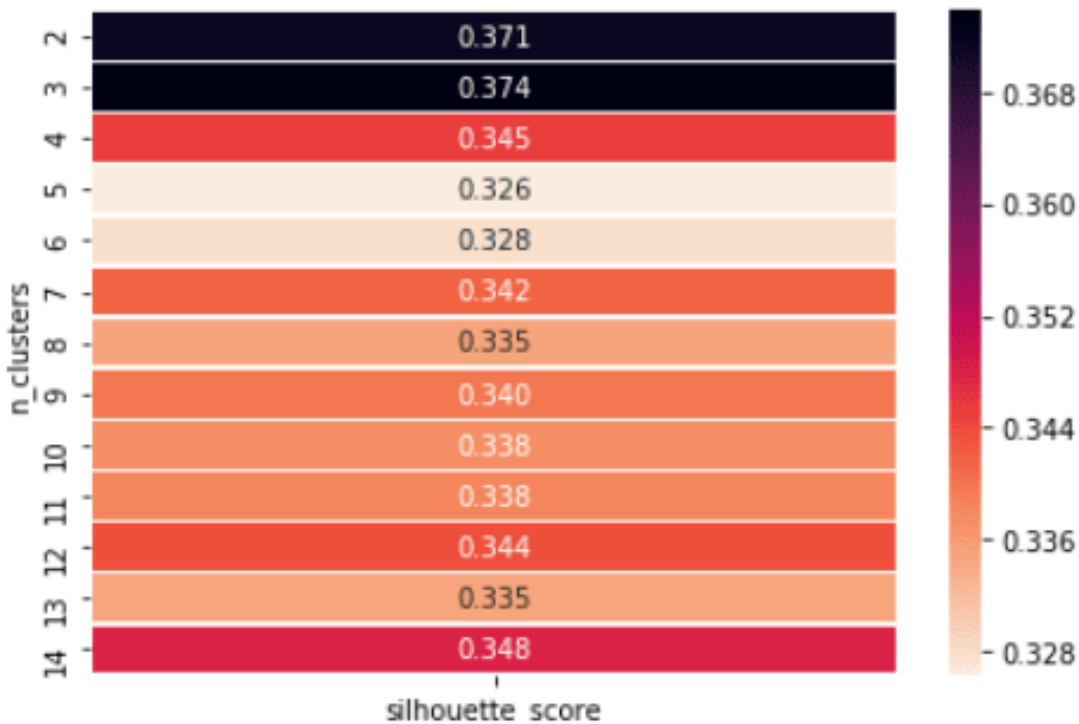


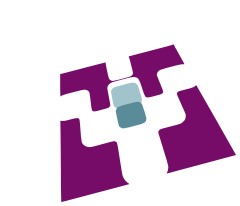


Silhouette score method to find k number of clusters

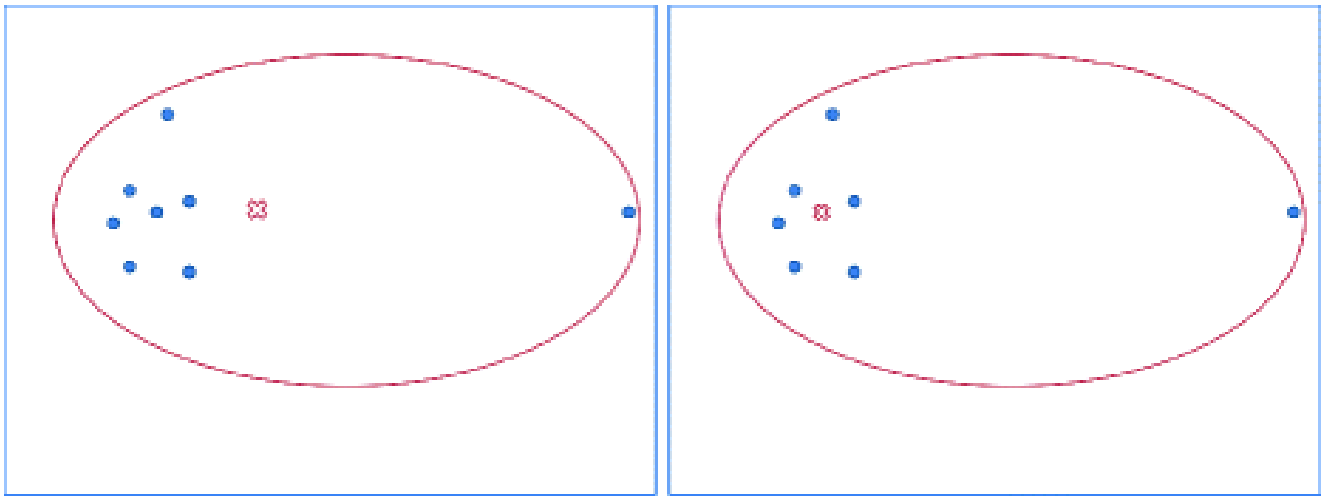
The silhouette value (from -1 to +1) is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).

A high value indicates that the point is well matched to its own cluster and poorly matched to neighboring clusters. If most points have a high value, then the clustering configuration is appropriate.





The concept of "medoid"

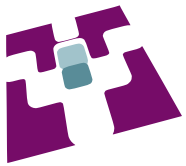


(a) Mean (b) Medoid

- The centroid may be totally artificial, it may not correspond to the real configuration of the dataset.
- The concept of medoid is more appropriate in some circumstances. This is an observed data point which minimizes its distance to all the other points.

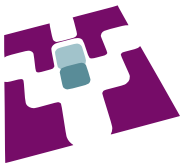
$$j^* = \arg\min_j \sum_{\boldsymbol{x}_i \in \boldsymbol{\mathcal{E}}} d(\boldsymbol{x}_i, \boldsymbol{m}_j)$$

- We are no longer limited to the Euclidean distance.
  ➠ Manhattan distance : $\delta(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \sum_{j=1}^{p} |\boldsymbol{x}_{ij} - \boldsymbol{x}_{i'j}|$



k-medoids Algorithm

- Medoid d'un groupe = individu le plus « central » du groupe
- Le seul changement par rapport à k-means est le remplacement des centres de gravité par des medoids.
- A chaque itération :
  1. choix, pour chaque point, du medoid $\boldsymbol{m}_{j^*}$ le plus proche : $j^* = \arg\min_j d(\boldsymbol{x}_i, \boldsymbol{m}_j)$
  2. constitution des groupes : C_j est constitué de tous les $\boldsymbol{x}_i$ qui sont plus proches de $\boldsymbol{m}_j$ que tout autre medoid
  3. recherche des medoids de ces (nouveaux) groupes : $m_j = \arg\min_{\boldsymbol{x}_l \in \boldsymbol{C}_j} \sum_{\boldsymbol{x}_p \in \boldsymbol{C}_j} d(\boldsymbol{x}_l, \boldsymbol{x}_p)$
- Une initialisation de même nature que k-means++ peut être employée
- Robustesse apportée par l'utilisation de medoids plutôt que de centres de gravité mais avec une complexité en $O(|\mathcal{E}|^2)$



Hierarchical Clustering

- Build a hierarchy of clusters, **also called the cluster dendrogram**;

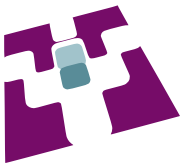
  The clusters in the hierarchy range from the fine-grained to the coarse-grained – the lowest level of the tree (the leaves) consists of each point in its own cluster, whereas the highest level (the root) consists of all points in one cluster.

- The number of clusters k is not required as input;

- Use a distance matrix as clustering criteria;

- Work in a bottom-up manner.

➠ Permet d'examiner l'ordre des agrégations de groupes, les rapports des similarités entre groupes, etc.



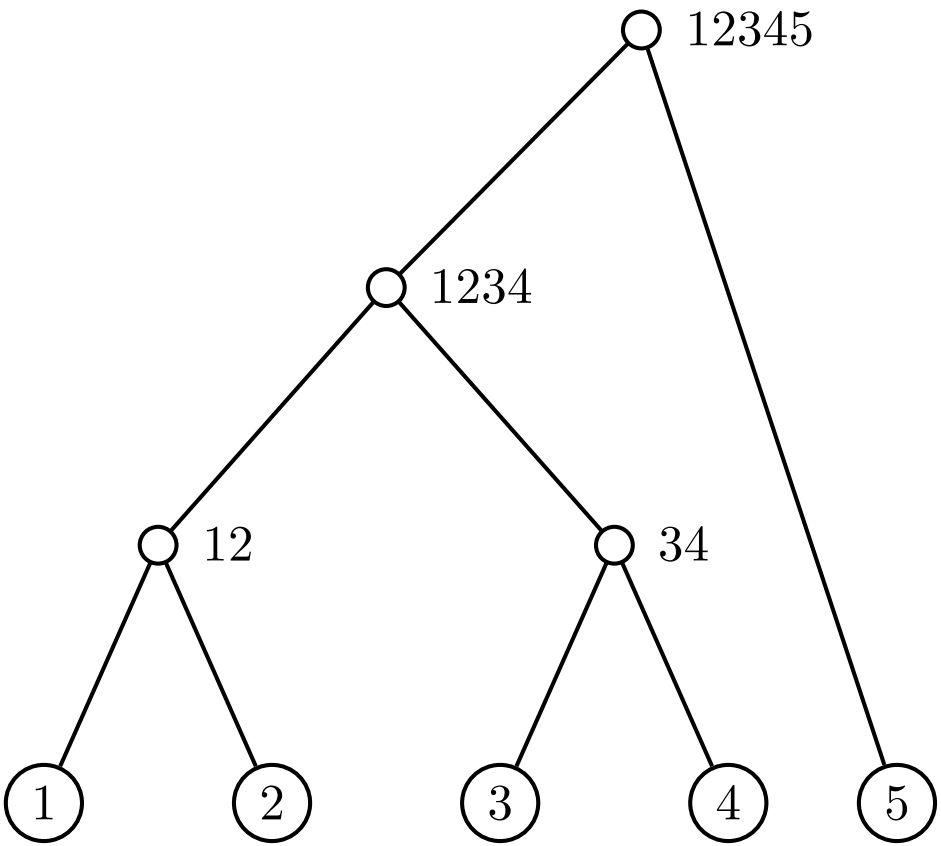
Hierarchical Clustering: Nested Partitions

Given a dataset $\boldsymbol{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$, a clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$ is a partition of $\boldsymbol{D}$.

A clustering $\mathcal{A} = \{A_1, \ldots, A_r\}$ is said to be nested in another clustering $\mathcal{B} = \{B_1, \ldots, B_s\}$ if and only if $r > s$, and for each cluster $A_i \in \mathcal{A}$, there exists a cluster $B_j \in \mathcal{B}$, such that $A_i \subseteq B_j$.

Hierarchical clustering yields a sequence of n nested partitions $\mathcal{C}_1, \ldots, \mathcal{C}_n$. The clustering $\mathcal{C}_{t-1}$ is nested in the clustering $\mathcal{C}_t$. The cluster dendrogram is a rooted binary tree that captures this nesting structure, with edges between cluster $C_i \in \mathcal{C}_{t-1}$ and cluster $C_j \in \mathcal{C}_t$ if C_i is nested in C_j, that is, if $C_i \subset C_j$.

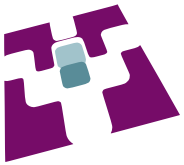
Hierarchical Clustering Dendrogram



The dendrogram represents the following sequence of nested partitions:

Clustering	**Clusters**
$\mathcal{C}_1$	$\{1\},\{2\},\{3\},\{4\},\{5\}$
$\mathcal{C}_2$	$\{12\},\{3\},\{4\},\{5\}$
$\mathcal{C}_3$	$\{12\},\{34\},\{5\}$
$\mathcal{C}_4$	$\{1234\},\{5\}$
$\mathcal{C}_5$	$\{12345\}$

with $\mathcal{C}_{t-1} \subset \mathcal{C}_t$ for $t = 2, \ldots, 5$. We assume that 1 and 2 are merged before 3 and 4.



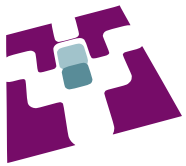
Agglomerative Hierarchical Clustering

In agglomerative hierarchical clustering, we begin with each of the n points in a separate cluster. We repeatedly merge the two closest clusters until all points are members of the same cluster.

Given a set of clusters $\mathcal{C} = \{C_1, C_2, .., C_m\}$, we find the **closest** pair of clusters C_i and C_j and merge them into a new cluster $C_{ij} = C_i \cup C_j$.

Next, we update the set of clusters by removing C_i and C_j and adding C_{ij}, as follows $\mathcal{C} = (\mathcal{C} \setminus \{C_i, C_j\}) \cup \{C_{ij}\}$.

We repeat the process until $\mathcal{C}$ contains only one cluster. If specified, we can stop the merging process when there are exactly k clusters remaining.



Agglomerative Hierarchical Clustering Algorithm

AgglomerativeClustering($\boldsymbol{D}, k$):

1 $\mathcal{C} \leftarrow \{C_i = \{\boldsymbol{x}_i\} \mid \boldsymbol{x}_i \in \boldsymbol{D}\}$ // Each point in separate cluster

2 $\Delta \leftarrow \{\delta(\boldsymbol{x}_i, \boldsymbol{x}_j) \colon \boldsymbol{x}_i, \boldsymbol{x}_j \in \boldsymbol{D}\}$ // Compute distance matrix

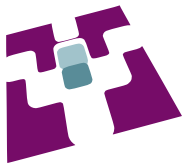
3 **repeat**

4 Find the closest pair of clusters $C_i, C_j \in \mathcal{C}$

5 $C_{ij} \leftarrow C_i \cup C_j$ // Merge the clusters

6 $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C_i, C_j\}) \cup \{C_{ij}\}$ // Update the clustering

7 **until** $|\mathcal{C}| = k$



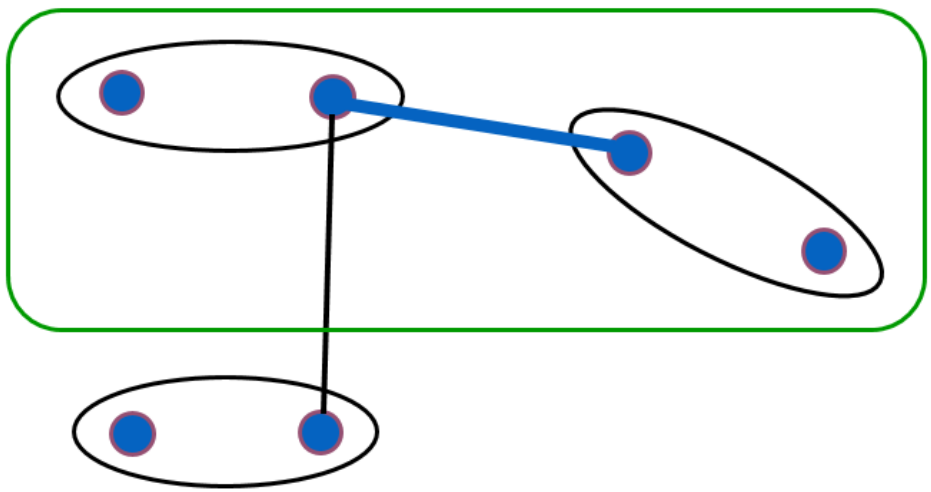
Distance between Clusters

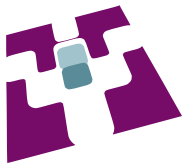
A typical distance between two points is the Euclidean distance

$$\delta(\boldsymbol{x}_i, \boldsymbol{x}_k) = \left(\sum_{j=1}^{d}(x_{ij} - x_{kj})^2\right)^{1/2}$$

Single Link: The minimum distance between a point inMinimal distance among the pairs composed of points from the two clusters C_i and a point in C_j

$$\delta(C_i, C_j) = \min\{\delta(\boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j\}$$





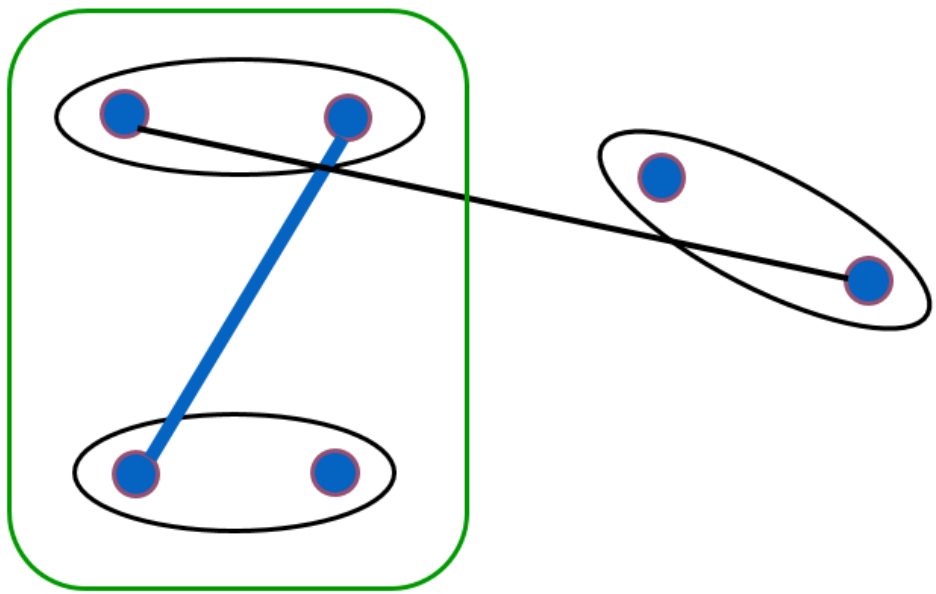
Distance between Clusters

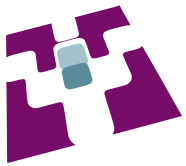
A typical distance between two points is the Euclidean distance

$$\delta(\boldsymbol{x}_i, \boldsymbol{x}_k) = \left(\sum_{j=1}^{d} (x_{ij} - x_{kj})^2\right)^{1/2}$$

Complete Link: The maximum distance between points in the two clusters:

$$\delta(C_i, C_j) = \max\{\delta(\boldsymbol{x}, \boldsymbol{y}) \mid \boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j\}$$





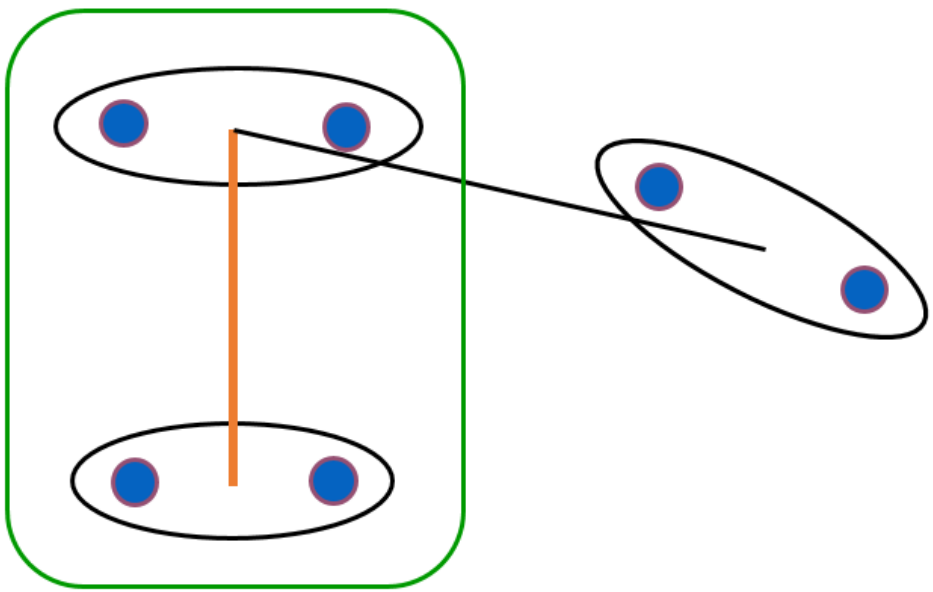
Distance between Clusters

A typical distance between two points is the Euclidean distance

$$\delta(\boldsymbol{x}_i, \boldsymbol{x}_k) = \left(\sum_{j=1}^{d} (x_{ij} - x_{kj})^2\right)^{1/2}$$

Group Average: The average pairwise distance between points in C_i and C_j:

$$\delta(C_i, C_j) = \frac{\sum_{\boldsymbol{x} \in C_i} \sum_{\boldsymbol{y} \in C_j} \delta(\boldsymbol{x}, \boldsymbol{y})}{n_i \cdot n_j}$$



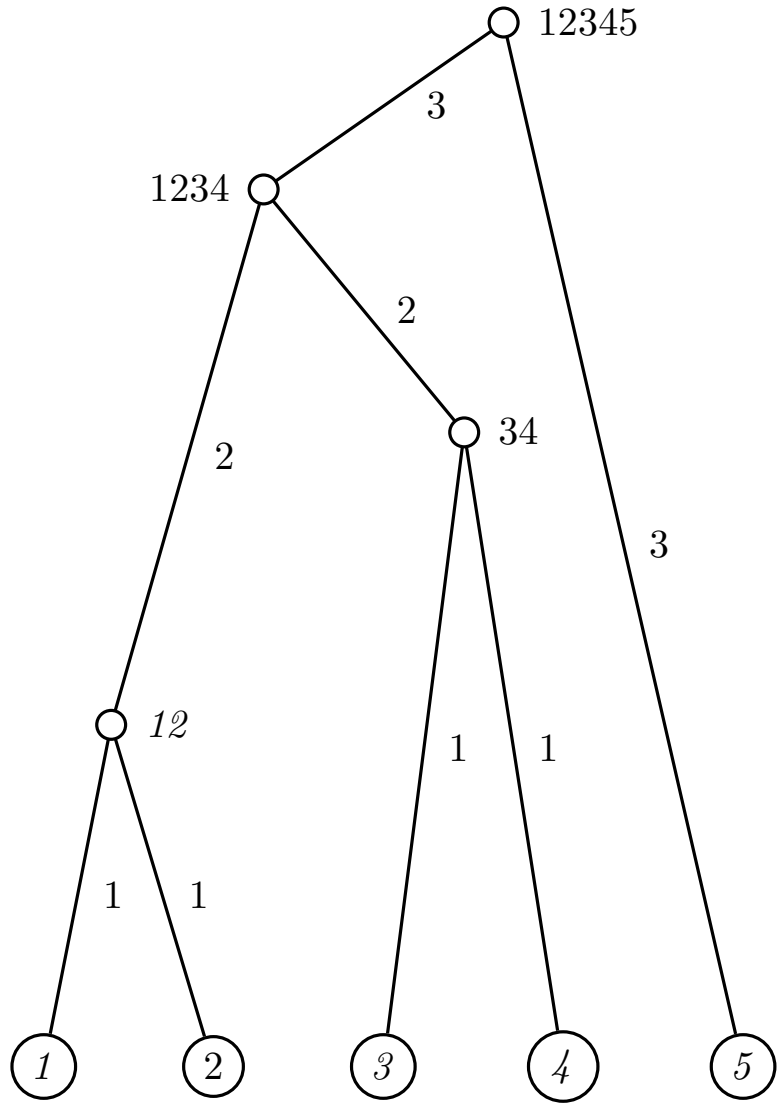
Single Link Agglomerative Clustering

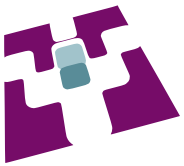
δ	*5*
1234	(3)

δ	34	5
12	(2)	3
34		3

δ	3	4	5
12	3	2	3
3		(1)	3
4			5

δ	2	3	4	5
1	(1)	3	2	4
2		3	2	3
3			1	3
4				5





The DBSCAN Approach
Neighborhood and Core Points

Define a ball of radius ϵ centered at point $\boldsymbol{x} \in \mathbb{R}^d$, called the ϵ-neighborhood of $\boldsymbol{x}$:

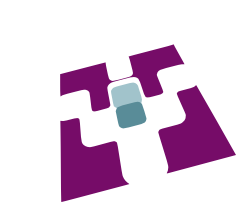
$$N_\epsilon(\boldsymbol{x}) = \{\boldsymbol{y} \mid \ \delta(\boldsymbol{x}, \boldsymbol{y}) \leq \epsilon\}$$

Here $\delta(\boldsymbol{x}, \boldsymbol{y})$ represents the distance between points $\boldsymbol{x}$ and $\boldsymbol{y}$. which is usually assumed to be the Euclidean

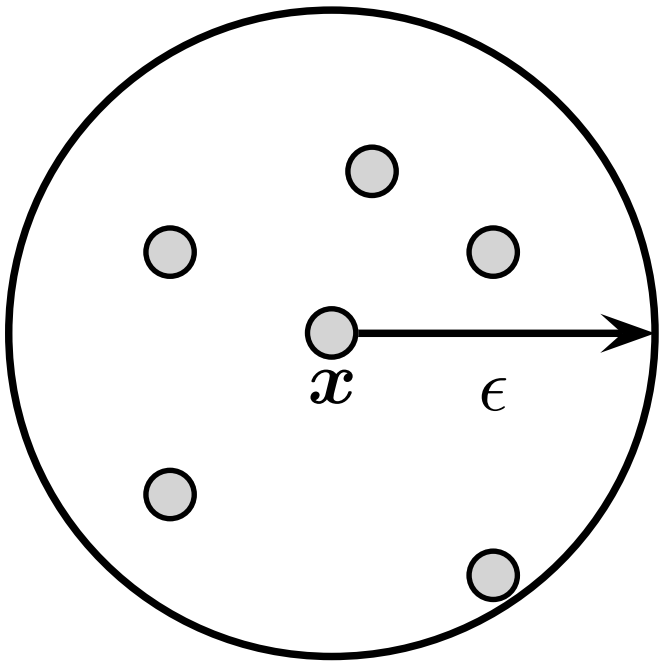
We say that $\boldsymbol{x}$ is a **core point** if there are at least *minpts* points in its ϵ-neighborhood, i.e., if $|N_\epsilon(\boldsymbol{x})| \geq \mathit{minpts}$.

A **border point** does not meet the *minpts* threshold, i.e., $|N_\epsilon(\boldsymbol{x})| < \mathit{minpts}$, but it belongs to the ϵ-neighborhood of some core point $\boldsymbol{z}$, that is, $\boldsymbol{x} \in N_\epsilon(\boldsymbol{z})$.

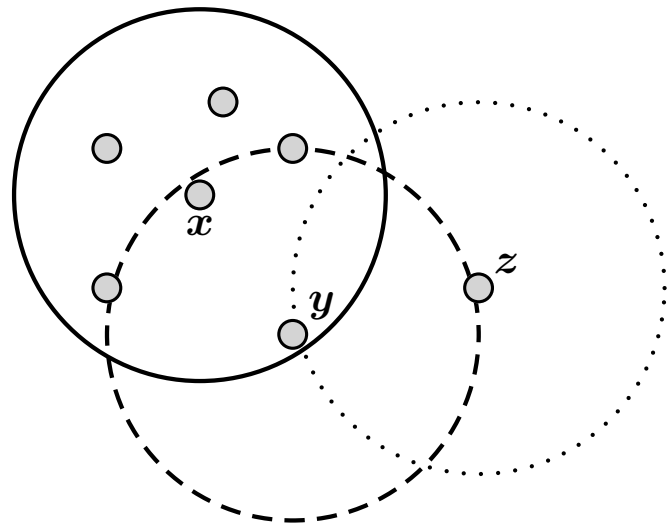
If a point is neither a core nor a border point, then it is called a **noise point** or an outlier.



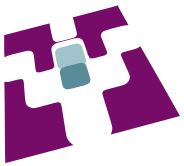
Core, Border and Noise Points



(a) Neighborhood of a Point



(b) Core, Border, and Noise Points



The DBSCAN Approach
Reachability and Density-based Cluster

A point $\boldsymbol{x}$ is **directly density reachable** from another point $\boldsymbol{y}$ if $\boldsymbol{x} \in N_\epsilon(\boldsymbol{y})$ and $\boldsymbol{y}$ is a core point.

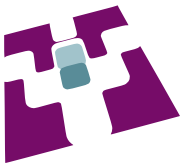
A point $\boldsymbol{x}$ is **density reachable** from $\boldsymbol{y}$ if there exists a chain of points, $(\boldsymbol{x}_1, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, such that

- $\boldsymbol{x}_1 = \boldsymbol{x}$;
- $\boldsymbol{x}_{i+1}$ is directly density reachable from $\boldsymbol{x}_i$ for all $i = 1, \ldots, n-1$;
- $\boldsymbol{x}_n = \boldsymbol{y}$.

In other words, there is set of core points leading from $\boldsymbol{y}$ to $\boldsymbol{x}$.

Two points $\boldsymbol{x}$ and $\boldsymbol{y}$ are **density connected** if there exists a core point $\boldsymbol{z}$, such that both $\boldsymbol{x}$ and $\boldsymbol{y}$ are density reachable from $\boldsymbol{z}$.

A **density-based cluster** is defined as a maximal set of density connected points.



DBSCAN Density-based Clustering Algorithm

DBSCAN computes the ϵ-neighborhood $N_\epsilon(\boldsymbol{x}_i)$ for each point $\boldsymbol{x}_i$ in the dataset $\boldsymbol{D}$, and checks if it is a core point. It also sets the cluster id $id(\boldsymbol{x}_i) = \emptyset$ for all points, indicating that they are not assigned to any cluster.

Starting from each unassigned core point, the method recursively finds all its density connected points, which are assigned to the same cluster.

Some border point may be reachable from core points in more than one cluster; they may either be arbitrarily assigned to one of the clusters or to all of them (if overlapping clusters are allowed).

Those points that do not belong to any cluster are treated as outliers or noise.

Each DBSCAN cluster is a maximal connected component over the core point graph.

DBSCAN is sensitive to the choice of ϵ, in particular if clusters have different densities. The overall complexity of DBSCAN is $O(n^2)$.

DBSCAN Algorithm

dbscan ($\boldsymbol{D}$, ϵ, *minpts*)**:**

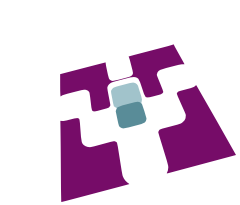
$Core \leftarrow \emptyset$
foreach $\boldsymbol{x}_i \in \boldsymbol{D}$ **do** // Find the core points
    Compute $N_\epsilon(\boldsymbol{x}_i)$
    $id(\boldsymbol{x}_i) \leftarrow 0$ // cluster id for $\boldsymbol{x}_i$
    if $N_\epsilon(\boldsymbol{x}_i) \geq minpts$ **then** $Core \leftarrow Core \cup \{\boldsymbol{x}_i\}$
$k \leftarrow 0$ // cluster id

foreach $\boldsymbol{x}_i \in Core$, *such that* $id(\boldsymbol{x}_i) = 0$ **do**
    $k \leftarrow k + 1$
    $id(\boldsymbol{x}_i) \leftarrow k$ // assign $\boldsymbol{x}_i$ to cluster id k
    DensityConnected $(\boldsymbol{x}_i, k)$
$\mathcal{C} \leftarrow \{C_i\}_{i=1}^{k}$, where $C_i \leftarrow \{\boldsymbol{x} \in \boldsymbol{D} \mid id(\boldsymbol{x}) = i\}$
$Noise \leftarrow \{\boldsymbol{x} \in \boldsymbol{D} \mid id(\boldsymbol{x}) = \emptyset\}$
$Border \leftarrow \boldsymbol{D} \setminus \{Core \cup Noise\}$
return $\mathcal{C}$, *Core*, *Border*, *Noise*

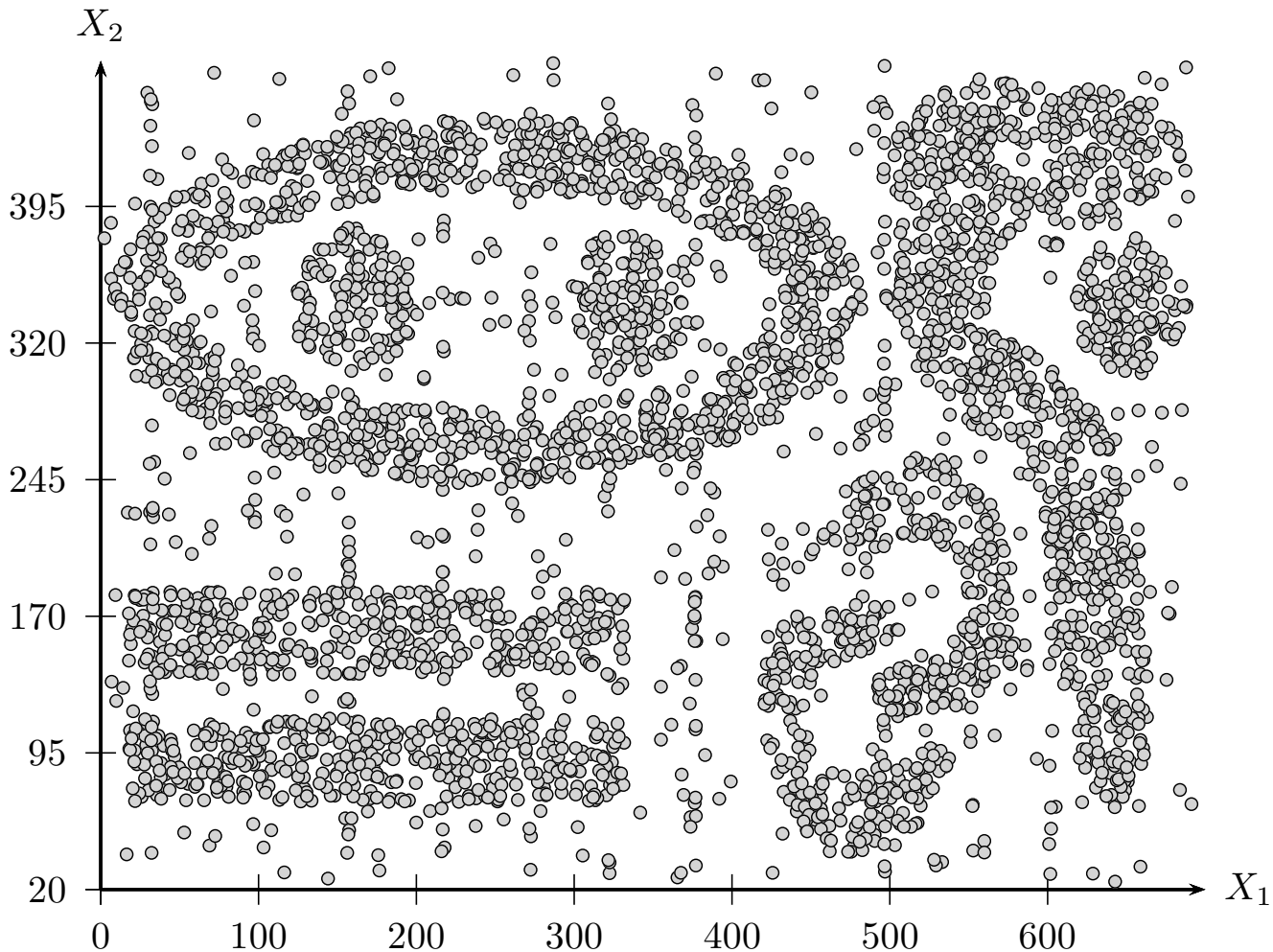
DensityConnected ($\boldsymbol{x}$, k)**:**

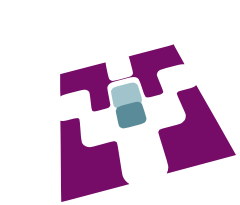
foreach $\boldsymbol{y} \in N_\epsilon(\boldsymbol{x})$ **do**
    $id(\boldsymbol{y}) \leftarrow k$ // assign $\boldsymbol{y}$ to cluster id k

    if $\boldsymbol{y} \in Core$ **then** DensityConnected $(\boldsymbol{y}, k)$



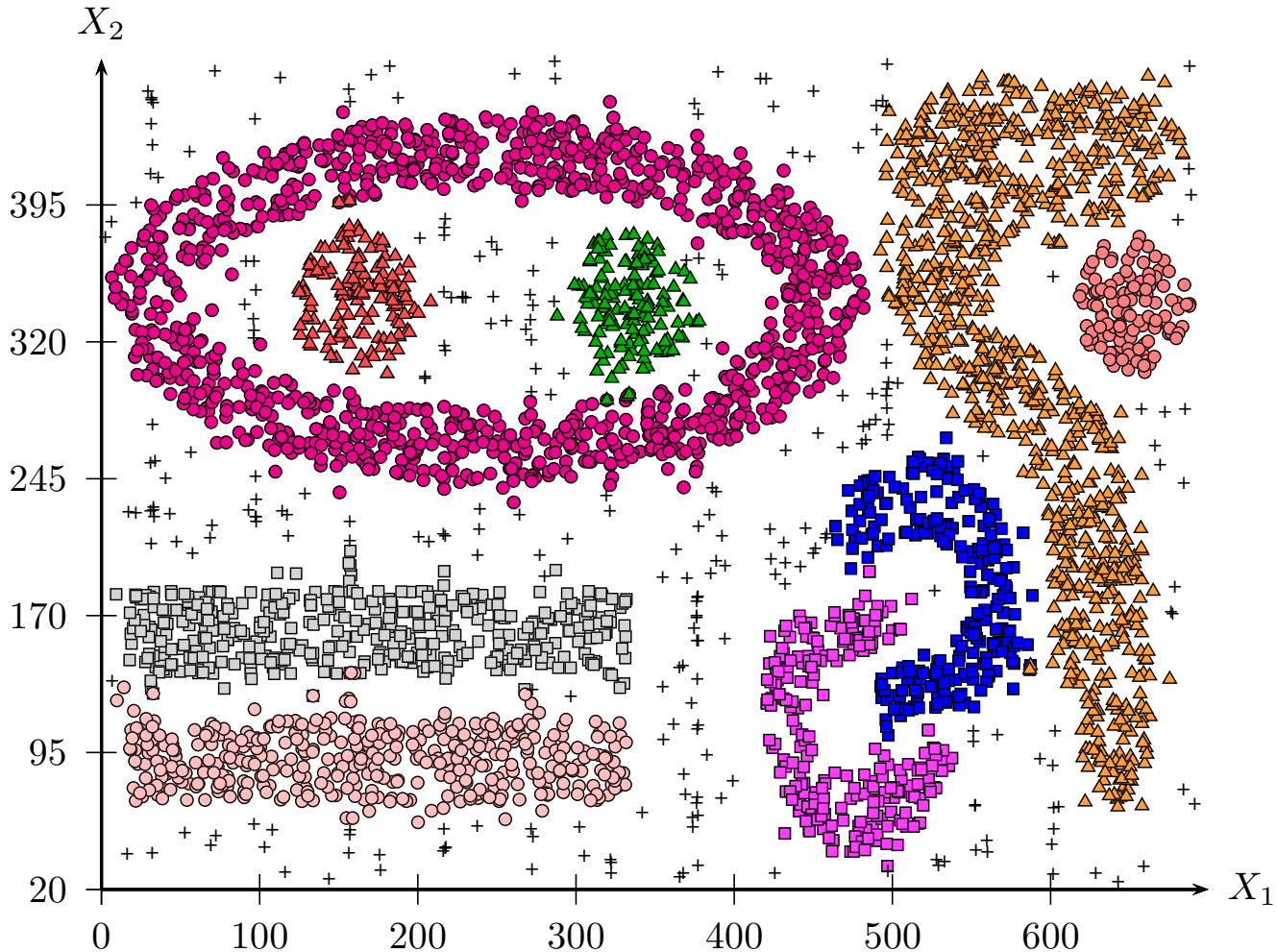
Density-based Clusters – Example

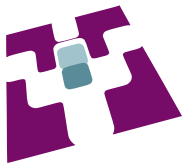




Density-based Clusters – Example

$\epsilon = 15$ and *minpts* $= 10$





References

- Arthur, D. and S. Vassilvitskii. K-means++: The advantages of careful seeding. In Proceedings of SODA'07, pages 1027–1035, USA, 2007.
- Kaufman, L. and Rousseeuw, P.J. Clustering by means of Medoids. ed. Y. Dodge, North-Holland, 1987, pp. 405–416.
- Shamir, O. and N. Tishby. Stability and model selection in k-means clustering. Machine Learning, 80(2):213–243, 2010.
- Cours Cnam RCP208 – Classification automatique `http://cedric.cnam.fr/vertigo/Cours/ml/`
- Chapters 13-15 : Data Mining and Machine Learning - Fundamental Concepts and Algorithms. Mohammed J. Zaki and Wagner Meira, JR. Cambridge University Press, 2020.
- Clustering Approaches – Marc Plantevit' slides (LIRIS)