Clustering

The clustering problem

- Intuitively, by clustering we mean the process that partitions a set of elements into disjoint subsets so that the following conditions are satisfied:
  (1) objects that are similar to each other belong to the same subset;
  (2) different elements belong to different subsets.

- Sometimes it is useful to obtain an initial intuition about the data by grouping them into meaningful subsets:
  - Computational biologists group genes according to the similarity of their expression patterns;
  - Retailers group customers according to their preferences in order to optimize marketing strategies;
  - Astronomers group stars according to their spatial proximity.

The clustering problem

- We will call the desired partition a **clustering** and each of its elements a **cluster**.

- We observe that the similarity relation is not transitive, whereas belonging to the same cluster is an equivalence relation.

- Consider the following example: given m elements $x_1, \ldots, x_m$ such that x_i is very similar to x_{i+1} for every i, but x_1 and x_m are very different from each other. If we want to make sure that two similar elements are in the same cluster, then we should consider a cluster containing all the elements $x_1, \ldots, x_m$, which, however, also contains two elements that are very different from each other, namely x_1 and x_m.
- Consider the following elements:

.....

.....

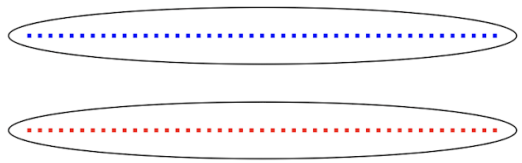
The clustering problem

- We will call the desired partition a **clustering** and each of its elements a **cluster**.

- We observe that the similarity relation is not transitive, whereas belonging to the same cluster is an equivalence relation.

- Consider the following example: given m elements $x_1, \ldots, x_m$ such that x_i is very similar to x_{i+1} for every i, but x_1 and x_m are very different from each other. If we want to make sure that two similar elements are in the same cluster, then we should consider a cluster containing all the elements $x_1, \ldots, x_m$, which, however, also contains two elements that are very different from each other, namely x_1 and x_m.

- A clustering that emphasizes not separating nearby points will be of the following type:

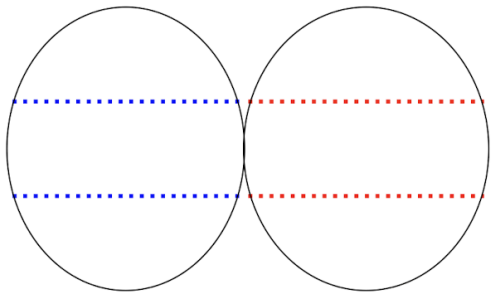


The clustering problem

- We will call the desired partition a **clustering** and each of its elements a **cluster**.

- We observe that the similarity relation is not transitive, whereas belonging to the same cluster is an equivalence relation.

- Consider the following example: given m elements $x_1, \ldots, x_m$ such that x_i is very similar to x_{i+1} for every i, but x_1 and x_m are very different from each other. If we want to make sure that two similar elements are in the same cluster, then we should consider a cluster containing all the elements $x_1, \ldots, x_m$, which, however, also contains two elements that are very different from each other, namely x_1 and x_m.
- A clustering that emphasizes separating distant points will be of the following type:



The clustering problem

- Another problem comes from the fact that clustering is an unsupervised problem, so a priori we do not know how the input data should be partitioned.
- The same set of points can be partitioned in several different ways depending on the similarity function used.
- Since we do not have a correct partition available with which to compare the result obtained by a given clustering algorithm, other measures, called "internal" measures, are used.
- Obtaining a high value for an internal measure does not necessarily imply that the information obtained from that result is truthful.

Clustering model

- **Input**:
  - A set of points $\mathcal{X}$ to be clustered;
  - a distance function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ that is symmetric and satisfies $d(x, x) = 0$ (possibly also the triangle inequality), or a similarity function $s : \mathcal{X} \times \mathcal{X} \to [0, 1]$ that is symmetric and such that $s(x, x) = 1$;
  - Some models also require as input the number $k \in \mathbb{N}$ of clusters into which the set must be partitioned.

- **Output**:
  - a clustering $\mathcal{C} = (\mathcal{C}_1, \ldots, \mathcal{C}_2)$ where $\mathcal{C}_i \subseteq \mathcal{X}$ with $\bigcap_i \mathcal{C}_i = \emptyset$ and $\bigcup_i \mathcal{C}_i = \mathcal{X}$.

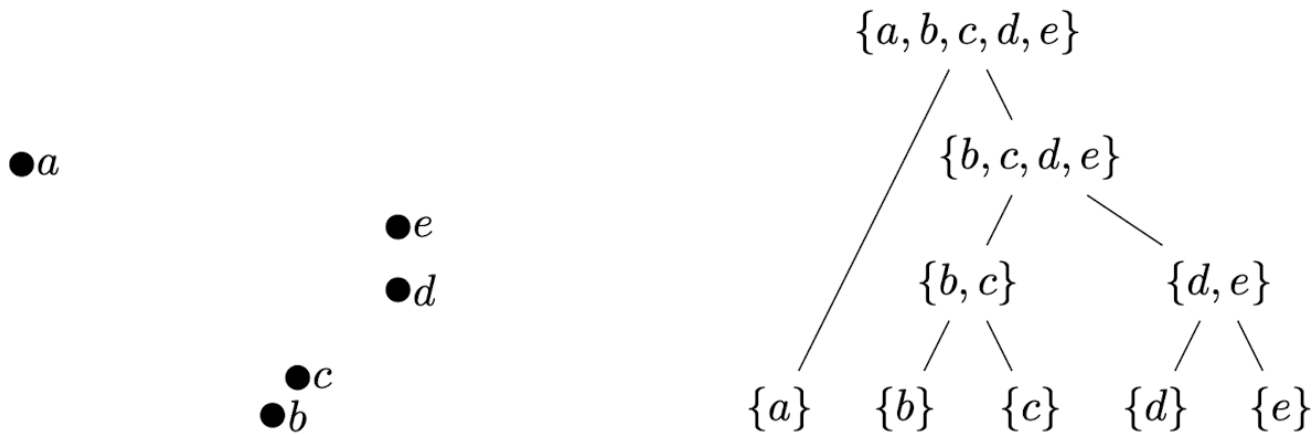
Linkage-based clustering algorithms

- These are the simplest clustering algorithms. The procedure proceeds in rounds:
  - We start from the trivial clustering in which each element of $\mathcal{X}$ forms a cluster;
  - Iteratively, the closest pairs of clusters are merged, obtaining another clustering with fewer clusters;
  - At the end, we obtain the trivial clustering consisting of a single cluster containing all the elements of $\mathcal{X}$.

- This type of algorithm requires two parameters:
  - A distance or similarity function between clusters;
  - A stopping criterion to terminate before obtaining the trivial clustering.

Representation using dendrograms

- If no stopping criterion is used, it is possible to represent a linkage-based algorithm using a dendrogram:
- The leaves correspond to the initial clustering and the root to the final clustering:



Choice of cluster distance and stopping criterion

- Starting from the function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$, it is possible to obtain a distance function between clusters $\mathcal{C}_1, \mathcal{C}_2$ that extends the function d:
  - **Single-Linkage** $D(\mathcal{C}_1, \mathcal{C}_2) = \min\{d(x, y) : x \in \mathcal{C}_1, y \in \mathcal{C}_2\}$
  - **Average-Linkage** $D(\mathcal{C}_1, \mathcal{C}_2) = \frac{1}{|\mathcal{C}_1||\mathcal{C}_2|} \sum_{x \in \mathcal{C}_1, y \in \mathcal{C}_2} d(x, y)$
  - **Max-Linkage** $D(\mathcal{C}_1, \mathcal{C}_2) = \max\{d(x, y) : x \in \mathcal{C}_1, y \in \mathcal{C}_2\}$

Choice of cluster distance and stopping criterion

- Starting from the function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$, it is possible to obtain a distance function between clusters $\mathcal{C}_1, \mathcal{C}_2$ that extends the function d:
  - **Single-Linkage** $D(\mathcal{C}_1, \mathcal{C}_2) = \min\{d(x, y) : x \in \mathcal{C}_1, y \in \mathcal{C}_2\}$
  - **Average-Linkage** $D(\mathcal{C}_1, \mathcal{C}_2) = \frac{1}{|\mathcal{C}_1||\mathcal{C}_2|} \sum_{x \in \mathcal{C}_1, y \in \mathcal{C}_2} d(x, y)$
  - **Max-Linkage** $D(\mathcal{C}_1, \mathcal{C}_2) = \max\{d(x, y) : x \in \mathcal{C}_1, y \in \mathcal{C}_2\}$
- As a stopping criterion, one of the following is used:
  - Fix the number of clusters k; once this number is reached, the algorithm stops;
  - Fix an upper bound for the distance between clusters: given $r \in \mathbb{R}^+$, the algorithm stops once all pairwise distances between clusters are greater than r.

Clustering through cost minimization

- Another widely used approach consists of defining a parametric set of clusterings on the set $\mathcal{X}$ and a cost function on that set.
- The objective is to find the clustering with minimum cost among all those belonging to the parametric set.
- More precisely, given a set $\mathcal{X}$, a distance function d on it, and a clustering $\mathcal{C}$, we define a function $G(\mathcal{X}, d, \mathcal{C})$ and we seek its minimum.
- Many of the most commonly used functions G make the corresponding optimization problem very difficult to solve (NP-hard).

Examples of cost functions

- Consider a set $\mathcal{X}$ embedded in a metric space $(\mathcal{X}', d)$ and the centroids of a given clustering $\mathcal{C}$ of $\mathcal{X}$:

$$\mu_i(\mathcal{C}_i) = \arg\min_{\mu \in \mathcal{X}'} \sum_{x \in \mathcal{C}_i} d(x, \mu)^2$$

- **k-means clustering**: this is one of the most widely used cost functions and consists of minimizing the distance of each point from the centroid of the cluster to which it is assigned.

$$\begin{aligned} G_{k\text{-means}}(\mathcal{X}, d, \mathcal{C}) &= \sum_{i=1}^{k} \sum_{x \in \mathcal{C}_i} d(x, \mu_i)^2 \\ &= \min_{\mu_1, \ldots, \mu_k \in \mathcal{X}'} \sum_{i=1}^{k} \sum_{x \in \mathcal{C}_i} d(x, \mu_i)^2 \end{aligned}$$

Examples of cost functions

- **k-medoids clustering**: this is similar to the k-means function, but the centroids are constrained to belong to the set $\mathcal{X}$:

$$G_{k\text{-medoids}}(\mathcal{X}, d, \mathcal{C}) = \min_{\mu_1,\ldots,\mu_k \in \mathcal{X}} \sum_{i=1}^{k} \sum_{x \in \mathcal{C}_i} d(x, \mu_i)^2$$

- **k-median clustering**: as in k-medoids, but the distance is considered without the square:

$$G_{k\text{-median}}(\mathcal{X}, d, \mathcal{C}) = \min_{\mu_1,\ldots,\mu_k \in \mathcal{X}} \sum_{i=1}^{k} \sum_{x \in \mathcal{C}_i} d(x, \mu_i)$$

The k-means algorithm

- Solving the optimization problem for k-means clustering is very complicated.
- The following algorithm finds an approximate solution to the k-means problem:

Algorithm k-means algorithm

---

input: $\mathcal{X}$, k
initialization: choose random centroids $\mu_1, \ldots, \mu_k$
while The clusters $\mathcal{C}_i$ do not change **do**
  $\forall i \in \{1, \ldots, k\} \quad \mathcal{C}_i = \{x \in \mathcal{X} | i = \arg\min_j \|x - \mu_j\|\}$
  $\forall i \in \{1, \ldots, k\}$ update $\mu_i = \frac{1}{|\mathcal{C}_i|} \sum_{x \in \mathcal{C}_i} x$
end while

The k-means algorithm

- Solving the optimization problem for k-means clustering is very complicated.
- The following algorithm finds an approximate solution to the k-means problem:

Algorithm k-means algorithm

---

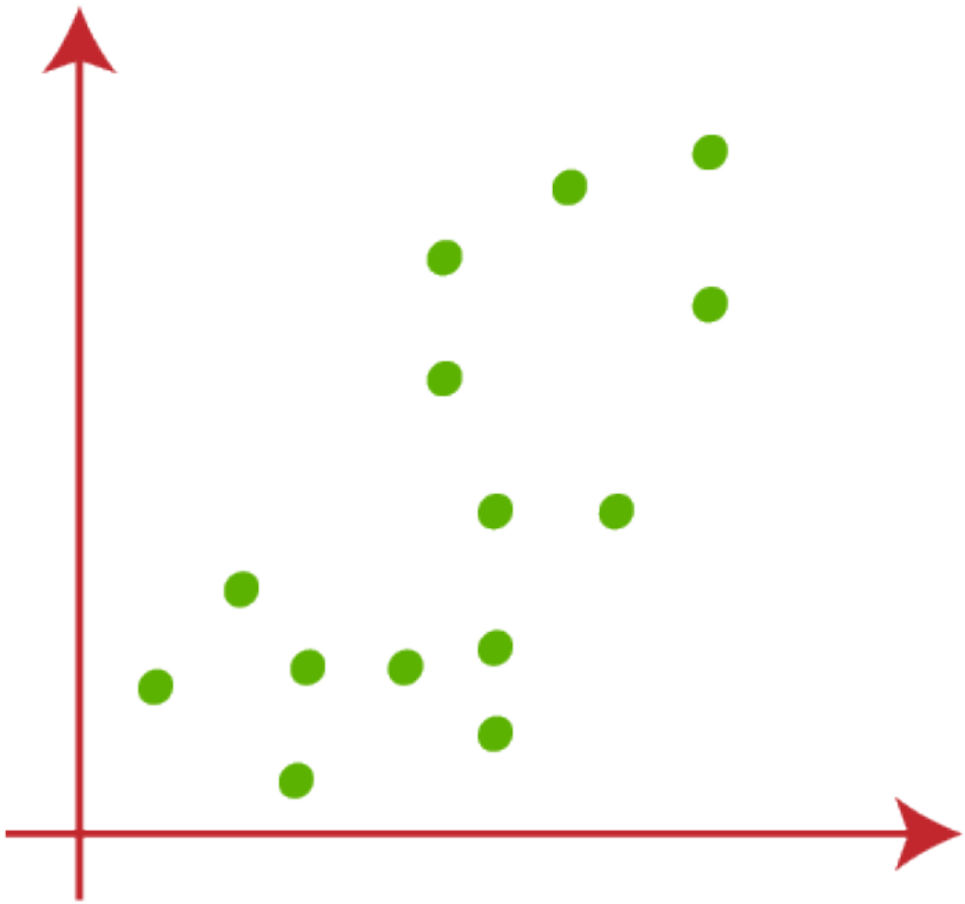
input: $\mathcal{X}$, k
initialization: choose random centroids $\mu_1, \ldots, \mu_k$
while The clusters $\mathcal{C}_i$ do not change **do**
  $\forall i \in \{1, \ldots, k\} \quad \mathcal{C}_i = \{x \in \mathcal{X} | i = \arg\min_j \|x - \mu_j\|\}$
  $\forall i \in \{1, \ldots, k\}$ update $\mu_i = \frac{1}{|\mathcal{C}_i|} \sum_{x \in \mathcal{C}_i} x$
end while

Lemma

Each iteration of the k-means algorithm does not increase the objective function $G_{k\text{-}means}$.

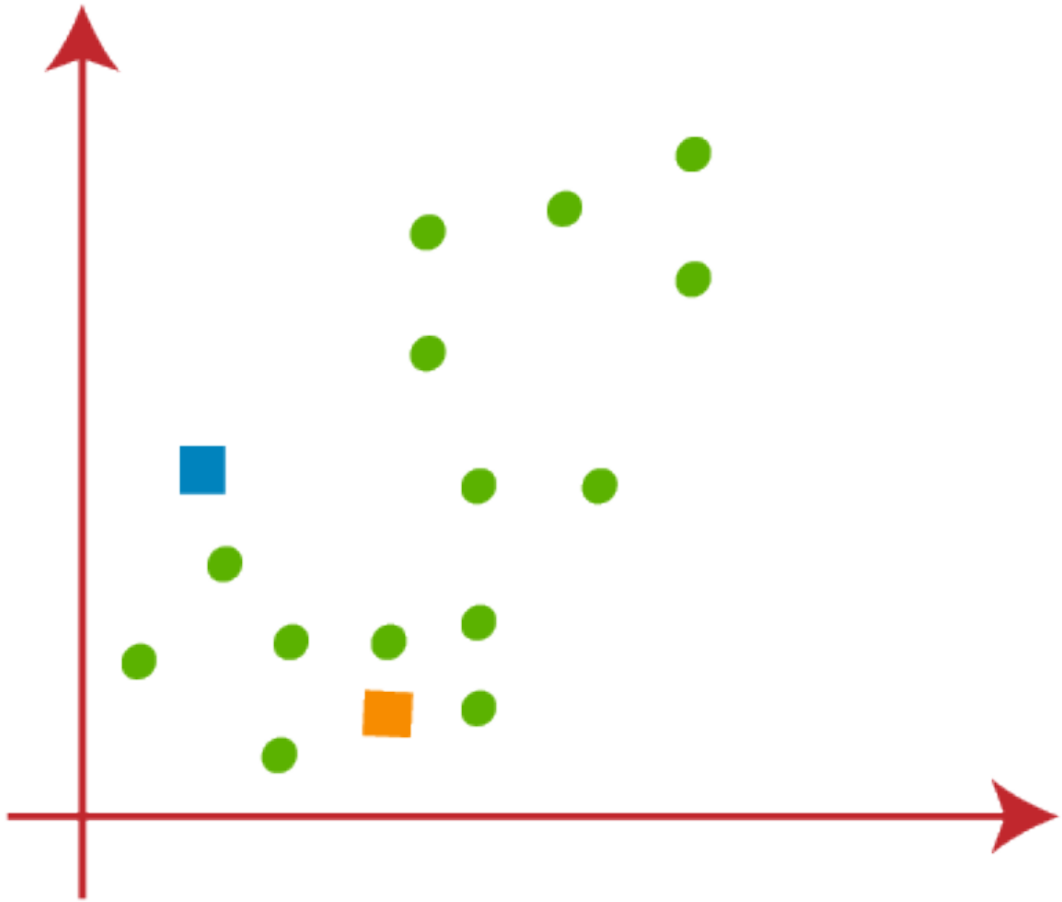
Example

- Consider the following set of points to which we want to apply the k-means algorithm:



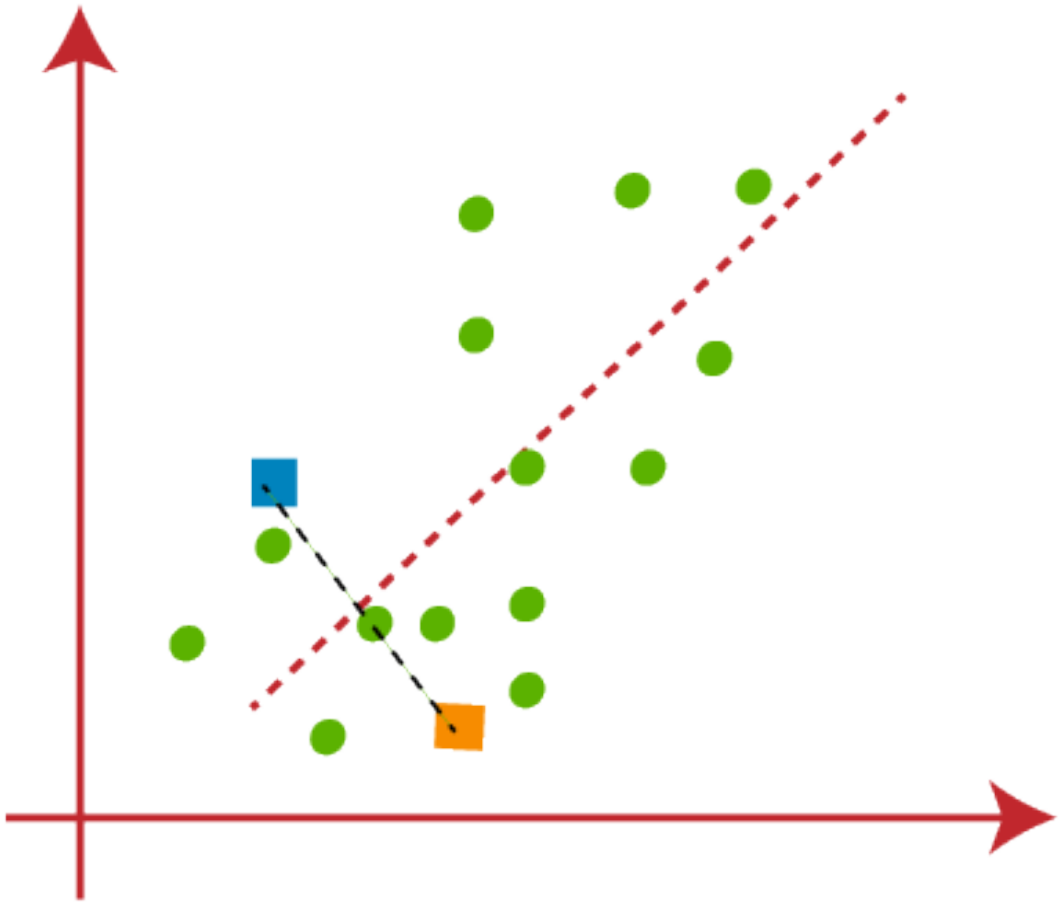
Example

- We choose two initial centroids.



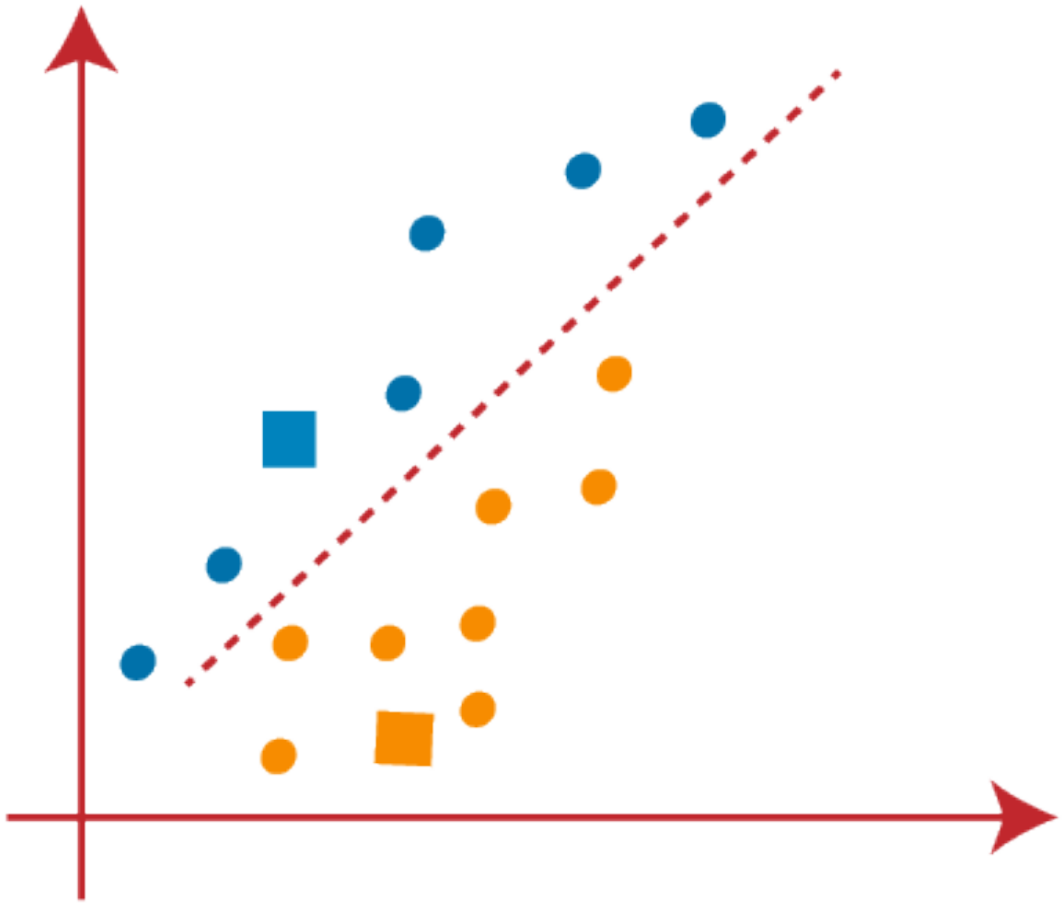
Example

- We divide the space into two regions according to the distance from the two chosen centroids.



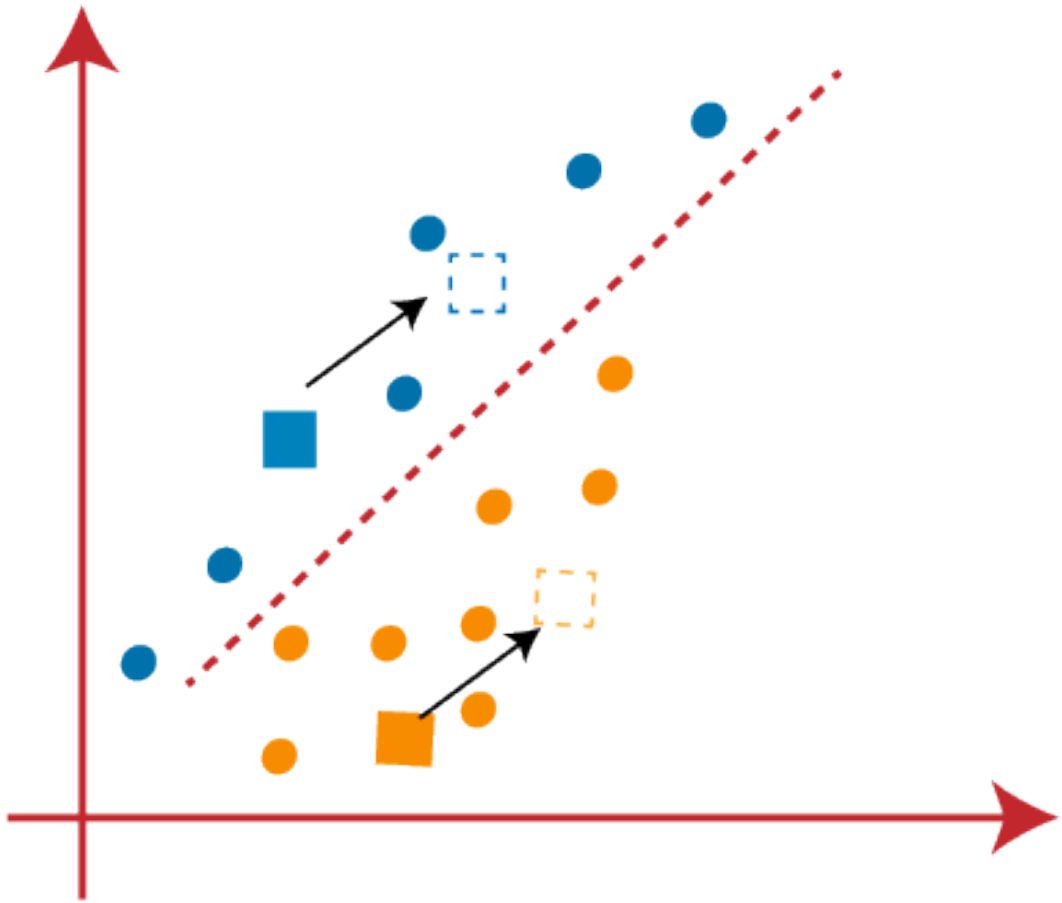
Example

- We assign the points to two clusters according to their distance from the two centroids.



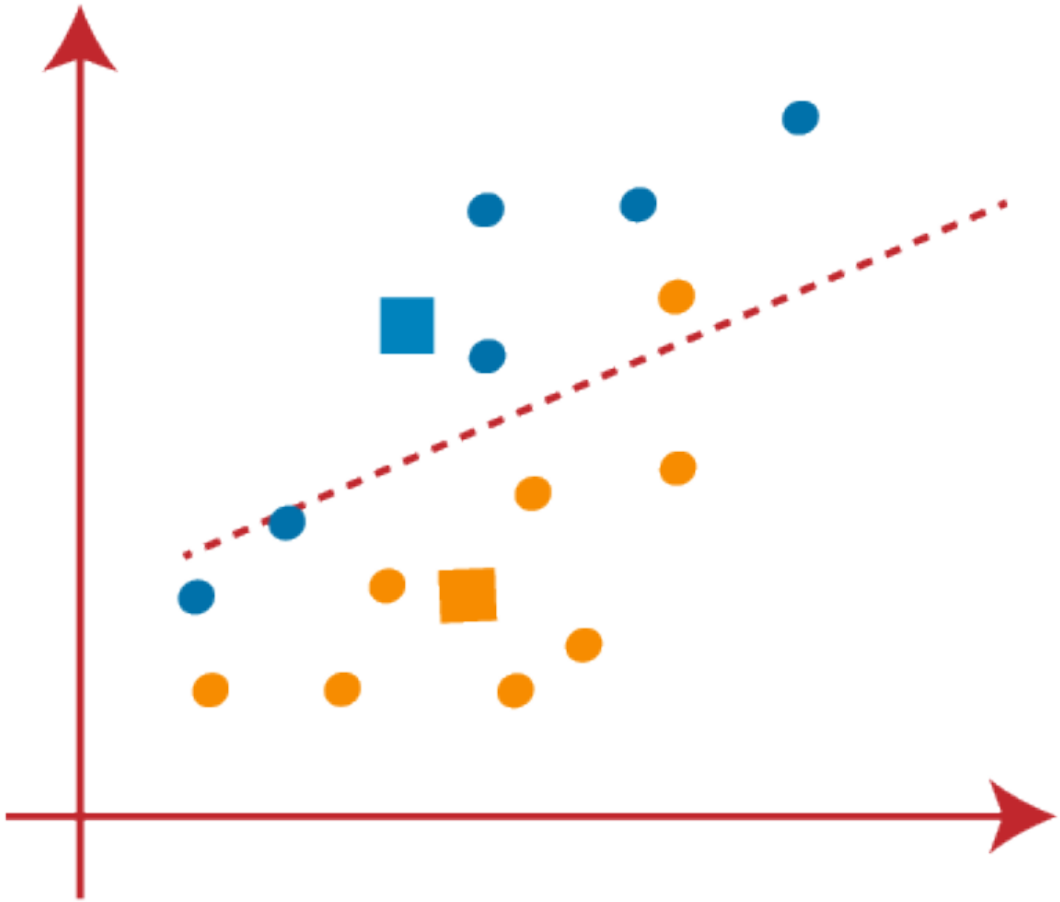
Example

- We update the centroids by computing them as the mean of the elements of the two clusters.



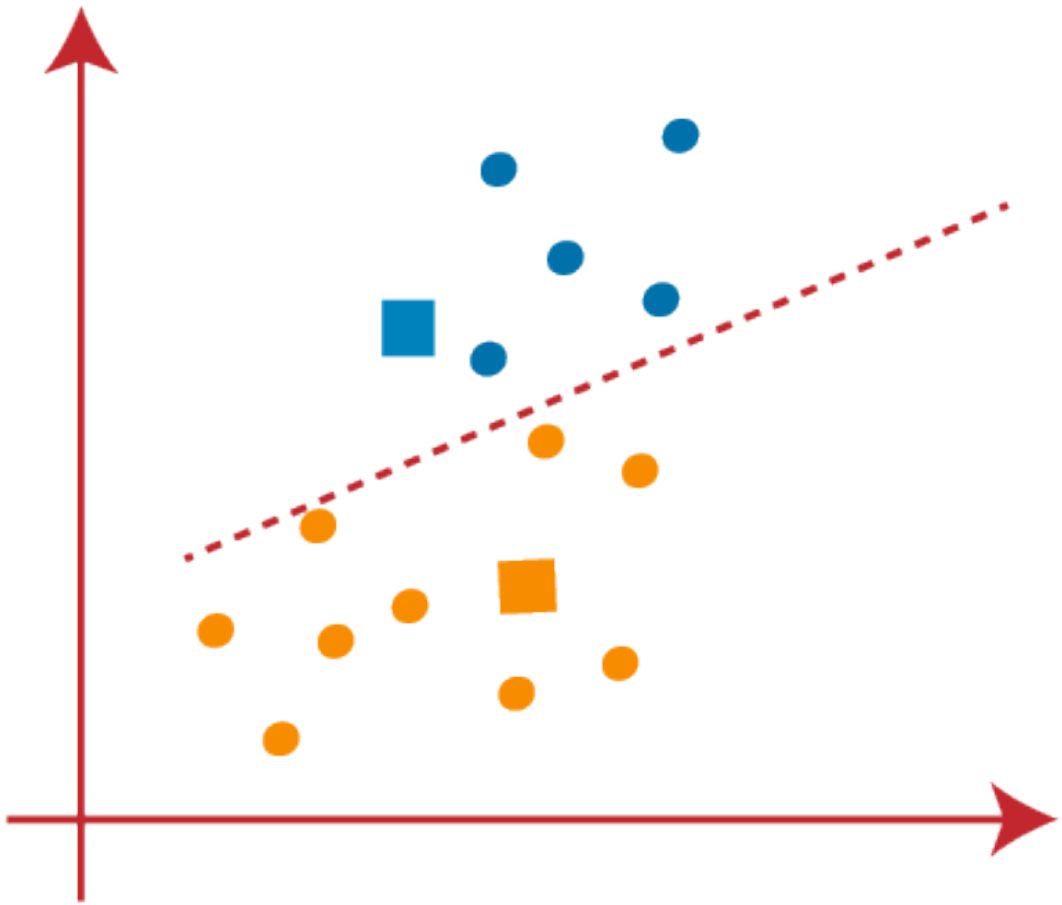
Example

- We divide the space again using the distance from the two new centroids as the criterion.



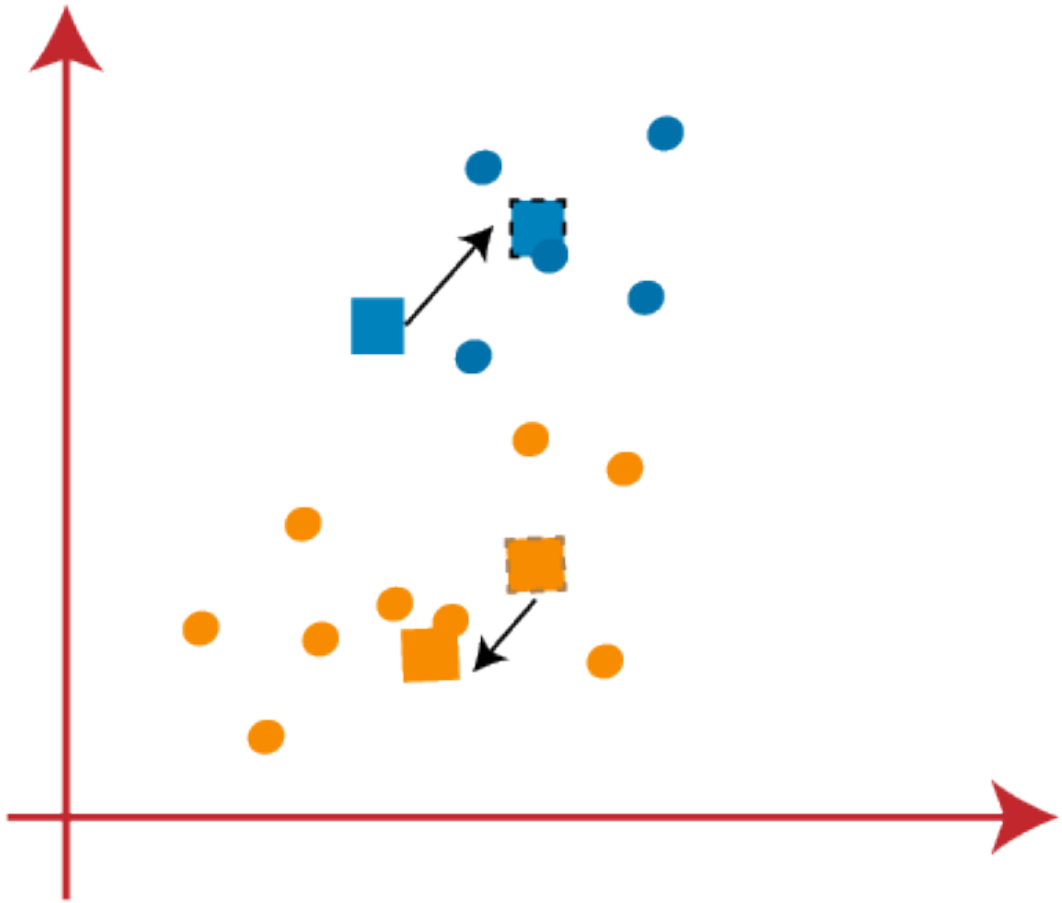
Example

- We assign the points to clusters using the same criterion.



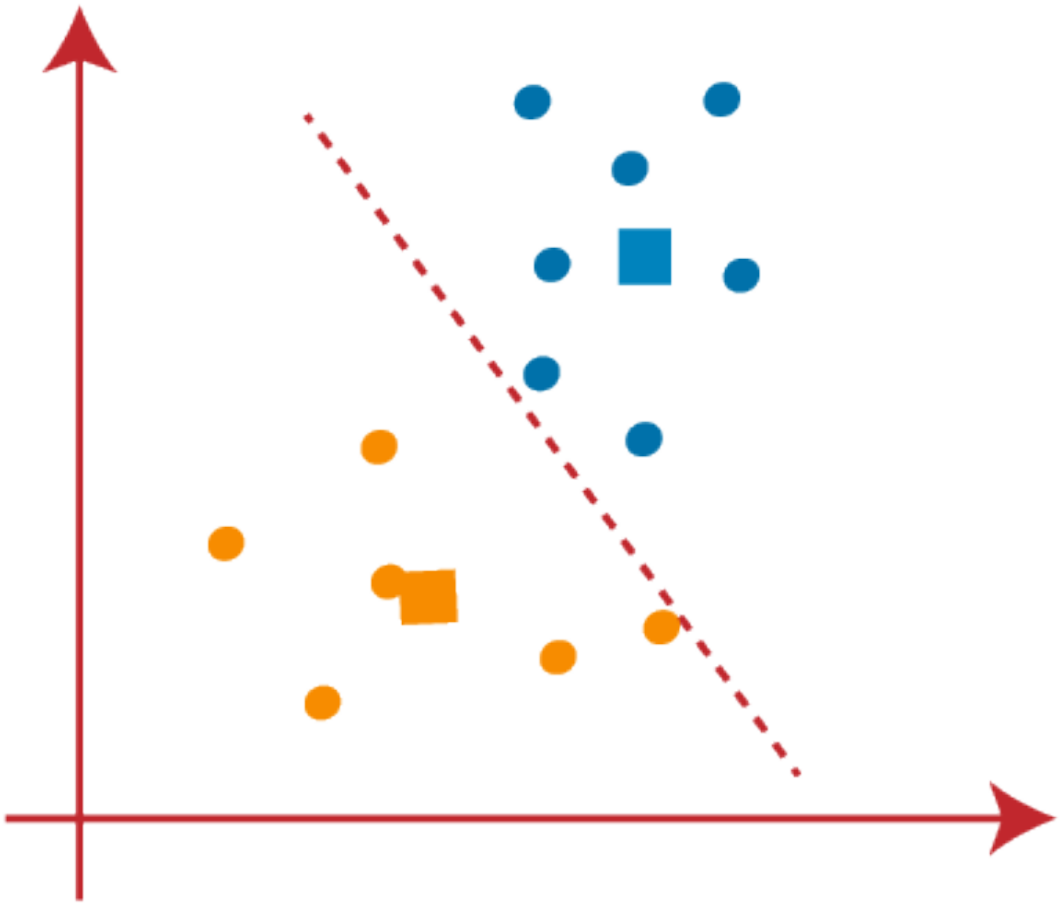
Example

- We update the centroids again.



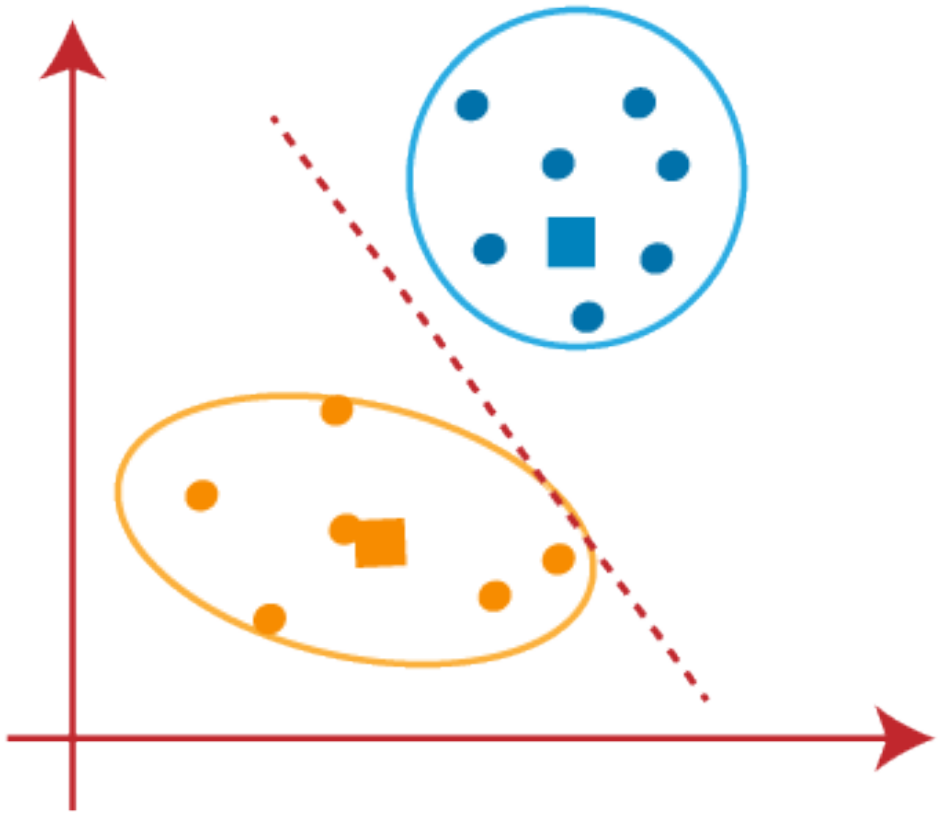
Example

- We divide the space according to the distance from the new centroids.



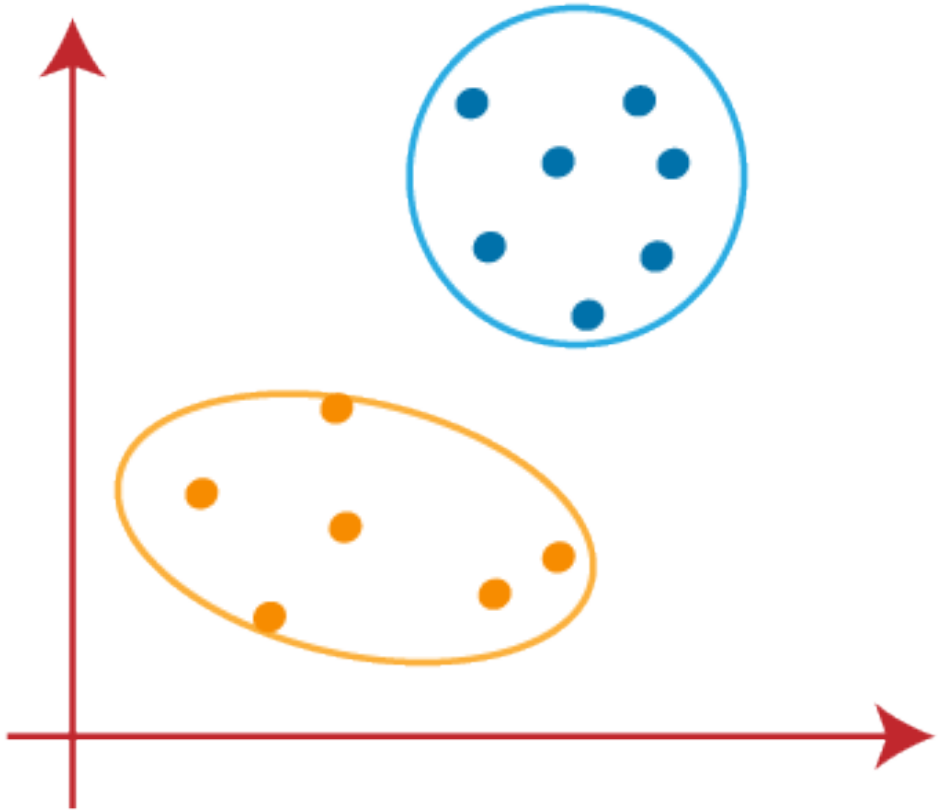
Example

- We update the clusters.



Example

- If there are no changes in the clusters from the previous iteration, we terminate.



Spectral clustering

- Sometimes it is convenient to represent the relationships between the elements of $\mathcal{X}$ using a similarity graph $G(V, E, s)$, where
  - V = set of vertices = $\mathcal{X}$;
  - E = set of edges = $\mathcal{X} \times \mathcal{X}$;
  - s = similarity function $\mathcal{X} \times \mathcal{X} \to [0, 1]$;
  - $W \in \mathbb{R}^{m \times m}$ similarity matrix (or weight matrix) with $W_{i,j} = s(x_i, x_j)$.

Spectral clustering

- Sometimes it is convenient to represent the relationships between the elements of $\mathcal{X}$ using a similarity graph $G(V, E, s)$, where
  - V = set of vertices = $\mathcal{X}$;
  - E = set of edges = $\mathcal{X} \times \mathcal{X}$;
  - s = similarity function $\mathcal{X} \times \mathcal{X} \to [0, 1]$;
  - $W \in \mathbb{R}^{m \times m}$ similarity matrix (or weight matrix) with $W_{i,j} = s(x_i, x_j)$.
- The goal is to find a partition of the graph such that edges between different clusters have small weights and edges within the same cluster have large weights.
- This type of algorithm focuses on generating clusterings in which elements belonging to different clusters are different from each other.

Graph cuts

- Given a graph $G(V, E, s)$ and its similarity matrix W, the most direct method for obtaining a partition is to solve the min-cut problem:

$$\mathcal{C} = \arg\min \operatorname{cut}(\mathcal{C}_1, \ldots, \mathcal{C}_k) = \arg\min \sum_{i=1}^{k} \sum_{x \in \mathcal{C}_i, y \notin \mathcal{C}_i} W_{x,y}$$

- For $k = 2$, a solution can be obtained efficiently; however, the solution tends to separate individual vertices from the rest of the graph.

Graph cuts

- Given a graph $G(V, E, s)$ and its similarity matrix W, the most direct method for obtaining a partition is to solve the min-cut problem:

$$\mathcal{C} = \arg\min \operatorname{cut}(\mathcal{C}_1, \ldots, \mathcal{C}_k) = \arg\min \sum_{i=1}^{k} \sum_{x \in \mathcal{C}_i, y \notin \mathcal{C}_i} W_{x,y}$$

- For $k = 2$, a solution can be obtained efficiently; however, the solution tends to separate individual vertices from the rest of the graph.
- To overcome this problem, the following normalization of the objective function is used:

$$\text{Ratiocut}(\mathcal{C}_1, \ldots, \mathcal{C}_k) = \sum_{i=1}^{k} \frac{1}{|\mathcal{C}_i|} \sum_{x \in \mathcal{C}_i, y \notin \mathcal{C}_i} W_{x,y}$$

- Minimizing the Ratiocut function is computationally difficult. Spectral clustering makes it possible to relax the problem and obtain an approximate solution.

Graph Laplacian and graph-cut relaxation

- The tool used for spectral clustering is the **Laplacian matrix** of a graph:

Definition

Given a graph $G(V, E, s)$ and its similarity matrix W, the unnormalized Laplacian of G is the $m \times m$ matrix

$$L = D - W$$

where D is a diagonal matrix called the degree matrix, defined as

$$D_{i,i} = \sum_{j=1}^{m} W_{i,j}.$$

Graph Laplacian and graph-cut relaxation

- The following lemma highlights the relationship between the Laplacian and the Ratiocut function.

Lemma

Let $\mathcal{C}$ be a clustering and let $H \in \mathbb{R}^{m \times k}$ be the matrix defined as

$$H_{i,j} = \frac{1}{\sqrt{|\mathcal{C}_j|}} \mathbf{1}_{i \in \mathcal{C}_j}.$$

Then the columns of H are orthonormal and

$$Ratiocut(\mathcal{C}) = \mathsf{trace}(H^\top L H).$$

Unnormalized spectral clustering

- To minimize Ratiocut, we can look for a matrix H whose columns are orthonormal and such that

$$H_{i,j} = \begin{cases} 0 & \text{if} \quad x_i \notin \mathcal{C}_j \\ \frac{1}{\sqrt{|\mathcal{C}_j|}} & \text{if} \quad x_i \in \mathcal{C}_j \end{cases}$$

- The corresponding optimization problem is an integer optimization problem, and is therefore difficult to solve.

Unnormalized spectral clustering

- To minimize Ratiocut, we can look for a matrix H whose columns are orthonormal and such that

$$H_{i,j} = \begin{cases} 0 & \text{if} \quad x_i \notin \mathcal{C}_j \\ \frac{1}{\sqrt{|\mathcal{C}_j|}} & \text{if} \quad x_i \in \mathcal{C}_j \end{cases}$$

- The corresponding optimization problem is an integer optimization problem, and is therefore difficult to solve.
- We relax the problem and look for an orthonormal matrix H that minimizes $\mathsf{trace}(H^\top L H)$.

Algorithm Unnormalized spectral clustering

---

input: $W \in \mathbb{R}^{m,m}, k$
initialization: compute the unnormalized Laplacian L
Let $U \in \mathbb{R}^{m,k}$ be the matrix whose columns are the eigenvectors of L corresponding to the k smallest eigenvalues of L
Let $u_1, \ldots, u_m$ be the rows of U
Apply k-means to the points $u_1, \ldots, u_m$ to obtain a clustering $\mathcal{C}$

Spatial density-based clustering

- Another type of clustering algorithm separates the elements of $\mathcal{X}$ according to their mutual distance.
- This type of algorithm makes it possible to identify outlier elements.
- As an example, consider the DBSCAN algorithm (Density-Based Spatial Clustering of Applications with Noise).
- DBSCAN depends on two parameters:
  - MinPts = the minimum number of points around a given element x for a circular region centered at x to be considered dense;
  - ε = the radius of the circular region centered at x under consideration.

DBSCAN algorithm

- The DBSCAN algorithm divides the elements of $\mathcal{X}$ into three categories:
  - Core points: points x such that a ball of radius ε centered at x contains at least MinPts elements inside it;
  - Border points: elements that are not core points, but that belong to the ball of radius ε centered at a core point;
  - Noise points: elements that are neither core points nor border points.
- We will also use the following notions:
  - An element x is said to be directly density-reachable from a core point y if x belongs to the ball of radius ε centered at y;
  - An element x is density-connected to an element y if there exists a chain of elements $z_1, \ldots, z_m$ with $z_1 = y$ and $z_m = x$ such that z_{i+1} is directly density-reachable from z_i.

DBSCAN algorithm

- The following pseudocode summarizes the DBSCAN algorithm.

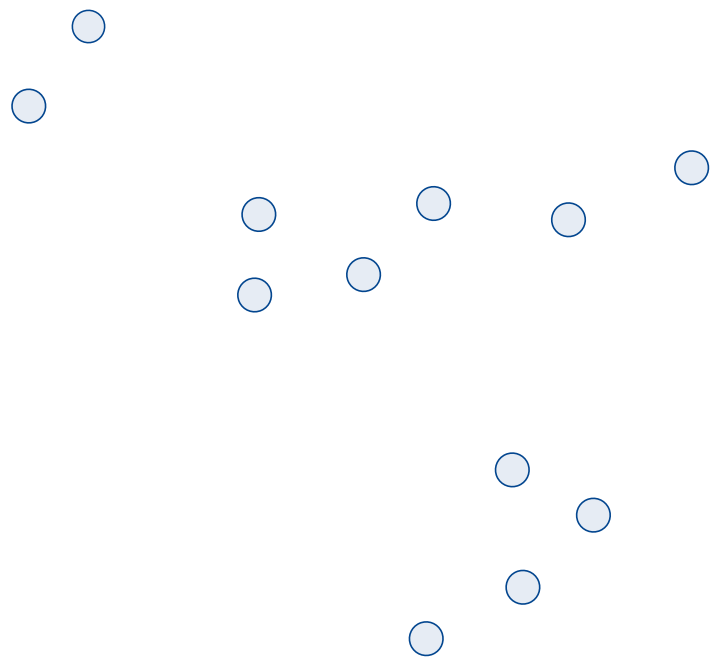
Algorithm DBSCAN algorithm

Determine all core, border, and noise points
for x_i core point **do**
  create a new cluster $\mathcal{C}_i$
  add to $\mathcal{C}_i$ all points that are not assigned to any other cluster and that are density-connected to the current points of $\mathcal{C}_i$
end for
assign all border points to the cluster corresponding to the nearest core point

---

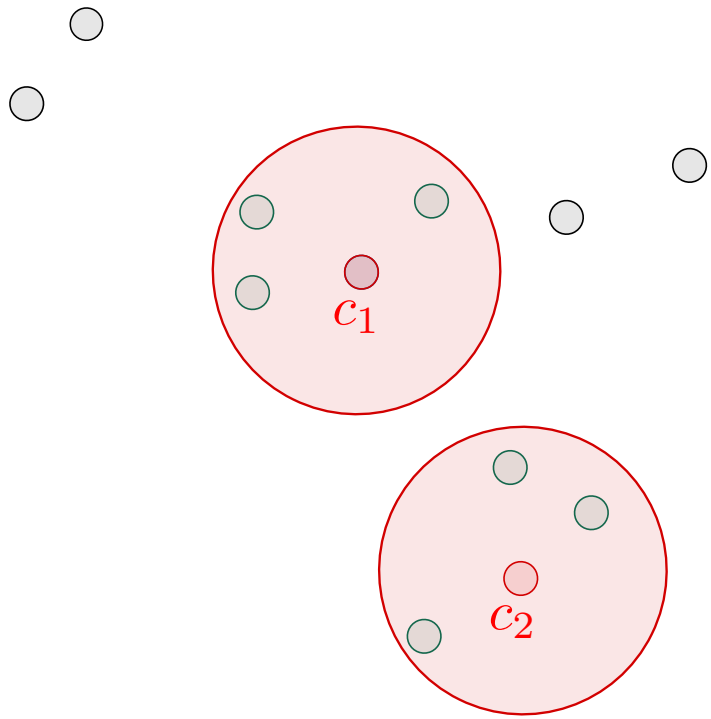
Example

- Consider the following set of points to which we apply the DBSCAN algorithm with $\mathsf{MinPts} = 3$ and $\varepsilon = 1$.



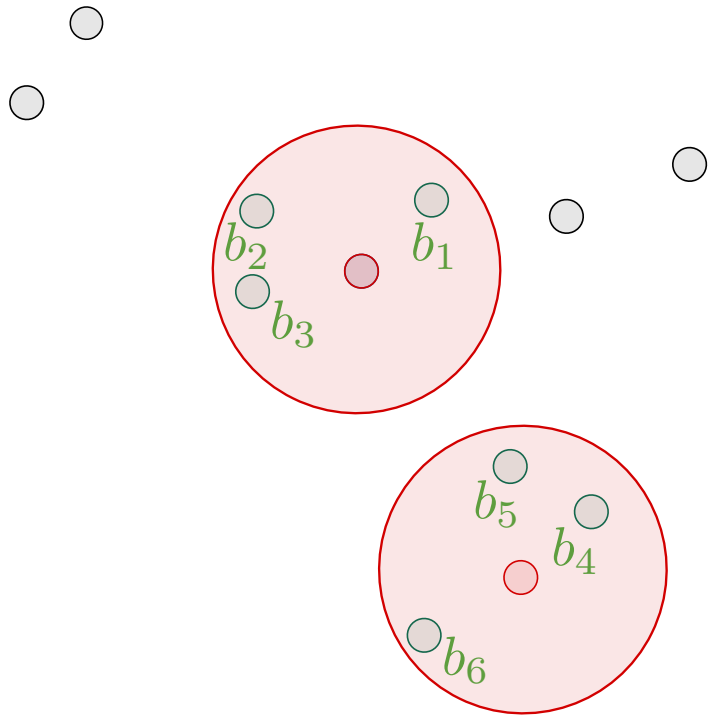
Example

- We identify the core points, that is, the points c such that a ball of radius ε centered at c contains at least 3 other points.



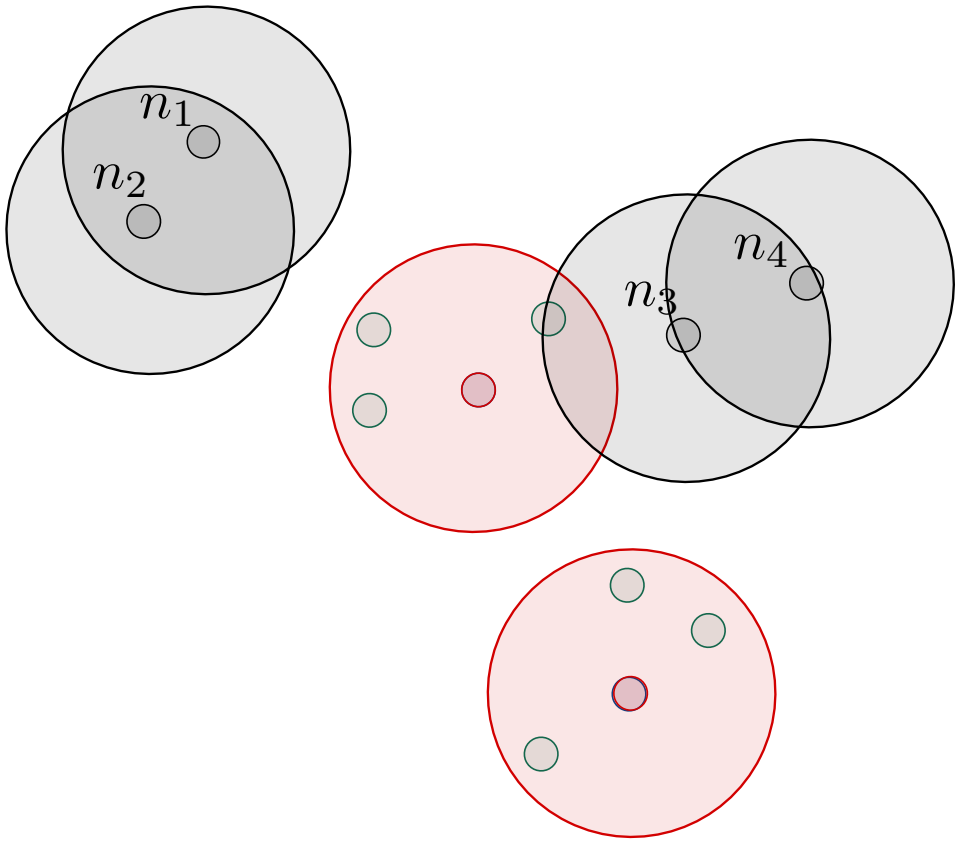
Example

- Border points, instead, are points that are not core points but belong to the circular neighborhood of a core point.



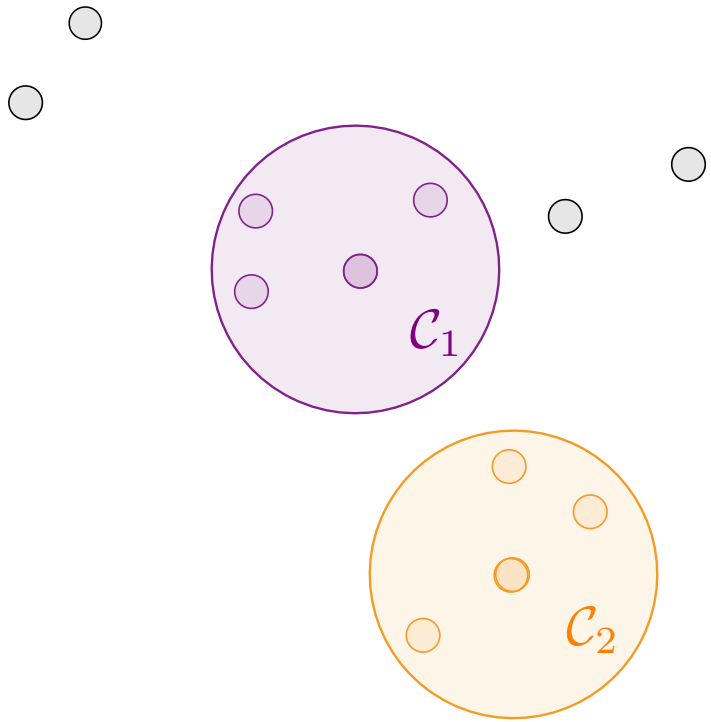
Example

- Noise points are those points that are neither core points nor border points.



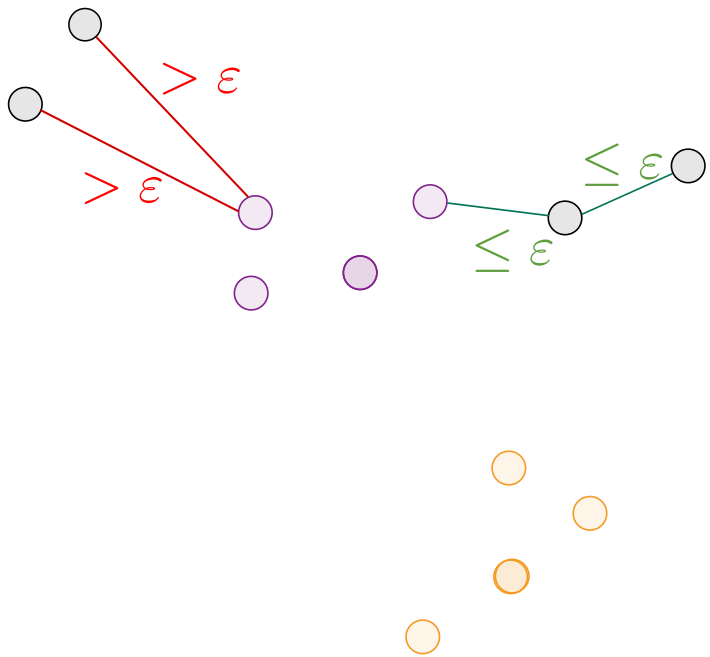
Example

- For each core point c, we build a new cluster $\mathcal{C}$ and first assign to it all the points that belong to the circular neighborhood centered at c with radius ε.



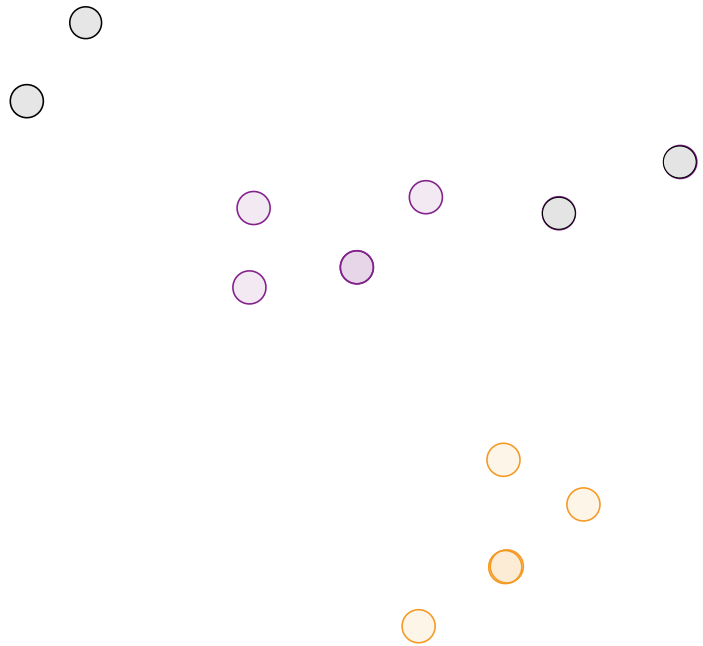
Example

- For each point that has not been assigned to any cluster, we check whether it is density-connected to some point assigned to one of the clusters.



Example

- All points that have not been assigned to any cluster remain as noise points, and any border points that have not been assigned are assigned to the cluster corresponding to the nearest core point.



Measures for evaluating clustering quality

- To evaluate the quality of a clustering, we cannot use measures that depend on a priori information; instead, we can only use internal measures that depend on the clustering we obtained.
- Some of the main measures used are:
  - The Davies-Bouldin index, which measures the ratio between distances between different clusters and distances between elements belonging to the same cluster:

$$DB = \frac{1}{k} \sum_{i=1}^{k} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(\mu_i, \mu_j)} \right)$$

    where σ_i is the average distance of the points belonging to cluster $\mathcal{C}_i$.

Measures for evaluating clustering quality

- The Silhouette coefficient measures how similar elements belonging to the same cluster are compared with elements belonging to different clusters. For each cluster $\mathcal{C}_i$ and for each element $x_j \in \mathcal{C}_i$, we define two quantities

$$a(x_j) = \frac{1}{|\mathcal{C}_i| - 1} \sum_{x_l \in \mathcal{C}_i, l \neq j} d(x_j, x_l), \quad b(x_j) = \min_{r \neq i} \frac{1}{|\mathcal{C}_r|} \sum_{x_l \in \mathcal{C}_r} d(x_j, x_l)$$

  and combine them to obtain

$$s(x_j) = \begin{cases} 1 - \frac{a(x_j)}{b(x_j)} & a(x_j) < b(x_j) \\ 0 \quad a(x_j) = b(x_j) & \\ \frac{b(x_j)}{a(x_j)} - 1 & a(x_j) > b(x_j) \end{cases}$$

- The Silhouette coefficient is defined as

$$SC = \max_i \tilde{s}(i)$$

  where $\tilde{s}(i)$ is the average of all values $s(x_j)$ for $x_j \in \mathcal{C}_i$.