Geometric Data Analysis

Partitional Clustering

MAT 6480W / STT 6705V

Guy Wolf
guy.wolf@umontreal.ca

Université de Montréal
Fall 2019

Outline

1. Partitional clustering
2. Lazy classifiers
   - *k*-nearest neighbor classification
   - Voronoi diagrams
3. Centroid-based classification
   - *k*-means
   - Choosing the number of centroids
4. Extensions and alternatives to *k-means*
   - *k-modes*
   - Shake & bake
   - *k*-medoids & PAM

Partitional clustering

Partitional clustering aims to directly partition the space into mutually exclusive “cells”, where each cluster is contained in a single cell.

Partitional clustering

Partitional clustering aims to directly partition the space into mutually exclusive “cells”, where each cluster is contained in a single cell.

Observation: classification methods essentially partition the data based on supervised learning.

Partitional clustering

Partitional clustering aims to directly partition the space into mutually exclusive “cells”, where each cluster is contained in a single cell.

Observation: classification methods essentially partition the data based on supervised learning.
Question: can we extend their principles to unsupervised settings?

Partitional clustering

Partitional clustering aims to directly partition the space into mutually exclusive "cells", where each cluster is contained in a single cell.

Observation: classification methods essentially partition the data based on supervised learning.
Question: can we extend their principles to unsupervised settings?
Challenge: all classifiers learned so far relied heavily on training with labeled data.

Partitional clustering

Partitional clustering aims to directly partition the space into mutually exclusive "cells", where each cluster is contained in a single cell.

Observation: classification methods essentially partition the data based on supervised learning.
Question: can we extend their principles to unsupervised settings?
Challenge: all classifiers learned so far relied heavily on training with labeled data.

Can we classify without training and extend this approach to work in unsupervised settings?

Lazy Classifiers

k-nearest neighbor classification

Can we classify without training?

Lazy Classifiers

k-nearest neighbor classification

Can we classify without training?

Eager & lazy learners in classification

Eager learners train an efficient classification model from labeled data, and then apply this model to new data.

Lazy learners do not train a model at all, but instead infer the class of new data points, as they arrive, from entire labeled training dataset.

Lazy Classifiers

k-nearest neighbor classification

Can we classify without training?

Eager & lazy learners in classification

Eager learners train an efficient classification model from labeled data, and then apply this model to new data.

Lazy learners do not train a model at all, but instead infer the class of new data points, as they arrive, from entire labeled training dataset.

All the classifiers we encountered so far (decision trees, SVM, etc.) are eager learners.

Lazy Classifiers

k-nearest neighbor classification

One of the most popular lazy learners is the k Nearest Neighbors (kNN) classifier:

kNN classifier

Training: none; use the entire dataset as the classification model.
Classification: given the input

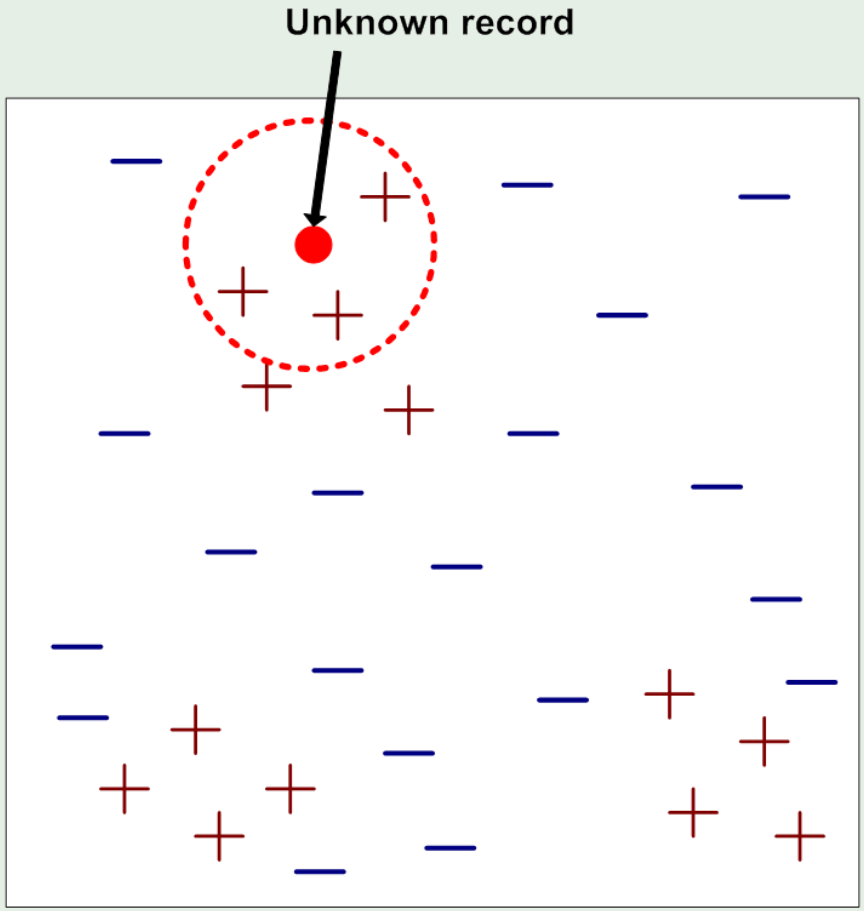
- new point y,
- training data $X = \{x_1, \ldots, x_N\}$ with labels $c_1, \ldots, c_N$,
- distance metric $d(x, y)$, and
- an integer $1 \leq k < N$,

find the k nearest neighbors $x_{j_1}, \ldots, x_{j_k} \in X$ to y with respect to the given distance metric, and choose the class that is most common in $c_{j_1}, \ldots, c_{j_k}$.

Lazy Classifiers

k-nearest neighbor classification

Example

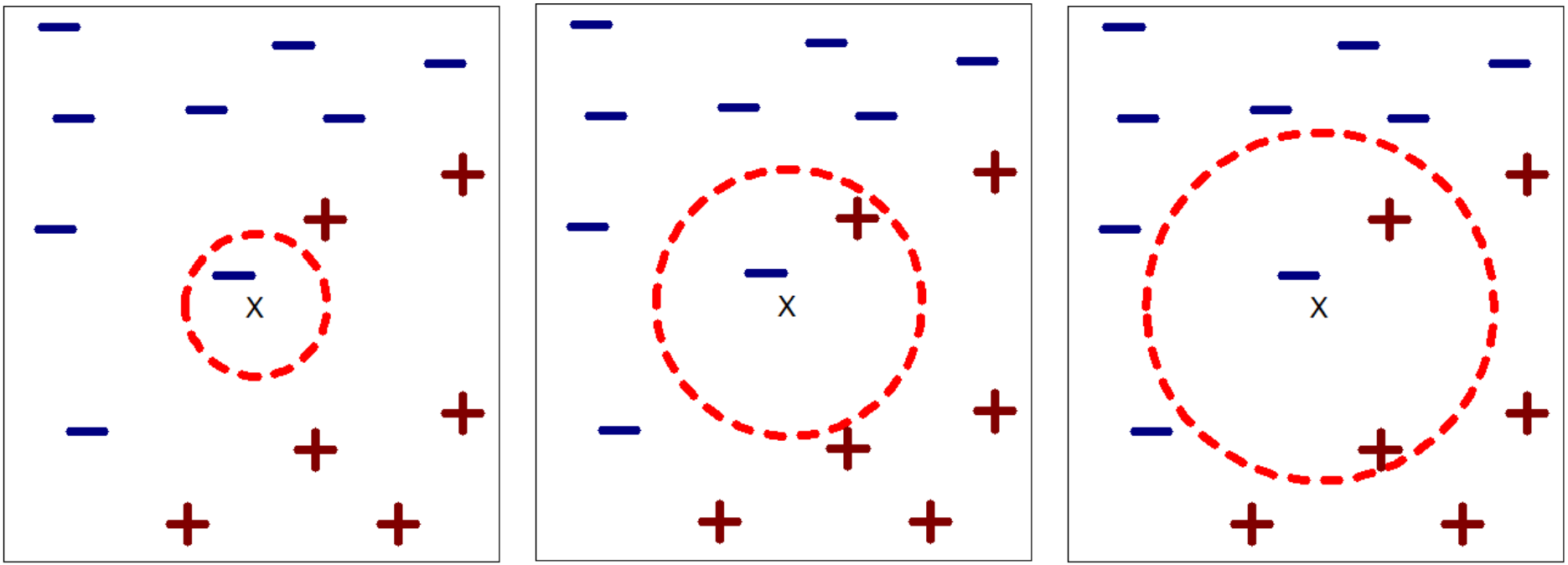


The new data point in this case will be classified as '+'.

Lazy Classifiers

k-nearest neighbor classification

Notice that the classification result my be very sensitive to the exact choice of k:



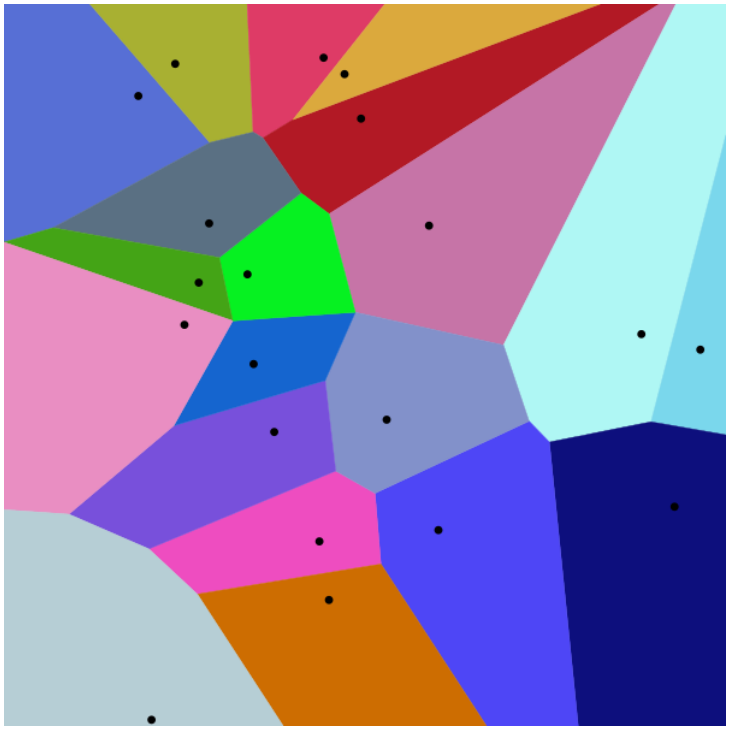
(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

The classification results in these cases will be '$-$' for $k = 1$, uncertain for $k = 2$, and '$+$' for $k = 3$.

Lazy classifiers

Voronoi diagrams

Consider $k = 1$ and N classes, with one training point from each class. The resulting classifier produces a partition of the space into convex cells, with the training samples as their centroids. In two dimensions such a partition is called a Voronoi diagram.



Euclidean Voronoi diagram

Manhattan Voronoi diagram

Centroid-based classification

k-means

The k-means algorithm aims to find k clusters by using a Voronoi-like partition of the space, where each cluster is contained within a single cell.

Observation: given any set of k "centroids", we can cluster points around them by applying 1-NN classification.
Question: how do we get to having a good set of k centroids?

Given a cluster $C = \{x_1, \ldots, x_\ell\}$, we want its centroid c to have a minimal average (squared) distance to other points in the cluster.

Example (Euclidean distance centroid)

We consider points in $\mathbb{R}^n$ and use the sum of squared errors (a.k.a. scatter)

$$\mathrm{SSE}(\vec{c}, \vec{x}_1, \ldots, \vec{x}_\ell) = \sum_{i=1}^{\ell} \|\vec{x}_i - \vec{c}\|^2 = \sum_{j=1}^{n} \sum_{i=1}^{\ell} (x_i[j] - c[j])^2$$

as a proxy for the average squared distance from the centroid to other points, and compute its minimum by taking:

$$0 = \frac{\partial SSE}{\partial c[j]} = \sum_{i=1}^{\ell} \frac{\partial}{\partial c[j]} (x_i[j] - c[j])^2 = 2 \sum_{i=1}^{\ell} (x_i[j] - c[j])$$

$$\Rightarrow \ell c[j] = \sum_{i=1}^{\ell} x_i[j] \Rightarrow \vec{c} = \mathrm{mean}\{\vec{x}_1, \ldots, \vec{x}_\ell\}$$

Example (Manhattan distance centroid)

We consider points in $\mathbb{R}^n$ and use the sum of absolute errors

$$\mathrm{SAE}(\vec{c}, \vec{x}_1, \ldots, \vec{x}_\ell) = \sum_{i=1}^{\ell} \|\vec{x}_i - \vec{c}\|_1 = \sum_{j=1}^{n} \sum_{i=1}^{\ell} |x_i[j] - c[j]|$$

as a proxy for the average distance from the centroid to other points, and compute its minimum by taking:

$$0 = \frac{\partial SAE}{\partial c[j]} = \sum_{i=1}^{\ell} \frac{\partial}{\partial c[j]} |x_i[j] - c[j]| = \sum_{i=1}^{\ell} \mathrm{sign}(x_i[j] - c[j])$$

$$\Rightarrow \#\{x_i[j] \leq c[j]\} = \#\{x_i[j] \geq c[j]\} \Rightarrow \vec{c} = \mathrm{median}\{\vec{x_1}, \ldots, \vec{x_\ell}\}$$

Centroid-based classification

k-means

The k-means algorithm combines the notion of building clusters around centroids and inferring centroids from clusters into one algorithm that iterates these two steps until convergence is reached:

k-means

Initialization: choose k random centroids $c_1, \ldots, c_k \in X$

Repeat the following steps:

- **Assignment:** perform 1-NN classification to (re)assign each $x \in X$ to one of the cluster centroids
- **Update:** recompute each c_j to be the centroid of all the points assigned to its cluster

Until the centroids/clusters stabilize (i.e., do not change)

k-means

Initialization: choose k random centroids $c_1, \ldots, c_k \in X$

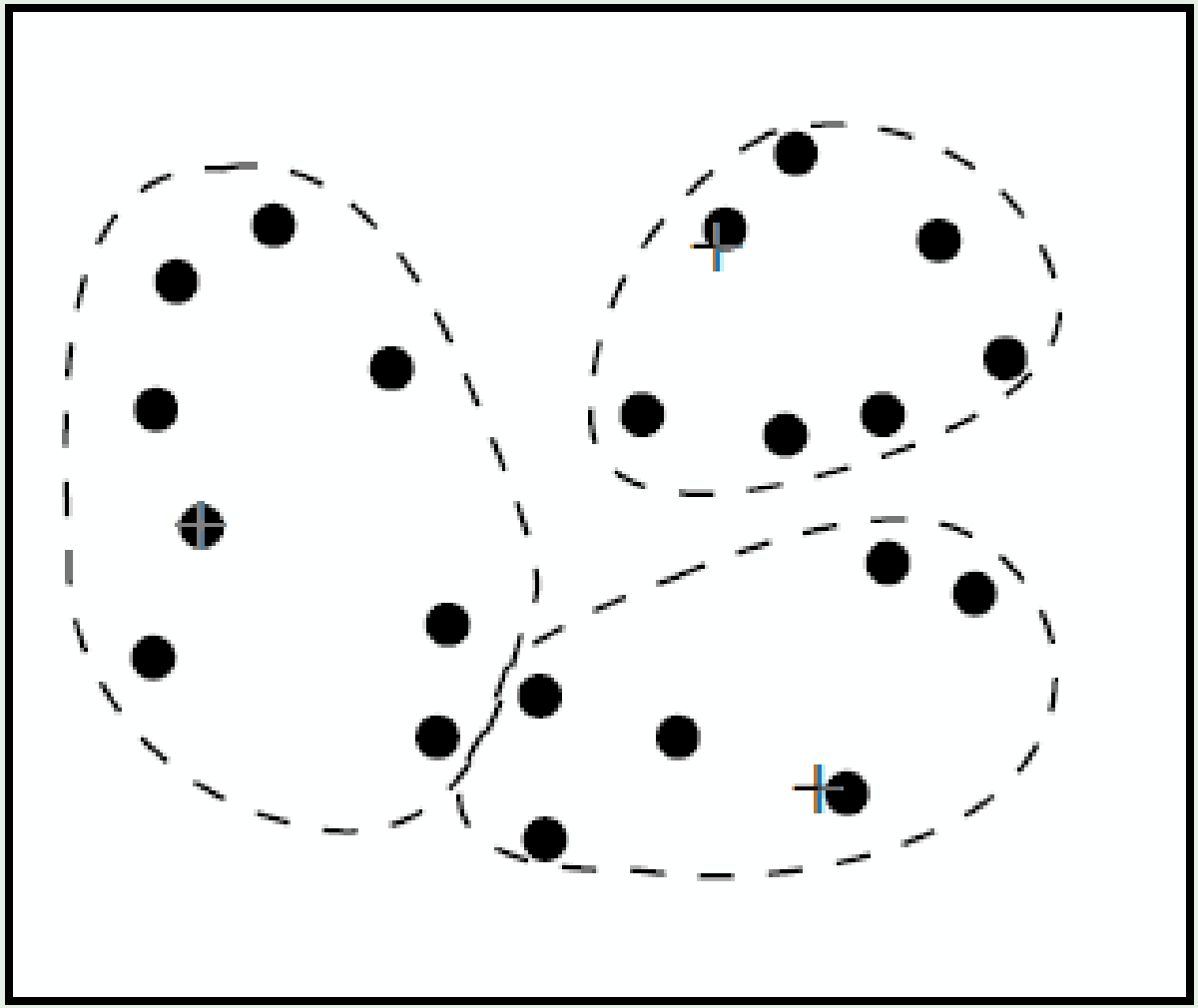
Repeat the following steps:

- **Assignment:** perform 1-NN classification to (re)assign each $x \in X$ to one of the cluster centroids
- **Update:** recompute each c_j to be the centroid of all the points assigned to its cluster

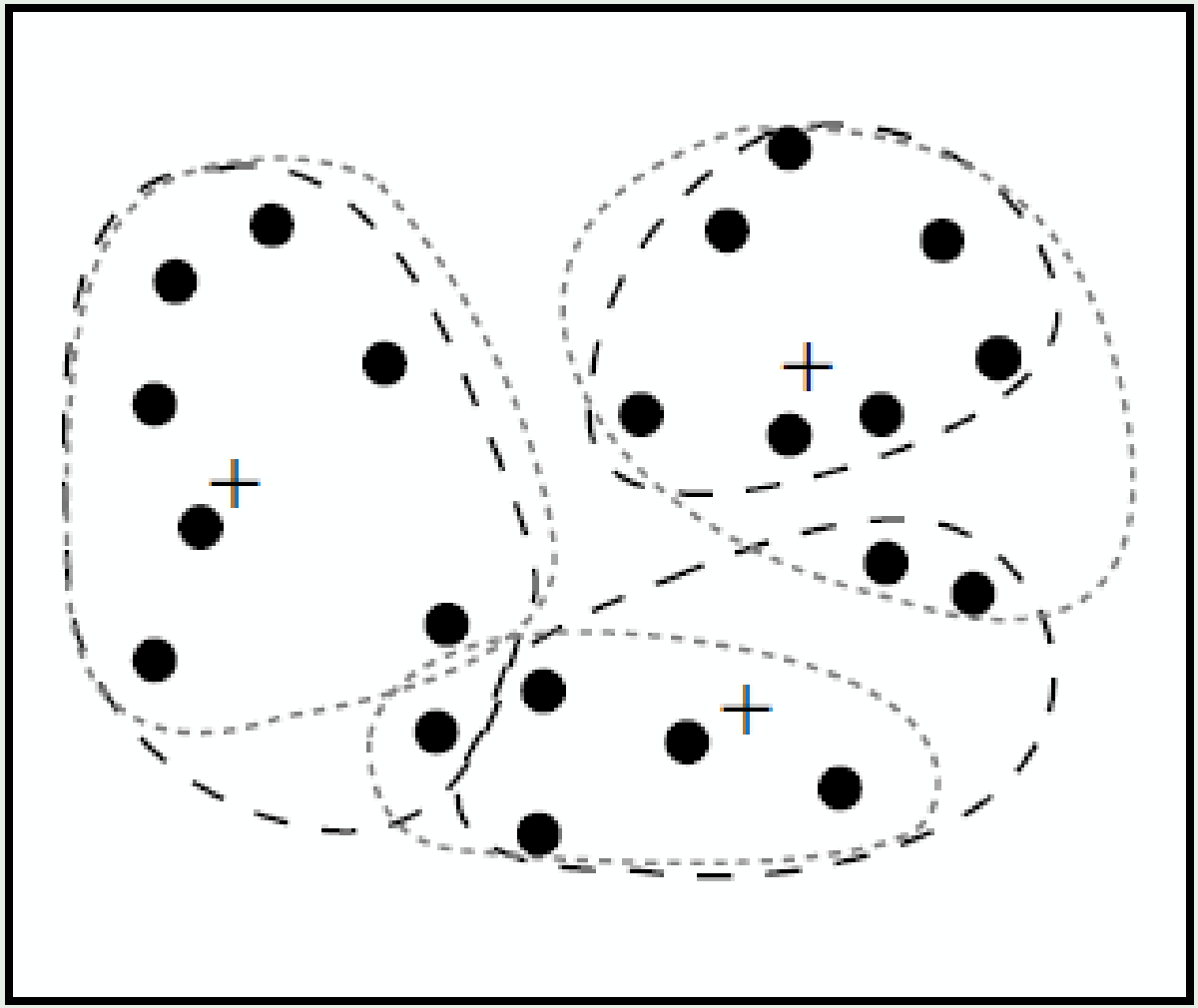
Until the centroids/clusters stabilize (i.e., do not change)

- This algorithm tries to optimize the total SSE/SAE (or equivalent measure) of all the clusters.
- Convergence is guaranteed by SSE/SAE decreasing in each step.
- However, it often finds a local minimum rather than a global one.

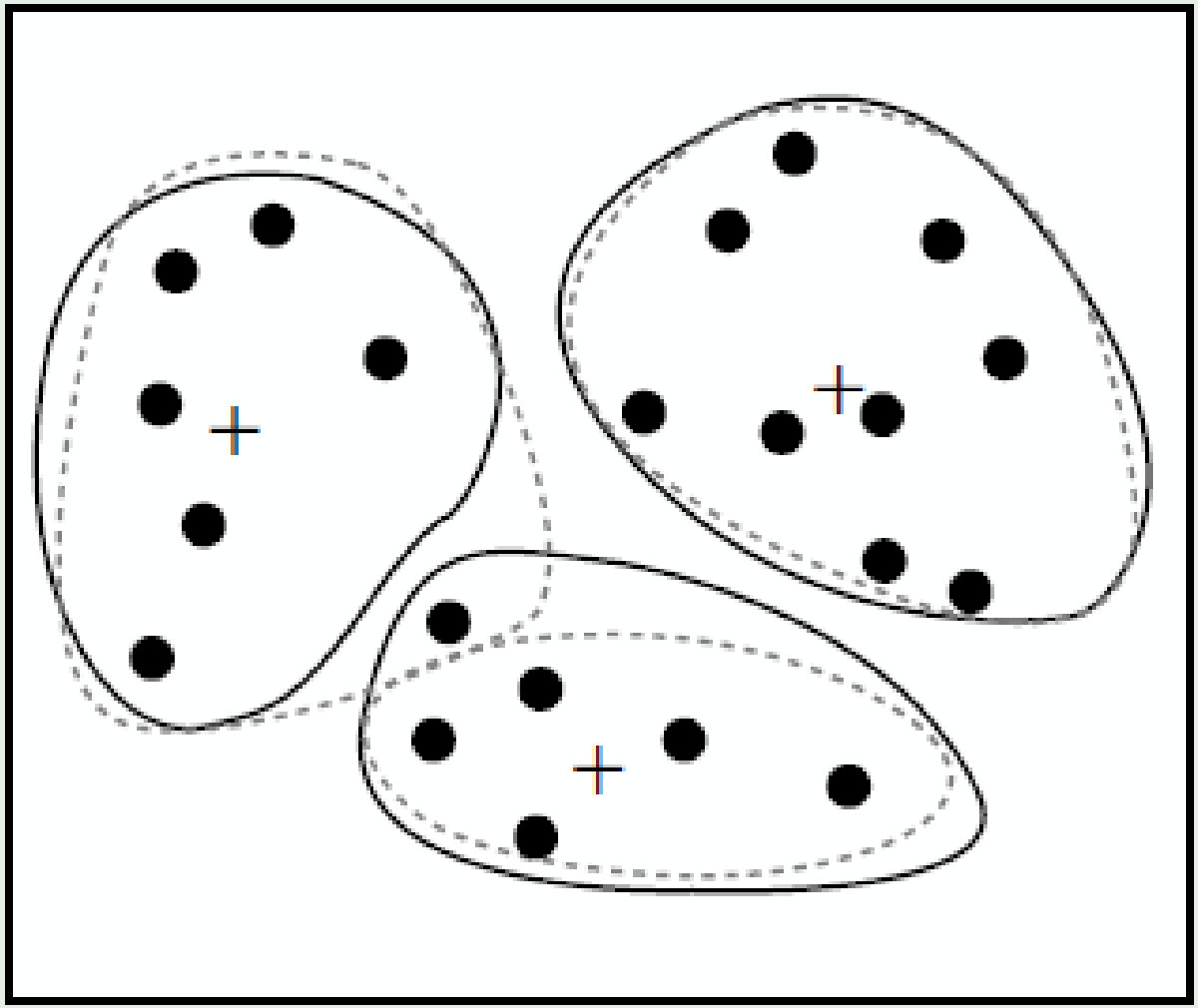
Example



Example



Example

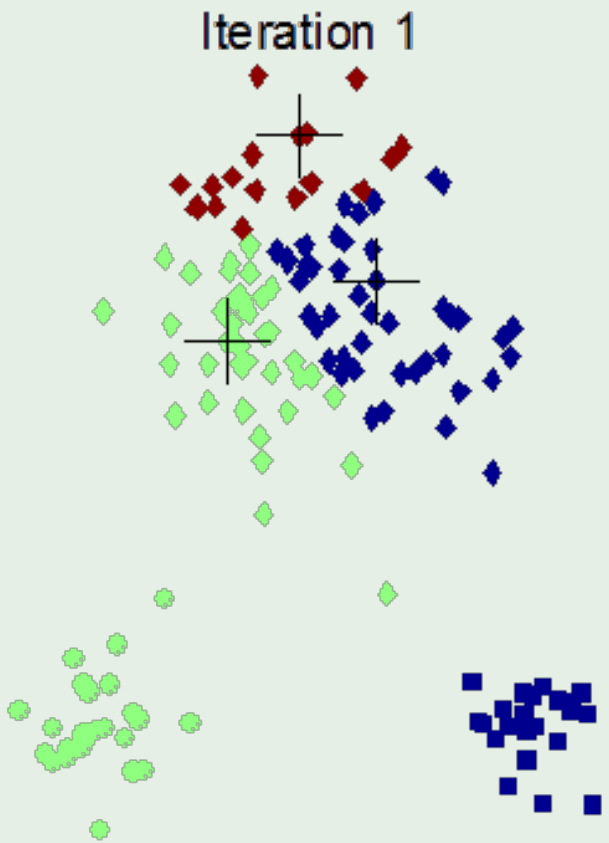


Centroid-based classification

k-means

While simple and popular, it is extremely sensitive to its initialization:

Example (Good initialization)

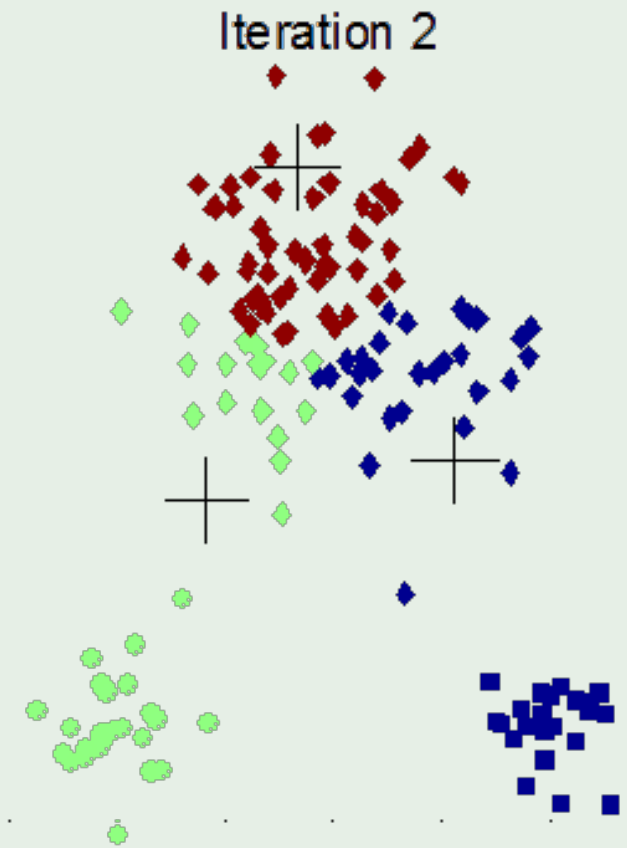


Centroid-based classification

k-means

While simple and popular, it is extremely sensitive to its initialization:

Example (Good initialization)

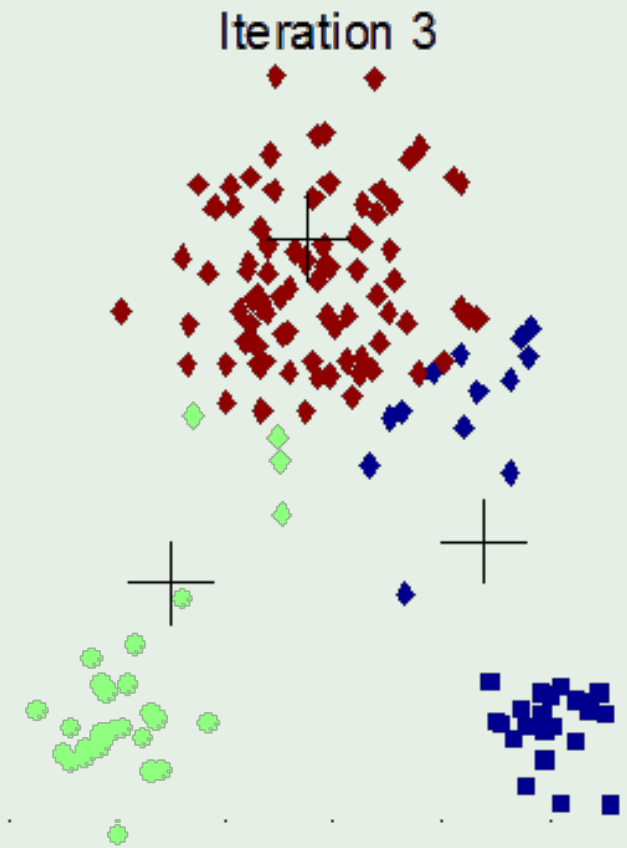


Centroid-based classification

k-means

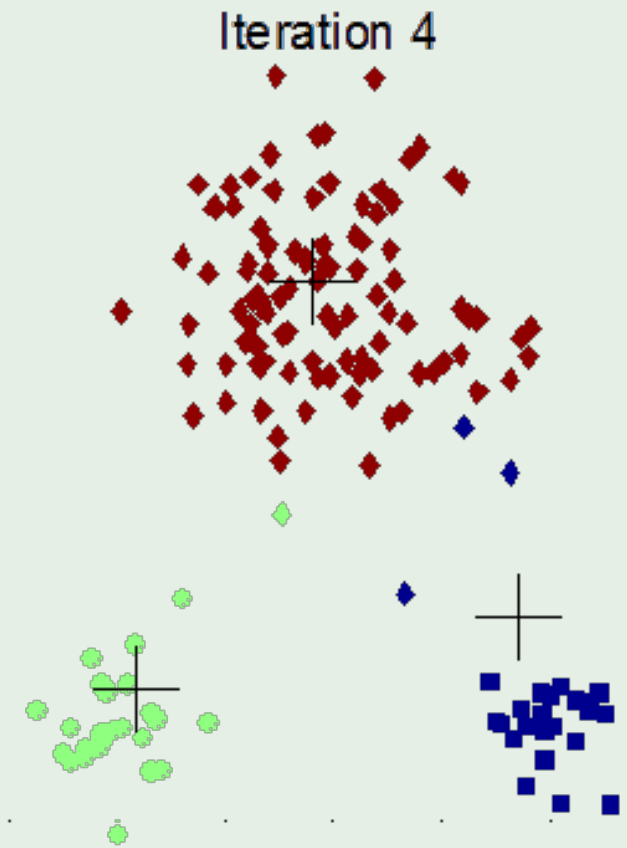
While simple and popular, it is extremely sensitive to its initialization:

Example (Good initialization)



While simple and popular, it is extremely sensitive to its initialization:

Example (Good initialization)

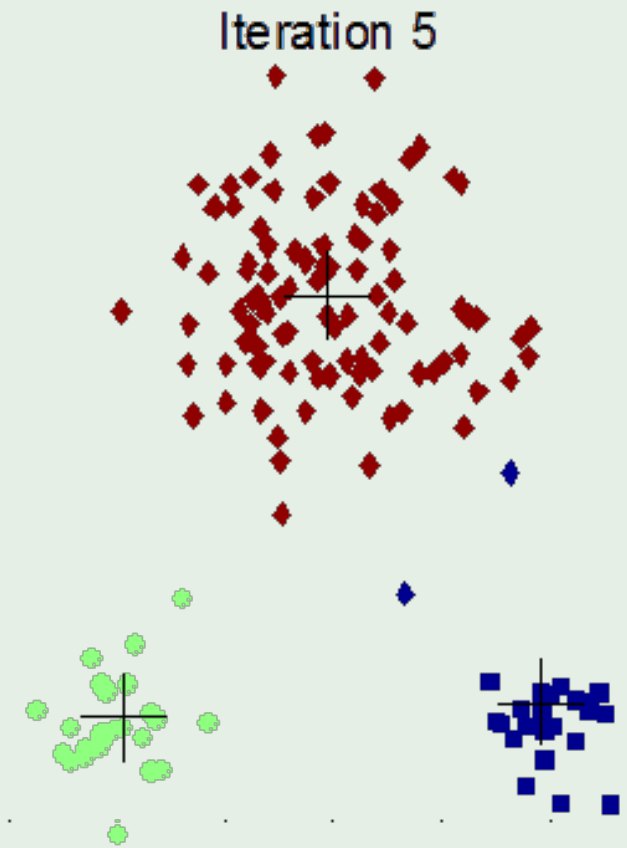


Centroid-based classification

k-means

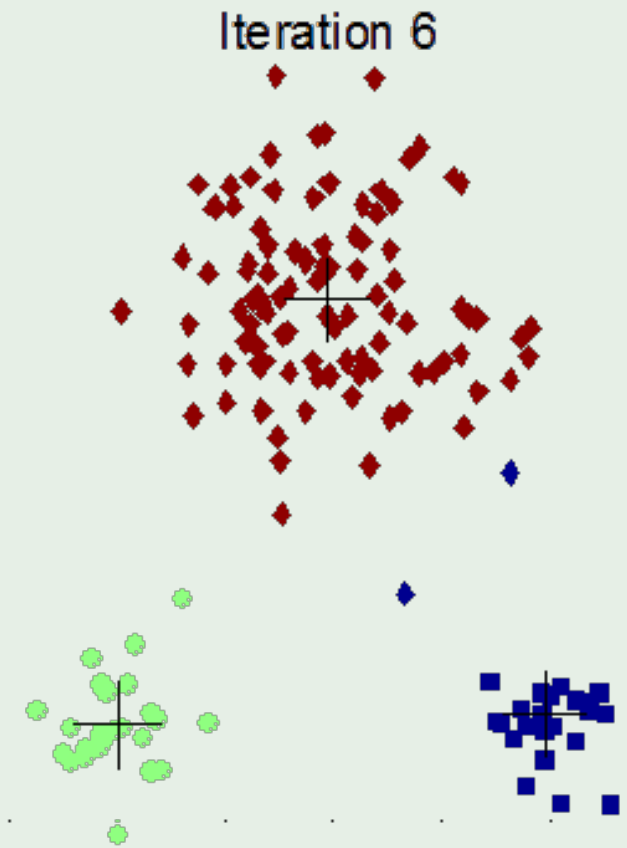
While simple and popular, it is extremely sensitive to its initialization:

Example (Good initialization)



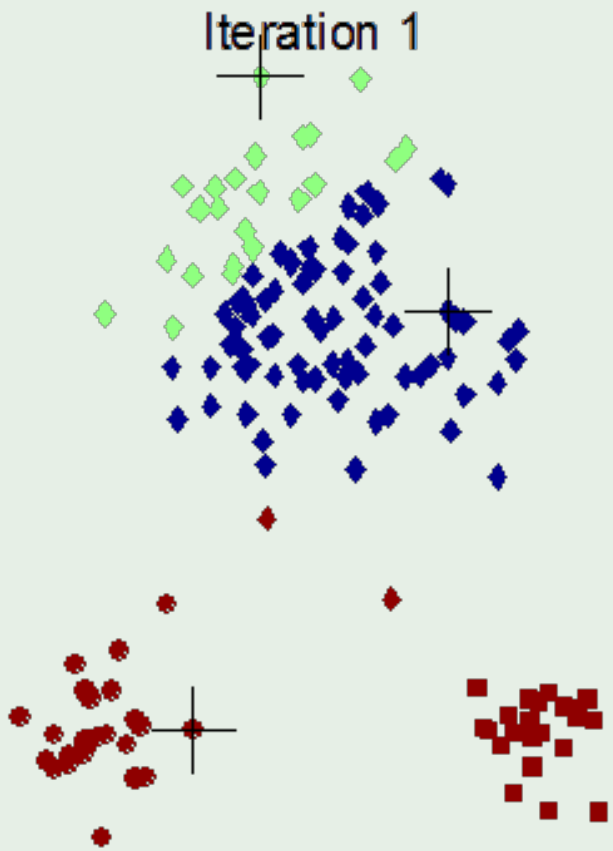
While simple and popular, it is extremely sensitive to its initialization:

Example (Good initialization)



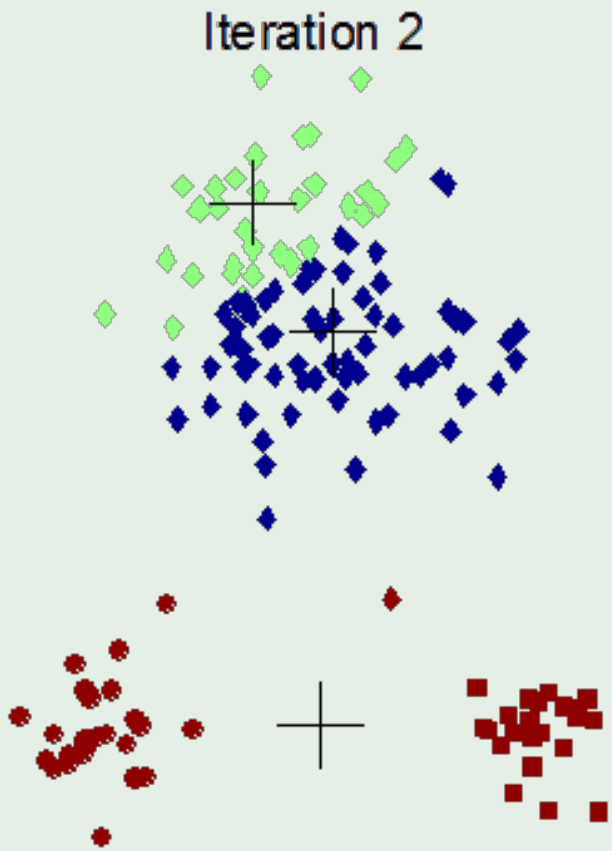
While simple and popular, it is extremely sensitive to its initialization:

Example (Bad initialization)



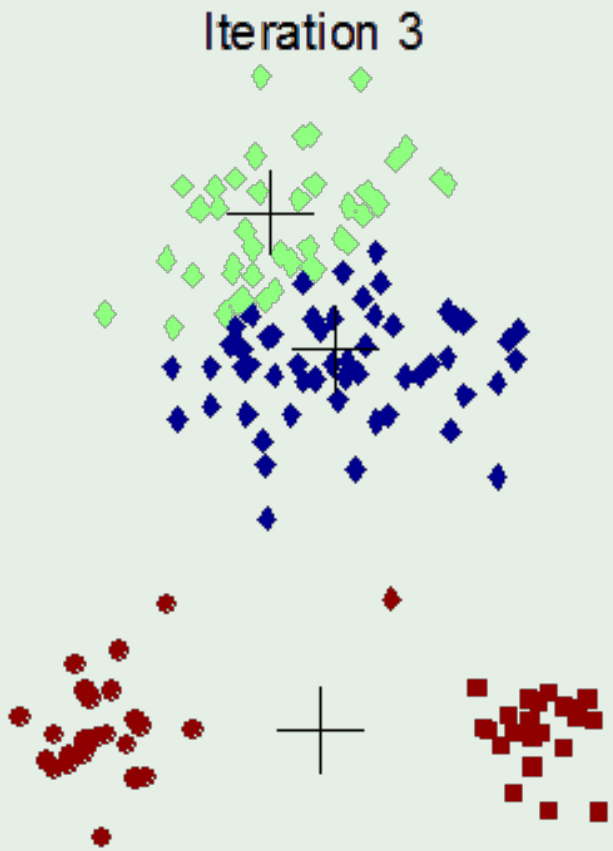
While simple and popular, it is extremely sensitive to its initialization:

Example (Bad initialization)



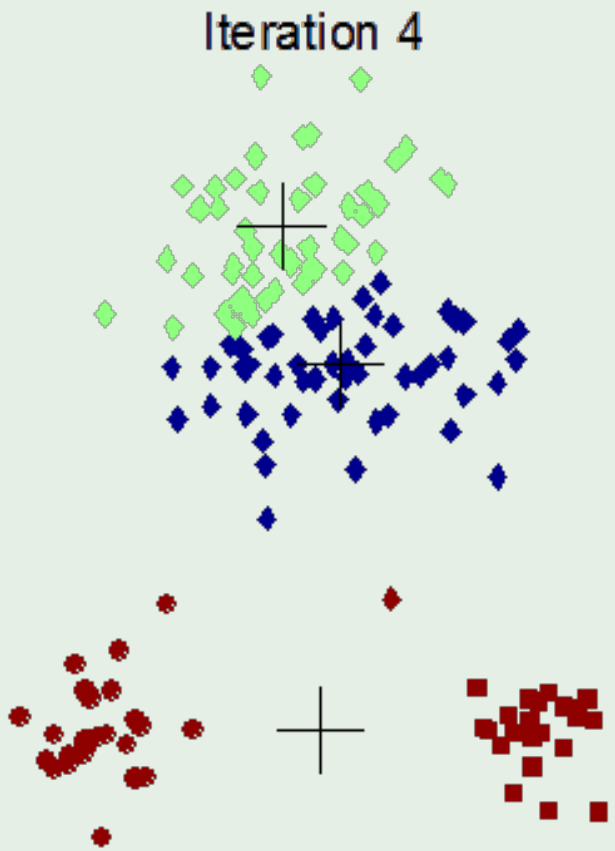
While simple and popular, it is extremely sensitive to its initialization:

Example (Bad initialization)



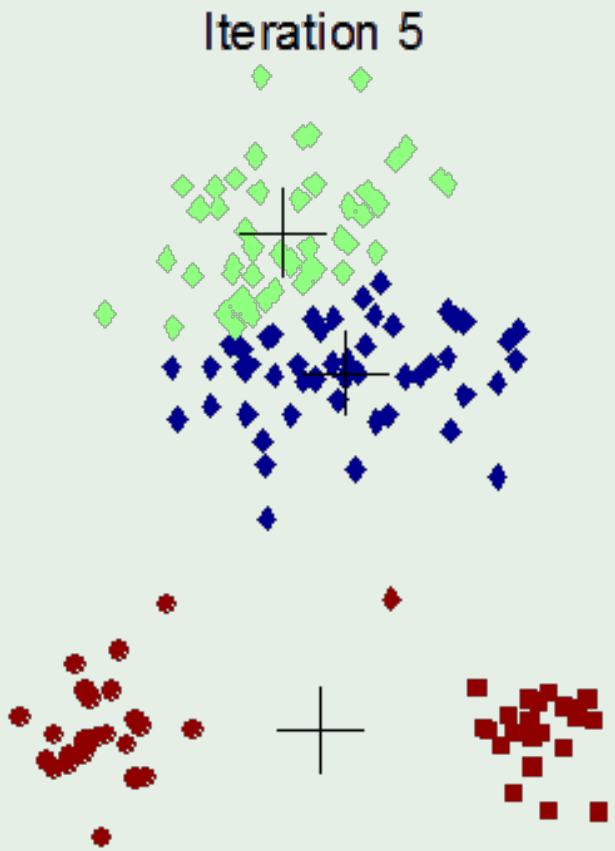
While simple and popular, it is extremely sensitive to its initialization:

Example (Bad initialization)



While simple and popular, it is extremely sensitive to its initialization:

Example (Bad initialization)



Centroid-based classification

k-means

Since k-means performances highly depend on the initialization, it is often run multiple times with random initializations.

Similarly, if the number of clusters is unknown, several values of k can be used in multiple runs.

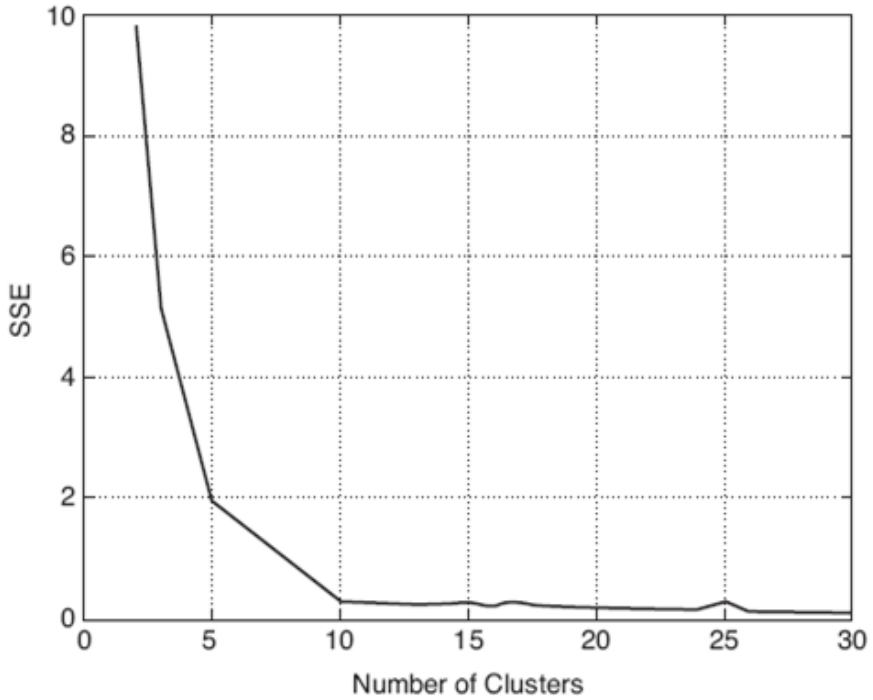
Finally, the quality of each clustering partition is computed (e.g., with total SSE/SAE) and the best partition is chosen.

Notice, however, that the SSE/SAE of clusters will typically decrease when we increase k, since the resulting clusters will be smaller.

Centroid-based classification

Choosing the number of centroids

The "knee method" is a useful SSE-based heuristic for choosing k:

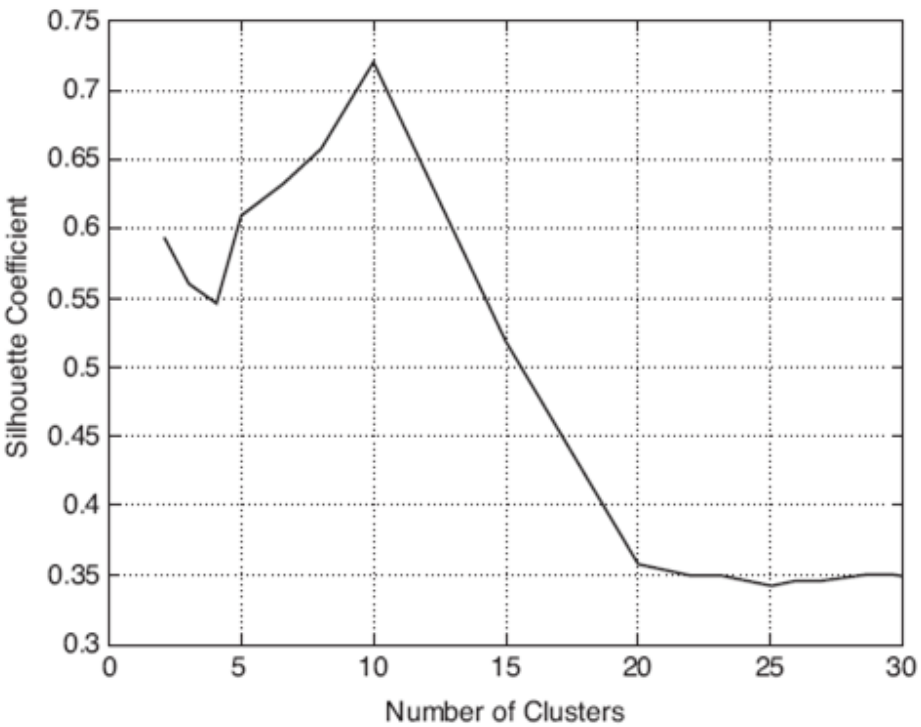


Choose k on the "knee" of the curve where its impact on SSE stabilizes (e.g., between 5 and 10 in this plot).

Centroid-based classification

Choosing the number of centroids

Alternatively, one can use the mean silhouette coefficient curve:



Let $a(x)$ be the mean distance of x from points in its cluster C:

$$a(x) = \text{mean}\{\text{dist}(x, y) | y \in C\}$$

Let $b(x)$ be the minimal mean distance of x from other clusters:

$$b(x) = \min_{C' \neq C} [\text{mean}\{\text{dist}(x, y) | y \in C'\}]$$

The silhouette coefficient of x is

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

In this case k can be chosen as the peak of the mean silhouette curve (e.g., 10 in this plot).

Centroid-based classification

Choosing the number of centroids

Other simple steps that may be used to improve & stabilize k-means:

Preprocessing:

- Normalization / standardization
- Dimensionality reduction
- Outlier elimination

Postprocessing:

- Eliminating small clusters
- Splitting "loose" clusters with high SSE/SAE
- Merging close clusters with low combined SSE/SAE

These steps allow a greater initial k to be used, and then reduce the number of clusters in postprocessing.

Extensions and alternatives to k-means

k-modes

The k-means algorithm was formulated with **numerical attributes** in mind, but it can be extended to **nominal data** as well using the k-modes algorithm.

Recall that the basic operations in this approach are finding **nearest neighbors** and computing **centroids**, and both rely on **minimizing distances** between centroids and data points in their clusters.

The k-modes algorithm uses the **mismatch distance** $d(x, y) = \#\{x[j] \neq y[j]\}$ for both 1-NN and SAE computations. It can be shown that the SAE of a cluster is minimized by **setting c[j] to be the mode** of the j-th attribute in the cluster.

The algorithm itself is identical to k-means, with the appropriate distance and centroid computations for nominal data.

Extensions and alternatives to k-means

Shake & bake

An extension of k-means was presented in (David & Averbuch, 2012)[1] as part of the LDF hierarchical clustering. A simplified version is presented here:

Shake step - random centroid selection

Initialize the set of centroid candidates $V \leftarrow X$

Repeat the following steps

- Choose a random point $v \in V$ as a new centroid
- $V \leftarrow V \setminus \{u \in V | \text{dist}(u, v) < \varepsilon\}$ (for a configurable $\varepsilon > 0$)

Until $V = \emptyset$

A partition can then be obtained with k-means ($k = \#\text{chosen centroids}$).

[1] *Applied and Computational Harmonic Analysis*, 33(1):1–23, 2012.

Bake step - new clustering distance metric

Input: a set of clustering partitions $C_1, \ldots, C_\ell : X \to \mathbb{N}$
For $j = 1, \ldots, \ell$, define

$$D_j(x, y) = \begin{cases} 0 & x = y \\ 0.5 & C_j(x) = C_j(y) \\ 1 & C_j(x) \neq C_j(y) \end{cases}$$

Set $D(x, y) = \ell^{-1} \sum_{j=1}^{\ell} D_j(x, y)$

The final clustering is obtained by using this metric to choose centroids and run k-means.

Extensions and alternatives to k-means

k-medoids & PAM

The k-means algorithm has two major drawbacks (even for center-based convex clustering):

1. It is sensitive to outliers and noise
2. It requires a good way of choosing centroids

A variation of k-means called k-medoids aims to alleviate these drawbacks by forcing selected centroids (now called "medoids") to be actual points observed in the data.

This approach can be realized in several ways (e.g., naïvely "correct" centroids to nearest data point in the basic k-means implementation).

Extensions and alternatives to k-means

k-medoids & PAM

PAM (Partition Around Medoids) is a popular realization of k-medoids that does not require any centroid computation:

PAM-based k-medoids

Initialization: choose k random medoids $c_1, \ldots, c_k \in X$

Repeat the following steps:

- **Assignment:** perform 1-NN classification to (re)assign each $x \in X$ to one of the cluster medoids
- **Update:** use random/arbitrary order to scan $X \setminus \{c_1, \ldots, c_k\}$
  For each $y \in X \setminus \{c_1, \ldots, c_k\}$ **do** the following steps:
  - **For** $i = 1, \ldots, k$, let $\Delta_i =$ SSE/SAE diff. of setting '$c_i \leftarrow y$'
  - **If** $\min_i \Delta_i = \Delta_{i'} < 0$, **then** set $c_{i'} \leftarrow y$

Until the medoids stabilize (i.e., do not change)

CLARA & CLARANS use sampling to improve the scalability of PAM

Summary

Partitional approaches are distance-based and tend to construct convex-shaped centroid-based clusters.

- Can be regarded as generalization of lazy classifiers to descriptive tasks
- Depend on having appropriate distance notion

The most popular examples of such approaches are k-means and its variations.

- Classic versions derived via SSE / SAE
- Sensitive to initialization
- Scalability is a challenge, especially for non-numerical data