Clustering

Organizing data points into **similar groups**

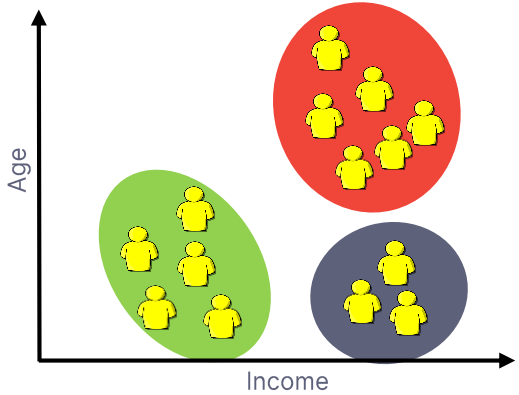
Faculty of Mathematics and Computer Science, University of Bucharest
and
Sparktech Software

Academic Year 2018/2019, 1st Semester

Reminder – Unsupervised Learning

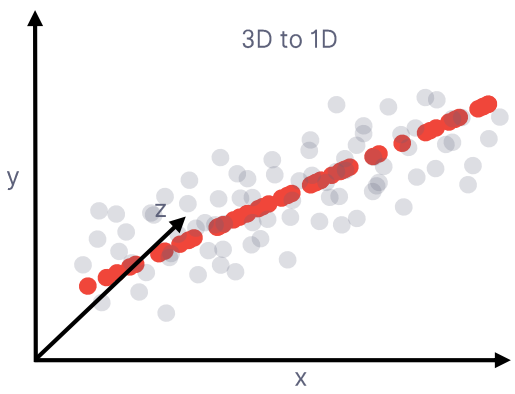
- Unlike *Supervised Learning*, there is no expected label in the training phase.
- We are trying to **discover structure** in the data.

Clustering
Grouping *similar* items



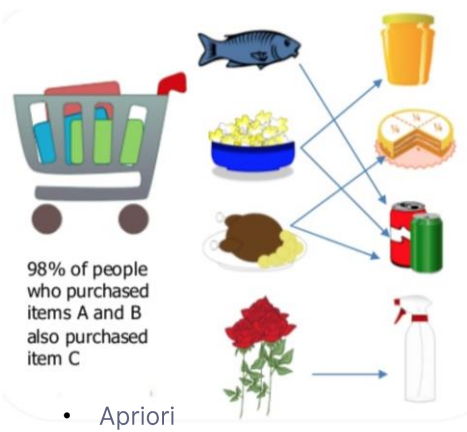
- K-means
- DBSCAN
- Hierarchical Clustering

Dimensionality Reduction
Reducing the number of features



- Principal Component Analysis (PCA)
- T-SNE
- Self-organizing Maps (SOMs)

Association Rule Mining
Finding frequent patterns and correlations



- Apriori
- FP-growth

Clustering

- **Clustering** of **Cluster Analysis** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are *more similar* to each other than to objects in other groups.
- There are several types of models:
  - *Centroid-based* – Each cluster is represented by a prototype point (a “center”) (e.g. K-means)
  - *Density-based* – Clusters a considered denser regions of space. (e.g. DBSCAN)
  - *Hierarchical models* – There is a hierarchical relationship between clusters.
  - *Distribution models* – Clusters are modeled using statistical distributions. (e.g. GMM)
  - *Graph-based* – Clusters are considered cliques in a graph (e.g. HCS)
  - *Others*
- Based on the relation between objects and clusters:
  - *Hard-clustering* (or *Partitioning*) – One point can only be in one cluster.
  - *Soft-clustering* (*or Fuzzy-clustering*) – A point can have a degree of membership in multiple clusters.

K-Means

Centroid-based partitioning into a **fixed** number of clusters

Faculty of Mathematics and Computer Science, University of Bucharest
and
Sparktech Software

Academic Year 2018/2019, 1st Semester

K-means

- **K-means** is a clustering algorithm which aims to *partition* the data points into a *fixed* number of clusters *k*.
- It is a *centroid-based* method, which means that each cluster is represented by a *prototype point* (called a **centroid**) and every point is assigned to the cluster with the *closest* centroid.
- The goal is to find a clustering which minimizes the **within-cluster sum of squares** (i.e. *variance* of clusters).
- K-means uses an iterative method and it converges to a *local optimum*.
    - Finding the global optimum is *NP-hard*

K-means Algorithm

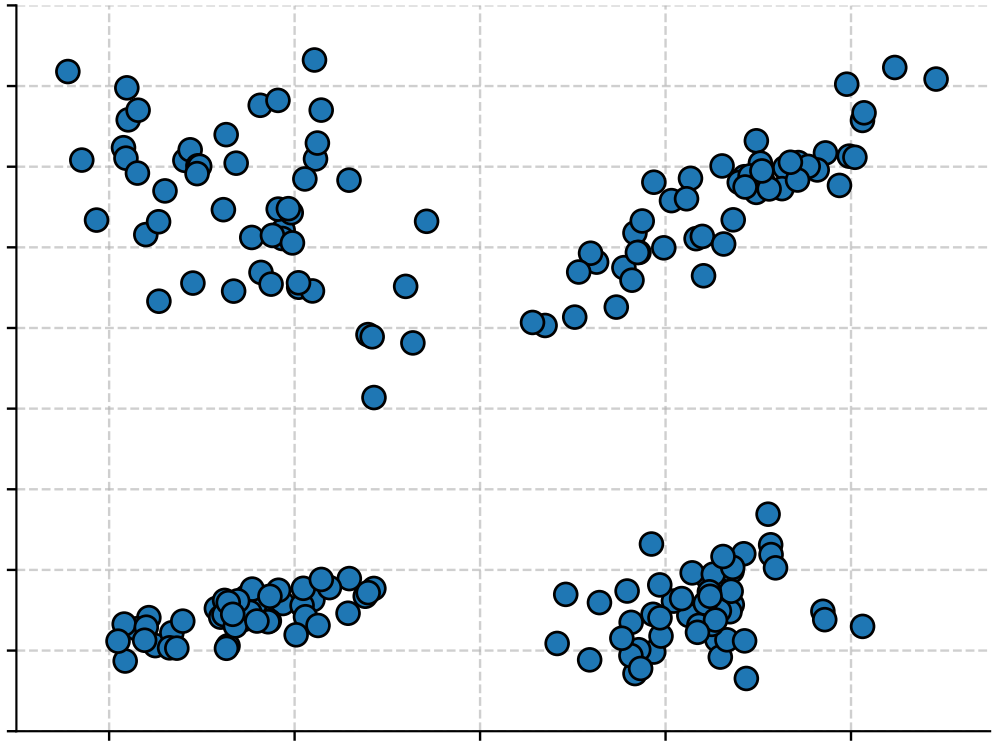
```
initialization:
    - select k random cluster centers
repeat:
    “assignment” step:
        - assign each point to the cluster of the nearest center
    “update” step:
        - move the cluster centers to the mean point of each cluster
```

K-means Algorithm

```
initialization:
    - select k random cluster centers
repeat:
    “expectation” step:
        - assign each point to the cluster of the nearest center
    “maximization” step:
        - move the cluster centers to the mean point of each cluster
```

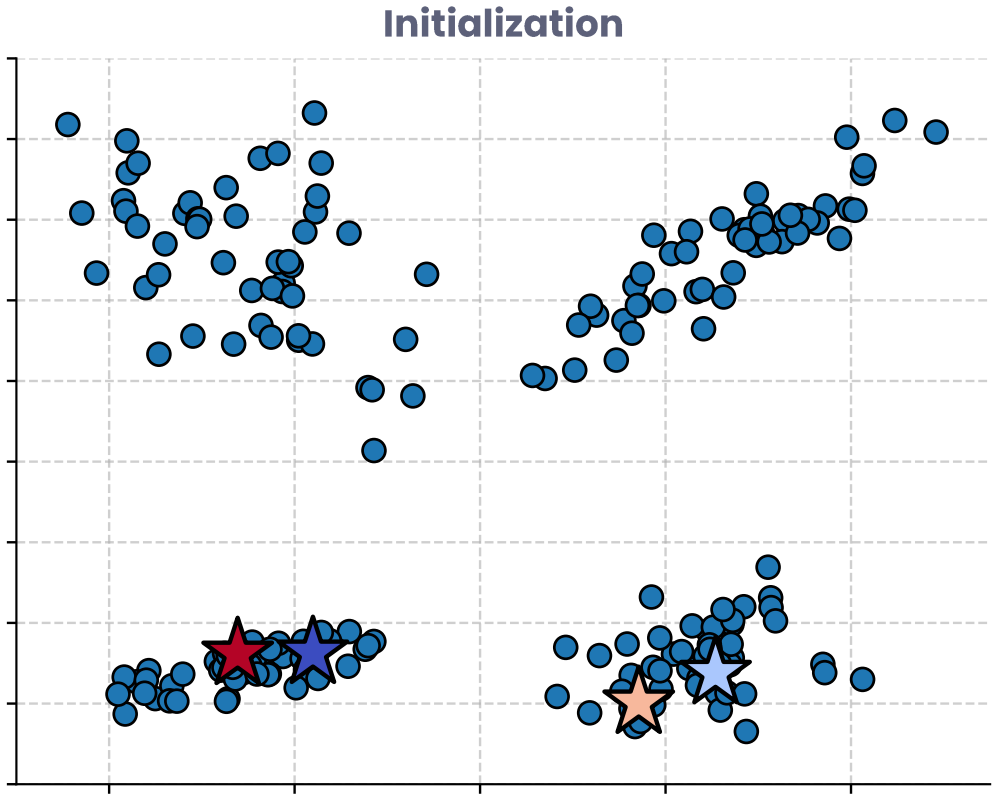
K-means is an instance of the *Expectation-Maximization (EM)* algorithm

K-means Algorithm



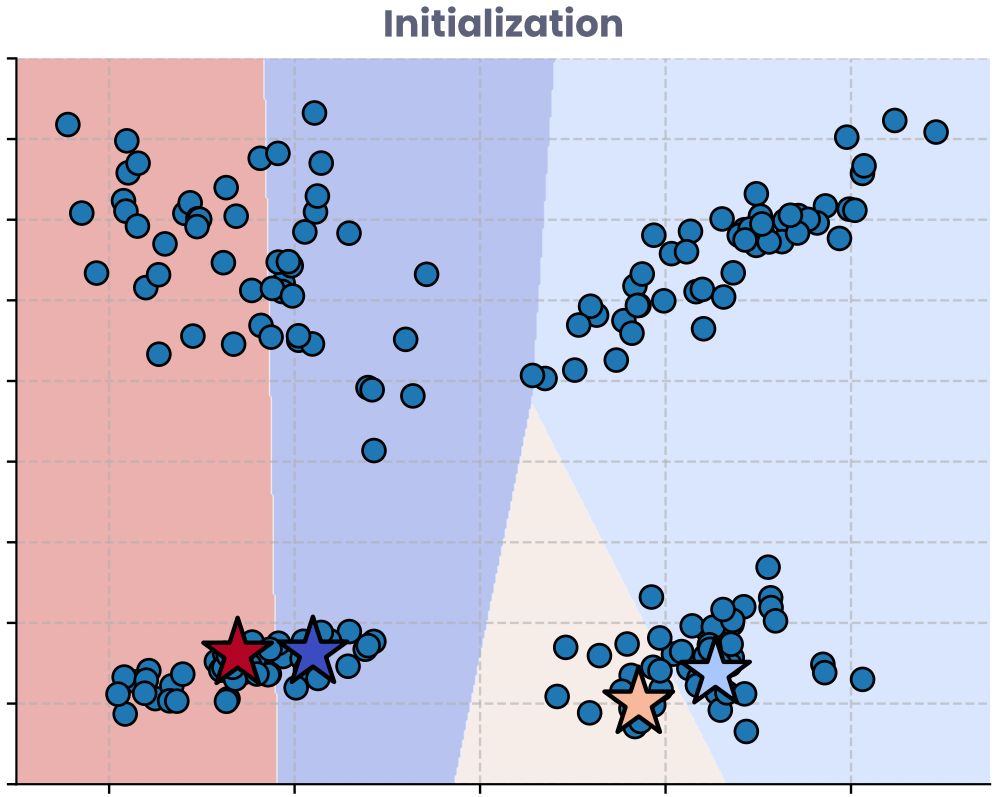
K-means Algorithm

- Choose $k = 4$ random cluster centers.



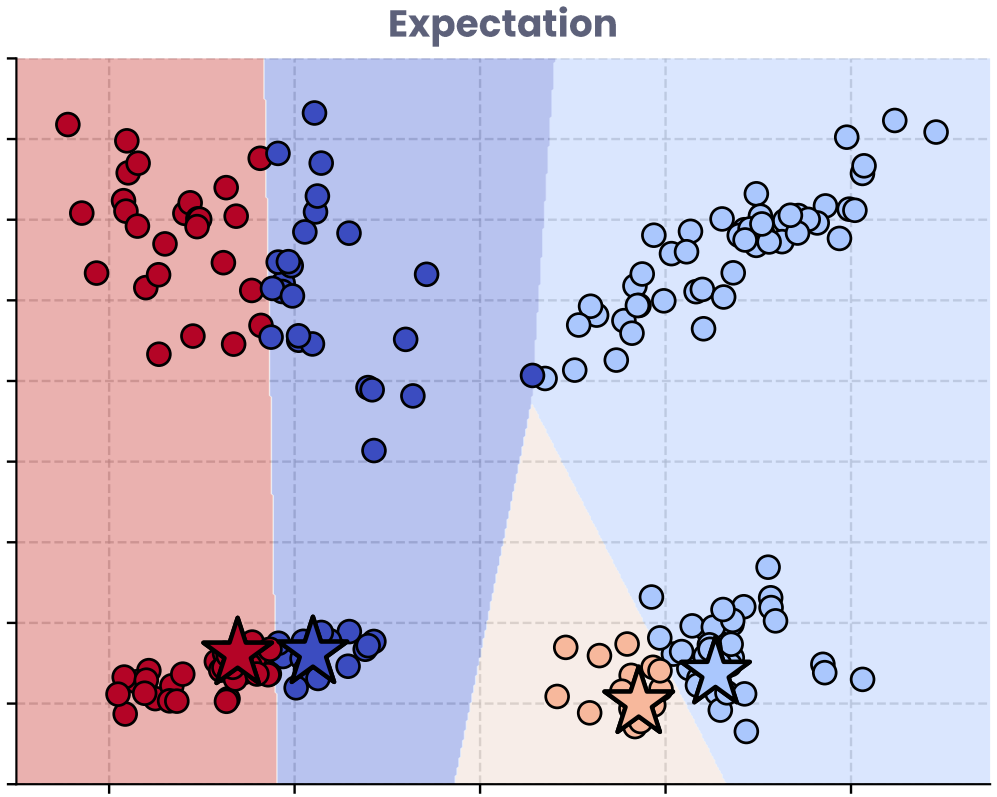
K-means Algorithm

- Choose $k = 4$ random cluster centers.
- **Each centroid has an area of space in which every point is closest to it (its "Voronoi" cell).**



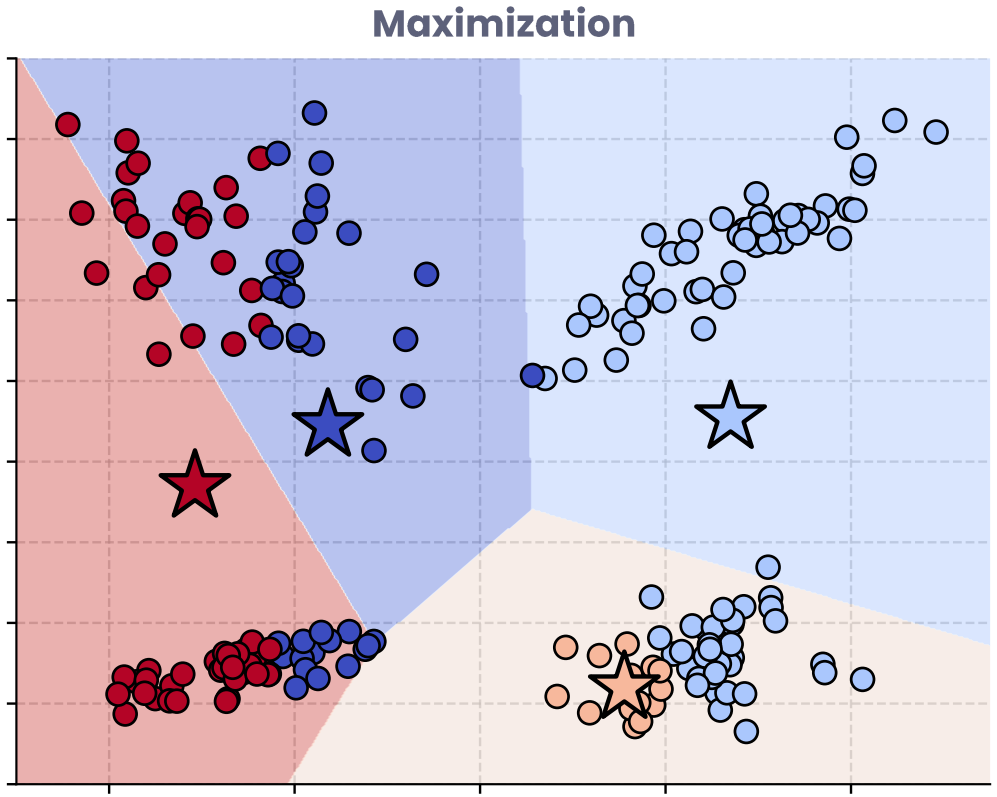
K-means Algorithm

- Choose $k = 4$ random cluster centers.
- Each centroid has an area of space in which every point is closest to it (its “Voronoi” cell).
- **Assign each point to the cluster with the closest center (“Expectation”).**



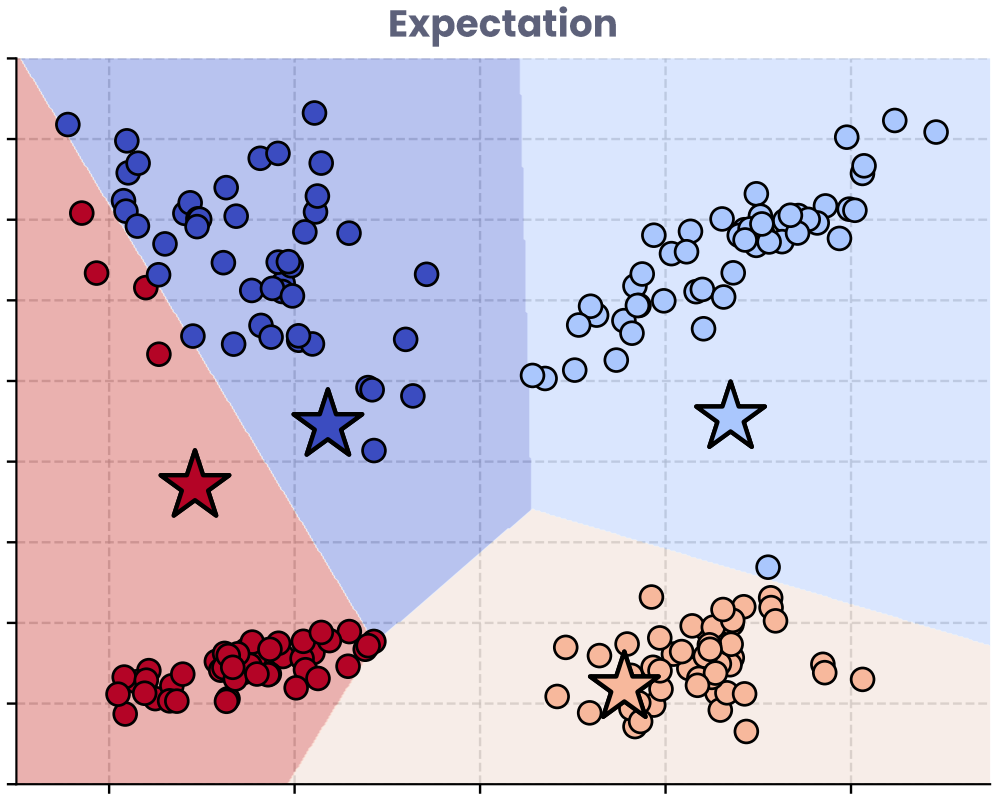
K-means Algorithm

- Choose $k = 4$ random cluster centers.
- Each centroid has an area of space in which every point is closest to it (its “Voronoi” cell).
- Assign each point to the cluster with the closest center (“Expectation”).
- **Move each centroid to the true mean of its cluster (“Maximization”).**



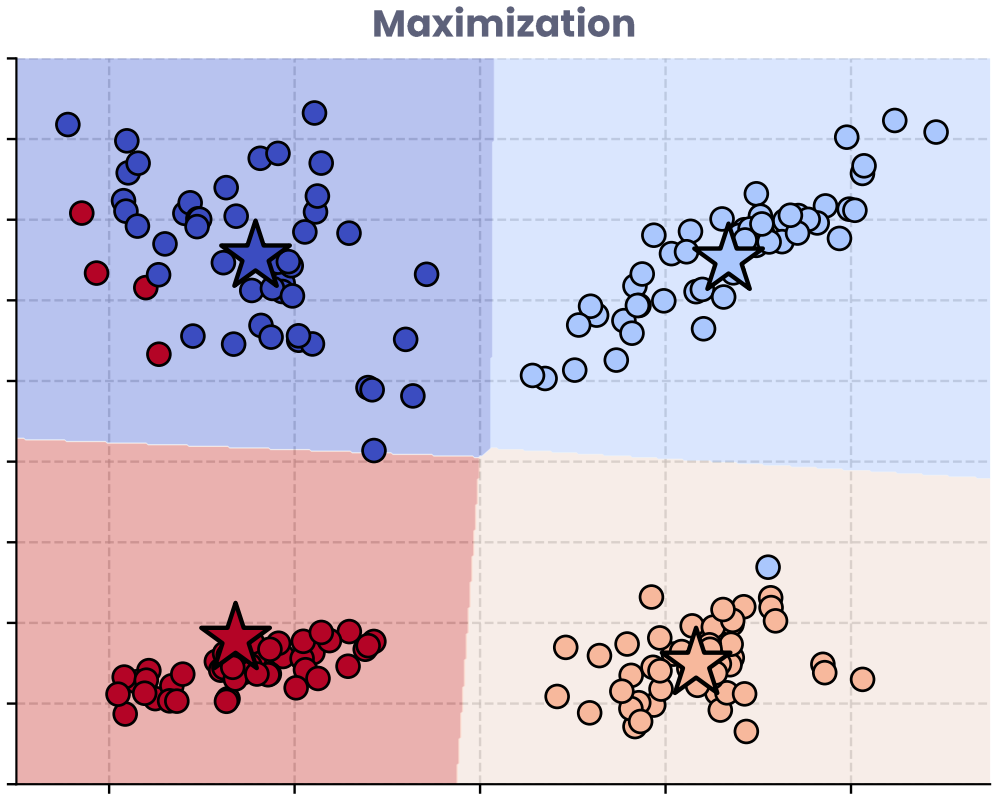
K-means Algorithm

- Choose $k = 4$ random cluster centers.
- Each centroid has an area of space in which every point is closest to it (its “Voronoi” cell).
- **Assign each point to the cluster with the closest center (“Expectation”).**
- Move each centroid to the true mean of its cluster (“Maximization”).



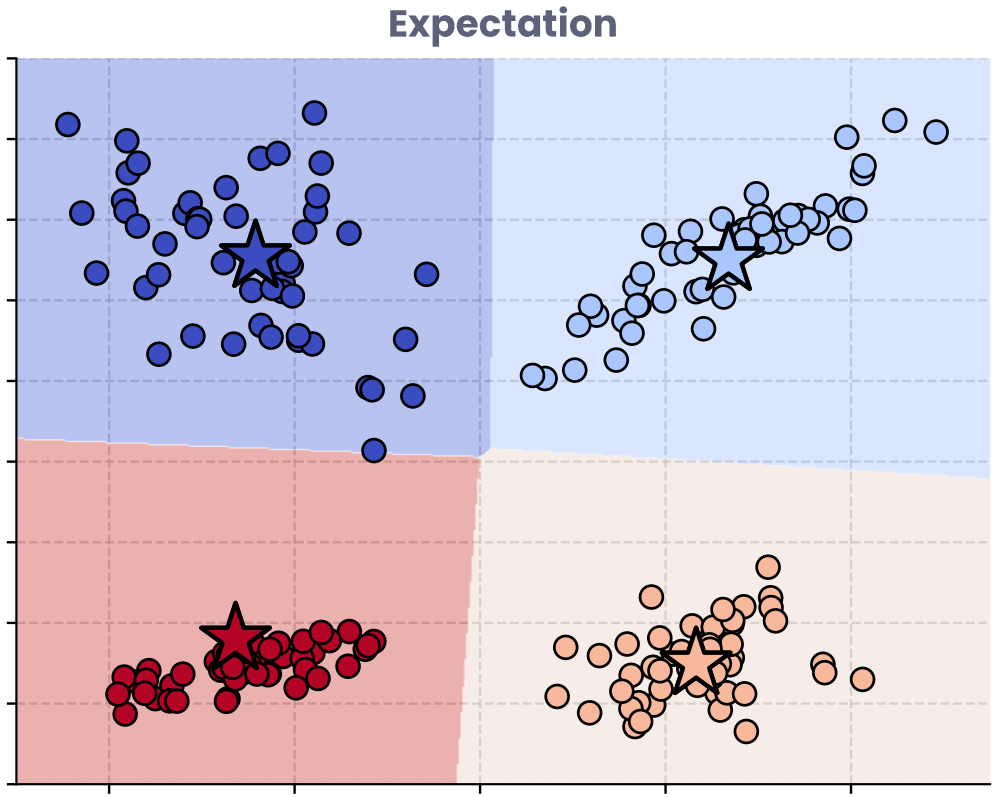
K-means Algorithm

- Choose $k = 4$ random cluster centers.
- Each centroid has an area of space in which every point is closest to it (its "Voronoi" cell).
- Assign each point to the cluster with the closest center ("Expectation").
- **Move each centroid to the true mean of its cluster ("Maximization").**



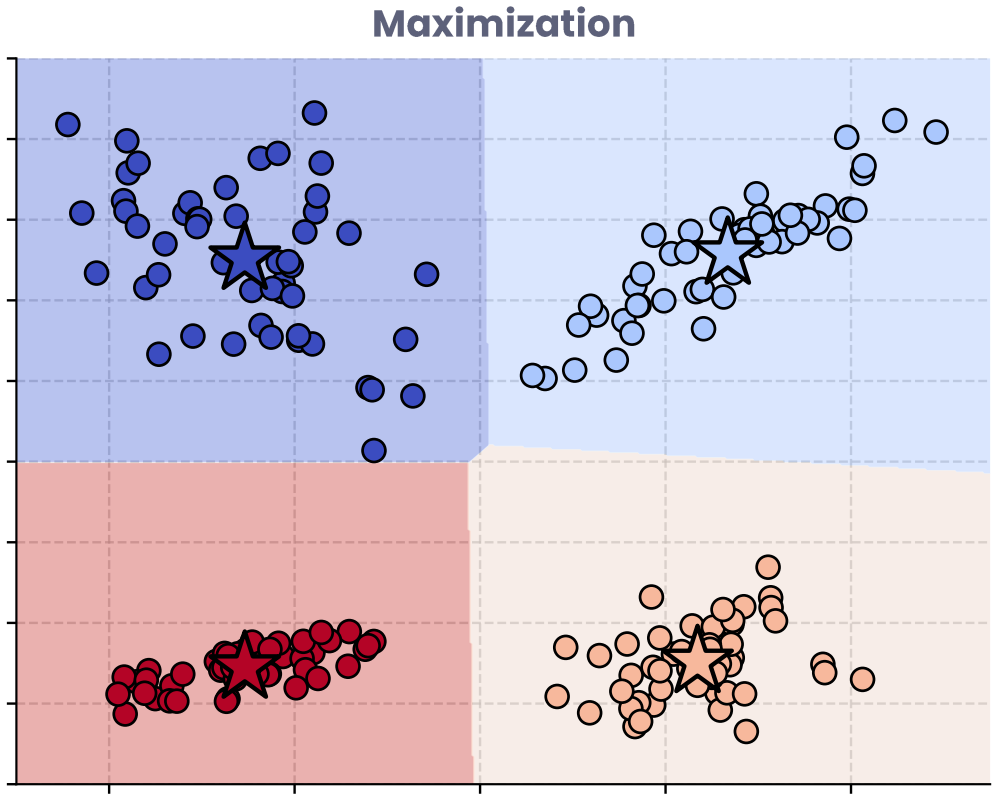
K-means Algorithm

- Choose $k = 4$ random cluster centers.
- Each centroid has an area of space in which every point is closest to it (its “Voronoi” cell).
- **Assign each point to the cluster with the closest center (“Expectation”).**
- Move each centroid to the true mean of its cluster (“Maximization”).



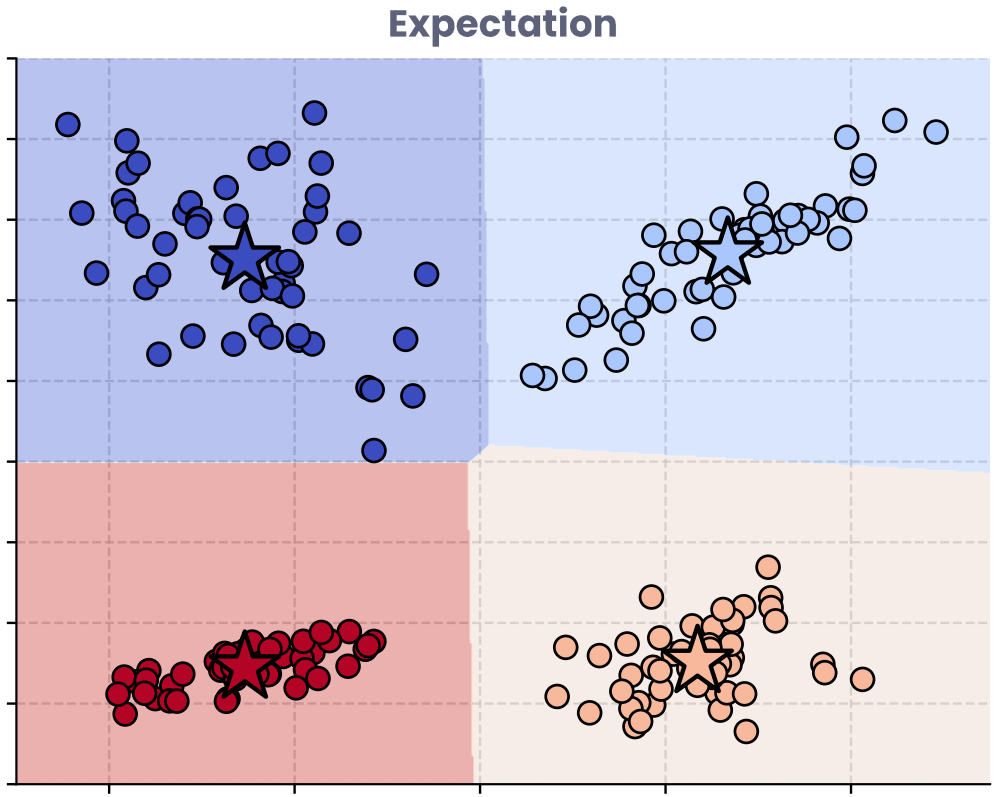
K-means Algorithm

- Choose $k = 4$ random cluster centers.
- Each centroid has an area of space in which every point is closest to it (its “Voronoi” cell).
- Assign each point to the cluster with the closest center (“Expectation”).
- **Move each centroid to the true mean of its cluster (“Maximization”).**



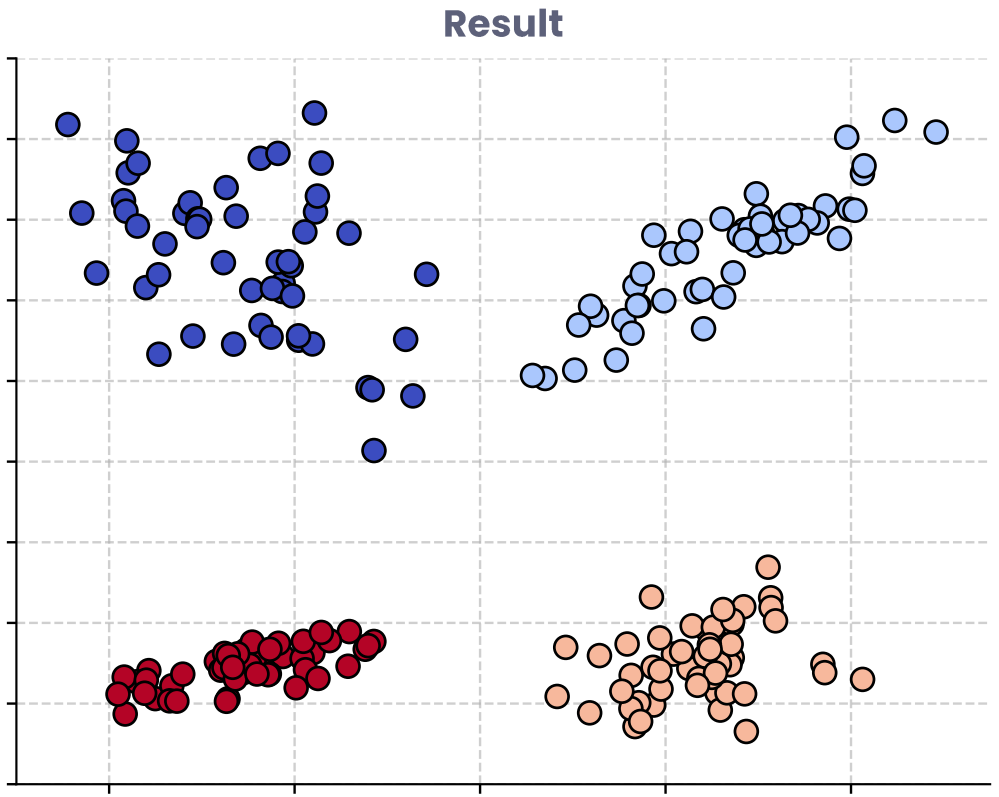
K-means Algorithm

- Choose $k = 4$ random cluster centers.
- Each centroid has an area of space in which every point is closest to it (its “Voronoi” cell).
- **Assign each point to the cluster with the closest center (“Expectation”).**
- Move each centroid to the true mean of its cluster (“Maximization”).



K-means Algorithm

- Choose $k = 4$ random cluster centers.
- Each centroid has an area of space in which every point is closest to it (its “Voronoi” cell).
- Assign each point to the cluster with the closest center (“Expectation”).
- Move each centroid to the true mean of its cluster (“Maximization”).
- **Assignment did not change anything, so stop.**



Mathematical Definition

Mathematical Definition

- For $X = \{\vec{x}^{(1)}, \ldots, \vec{x}^{(m)}\} \subset \mathbb{R}^n$ and $k \in \mathbb{N}^+$

Mathematical Definition

- For $X = \{\vec{x}^{(1)}, \ldots, \vec{x}^{(m)}\} \subset \mathbb{R}^n$ and $k \in \mathbb{N}^+$, we define the *distortion* measure:

$$J = \sum_{j=1}^{k} \sum_{\vec{x} \in C_j} \left\| \vec{x} - \vec{\mu}^{(j)} \right\|^2$$

Mathematical Definition

- For $X = \{\vec{x}^{(1)}, \dots, \vec{x}^{(m)}\} \subset \mathbb{R}^n$ and $k \in \mathbb{N}^+$, we define the *distortion* measure:

$$J = \sum_{j=1}^{k} \sum_{\vec{x} \in C_j} \left\| \vec{x} - \vec{\mu}^{(j)} \right\|^2 = \sum_{i=1}^{m} \sum_{j=1}^{k} z_{ij} \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\|^2$$

Where:

- $z_{ij} = \begin{cases} 1 & \text{if } \vec{x}^{(i)} \in C_j \\ 0 & \text{otherwise} \end{cases}$
- $\vec{\mu}^{(j)} \in \mathbb{R}^n$ is the centroid of cluster C_j

$$\sum_{j=1}^{k} z_{ij} = 1, \forall i$$
(it is a *partition*)

Mathematical Definition

- For $X = \{\vec{x}^{(1)}, \ldots, \vec{x}^{(m)}\} \subset \mathbb{R}^n$ and $k \in \mathbb{N}^+$, we define the *distortion* measure:

$$J = \sum_{j=1}^{k} \sum_{\vec{x} \in C_j} \left\| \vec{x} - \vec{\mu}^{(j)} \right\|^2 = \sum_{i=1}^{m} \sum_{j=1}^{k} z_{ij} \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\|^2$$

Where:

  - $z_{ij} = \begin{cases} 1 & \text{if } \vec{x}^{(i)} \in C_j \\ 0 & \text{otherwise} \end{cases}$
  - $\vec{\mu}^{(j)} \in \mathbb{R}^n$ is the centroid of cluster C_j

$\sum_{j=1}^{k} z_{ij} = 1, \forall i$
(it is a *partition*)

- **Goal:** Minimize J with respect to z_{ij} and $\vec{\mu}^{(j)}$

Finding the global optimum is NP-hard

Algorithm (2)

initialization:

- choose random values for $\vec{\mu}^{(j)}$, for all $j \in \{1, 2, \ldots, k\}$

repeat:

"expectation" step:

- minimize J w.r.t. z_{ij}, keeping $\vec{\mu}^{(j)}$ fixed

"maximization" step:

- minimize J w.r.t. $\vec{\mu}^{(j)}$, keeping z_{ij} fixed

Algorithm (2)

- "**Expectation**" step:

  Minimize

$$J = \sum_{i=1}^{m} \sum_{j=1}^{k} z_{ij} \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\|^2$$

  With respect to z_{ij}, keeping $\vec{\mu}^{(j)}$ fixed.

Algorithm (2)

- "**Expectation**" step:

  Minimize

  $$J = \sum_{i=1}^{m} \sum_{j=1}^{k} z_{ij} \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\|^2$$

  With respect to z_{ij}, keeping $\vec{\mu}^{(j)}$ fixed.

- There must be only one $z_{ij} = 1$, $\forall i$ (i.e. $\vec{x}^{(i)}$ must be in only one cluster).

Algorithm (2)

- "**Expectation**" step:

  Minimize $$J = \sum_{i=1}^{m} \sum_{j=1}^{k} z_{ij} \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\|^2$$

  With respect to z_{ij}, keeping $\vec{\mu}^{(j)}$ fixed.

- There must be only one $z_{ij} = 1$, $\forall i$ (i.e. $\vec{x}^{(i)}$ must be in only one cluster).
  - To minimize J, we must set $z_{ij^*} \coloneqq \begin{cases} 1 & \text{if } j^* = \operatorname{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| \\ 0 & \text{otherwise} \end{cases}$

Algorithm (2)

- "**Expectation**" step:

  Minimize $$J = \sum_{i=1}^{m} \sum_{j=1}^{k} z_{ij} \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\|^2$$

  With respect to z_{ij}, keeping $\vec{\mu}^{(j)}$ fixed.

- There must be only one $z_{ij} = 1$, $\forall i$ (i.e. $\vec{x}^{(i)}$ must be in only one cluster).
  - To minimize J, we must set $z_{ij^*} \coloneqq \begin{cases} 1 & \text{if } j^* = \operatorname{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| \\ 0 & \text{otherwise} \end{cases}$
- In order words, **we assign $\vec{x}^{(i)}$ to the cluster with the closest center.**

Algorithm (2)

- "**Maximization**" step:

Minimize

$$J = \sum_{i=1}^{m} \sum_{j=1}^{k} z_{ij} \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\|^2$$

With respect to $\vec{\mu}^{(j)}$, keeping z_{ij} fixed.

Algorithm (2)

- "**Maximization**" step:

  Minimize $$J = \sum_{i=1}^{m} \sum_{j=1}^{k} z_{ij} \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\|^2$$

  With respect to $\vec{\mu}^{(j)}$, keeping z_{ij} fixed.

- J is a quadratic function of $\vec{\mu}^{(j)} \Longrightarrow$ minimize by setting $\frac{\partial J}{\partial \vec{\mu}^{(j)}} = 0$

Algorithm (2)

- "**Maximization**" step:

Minimize

$$J = \sum_{i=1}^{m} \sum_{j=1}^{k} z_{ij} \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\|^2$$

With respect to $\vec{\mu}^{(j)}$, keeping z_{ij} fixed.

- J is a quadratic function of $\vec{\mu}^{(j)} \Longrightarrow$ minimize by setting $\frac{\partial J}{\partial \vec{\mu}^{(j)}} = 0$

$$\frac{\partial J}{\partial \vec{\mu}^{(j)}} = 2 \sum\nolimits_i z_{ij} \left(\vec{x}^{(i)} - \vec{\mu}^{(j)} \right) = 2 \sum\nolimits_i z_{ij} \vec{x}^{(i)} - 2 \sum\nolimits_i z_{ij} \vec{\mu}^{(j)} = 0$$

Algorithm (2)

- "**Maximization**" step:

Minimize

$$J = \sum_{i=1}^{m} \sum_{j=1}^{k} z_{ij} \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\|^2$$

With respect to $\vec{\mu}^{(j)}$, keeping z_{ij} fixed.

- J is a quadratic function of $\vec{\mu}^{(j)} \Longrightarrow$ minimize by setting $\frac{\partial J}{\partial \vec{\mu}^{(j)}} = 0$

$$\frac{\partial J}{\partial \vec{\mu}^{(j)}} = 2 \sum_i z_{ij} \left(\vec{x}^{(i)} - \vec{\mu}^{(j)} \right) = 2 \sum_i z_{ij} \vec{x}^{(i)} - 2 \sum_i z_{ij} \vec{\mu}^{(j)} = 0 \Longrightarrow \vec{\mu}^{(j)} = \frac{\sum_i z_{ij} \vec{x}^{(i)}}{\sum_i z_{ij}} = \frac{1}{|C_j|} \sum_{\vec{x} \in C_j} \vec{x}$$

Algorithm (2)

- "**Maximization**" step:

Minimize

$$J = \sum_{i=1}^{m} \sum_{j=1}^{k} z_{ij} \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\|^2$$

With respect to $\vec{\mu}^{(j)}$, keeping z_{ij} fixed.

- J is a quadratic function of $\vec{\mu}^{(j)} \Longrightarrow$ minimize by setting $\frac{\partial J}{\partial \vec{\mu}^{(j)}} = 0$

$$\frac{\partial J}{\partial \vec{\mu}^{(j)}} = 2 \sum_i z_{ij} \left(\vec{x}^{(i)} - \vec{\mu}^{(j)} \right) = 2 \sum_i z_{ij} \vec{x}^{(i)} - 2 \sum_i z_{ij} \vec{\mu}^{(j)} = 0 \Longrightarrow \vec{\mu}^{(j)} = \frac{\sum_i z_{ij} \vec{x}^{(i)}}{\sum_i z_{ij}} = \frac{\mathbf{1}}{|\boldsymbol{C_j}|} \sum_{\vec{x} \in \boldsymbol{C_j}} \vec{\boldsymbol{x}}$$

- In other words, **we set $\vec{\mu}^{(j)}$ to the mean of all points in C_j**

Algorithm (2)

- "**Maximization**" step:

Minimize

$$J = \sum_{i=1}^{m} \sum_{j=1}^{k} z_{ij} \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\|^2$$

With respect to $\vec{\mu}^{(j)}$, keeping z_{ij} fixed.

- J is a quadratic function of $\vec{\mu}^{(j)} \Longrightarrow$ minimize by setting $\frac{\partial J}{\partial \vec{\mu}^{(j)}} = 0$

$$\frac{\partial J}{\partial \vec{\mu}^{(j)}} = 2 \sum_i z_{ij} \left(\vec{x}^{(i)} - \vec{\mu}^{(j)} \right) = 2 \sum_i z_{ij} \vec{x}^{(i)} - 2 \sum_i z_{ij} \vec{\mu}^{(j)} = 0 \Longrightarrow \vec{\mu}^{(j)} = \frac{\sum_i z_{ij} \vec{x}^{(i)}}{\sum_i z_{ij}} = \frac{1}{|C_j|} \sum_{\vec{x} \in C_j} \vec{x}$$

- In other words, **we set $\vec{\mu}^{(j)}$ to the mean of all points in C_j**

This is for one particular $\vec{\mu}^{(j)}$, we do it for all j

Parameters and Evaluation

How to choose k?

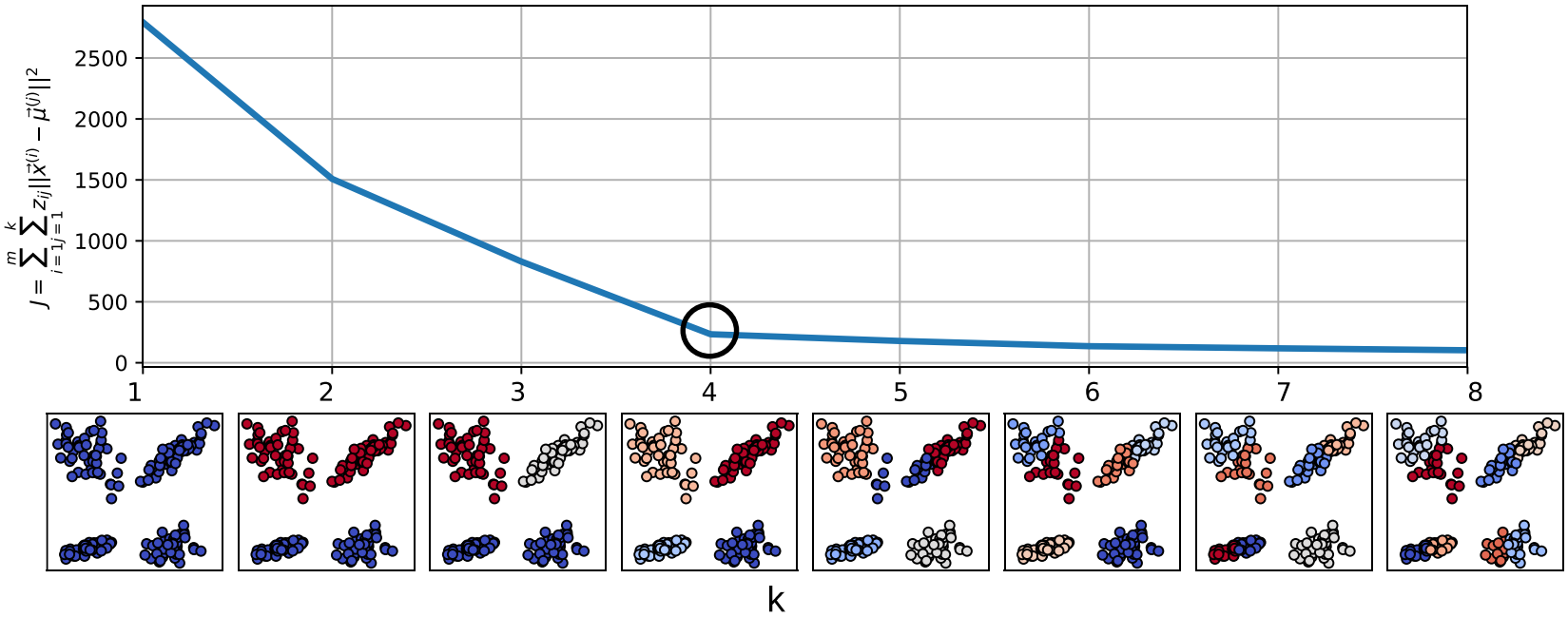
- Number of clusters *k* is a *hyperparameter*. How do we get a good k?

How to choose k?

- Number of clusters *k* is a *hyperparameter*. How do we get a good k?
- **Elbow method**
  - Start with a small *k* value and increase it until adding another cluster does not result in a much lower *distortion value*.
  - In other words, the new cluster does not explain much more of the *variance* in the data.
- **Silhouette Coefficient**
  - A measure of how *tight* each cluster is and how *far apart* cluster are from one another.
  - Choose a *k* value which results in a clustering with a large silhouette coefficient.

The Elbow Method

- Choose *k* such that adding another cluster wouldn't *explain much more of the variance* in the data (i.e. does not give a much lower distortion value):



Silhouette Coefficient

- Measures the *tightness of clusters* and *separation between clusters*:

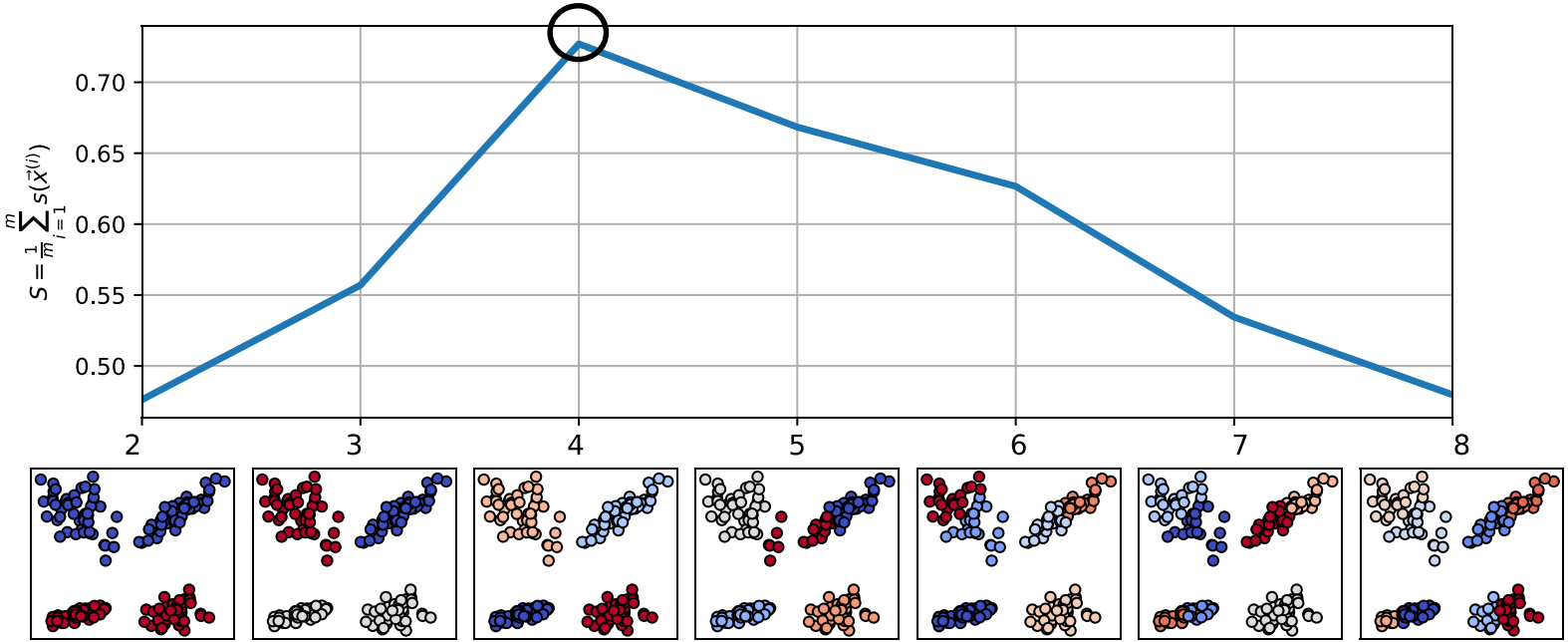
$$s\left(\vec{x}^{(i)}\right) = \frac{b\left(\vec{x}^{(i)}\right) - a\left(\vec{x}^{(i)}\right)}{\max\left(a(\vec{x}^{(i)}), b(\vec{x}^{(i)})\right)}$$

Where:

- $a\left(\vec{x}^{(i)}\right)$ – average distance between $\vec{x}^{(i)}$ and all other points in the same cluster
- $b\left(\vec{x}^{(i)}\right)$ – lowest average distance to all points in any other cluster
    - The average distance to the closest “neighboring” cluster.

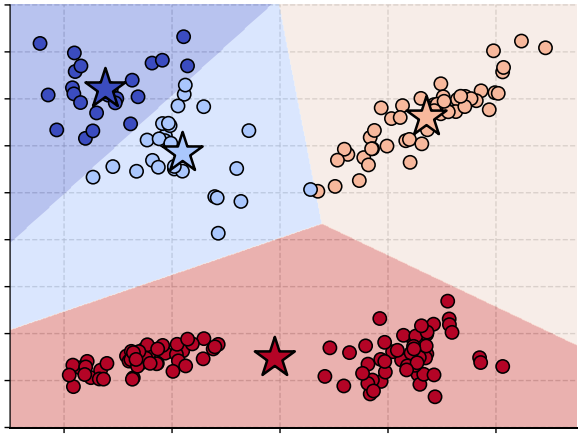
Silhouette Coefficient

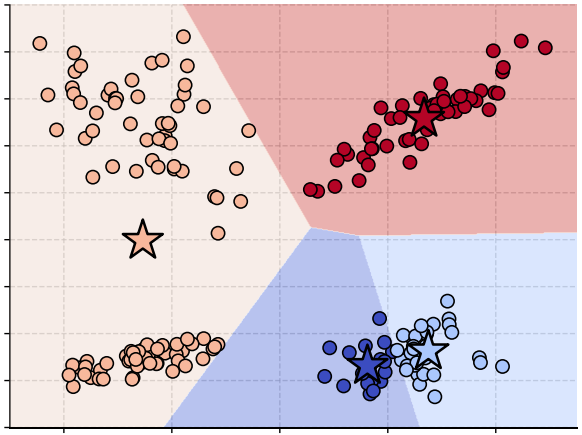
- Choose *k* which gives the highest mean silhouette:



Local minima

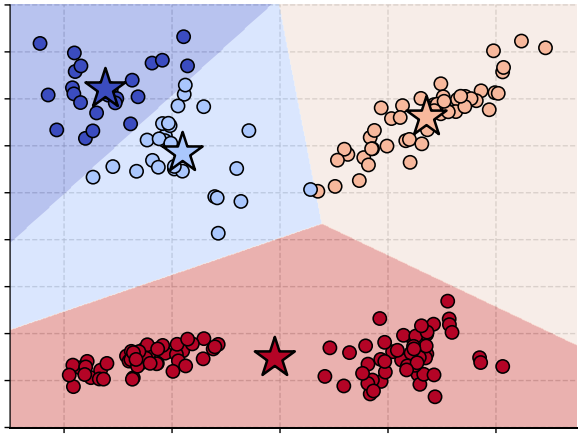
- K-means can get stuck in *local minima*.
- Both of the following states are stable (further iterations will not change anything):

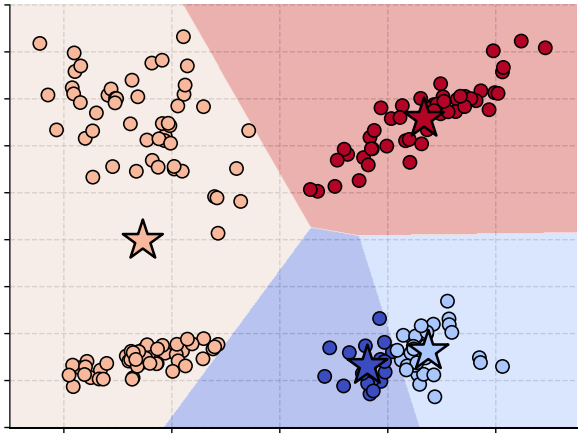




Local minima

- K-means can get stuck in *local minima*.
- Both of the following states are stable (further iterations will not change anything):

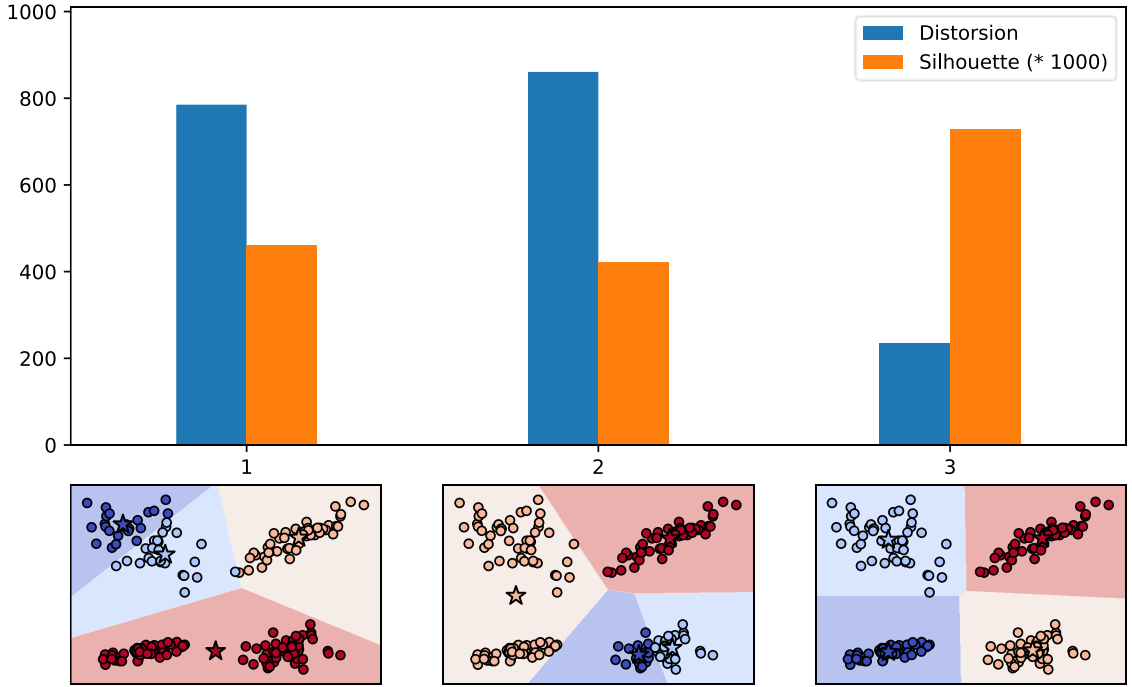




- Possible solutions:
  - Run the algorithm multiple times and pick the result with the *lowest distortion* (or *highest silhouette*)
  - Use a better initialization method.

Local minima

- Run the algorithm multiple times and pick the result with the *lowest distortion* (or *highest silhouette*):



Local minima

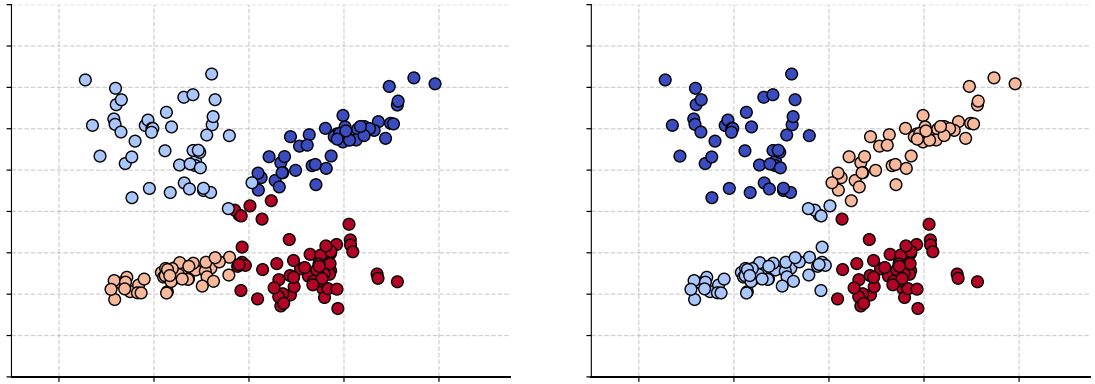
- Use a better initialization method:
  - *"**k-means++: the advantages of careful seeding**"*, Arthur, D., Vassilvitskii, S., 2007.

```
1 choose first center uniformly at random from the data points
2 repeat until all k centers have been chosen:
3     - compute D(x⃗^(i)), the distance from x⃗^(i) to the nearest chosen center
4     - choose a new center at random with probability P(x⃗^(i)) ~ D(x⃗^(i))^2
5 run standard k-means algorithm
```

  - The idea is to try to spread out the initial cluster centers.

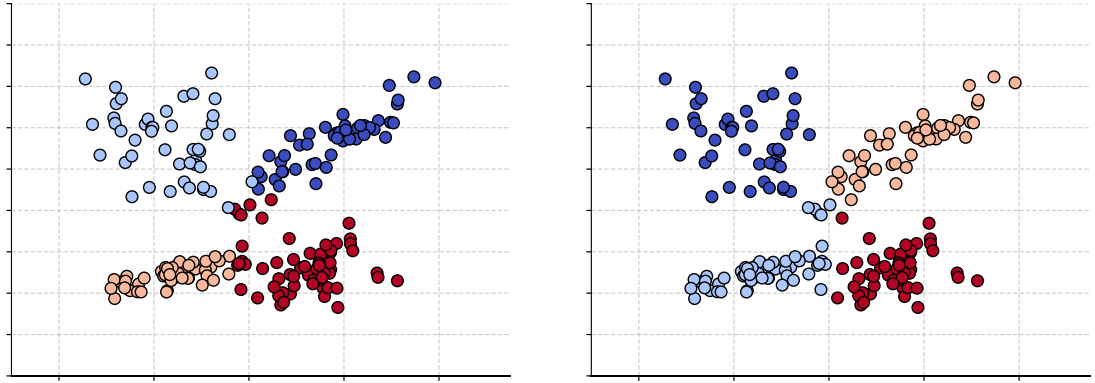
Comparing clusterings

- How similar are these two clusterings?



Comparing clusterings

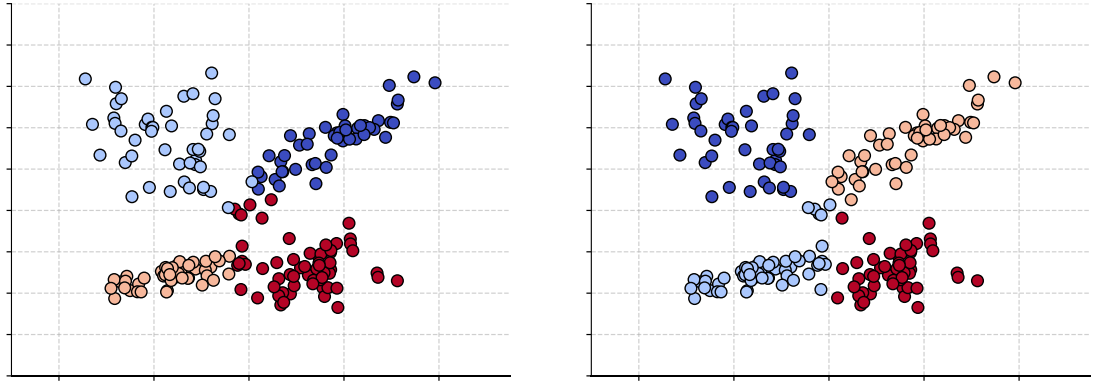
- How similar are these two clusterings?



The actual “label” assigned to each cluster (represented here by color) doesn’t matter, only the grouping of points matters.

Comparing clusterings

- How similar are these two clusterings?



The actual "label" assigned to each cluster (represented here by color) doesn't matter, only the grouping of points matters.

- **Rand Index** measures how often two clustering agree in terms of grouping points:

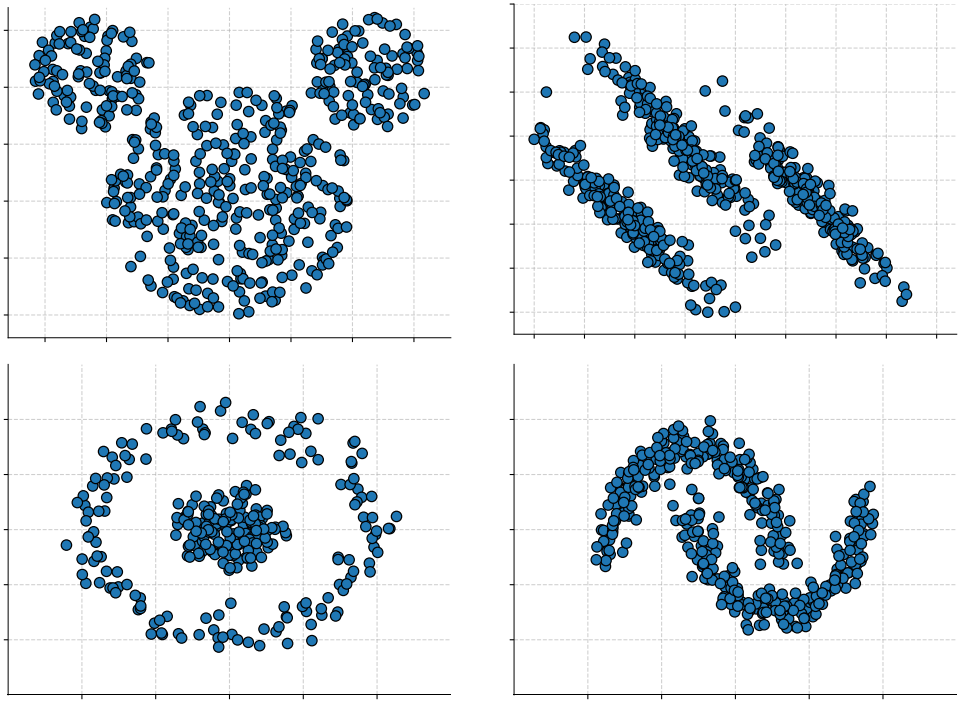
$$R = \frac{a + b}{n(n-1)/2}$$

  - a is the number of pairs of points which are in the same cluster in both clusterings.
  - b is the number of pairs of points which are in different clusters in both clusterings.

Adjusted Rand Index is a form of Rand index which also take into account that clusterings might agree by chance.

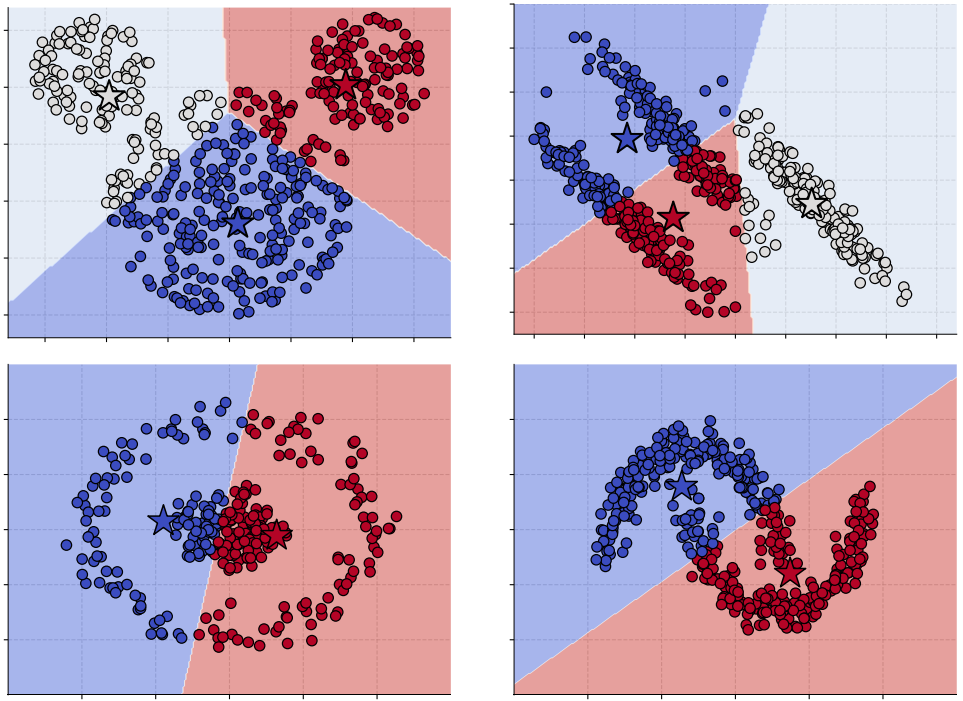
K-means Limitations

- How will K-means handle these datasets?



K-means Limitations

- How will K-means handle these datasets?
- Not so good…
  - K-means only produces *convex clusters*.
  - It doesn't handle *non-spherical clusters* very well.
  - It tends to produce *clusters of equals sizes*.



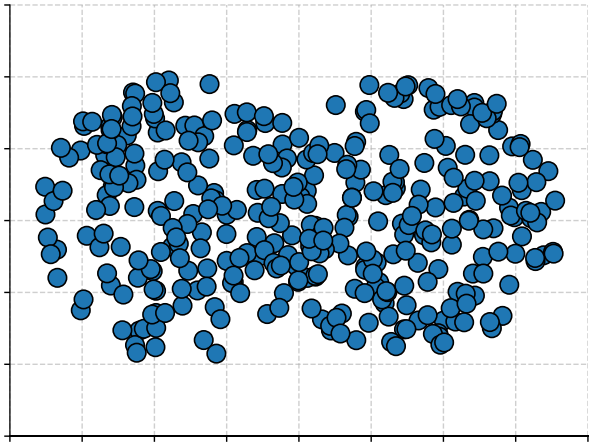
K-means Variations

Soft K-means

- K-means "*expectation*" step: $z_{ij^*} \coloneqq \begin{cases} 1 & \text{if } j^* = \operatorname{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| \\ 0 & \text{otherwise} \end{cases}$ (i.e. assign point $\vec{x}^{(i)}$ to the cluster with the closest centroid)
- This will produce a *partition* (or *hard-clustering*), which means a point is in only one cluster.

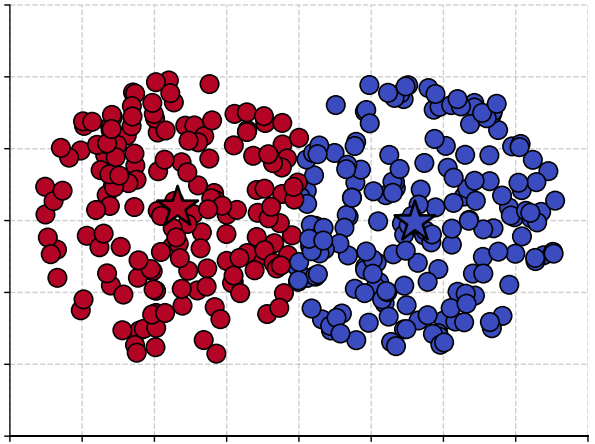
Soft K-means

- K-means "*expectation*" step: $z_{ij^*} \coloneqq \begin{cases} 1 & \text{if } j^* = \text{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| \\ 0 & \text{otherwise} \end{cases}$ (i.e. assign point $\vec{x}^{(i)}$ to the cluster with the closest centroid)
- This will produce a *partition* (or *hard-clustering*), which means a point is in only one cluster.
- Sometimes, in practice, clusters might have overlapping regions, in which there is no clear-cut reason for assigning points to one cluster or the other.



Soft K-means

- K-means "*expectation*" step: $z_{ij^*} \coloneqq \begin{cases} 1 & \text{if } j^* = \text{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| \\ 0 & \text{otherwise} \end{cases}$ (i.e. assign point $\vec{x}^{(i)}$ to the cluster with the closest centroid)
- This will produce a *partition* (or *hard-clustering*), which means a point is in only one cluster.
- Sometimes, in practice, clusters might have overlapping regions, in which there is no clear-cut reason for assigning points to one cluster or the other.
- With *hard-clustering*, the assignment in such regions will mostly be due to chance from random initialization.



Soft K-means

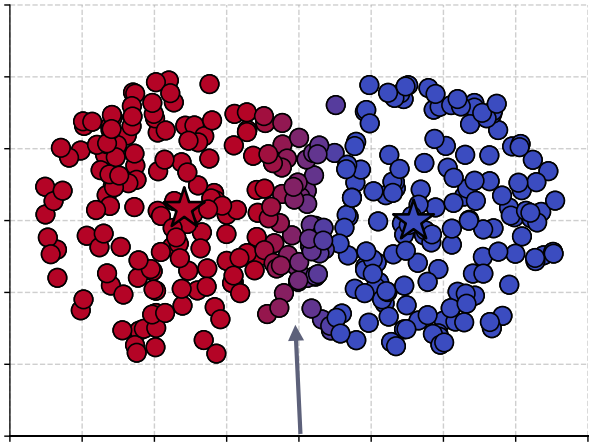
- K-means "*expectation*" step: $z_{ij^*} \coloneqq \begin{cases} 1 & \text{if } j^* = \text{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| \\ 0 & \text{otherwise} \end{cases}$ (i.e. assign point $\vec{x}^{(i)}$ to the cluster with the closest centroid)
- This will produce a *partition* (or *hard-clustering*), which means a point is in only one cluster.
- Sometimes, in practice, clusters might have overlapping regions, in which there is no clear-cut reason for assigning points to one cluster or the other.
- With *hard-clustering*, the assignment in such regions will mostly be due to chance from random initialization.
- **Soft K-means** redefines the "expectation" step such that $z_{ij} \in \mathbb{R}$ is the *degree of membership* of $\vec{x}^{(i)}$ to cluster C_j :

$$z_{ij^*} \coloneqq \frac{e^{-\beta \left\| \vec{x}^{(i)} - \vec{\mu}^{(j^*)} \right\|}}{\sum_j e^{-\beta \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\|}}$$

Expectation

$$\vec{\mu}^{(j)} = \frac{\sum_i z_{ij} \vec{x}^{(i)}}{\sum_i z_{ij}}$$

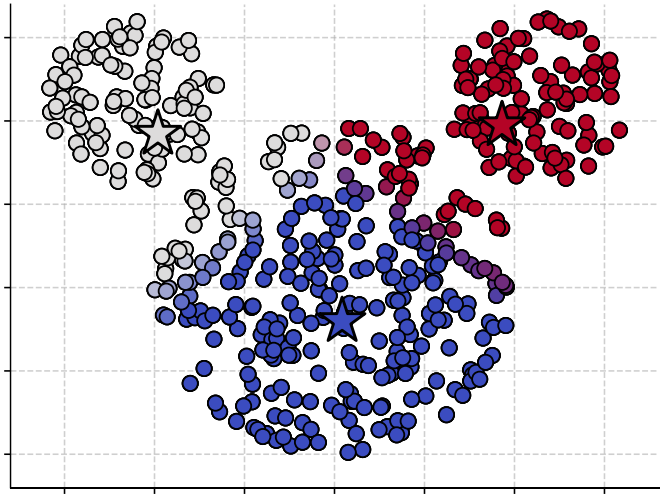
Maximization

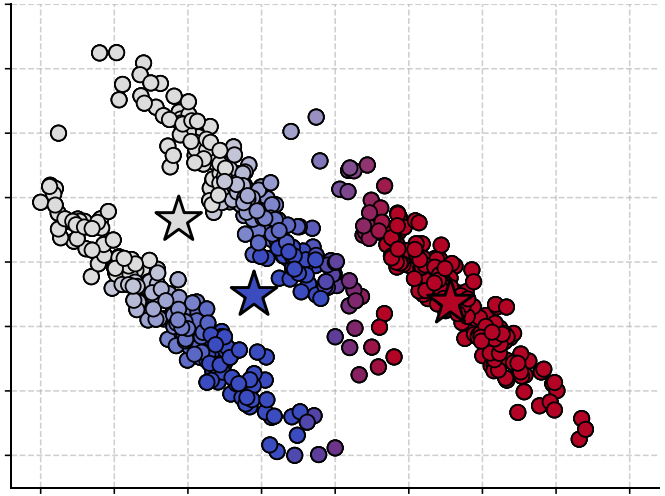


Points in this area have ~0.5 membership in both clusters.

Soft K-means

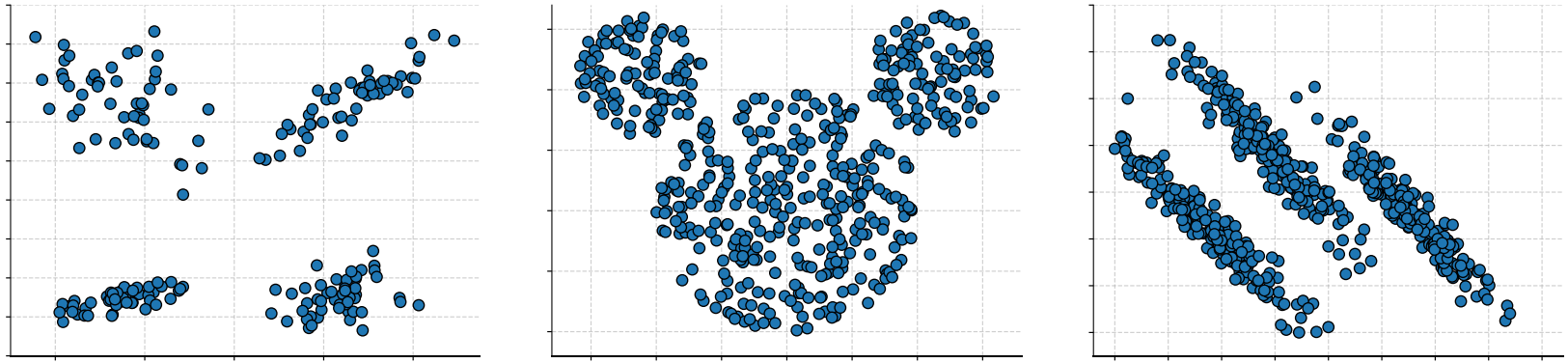
- Soft K-means does not solve the unequal-size and non-spherical clusters issues.





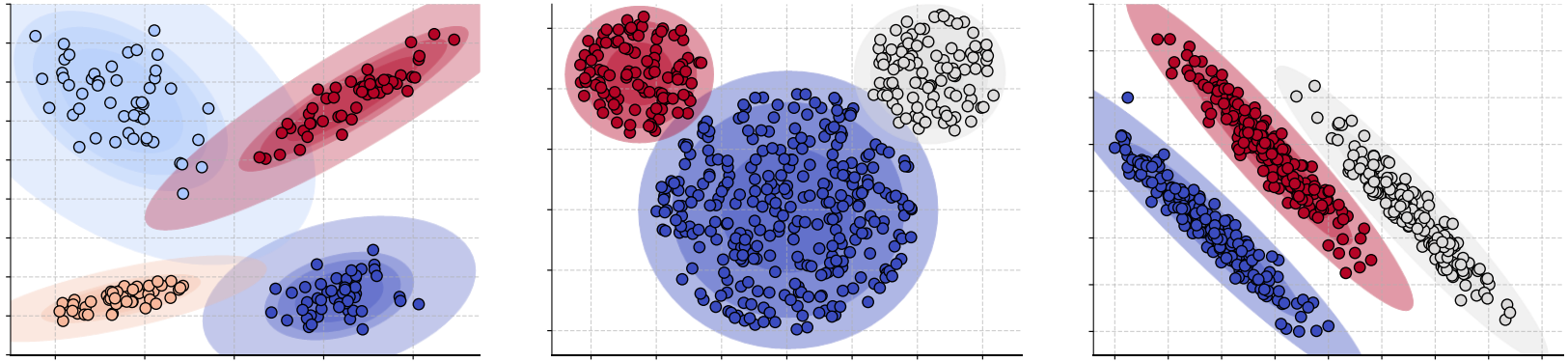
Gaussian Mixture Models

- **GMMs** are *probabilistic models* which assumes that data points are generated by a mixture of *normal distributions* (a.k.a. *Gaussians*).
- A *expectation-maximization* algorithm can be used to fit the Gaussians by maximizing the *likelihood* of data.



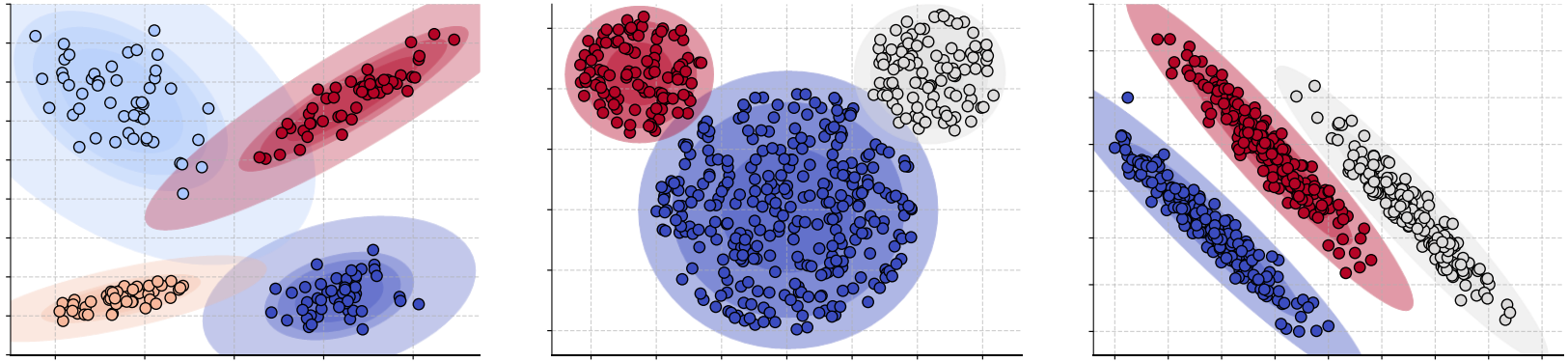
Gaussian Mixture Models

- **GMMs** are *probabilistic models* which assumes that data points are generated by a mixture of *normal distributions* (a.k.a. *Gaussians*).
- A *expectation-maximization* algorithm can be used to fit the Gaussians by maximizing the *likelihood* of data.



Gaussian Mixture Models

- **GMMs** are *probabilistic models* which assumes that data points are generated by a mixture of *normal distributions* (a.k.a. *Gaussians*).
- A *expectation-maximization* algorithm can be used to fit the Gaussians by maximizing the *likelihood* of data.



- It can be viewed as an extension of *Soft K-means* in which each cluster has both a *mean* and a *covariance matrix* (which gives the non-spherical shape).

Getting rid of centroids

- **Expectation** step means setting $z_{ij^*} = \begin{cases} 1 & \text{if } j^* = \text{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| \\ 0 & \text{otherwise} \end{cases}$

Getting rid of centroids

- **Expectation** step means setting $z_{ij^*} = \begin{cases} 1 & \text{if } j^* = \text{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| \\ 0 & \text{otherwise} \end{cases}$

$$\left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| = \sqrt{\left\langle \vec{x}^{(i)} - \vec{\mu}^{(j)}, \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\rangle}$$

Norm of a vector is the square root of the dot product with itself.

Getting rid of centroids

- **Expectation** step means setting $z_{ij^*} = \begin{cases} 1 & \text{if } j^* = \text{argmin}_j \|\vec{x}^{(i)} - \vec{\mu}^{(j)}\| \\ 0 & \text{otherwise} \end{cases}$

$$\|\vec{x}^{(i)} - \vec{\mu}^{(j)}\| = \sqrt{\langle \vec{x}^{(i)} - \vec{\mu}^{(j)}, \vec{x}^{(i)} - \vec{\mu}^{(j)} \rangle} = \sqrt{\langle \vec{x}^{(i)}, \vec{x}^{(i)} \rangle - 2\langle \vec{x}^{(i)}, \vec{\mu}^{(j)} \rangle + \langle \vec{\mu}^{(j)}, \vec{\mu}^{(j)} \rangle}$$

Norm of a vector is the square root of the dot product with itself.

Dot product is distributive.

Getting rid of centroids

- **Expectation** step means setting $z_{ij^*} = \begin{cases} 1 & \text{if } j^* = \operatorname{argmin}_j \left\|\vec{x}^{(i)} - \vec{\mu}^{(j)}\right\| \\ 0 & \text{otherwise} \end{cases}$

$$\left\|\vec{x}^{(i)} - \vec{\mu}^{(j)}\right\| = \sqrt{\left\langle \vec{x}^{(i)} - \vec{\mu}^{(j)}, \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\rangle} = \sqrt{\left\langle \vec{x}^{(i)}, \vec{x}^{(i)} \right\rangle - 2\left\langle \vec{x}^{(i)}, \vec{\mu}^{(j)} \right\rangle + \left\langle \vec{\mu}^{(j)}, \vec{\mu}^{(j)} \right\rangle}$$

$$= \sqrt{\left\|\vec{x}^{(i)}\right\|^2 - 2\left\langle \vec{x}^{(i)}, \frac{1}{|C_j|}\sum_{\vec{x}' \in C_j} \vec{x}' \right\rangle + \left\langle \frac{1}{|C_j|}\sum_{\vec{x}' \in C_j} \vec{x}', \frac{1}{|C_j|}\sum_{\vec{x}'' \in C_j} \vec{x}'' \right\rangle}$$

Getting rid of centroids

- **Expectation** step means setting $z_{ij^*} = \begin{cases} 1 & \text{if } j^* = \operatorname{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| \\ 0 & \text{otherwise} \end{cases}$

$$\left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| = \sqrt{\left\langle \vec{x}^{(i)} - \vec{\mu}^{(j)}, \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\rangle} = \sqrt{\left\langle \vec{x}^{(i)}, \vec{x}^{(i)} \right\rangle - 2\left\langle \vec{x}^{(i)}, \vec{\mu}^{(j)} \right\rangle + \left\langle \vec{\mu}^{(j)}, \vec{\mu}^{(j)} \right\rangle}$$

$$= \sqrt{\left\| \vec{x}^{(i)} \right\|^2 - 2\left\langle \vec{x}^{(i)}, \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \vec{x}' \right\rangle + \left\langle \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \vec{x}', \frac{1}{|C_j|} \sum_{\vec{x}'' \in C_j} \vec{x}'' \right\rangle}$$

$$= \sqrt{\left\| \vec{x}^{(i)} \right\|^2 - 2\frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \left\langle \vec{x}^{(i)}, \vec{x}' \right\rangle + \frac{1}{|C_j|^2} \sum_{\vec{x}' \in C_j} \sum_{\vec{x}'' \in C_j} \left\langle \vec{x}', \vec{x}'' \right\rangle}$$

Dot product is distributive.

Getting rid of centroids

- **Expectation** step means setting $z_{ij^*} = \begin{cases} 1 & \text{if } j^* = \operatorname{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| \\ 0 & \text{otherwise} \end{cases}$

$$\left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| = \sqrt{\left\langle \vec{x}^{(i)} - \vec{\mu}^{(j)}, \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\rangle} = \sqrt{\left\langle \vec{x}^{(i)}, \vec{x}^{(i)} \right\rangle - 2\left\langle \vec{x}^{(i)}, \vec{\mu}^{(j)} \right\rangle + \left\langle \vec{\mu}^{(j)}, \vec{\mu}^{(j)} \right\rangle}$$

$$= \sqrt{\left\| \vec{x}^{(i)} \right\|^2 - 2\left\langle \vec{x}^{(i)}, \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \vec{x}' \right\rangle + \left\langle \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \vec{x}', \frac{1}{|C_j|} \sum_{\vec{x}'' \in C_j} \vec{x}'' \right\rangle}$$

$$= \sqrt{\left\| \vec{x}^{(i)} \right\|^2 - 2\frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \left\langle \vec{x}^{(i)}, \vec{x}' \right\rangle + \frac{1}{|C_j|^2} \sum_{\vec{x}' \in C_j} \sum_{\vec{x}'' \in C_j} \left\langle \vec{x}', \vec{x}'' \right\rangle}$$

$$\operatorname{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| = \mathbf{argmin}_j \left(-\mathbf{2}\frac{\mathbf{1}}{|\boldsymbol{C_j}|} \sum_{\vec{x}' \in \boldsymbol{C_j}} \left\langle \vec{\boldsymbol{x}}^{(i)}, \vec{\boldsymbol{x}}' \right\rangle + \frac{\mathbf{1}}{|\boldsymbol{C_j}|^2} \sum_{\vec{x}' \in \boldsymbol{C_j}} \sum_{\vec{x}'' \in \boldsymbol{C_j}} \left\langle \vec{\boldsymbol{x}}', \vec{\boldsymbol{x}}'' \right\rangle \right)$$

$\left\| \vec{x}^{(i)} \right\|^2$ and the √ do not affect the *argmin*.

Getting rid of centroids

- **Expectation** step means setting $z_{ij^*} = \begin{cases} 1 & \text{if } j^* = \operatorname{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| \\ 0 & \text{otherwise} \end{cases}$

$$\left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| = \sqrt{\left\langle \vec{x}^{(i)} - \vec{\mu}^{(j)}, \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\rangle} = \sqrt{\left\langle \vec{x}^{(i)}, \vec{x}^{(i)} \right\rangle - 2\left\langle \vec{x}^{(i)}, \vec{\mu}^{(j)} \right\rangle + \left\langle \vec{\mu}^{(j)}, \vec{\mu}^{(j)} \right\rangle}$$

$$= \sqrt{\left\| \vec{x}^{(i)} \right\|^2 - 2\left\langle \vec{x}^{(i)}, \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \vec{x}' \right\rangle + \left\langle \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \vec{x}', \frac{1}{|C_j|} \sum_{\vec{x}'' \in C_j} \vec{x}'' \right\rangle}$$

$$= \sqrt{\left\| \vec{x}^{(i)} \right\|^2 - 2\frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \left\langle \vec{x}^{(i)}, \vec{x}' \right\rangle + \frac{1}{|C_j|^2} \sum_{\vec{x}' \in C_j} \sum_{\vec{x}'' \in C_j} \left\langle \vec{x}', \vec{x}'' \right\rangle}$$

$$\operatorname{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| = \mathbf{argmin}_j \left(-\mathbf{2}\frac{\mathbf{1}}{|\boldsymbol{C_j}|} \sum_{\vec{x}' \in \boldsymbol{C_j}} \left\langle \vec{\boldsymbol{x}}^{(i)}, \vec{\boldsymbol{x}}' \right\rangle + \frac{\mathbf{1}}{|\boldsymbol{C_j}|^2} \sum_{\vec{x}' \in \boldsymbol{C_j}} \sum_{\vec{x}'' \in \boldsymbol{C_j}} \left\langle \vec{\boldsymbol{x}}', \vec{\boldsymbol{x}}'' \right\rangle \right)$$

$\left\| \vec{x}^{(i)} \right\|^2$ and the √ do not affect the *argmin*.

- We can do k-means clustering *without computing centroids*.

Getting rid of centroids

- **Expectation** step means setting $z_{ij^*} = \begin{cases} 1 & \text{if } j^* = \operatorname{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| \\ 0 & \text{otherwise} \end{cases}$

$$\left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| = \sqrt{\left\langle \vec{x}^{(i)} - \vec{\mu}^{(j)}, \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\rangle} = \sqrt{\left\langle \vec{x}^{(i)}, \vec{x}^{(i)} \right\rangle - 2\left\langle \vec{x}^{(i)}, \vec{\mu}^{(j)} \right\rangle + \left\langle \vec{\mu}^{(j)}, \vec{\mu}^{(j)} \right\rangle}$$

$$= \sqrt{\left\| \vec{x}^{(i)} \right\|^2 - 2\left\langle \vec{x}^{(i)}, \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \vec{x}' \right\rangle + \left\langle \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \vec{x}', \frac{1}{|C_j|} \sum_{\vec{x}'' \in C_j} \vec{x}'' \right\rangle}$$

$$= \sqrt{\left\| \vec{x}^{(i)} \right\|^2 - 2\frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \left\langle \vec{x}^{(i)}, \vec{x}' \right\rangle + \frac{1}{|C_j|^2} \sum_{\vec{x}' \in C_j} \sum_{\vec{x}'' \in C_j} \left\langle \vec{x}', \vec{x}'' \right\rangle}$$

$$\operatorname{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| = \mathbf{argmin}_j \left(-\mathbf{2}\frac{\mathbf{1}}{|\boldsymbol{C_j}|} \sum_{\vec{x}' \in \boldsymbol{C_j}} \left\langle \vec{\boldsymbol{x}}^{(i)}, \vec{\boldsymbol{x}}' \right\rangle + \frac{\mathbf{1}}{|\boldsymbol{C_j}|^2} \sum_{\vec{x}' \in \boldsymbol{C_j}} \sum_{\vec{x}'' \in \boldsymbol{C_j}} \left\langle \vec{\boldsymbol{x}}', \vec{\boldsymbol{x}}'' \right\rangle \right)$$

$\left\| \vec{x}^{(i)} \right\|^2$ and the √ do not affect the *argmin*.

- We can do k-means clustering *without computing centroids*.
- The expression **only depends on dot product** on pairs of training samples!

Getting rid of centroids

- **Expectation** step means setting $z_{ij^*} = \begin{cases} 1 & \text{if } j^* = \operatorname{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| \\ 0 & \text{otherwise} \end{cases}$

$$\left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| = \sqrt{\left\langle \vec{x}^{(i)} - \vec{\mu}^{(j)}, \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\rangle} = \sqrt{\left\langle \vec{x}^{(i)}, \vec{x}^{(i)} \right\rangle - 2\left\langle \vec{x}^{(i)}, \vec{\mu}^{(j)} \right\rangle + \left\langle \vec{\mu}^{(j)}, \vec{\mu}^{(j)} \right\rangle}$$

$$= \sqrt{\left\| \vec{x}^{(i)} \right\|^2 - 2\left\langle \vec{x}^{(i)}, \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \vec{x}' \right\rangle + \left\langle \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \vec{x}', \frac{1}{|C_j|} \sum_{\vec{x}'' \in C_j} \vec{x}'' \right\rangle}$$

$$= \sqrt{\left\| \vec{x}^{(i)} \right\|^2 - 2\frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} \left\langle \vec{x}^{(i)}, \vec{x}' \right\rangle + \frac{1}{|C_j|^2} \sum_{\vec{x}' \in C_j} \sum_{\vec{x}'' \in C_j} \left\langle \vec{x}', \vec{x}'' \right\rangle}$$

$$\operatorname{argmin}_j \left\| \vec{x}^{(i)} - \vec{\mu}^{(j)} \right\| = \mathbf{argmin}_j \left(-\mathbf{2}\frac{\mathbf{1}}{|\boldsymbol{C_j}|} \sum_{\vec{x}' \in \boldsymbol{C_j}} \left\langle \vec{\boldsymbol{x}}^{(i)}, \vec{\boldsymbol{x}}' \right\rangle + \frac{\mathbf{1}}{|\boldsymbol{C_j}|^2} \sum_{\vec{x}' \in \boldsymbol{C_j}} \sum_{\vec{x}'' \in \boldsymbol{C_j}} \left\langle \vec{\boldsymbol{x}}', \vec{\boldsymbol{x}}'' \right\rangle \right)$$

$\left\| \vec{x}^{(i)} \right\|^2$ and the √ do not affect the *argmin*.

- We can do k-means clustering *without computing centroids*.
- The expression **only depends on dot product** on pairs of training samples!

*We can use a **kernel function.***

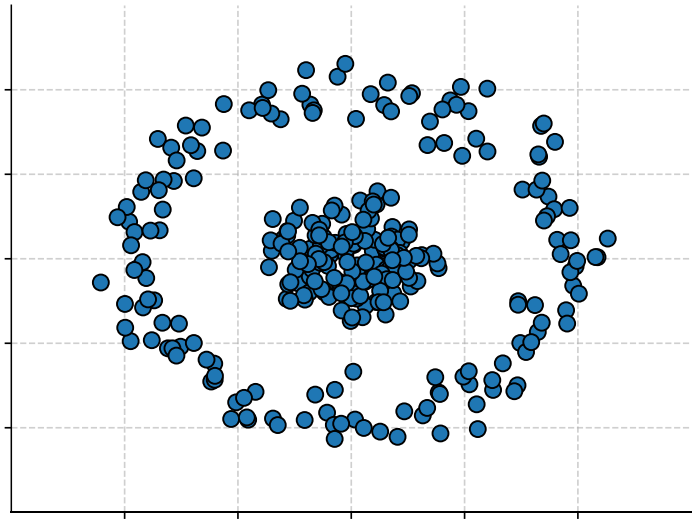
Kernel K-means

- Assign each point to a random cluster.
- Repeat until no change occurs:

$$z_{ij^*} \coloneqq \begin{cases} 1 & \text{if } j^* = \operatorname{argmin}_j \left(-2 \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} K(\vec{x}^{(i)}, \vec{x}') + \frac{1}{|C_j|^2} \sum_{\vec{x}' \in C_j} \sum_{\vec{x}'' \in C_j} K(\vec{x}', \vec{x}'') \right) \\ 0 & \text{otherwise} \end{cases}$$

Where:

- $z_{ij} = \begin{cases} 1 & \text{if } \vec{x}^{(i)} \in C_j \\ 0 & \text{otherwise} \end{cases}$
- $K: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a kernel function.



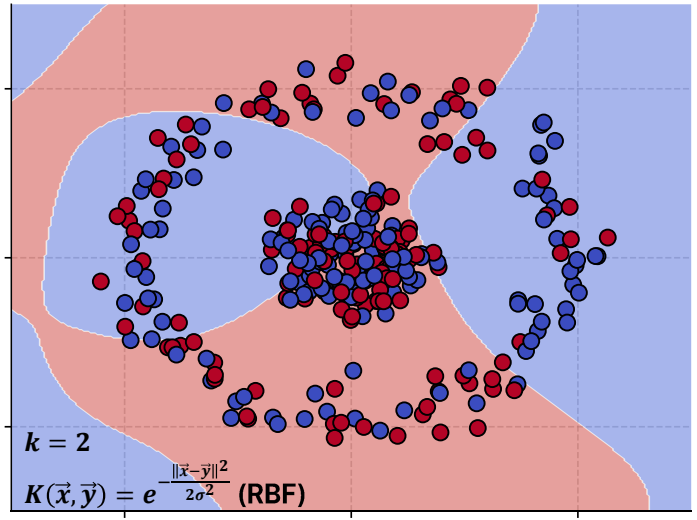
Kernel K-means

- **Assign each point to a random cluster.**
- Repeat until no change occurs:

$$z_{ij^*} \coloneqq \begin{cases} 1 & \text{if } j^* = \operatorname{argmin}_j \left(-2 \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} K(\vec{x}^{(i)}, \vec{x}') + \frac{1}{|C_j|^2} \sum_{\vec{x}' \in C_j} \sum_{\vec{x}'' \in C_j} K(\vec{x}', \vec{x}'') \right) \\ 0 & \text{otherwise} \end{cases}$$

Where:

- $z_{ij} = \begin{cases} 1 & \text{if } \vec{x}^{(i)} \in C_j \\ 0 & \text{otherwise} \end{cases}$
- $K: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a kernel function.



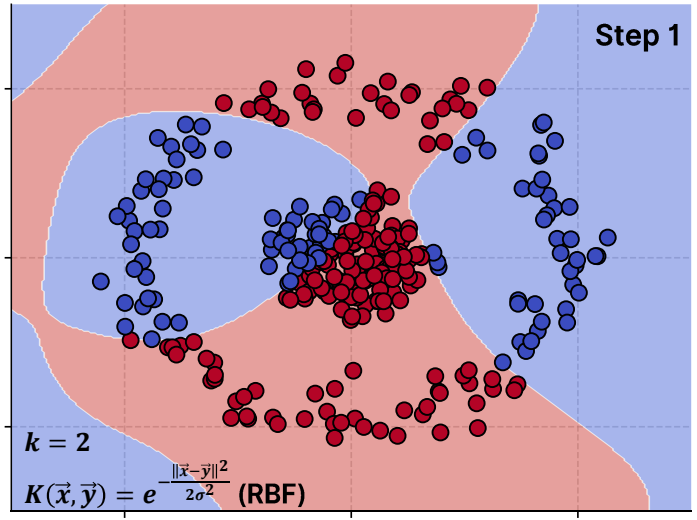
Kernel K-means

- Assign each point to a random cluster.
- **Repeat until no change occurs:**

$$\mathbf{z_{ij^*}} \coloneqq \begin{cases} \mathbf{1} & \textbf{if } j^* = \textbf{argmin}_j \left(-2 \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} K(\vec{x}^{(i)}, \vec{x}') + \frac{1}{|C_j|^2} \sum_{\vec{x}' \in C_j} \sum_{\vec{x}'' \in C_j} K(\vec{x}', \vec{x}'') \right) \\ \mathbf{0} & \textbf{otherwise} \end{cases}$$

Where:

- $z_{ij} = \begin{cases} 1 & \text{if } \vec{x}^{(i)} \in C_j \\ 0 & \text{otherwise} \end{cases}$
- $K: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a kernel function.



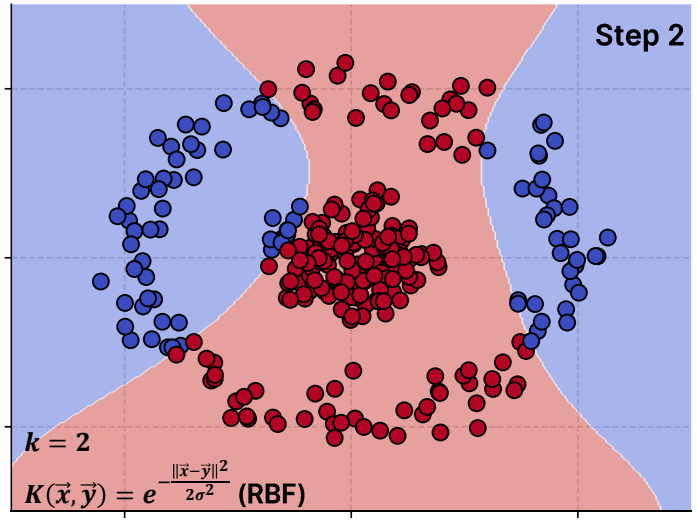
Kernel K-means

- Assign each point to a random cluster.
- **Repeat until no change occurs:**

$$\mathbf{z_{ij^*}} \coloneqq \begin{cases} \mathbf{1} & \textbf{if } j^* = \mathbf{argmin}_j \left(-2 \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} K(\vec{x}^{(i)}, \vec{x}') + \frac{1}{|C_j|^2} \sum_{\vec{x}' \in C_j} \sum_{\vec{x}'' \in C_j} K(\vec{x}', \vec{x}'') \right) \\ \mathbf{0} & \textbf{otherwise} \end{cases}$$

Where:

- $z_{ij} = \begin{cases} 1 & \text{if } \vec{x}^{(i)} \in C_j \\ 0 & \text{otherwise} \end{cases}$
- $K: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a kernel function.



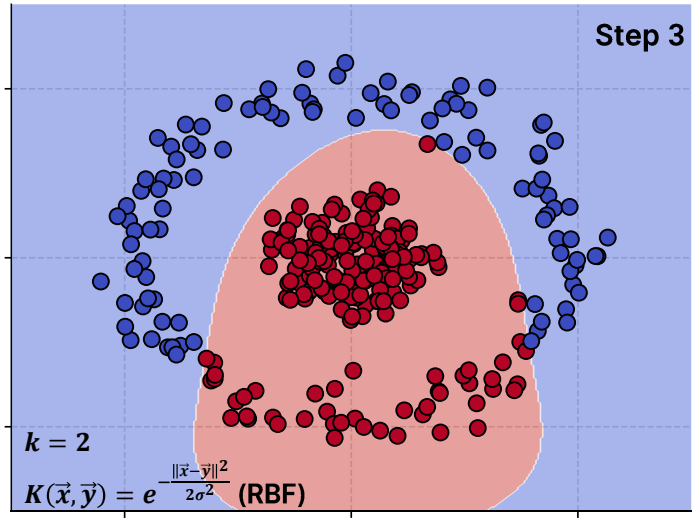
Kernel K-means

- Assign each point to a random cluster.
- **Repeat until no change occurs:**

$$\mathbf{z_{ij^*}} := \begin{cases} \mathbf{1} & \textbf{if } j^* = \textbf{argmin}_j \left(-2 \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} K(\vec{x}^{(i)}, \vec{x}') + \frac{1}{|C_j|^2} \sum_{\vec{x}' \in C_j} \sum_{\vec{x}'' \in C_j} K(\vec{x}', \vec{x}'') \right) \\ \mathbf{0} & \textbf{otherwise} \end{cases}$$

Where:

- $z_{ij} = \begin{cases} 1 & \text{if } \vec{x}^{(i)} \in C_j \\ 0 & \text{otherwise} \end{cases}$
- $K: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a kernel function.



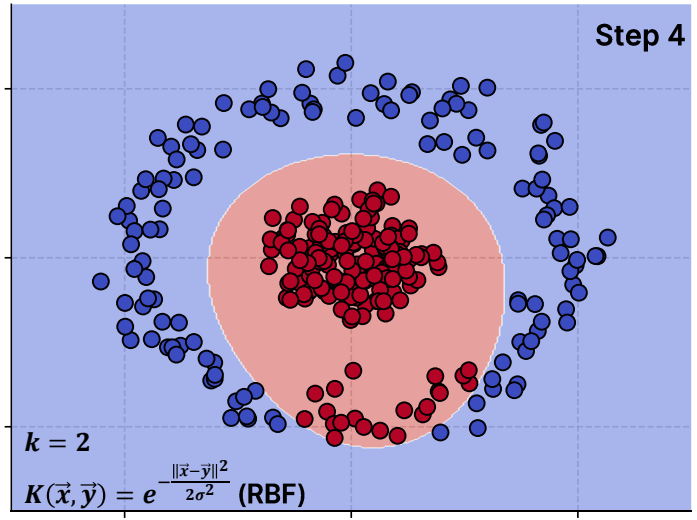
Kernel K-means

- Assign each point to a random cluster.
- **Repeat until no change occurs:**

$$\mathbf{z_{ij^*}} \coloneqq \begin{cases} \mathbf{1} & \textbf{if } j^* = \mathbf{argmin}_j \left(-2 \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} K(\vec{x}^{(i)}, \vec{x}') + \frac{1}{|C_j|^2} \sum_{\vec{x}' \in C_j} \sum_{\vec{x}'' \in C_j} K(\vec{x}', \vec{x}'') \right) \\ \mathbf{0} & \textbf{otherwise} \end{cases}$$

Where:

- $z_{ij} = \begin{cases} 1 & \text{if } \vec{x}^{(i)} \in C_j \\ 0 & \text{otherwise} \end{cases}$
- $K: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a kernel function.



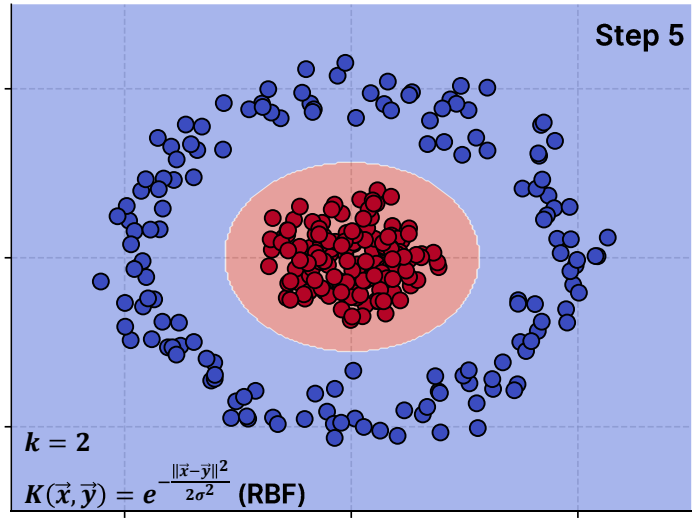
Kernel K-means

- Assign each point to a random cluster.
- **Repeat until no change occurs:**

$$\mathbf{z_{ij^*}} \coloneqq \begin{cases} \mathbf{1} & \textbf{if } j^* = \textbf{argmin}_j \left(-2 \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} K(\vec{x}^{(i)}, \vec{x}') + \frac{1}{|C_j|^2} \sum_{\vec{x}' \in C_j} \sum_{\vec{x}'' \in C_j} K(\vec{x}', \vec{x}'') \right) \\ \mathbf{0} & \textbf{otherwise} \end{cases}$$

Where:

- $z_{ij} = \begin{cases} 1 & \text{if } \vec{x}^{(i)} \in C_j \\ 0 & \text{otherwise} \end{cases}$
- $K: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a kernel function.



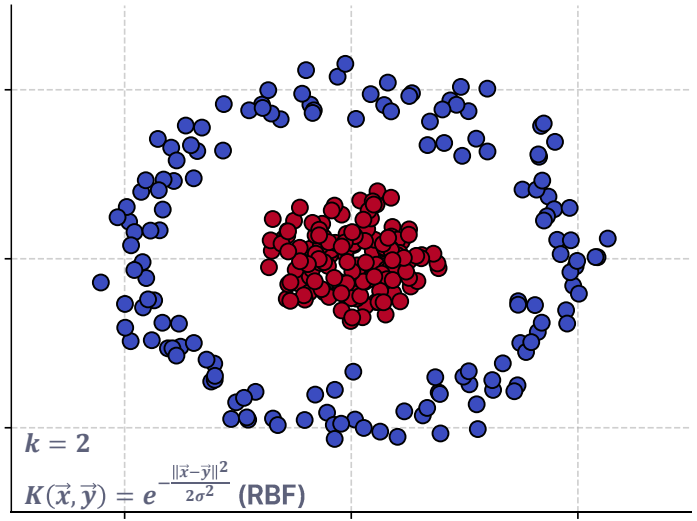
Kernel K-means

- Assign each point to a random cluster.
- Repeat until no change occurs:

$$z_{ij^*} \coloneqq \begin{cases} 1 & \text{if } j^* = \operatorname{argmin}_j \left(-2 \frac{1}{|C_j|} \sum_{\vec{x}' \in C_j} K(\vec{x}^{(i)}, \vec{x}') + \frac{1}{|C_j|^2} \sum_{\vec{x}' \in C_j} \sum_{\vec{x}'' \in C_j} K(\vec{x}', \vec{x}'') \right) \\ 0 & \text{otherwise} \end{cases}$$

Where:

- $z_{ij} = \begin{cases} 1 & \text{if } \vec{x}^{(i)} \in C_j \\ 0 & \text{otherwise} \end{cases}$
- $K: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a kernel function.



K-means in Python

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, silhouette_samples, adjusted_rand_score
from sklearn.mixture import GaussianMixture

km = KMeans(n_clusters = 4) # k = 4, by default it uses k-means++ initialization and does 10 runs
km.fit(X) # run the algorithm, compute the cluster centers
y = km.predict(X) # cluster assignment for the points it was fitted on
km.cluster_centers_
km.inertia_ # final distortion value

silhouette_score(X, y) # mean silhouette score over all samples
```

Summary

- **K-means** is a clustering algorithm which *partitions* the data points into a *fixed* number of clusters *k*.
- Each cluster is represented by a **centroid** and points are assigned to the cluster with the closest centroid.
- It uses an iterative **expectation-maximization** method to optimize the objective function and might get stuck in local optima.
- Number of clusters *k* is a *hyperparameter* which can be tuned by using the **elbow** method and the **silhouette** coefficient.
- K-means can only obtain *non-convex spherical* clusters and tends to produce clusters of *equal sizes*.
- **Soft K-means**, **Gaussian Mixture Models** and **Kernel K-means** are extensions which can deal with some of the limitations of k-means.

History of K-means

- ***"Sur la division des corps matériels en parties"***
  - First introduced the idea — Hugo Steinhaus, 1957
- ***"Least square quantization in PCM"***
  - First proposed the algorithm, but published it outside Bell Labs only in 1982 — S. P. Lloyd, 1957
  - Sometimes called Lloyd's algorithm
- ***"Cluster analysis of multivariate data: efficiency versus interpretability of classification"***
  - Basically published the same method as Lloyd — E.W. Forgy,1965
    - which is why it is sometimes referred to as Lloyd-Forgy
- ***"Some methods for classification and analysis of multivariate observations"***
  - First use of the term *"k-means"*. — J. MacQueen, 1967

Keywords

Clustering

K-Means | Expectation-Maximization

Cluster Centers (Centroids) | Partition

Elbow Method | Silhouette Coefficient | K-means++ | Rand Index

Soft K-means | Gaussian Mixture Models | Kernel K-means