HIERARCHICAL CLUSTERING AND k-MEANS
LECTURE: UNSUPERVISED LEARNING AND EVOLUTIONARY COMPUTATION USING R

Jakob Bossek

MALEO Group, Department of Computer Science, Paderborn University, Germany

17th Nov, 2024

Learning Goals

- Distingish supervised and unsupervised learning
- Formalise the problem of group identification (aka clustering)
- Learn about distance measures between sets of points
- Hierarchical clustering
- k-means clustering

Supervised Learning

We are given a set $\mathcal{X} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ of labelled data where $y_i \in \mathcal{Y} = \{C_1, \ldots, C_k\}$ are known class labels.

- **Goal**: Given a new observations x' without class label, predict which class it most likely belongs to
- Application: in insurance predict whether a customer will pay back a credit or not ($\mathcal{Y} = \{\text{Yes, No}\}$)

Unsupervised Learning

Supervised Learning

We are given unlabelled data $\mathcal{X} = \{x_1, \ldots, x_N\}$.

- **Goal**: find suitable grouping (learn about the data structure and its characteristics)
- Application: in marketing find homogeneous groups of customers $\rightsquigarrow$ customer segmentation

Reminder: k-partition

Mathematically rigorous formulation of the clustering problem

Definition (k-partition)

A *k-partition* of a set $\mathcal{X}$ is a decomposition of $\mathcal{X}$ into $k > 0$ *non-empty* subsets $C_1, \ldots, C_k$ such that the following holds:

1. $C_i \cap C_j = \emptyset$ for $1 \leq i \neq j \leq k$, i.e., the subsets are pairwise disjoint
2. $\bigcup_{i=1}^{k} C_i = \mathcal{X}$, i.e., the union of all the subsets is $\mathcal{X}$ itself. We say that the partition *covers* $\mathcal{X}$

Another dataset

(a) Iris setosa

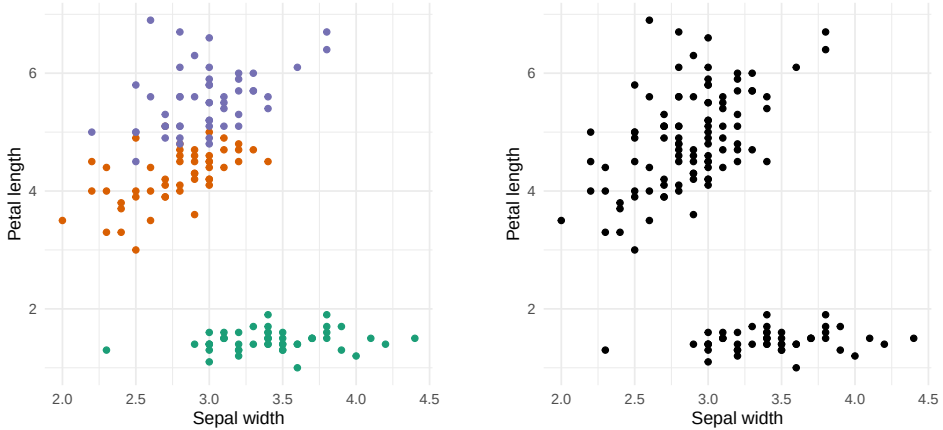
(b) Iris virginica.

(c) Iris versicolor.

Fisher's Iris flower data set (Fisher 1936)

- 50 samples from each of three species of Iris
- Five attributes: `sepal length`, `sepal width`, `petal length`, `petal width` and `species` (the known class label)
- Typical simple test case for machine learning algorithms.

What we see vs. what algorithm sees



Attention: Here, we know the class labels, but the algorithm does not!

Challenges

- **Attention**: true class labels are unknown![1]
- How to measure the "quality" of a grouping?
  - Access homo- / heterogenity by means of (dis)similarity measures. But what is the distance between clusters (i.e., groups of points)?
  - What if our data contains categorical variables?
  - What is the "right" number of clusters?
- How to interpret the results of a clustering algorithm? What is a good clustering?

[1] In this section we will often used labelled data to check the capability of the clustering algorithms to detect certain patterns.

Foundations

Distance/Dissimilarity

Definition (Distance function)

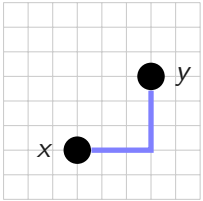
A function $d : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is called a *distance function* (or *dissimilarity function/measure* if the following conditions hold for all $x, y, z \in \mathbb{R}^p$

1. $d(x, y) \geq 0$ and $d(x, y) = 0$ if and only if $x = y$,
2. $d(x, y) = d(y, x)$ (symmetry),
3. $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality / Δ-inequality).

Typical distance measures

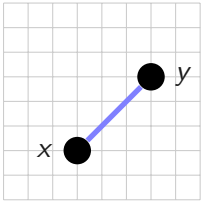
- Euclidean distance (L_2-norm)

$$d(x, y) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}.$$



- Manhattan-block distance (L_1-norm)

$$d(x, y) = \sum_{i=1}^{p} |x_i - y_i|.$$



Typical distance measures

Mahalanobis distance

Generalized squared distances (see outlier detection) defined as:

$$d(x, y) = (x - y)^T \cdot \Sigma^{-1} \cdot (x - y)$$

where $\Sigma \sim (p, p)$ is the (estimated) covariance matrix.
Effect of Σ^{-1}: conversion into "round" structure ("decorrelation")

Closeness/Similarity

Definition (Similarity function)

A function $s : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is called a *similarity function* (or *similarity measure*) if the following conditions hold:

1. $s(x, y) = s(y, x)$ (symmetry),
2. $s(x, y) \leq s(x, x)$ (no object can be more similar to another object than to itself),
3. optional, but often required, $s(x, y) \in [0, 1]$.

Typical closeness measures

Pearson's coefficient of correlation[2]

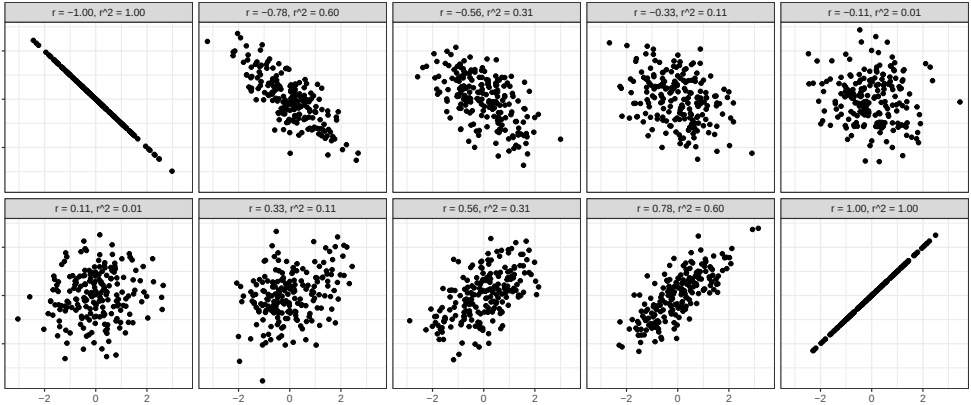
$$r_{xy} = \frac{s_{xy}}{s_x \cdot s_y} = \frac{\sum_{i=1}^{p}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{p}(x_i - \bar{x})^2 \cdot \sum_{i=1}^{p}(y_i - \bar{y})^2}} \in [-1, 1]$$

Problem: $r_{xy} \notin [0, 1] \rightsquigarrow$ square transformation

$$r_{xy}^2 = \left(\frac{s_{xy}}{s_x \cdot s_y}\right)^2 = \frac{\left(\sum_{i=1}^{p}(x_i - \bar{x}) \cdot (y_i - \bar{y})\right)^2}{\sum_{i=1}^{p}(x_i - \bar{x})^2 \cdot \sum_{i=1}^{p}(y_i - \bar{y})^2} \in [0, 1]$$

[2] We introduced it simply as the *correlation* (see math foundations).

Typical closeness measures



Bi-variate $\mathcal{N}(\mu, \Sigma)$-distributions with different specified Pearson correlation r_{xy}.

Conversion: similarities $\rightarrow$ dissimilarity

- If $d(\cdot, \cdot)$ is Euclidean we can compute a positive-semidefinite similarity function by

$$s(x, y) := \frac{1}{2}\left(d(x, 0)^2 + d(y, 0)^2 - d(x, y)^2\right)$$

- If d is a dissimilarity function than any non-decreasing function of d is a smilarity function. E.g.,

$$s(x, y) = \exp\left(-\frac{d(x, y)^2}{t}\right), t > 0$$

or

$$s(x, y) = \frac{1}{1 - d(x, y)}$$

Exercises

Imagine that our data points are not vectors in $\mathbb{R}^p$, but instead sub-sets of some universe U. Think of a suitable distance function

$$d : U \times U \to \mathbb{R}$$

which maps to sets $A, B \subset U$ to a distance.

Sample solutions

Hierarchical Clustering

Basics

The idea of the *hierarchical clustering algorithm* (HCA) is fairly simple:

Agglomerative approach[3]

- Start with one cluster per observation (i.e., we have N clusters)
- Repeat until only one cluster remains: fuse two "most similar" clusters

Divisive approach

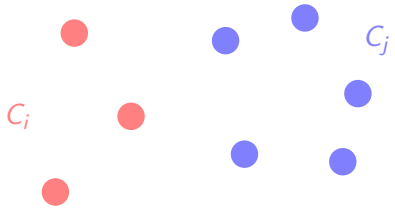
- Start with a single cluster (containing all observations)
- Repeat until N clusters are formed: split the two "most dissimilar" clusters

[3] Most often used in practice.

Major challenge

- So far, distance $d(x, y)$ between two observations $x, y \in \mathcal{X} \subset \mathbb{R}^p$
- Now: distance $D(C_i, C_j)$ between two sets $C_i, C_j \subseteq \mathcal{X} \subset \mathbb{R}^p$
  Given two clusters, how do we measure the distance or closeness of two sets of points?

$$d(C_i, C_j) := ?$$

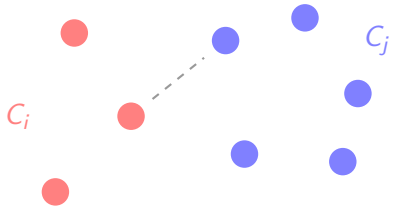


Distance between sets

Single-Linkage

Take the smallest inter-point distance:

$$D(C_i, C_j) := \min_{x \in C_i, y \in C_j} d(x, y).$$

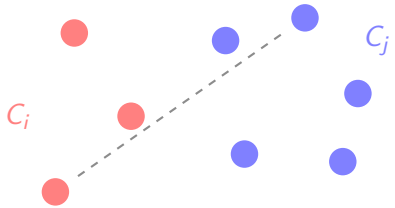


Distance between sets

Complete-Linkage

Take the largest inter-point distance:

$$D(C_i, C_j) := \max_{x \in C_i, y \in C_j} d(x, y).$$

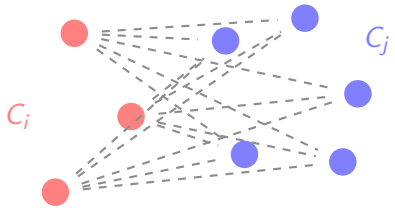


Distance between sets

Average-Linkage

Take the arithmetic mean of all inter-cluster distance:

$$D(C_i, C_j) := \frac{1}{|C_i| \cdot |C_j|} \sum_{x \in C_i} \sum_{y \in C_j} d(x, y)$$



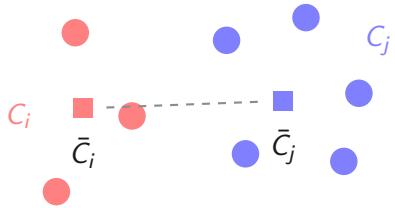
Distance between sets

Centroid-Approach

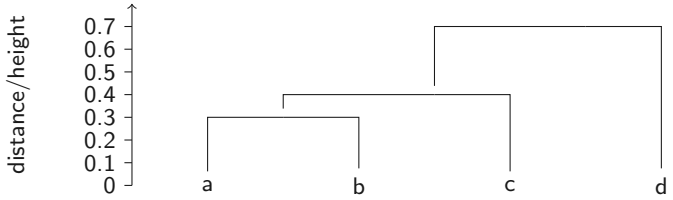
Take the distance between the centers of mass:

$$D(C_i, C_j) := d(\bar{C}_i, \bar{C}_j).$$

where $\bar{C}_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$ and $\bar{C}_j = \frac{1}{|C_j|} \sum_{x \in C_j} x$ are the *centroids*.



HCA example with single-linkage

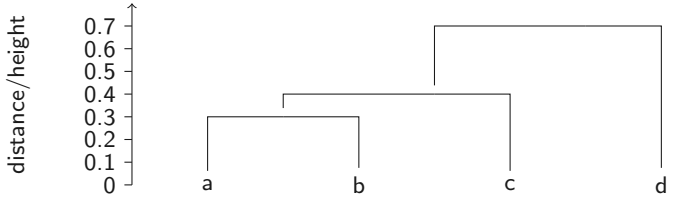


$$\begin{array}{c} \\ \{a\} \\ \{b\} \\ \{c\} \\ \{d\} \end{array} \begin{array}{c} \begin{array}{cccc} \{a\} & \{b\} & \{c\} & \{d\} \end{array} \\ \begin{pmatrix} 0 & 0.3 & 0.4 & 0.7 \\ 0.3 & 0 & 0.5 & 0.8 \\ 0.4 & 0.5 & 0 & 0.8 \\ 0.7 & 0.8 & 0.8 & 0 \end{pmatrix} \end{array}$$

Minimum distance is $D(\{a\}, \{b\}) = 0.3$
$\rightsquigarrow$ merge $\{a\}$ and $\{b\}$ at height 0.3
Update of distances:

$$\begin{aligned} D(\{a,b\},\{c\}) &= \min\{D(\{a\},\{c\}), D(\{b\},\{c\})\} \\ &= \min\{0.4, 0.5\} = 0.4 \\ D(\{a,b\},\{d\}) &= \min\{D(\{a\},\{d\}), D(\{b\},\{d\})\} \\ &= \min\{0.7, 0.8\} = 0.7 \end{aligned}$$

HCA example with single-linkage

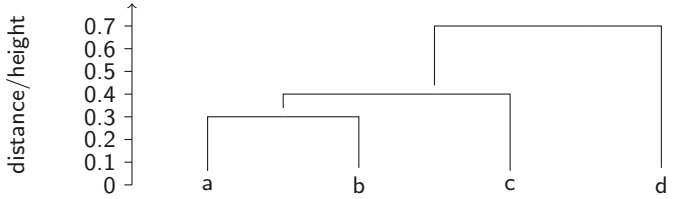


$$\begin{array}{c} \\ \{a,b\} \\ \{c\} \\ \{d\} \end{array} \begin{array}{c} \begin{array}{ccc} \{a,b\} & \{c\} & \{d\} \end{array} \\ \begin{pmatrix} 0 & 0.4 & 0.7 \\ 0.4 & 0 & 0.8 \\ 0.7 & 0.8 & 0 \end{pmatrix} \end{array}$$

Minimum distance: $D(\{a,b\},\{c\}) = 0.4$
$\rightsquigarrow$ merge $\{a,b\}$ and $\{c\}$ at height 0.4
Update of distances:

$$\begin{aligned} D(\{a,b,c\},\{d\}) &= \min\{D(\{a,b\},\{d\}), D(\{c\},\{d\})\} \\ &= \min\{0.7, 0.8\} = 0.7 \end{aligned}$$

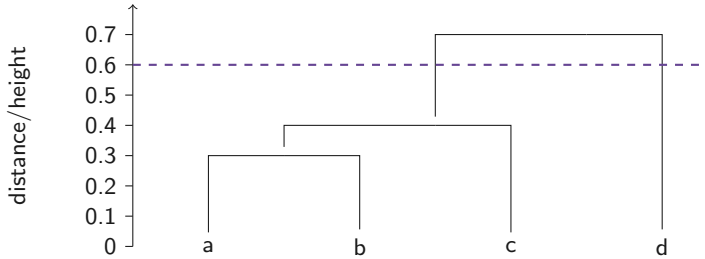
HCA example with single-linkage



$$\begin{array}{r} \\ \{a,b,c\} \\ \{d\} \end{array} \begin{array}{c} \begin{array}{cc} \{a,b,c\} & \{d\} \end{array} \\ \begin{pmatrix} 0 & 0.7 \\ 0.7 & 0 \end{pmatrix} \end{array}$$

Minimum distance: $D(\{a,b,c\},\{d\}) = 0.7$
$\rightsquigarrow$ merge $\{a,b,c\}$ and $\{d\}$ at height 0.7
Algorithm terminates!

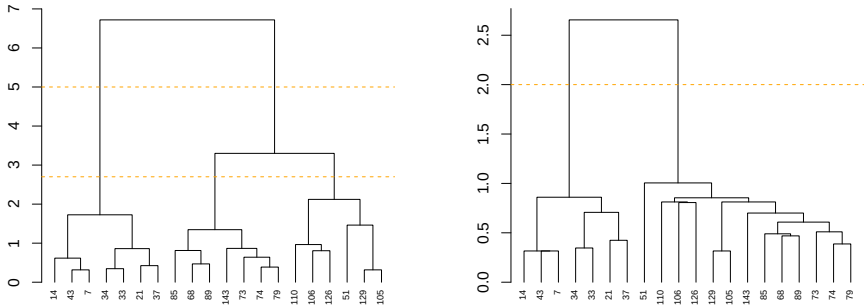
Dendrogram



- A *dendrogram* is a tree-like structure which is build bottom-up during (agglomerative) HCA
- Horizontal lines indicate the time clusters are merged
- Gap between horizontal lines: distance between clusters (large gaps indicates good separation)
- We can cut the dendrogram at an arbitrary height to obtain a clustering (a-posteriori; see, e.g., the dashed line)

HCA: example

HCA with single linkage on sample of 20 random flowers from `Iris`



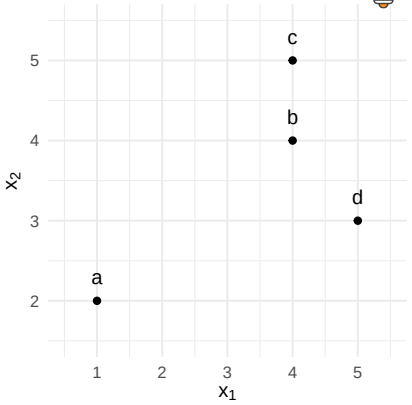
Complete linkage (left) and single linkage (right).

Observation: Two or three clusters seem plausible for complete linkage, just two for

Exercises

Consider the given set of observations:

1. Use the Manhattan-Block distance (L_1-norm) to derive the respective distance matrix.
2. Apply agglomerative hierarchical clustering with complete linkage to derive a clustering. Which number of clusters seems adequate?



Sample solutions

Sample solutions

Sample solutions

Hierarchical Clustering
Algorithm Complexity

- Standard algorithm **Hierarchical Agglomerative Clustering (HAC)** runs in $\mathcal{O}(N^3)$ and space $\Omega(N^2)$
- Special cases:
  - **SLINK** (Sibson 1973) for single-linkage: $\mathcal{O}(N^2)$
  - **CLINK** (Defays 1977) for complete-linkage: $\mathcal{O}(N^2)$
- General case: runtime improvements possible (e.g., $\mathcal{O}(N^2 \log N)$ by using more sophisticated/complex data-strucutures but often at the cost of additional space requirements

Hierarchical Clustering
Properties

Advantages ☺

- Flexible (bottom-up/top-down) approach due to option of (dis)similarity function[4]
- Number of cluster not needed a-priori
- Hierarchy and interpretability: dendrogram is a nice visual method two find the "right cut"
- Robust to small cluster or outliers

Disavantages ☹

- Too slow even for medium-sized data due to time- and space complexity
- Use of heuristics $\rightsquigarrow$ may stuck in local optimum
- Sensitive to the choice of linkage criteria
- Can struggle with non-convex clusters

[4] *Any* measure of distance can be used.

k-Means Clustering

k-Means Clustering

Different, so-called partition-based approach

- Specify the number of clusters k *a-priori*[5] and find a k-partition.
- **Core idea:** derive k clusters such that the distance between points within in cluster is rather low while the distance between points of different clusters tends to be high.
- *Within-cluster sum of squares (WCSS)* for cluster C:

$$W(C) = \frac{1}{|C|} \sum_{x,y \in C} ||x - y||^2. \tag{1}$$

- k-means aims to find k clusters $C_1, \ldots, C_k$ such that

$$\sum_{l=1}^{k} W(C_l) = \sum_{l=1}^{k} \frac{1}{|C_l|} \sum_{x,y \in C_l} ||x - y||^2 \rightarrow \text{min!}$$

k-Means Clustering

Remember this problem?

Definition (k-means clustering problem)

Given a set of observations $\mathcal{X} = \{x_1, \ldots, x_N\} \subset \mathbb{R}^p$ we aim to find a clustering $C_1, \ldots, C_k$ of $\mathcal{X}$ for a fixed $k > 1$ such that the following conditions hold:

1. $C_i \cap C_j = \emptyset$ for all $1 \leq i \neq j \leq k$
2. $\bigcup_{i=1}^{k} C_i = \mathcal{X}$
3. $C_1, \ldots, C_k = \arg\min_{C_1,\ldots,C_k} \sum_{i=1}^{k} \frac{1}{|C_i|} \sum_{x,y \in C_i} ||x - y||^2$

This is exactly the problem solved (approximately) by k-means!

Theorem

For each data set $\mathcal{X}$ and a cluster C it holds that

$$\frac{1}{|C|}\sum_{x,y\in C}||x-y||^2 = 2\sum_{x\in C}||x-\mu_I||^2 = 2\cdot|C|\cdot Var(C).$$

Here, $\mu_I = \frac{1}{|C|}\sum_{x\in C} x$ is the mean vector of the I^{th} cluster.[6] *I.e.*

- *The within-cluster sum of squares equals*
- *twice the sum of squared distances of the points assigned to the cluster which equals*
- *two times the (scaled) within-cluster variation.*

[6] A proof is given in the lecture notes.

$$\begin{aligned}
\frac{1}{|C|}\sum_{x,y\in C}||x-y||^2 &= \frac{1}{|C|}\sum_{x,y\in C_l}||(x-\mu_l)-(y-\mu_l)||^2 \\
&= \underbrace{\frac{1}{|C|}\sum_{x,y\in C}||x-\mu_l||^2}_{\text{independent of } y} + \underbrace{\frac{1}{|C|}\sum_{x,y\in C}||y-\mu_l||^2}_{\text{independent of } x} \\
&\quad - \frac{2}{|C_l|}\underbrace{\sum_{x,y\in C}(x-\mu_l)^T(y-\mu_l)}_{=0} \\
&= \frac{|C_l|}{|C_l|}\sum_{x\in C}||x-\mu_l||^2 + \frac{|C|}{|C|}\sum_{y\in C}||y-\mu_l||^2 + 0 \\
&= 2\sum_{x\in C}||x-\mu_l||^2
\end{aligned}$$

With this we can finally derive

$$2 \sum_{x \in C} ||x - \mu_I||^2 = 2 \cdot \underbrace{|C| \cdot \frac{1}{|C|}}_{=1} \sum_{x \in C} ||x - \mu_I||^2 = 2 \cdot |C| \cdot \mathrm{Var}(C)$$

which completes the proof. ♣

Lloyd's k-Means Algorithm

According to Lloyd's proposal (Lloyd 1982):

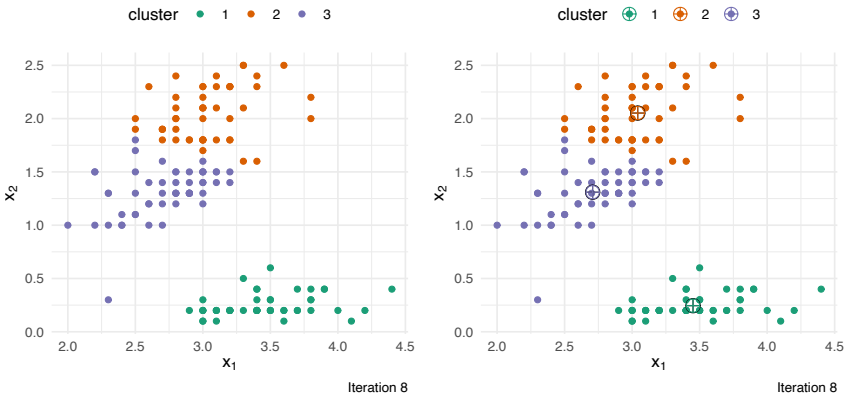
1. Assign each point $x \in \mathcal{X}$ uniformly at random to one of the k clusters $C_1, \ldots, C_k$.
2. Build cluster centers $\mu_l = \frac{1}{|C_l|} \sum_{x \in C_l} x$ for $l = 1, \ldots, k$.
3. Calculate the Euclidean distance of each point $x \in \mathcal{X}$ to each center.
4. Assign each point to its nearest cluster center, i.e.,

$$C_l = \{x \in \mathcal{X} \mid ||x - \mu_l||^2 = \min_{j=1,\ldots,k} ||x - \mu_j||^2\}.$$

5. Repeat from step (2) until the assignment is stable.[7]

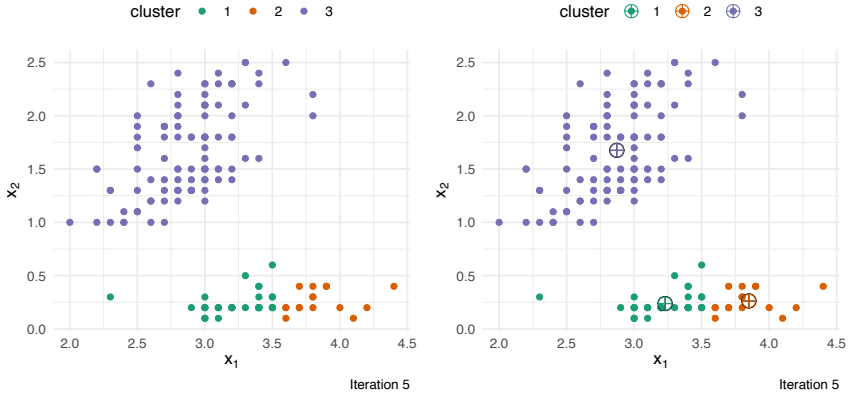
[7] I.e., no changes in the assignment phase occur.

k-means on iris



k-means on iris

Another run with a different RNG-seed[8]



[8] A Random Number Generator (RNG) expects a *seed* which detemines the initialization of pseudo-random number generation (and makes randomized experiments reproducible).

Why different results?

- **Sobering fact**: the k-means clustering problem is $\mathcal{NP}$-hard (Aloise et al. 2009) ☹
  I.e., we do not know an exact algorithm that finds an optimal solution to any k-means input in guaranteed reasonable (i.e., sub-exponential) time.
- We need to use heuristic algorithms to get "good" results in reasonable time.
- Lloyd's k-means algorithm (and all other practically relevant variants) are *heuristics*.
  I.e., they solve the problem approximately, but cannot guarantee to find the optimal k-means clustering.
- **Consequence**: We always need to run the k-means algorithm *multiple times*.

Can we make it more robust?

Well, yes and no!

Initialization methods

Lloyd Lloyd's *random partition* initialization (assign each point randomly) tends to generate centers (in the first iteration) that are *all* very close to the overall mean vector of the data.
Likely to get trapped in the same local optimum.

Forgy Sample k points from $\mathcal{X}$ uniformly at random as centers.
Simple, yet effective.

k-means++ Sample first center uniformly at random. Sample remaining points by biasing the probability distribution towards distant points.
Standard initialization. Works very well if the inut data adheres to k-means clustering model.

What is the best value of k?

This is a non-trivial problem!

Heuristic approach

Consider the *within-cluster variation (WCV)*

$$W_k = \sum_{l=1}^{k} W(C_l) = \sum_{l=1}^{k} \sum_{x_i \in C_l} (x_i - \mu_l)^2 = \frac{1}{2} \cdot \sum_{l=1}^{k} \frac{1}{|C_l|} \left(\sum_{x,y \in C_l} ||x - y||^2 \right)$$

where $W(C_l) = \sum_{x_i \in C_l} (x_i - \mu_l)^2$ is the *within-cluster variation of* C_l.

- For increasing k the value of W_k will usually decrease[9]

[9] Imagine why only "usually"?

The Elbow Method

1. Run k-means for varying $k = 1, 2, \ldots$
2. For each k, calculate the WCSS W_k
3. Plot k versus W_k in a line-plot
4. Search for a k^* where a heavy drop in W_k occurs (the "elbow", "bend" or "knee")

Idea: Indicator for huge drop in WCV from $k^* - 1$ to k^*.

Observation: $k = 2$ seems to be the best choice

The Gap Statistic Method I

Idea: "standardize the graph of $\log(W_k)$ by comparing it with its expectation under an appropriate null reference distribution of the data" Tibshirani, Guenther, and Hastie. 2001

1. Run k-means for varying $k = 1, 2, \ldots, K$ and calculate W_k
2. Generate B reference sets sampling uniformly at random within the bounds of the data[10]. This yield W^*_{kb} for $b = 1, \ldots, B$ and $k = 1, \ldots, K$
3. Calculate the *estimated gap statistic*

$$\text{Gap}(k) = \underbrace{\left(\frac{1}{B}\sum_{b=1}^{B}\log(W^*_{kb})\right)}_{=:W^*} - \log(W_k)$$

The Gap Statistic Method II

4. Compute standard deviations

$$\mathrm{sd}_k = \sqrt{\frac{1}{B}\sum_{b=1}^{B}\left(\log(W^*_{kb} - W^*\right)^2}$$

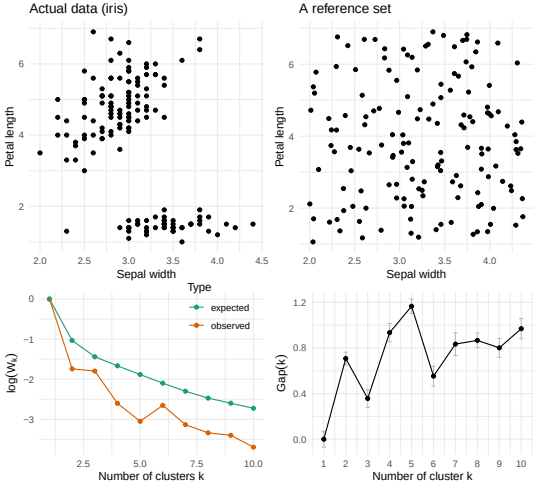
   and set $s_k = \mathrm{sd}_k \cdot \sqrt{1 + 1/B}$

5. Choose k^* such that

$$k^* = \arg\min_{k=1,\ldots,K} \mathrm{Gap}(k) \geq \mathrm{Gap}(k+1) - s_{k+1}.$$

[10] I.e., within $\min(x)$ and $\max(x)$.

The Gap Statistic Method



k-Means Clustering
Complexity

- General problem is $\mathcal{NP}$-hard
- Even $\mathcal{NP}$-hard for $k = 2$ in p-dim. Euclidean space
- Lloyd's k-means runs in time $\mathcal{O}(NpkL)$ where L is the number of iterations until convergence (or a fixed parameter)
- **Good news:** on data with clusters L is usually very small
  $\rightsquigarrow$ k-means is considered to have linear-time complexity in practise
- Since result depends on initialisation
  $\rightsquigarrow$ always perform $R > 1$ independent runs

k-Means Clustering
Properties

Advantages ☺

- Kind of captures the intuition of "good" clusters
- Quite fast (on average)

Disavantages ☹

- Number of clusters needs to be specified
- Works very well only on data with spherical clusters with similar size/extend
- Convergence to local-optima may yield "unintuitive" results
- Sensitive to (severe) outliers

What we learned today

- The problem of cluster identification
- Distance measures for set of points
- Bottom-up agglomerative clustering (HCA)
- k-means clustering

References I

Fisher, R. A. (1936). "The Use of Multiple Measurements in Taxonomic Problems". In: *Annals of Eugenics* 7.7, pp. 179–188.

Sibson, R. (Jan. 1973). "SLINK: An optimally efficient algorithm for the single-link cluster method". In: *The Computer Journal* 16.1, pp. 30–34. DOI: 10.1093/comjnl/16.1.30.

Defays, D. (Jan. 1977). "An efficient algorithm for a complete link method". In: *The Computer Journal* 20.4, pp. 364–366. DOI: 10.1093/comjnl/20.4.364.

Lloyd, S. (1982). "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2, pp. 129–137. DOI: 10.1109/TIT.1982.1056489.

Aloise, Daniel et al. (Jan. 2009). "NP-hardness of euclidean sum-of-squares clustering". In: *Machine Learning* 75.2, pp. 245–248. published.

Tibshirani, Robert, Walther Guenther, and Trevor Hastie. (2001). "Estimating the Number of Clusters in a Data Set via the Gap Statistic". In: *Journal of the Royal Statistical Society Series B*.