Multivariate Verfahren

7.1 Unsupervised Learning: Clustering

Hannah Schulz-Kümpel

based on lecture slides by Sabine Hoffmann

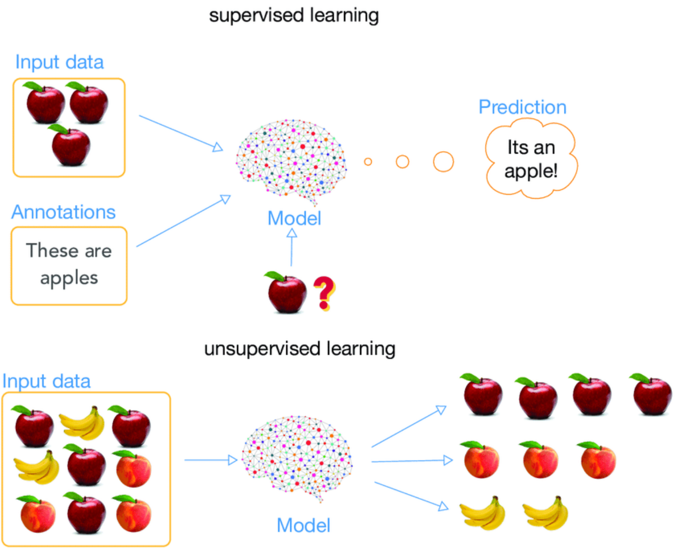
Institut für Statistik, LMU München

Sommersemester 2024

Contents

1. What is unsupervised Learning?
2. Clustering
    - Non-probabilistic methods
        - Hierarchical clustering
        - Partitional, distance-based clustering
    - Probabilistic methods
3. Cluster validation
    - Internal

What is unsupervised Learning? We recall:



What is unsupervised Learning? I

In contrast to supervised learning, *unsupervised learning methods*

- are applied to data that is **not labelled**:

  $$\mathcal{D} = \{x_1, \ldots, x_n\} \in \mathcal{X}^n$$

  (i.e. no y_is)

- and aim at "*making inferences about the structure of $\mathcal{D}$* ".

What is unsupervised Learning? II

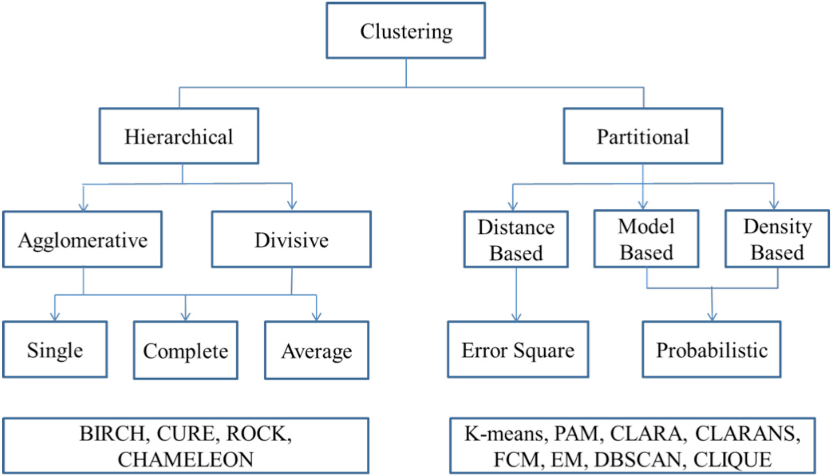
This goal is, admittedly, much vaguer than “finding the optimal parameter θ”, but unsupervised learning methods have a lot of relevant applications, like

- data visualization,
- exploratory data analysis,
- grouping objects $\longrightarrow$ *this lecture*,
- dimensionality reduction $\longrightarrow$ *following lecture(s)*

Clustering

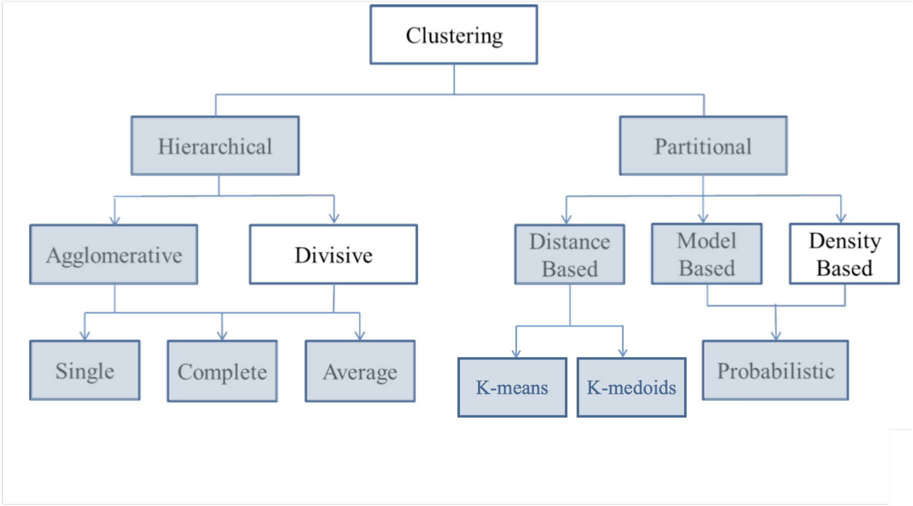
- *Clustering* refers to unsupervised learning methods aimed at grouping similar data points together.
- So we start with a sequence (or tupel) of data points $\mathcal{D} = \{x_1, \ldots, x_n\}$ with the goal of assigning each point to one of K separate clusters.
- Applications of clustering methods include: *Image Processing*, *Genomics*, *Anomaly Detection*, *Document Categorization*, etc.
- Can clustering algorithms also be used for supervised learning? *Yes!* See the very end of these slides.

One way to categorize Clustering methods



Source: Saxena, Amit Kumar et al. "*A review of clustering techniques and developments*." Neurocomputing 267 (2017): 664-681.

Clustering methods covered in this class

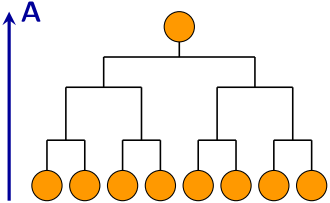


Hierarchical Clustering

Idea behind hierarchical clustering

Agglomerative methods

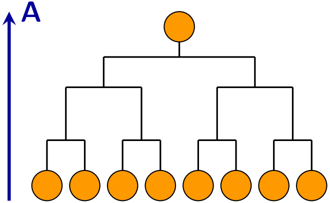
- Start with n objects as individual clusters.
- In each step, the two closest clusters are summarized.



Idea behind hierarchical clustering

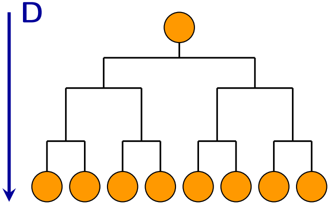
Agglomerative methods

- Start with n objects as individual clusters.
- In each step, the two closest clusters are summarized.



Divisive methods

- Start with all n objects in a cluster.
- A cluster is split in each step.

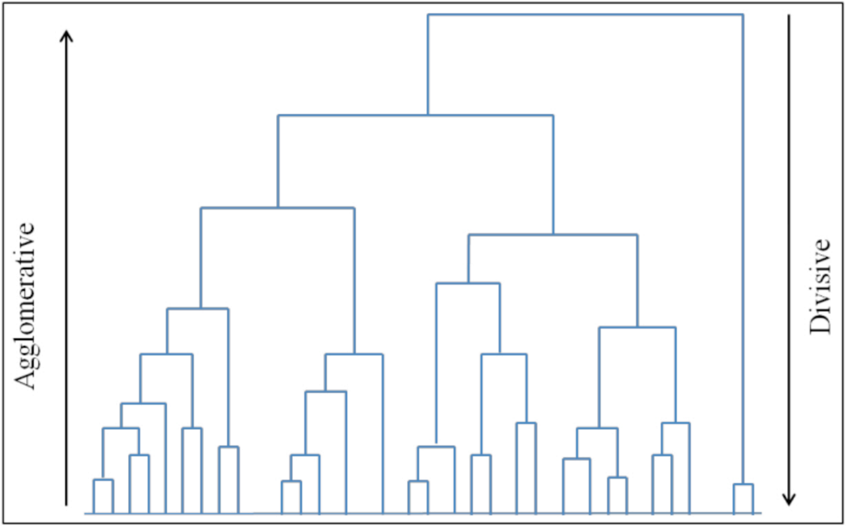


Hierarchical clustering

Form a hierarchy of partitions $\mathbb{C} = \{C_1, \dots, C_g\}$ according to one of the following two principles:

- Agglomerative methods
  Start with the partition $\mathbb{C}^{(0)} = \{\{x_1\}, \dots, \{x_n\}\}$, in which each observation forms its own cluster and successively *merge* the clusters.
- Divisive methods
  Start with the partition $\mathbb{C}^{(0)} = \{x_1, \dots, x_n\}$, where all observations form a single cluster and successively *divide* the clusters.

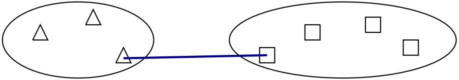
For both methods, the hierarchy of partitions $\mathbb{C}^{(0)}, \dots, \mathbb{C}^{(n)}$ may be visualized as a *dendogram*:



Source: Saxena, Amit Kumar et al. "*A review of clustering techniques and developments*." Neurocomputing 267 (2017): 664-681.

Linkage: Quantifying distances between classes

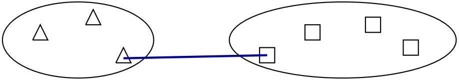
Linkage: Quantifying distances between classes



Single linkage:

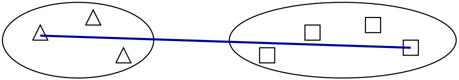
- Minimal distance between clusters
- Nearest Neighbor

Linkage: Quantifying distances between classes



Single linkage:

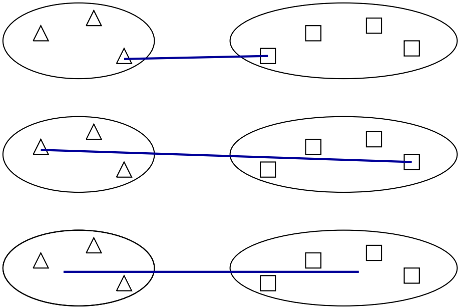
- Minimal distance between clusters
- Nearest Neighbor



Complete linkage:

- Maximal distance between objects

Linkage: Quantifying distances between classes



Single linkage:

- Minimal distance between clusters
- Nearest Neighbor

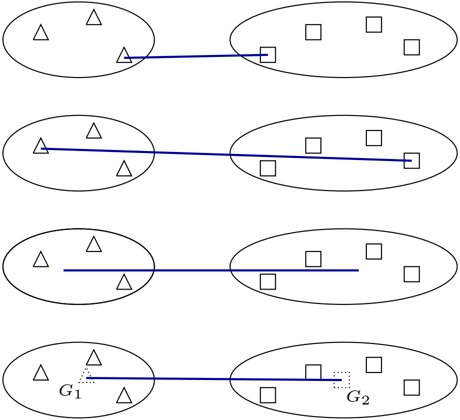
Complete linkage:

- Maximal distance between objects

Average linkage:

- Mean deviation of all pairwise distances

Linkage: Quantifying distances between classes



Single linkage:

- Minimal distance between clusters
- Nearest Neighbor

Complete linkage:

- Maximal distance between objects

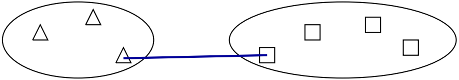
Average linkage:

- Mean deviation of all pairwise distances

Zentroid procedure:

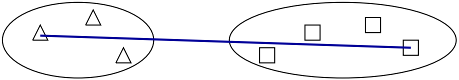
- Distance between cluster centroids

Linkage: Quantifying distances between classes



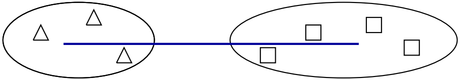
Single linkage:

- Minimal distance between clusters
- Nearest Neighbor



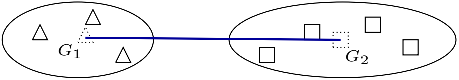
Complete linkage:

- Maximal distance between objects



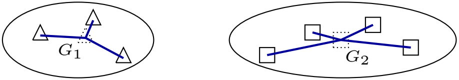
Average linkage:

- Mean deviation of all pairwise distances



Zentroid procedure:

- Distance between cluster centroids



Ward method:

- Consider the inertia within the clusters
- Analog to k-means (but not optimal)

Agglomerative hierarchical Clustering I

- In this class, we focus on **agglomerative hierarchical clustering methods**.
- Here, we require:
  - A distance measure to quantify the distance between objects
  - A distance measure to quantify the distance between classes
- The objects (classes) with the shortest distance are grouped together.
- This procedure is carried out until all objects are combined into one class.

Agglomerative hierarchical Clustering II

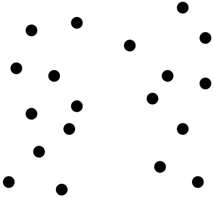
- Start partition: $\mathbb{C}^{(0)} = \{C_1^{(0)} = \{x_1\}, \ldots, C_n^{(0)} = \{x_n\}\}$
- In the νth step, merge those clusters

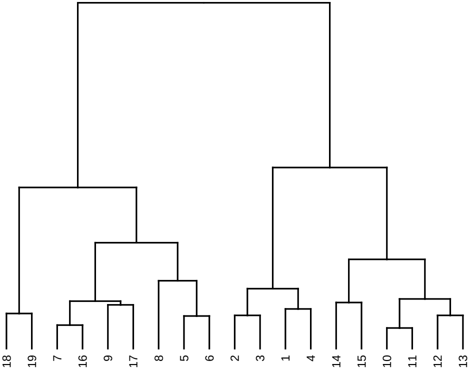
  $$C_r^{(\nu)},\ C_s^{(\nu)}, \quad r \neq s\,,$$

  which have the smallest distance D.
- The distance between the objects is determined by a distance measure, the distance between the clusters by the Linkage.

Agglomerative hierarcical clustering: Algorithm

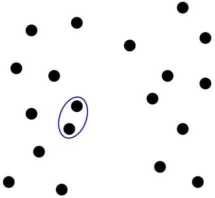
Step 1: Find the two elements that are closest to each other and combine them.

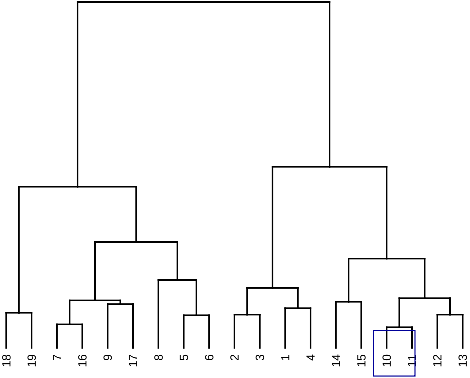




Agglomerative hierarcical clustering: Algorithm

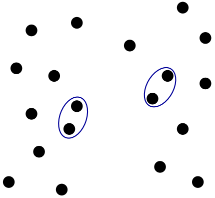
Step 1: Find the two elements that are closest to each other and combine them.

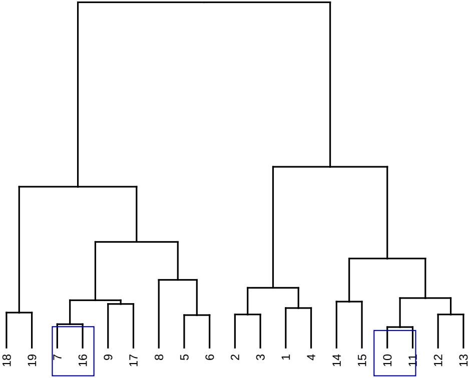




Agglomerative hierarcical clustering: Algorithm

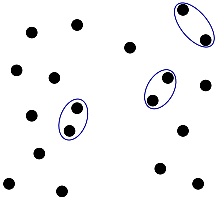
Step 1: Find the two elements that are closest to each other and combine them.

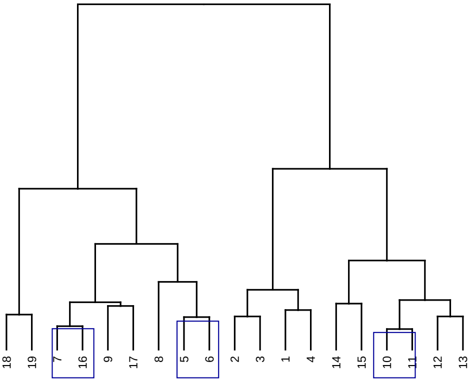




Agglomerative hierarcical clustering: Algorithm

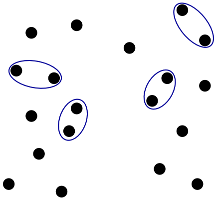
Step 1: Find the two elements that are closest to each other and combine them.

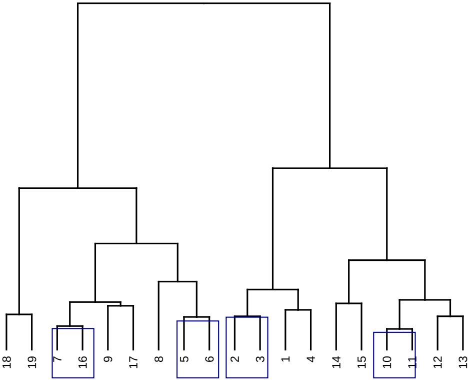




Agglomerative hierarcical clustering: Algorithm

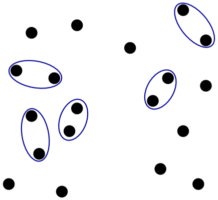
Step 1: Find the two elements that are closest to each other and combine them.

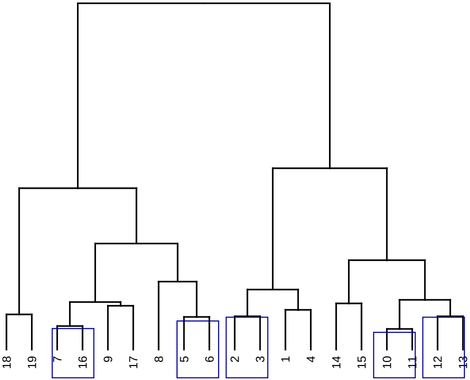




Agglomerative hierarcical clustering: Algorithm

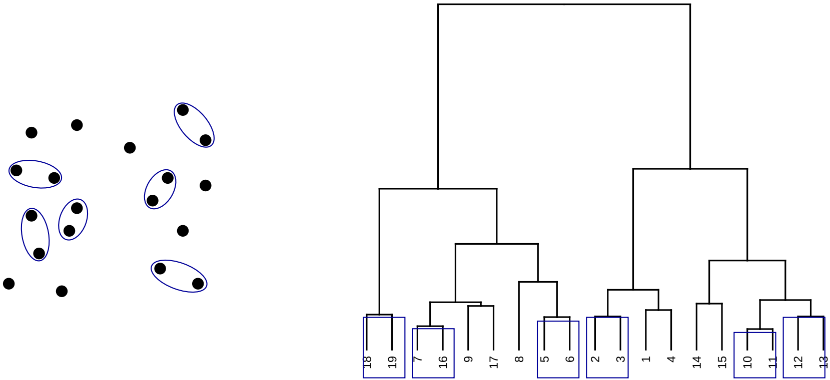
Step 1: Find the two elements that are closest to each other and combine them.





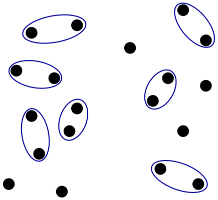
Agglomerative hierarcical clustering: Algorithm

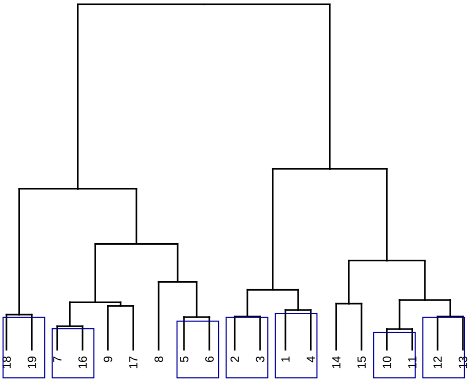
Step 1: Find the two elements that are closest to each other and combine them.



Agglomerative hierarcical clustering: Algorithm

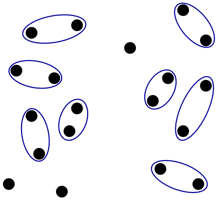
Step 1: Find the two elements that are closest to each other and combine them.

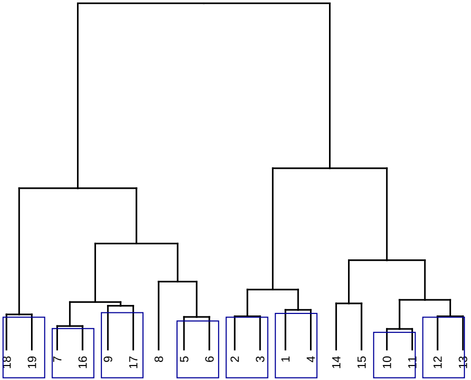




Agglomerative hierarcical clustering: Algorithm

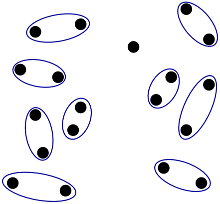
Step 1: Find the two elements that are closest to each other and combine them.

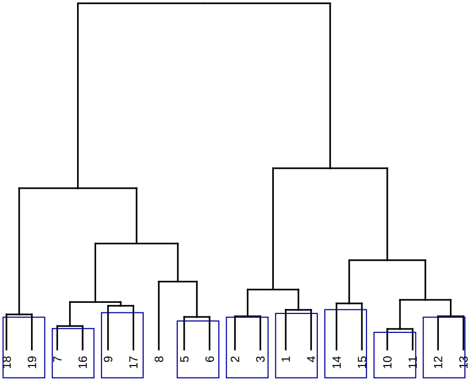




Agglomerative hierarcical clustering: Algorithm

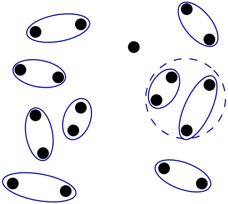
Step 1: Find the two elements that are closest to each other and combine them.

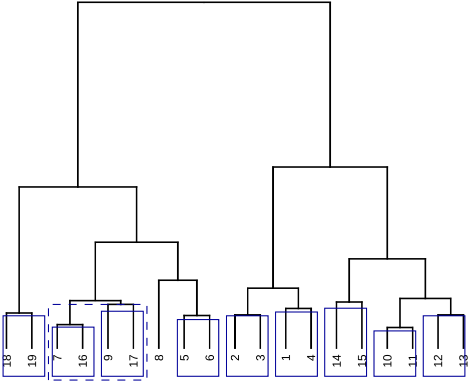




Agglomerative hierarcical clustering: Algorithm

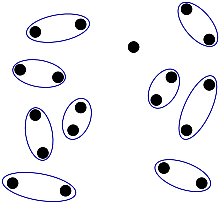
Step 2: Using some type of Linkage, determine the distance between the clusters.

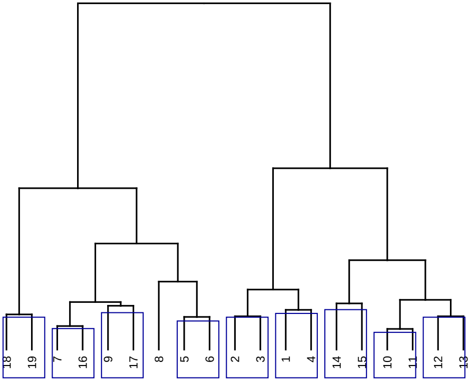




Agglomerative hierarcical clustering: Algorithm

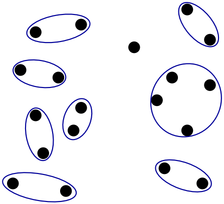
Step 2: Find the two clusters that are closest to each other and merge them.

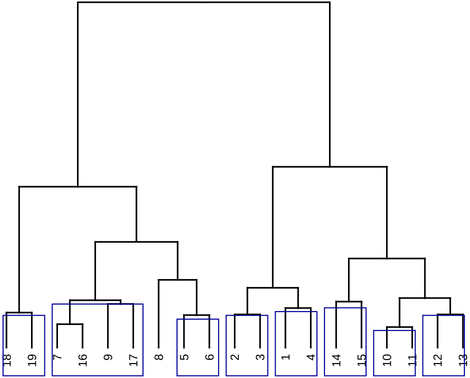




Agglomerative hierarcical clustering: Algorithm

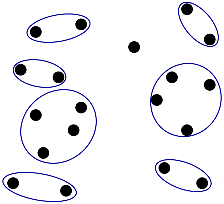
Step 2: Find the two clusters that are closest to each other and merge them.

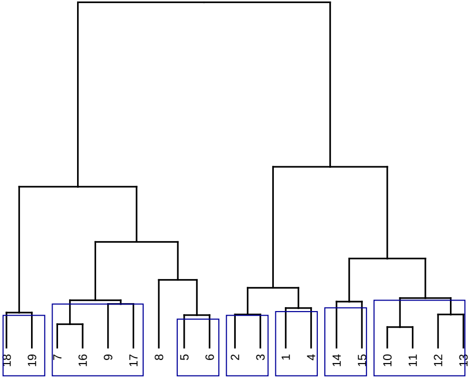




Agglomerative hierarcical clustering: Algorithm

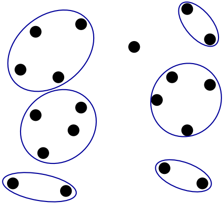
Step 2: Find the two clusters that are closest to each other and merge them.

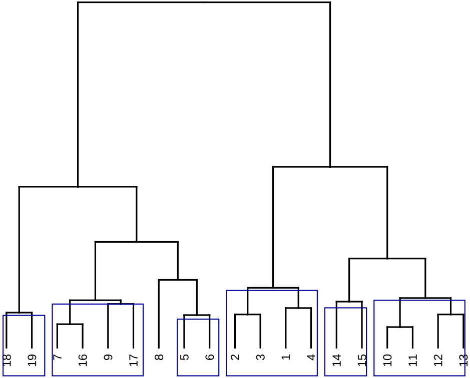




Agglomerative hierarcical clustering: Algorithm

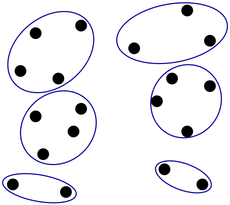
Step 2: Find the two clusters that are closest to each other and merge them.

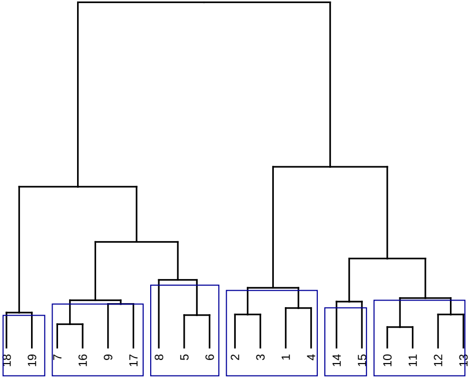




Agglomerative hierarcical clustering: Algorithm

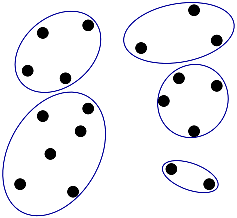
Step 2: Find the two clusters that are closest to each other and merge them.

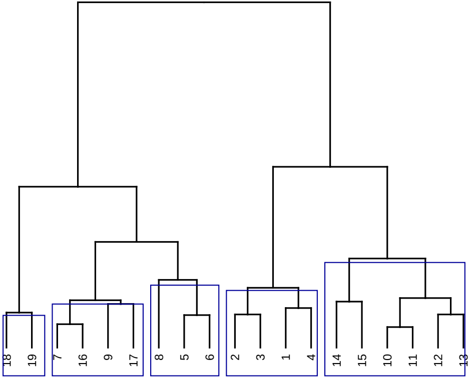




Agglomerative hierarcical clustering: Algorithm

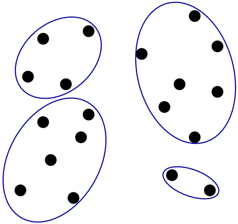
Step 2: Find the two clusters that are closest to each other and merge them.

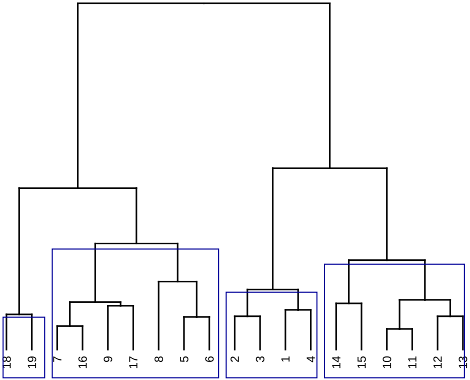




Agglomerative hierarcical clustering: Algorithm

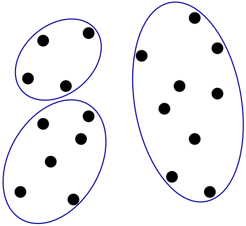
Step 2: Find the two clusters that are closest to each other and merge them.

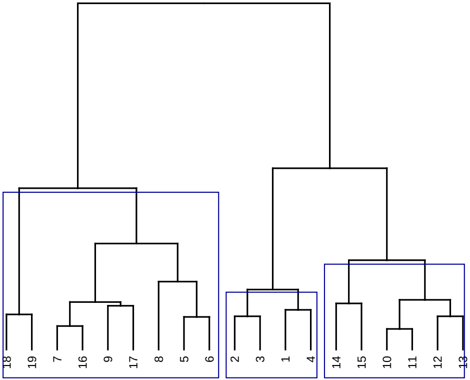




Agglomerative hierarcical clustering: Algorithm

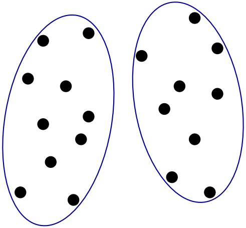
Step 2: Find the two clusters that are closest to each other and merge them.

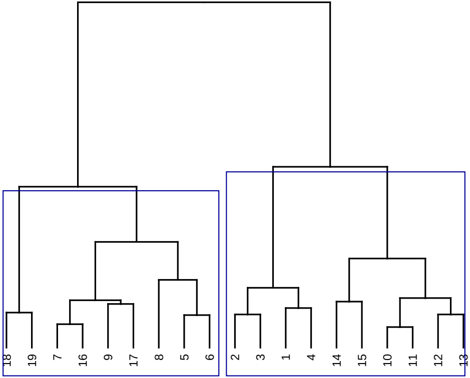




Agglomerative hierarcical clustering: Algorithm

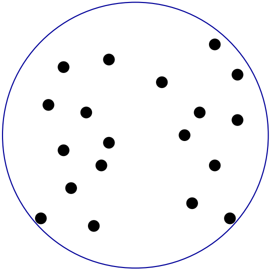
Step 2: Find the two clusters that are closest to each other and merge them.

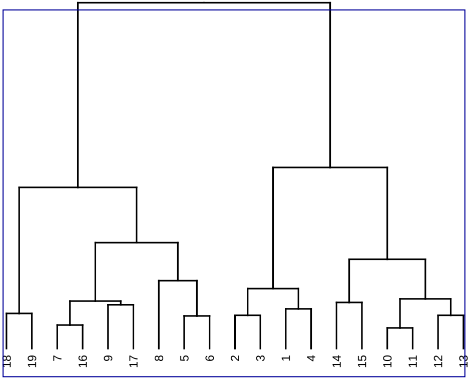




Agglomerative hierarcical clustering: Algorithm

Step 3: Stop when all objects are combined.





Example: agglomerative procedure

- Consider the age of 6 persons: 43, 38, 6, 47, 37, 9

- Determine the Euclidean distance between 2 people:

$$\mathbf{D} = \begin{pmatrix} 0 & 5 & 37 & 4 & 6 & 34 \\ 5 & 0 & 32 & 9 & 1 & 29 \\ 37 & 32 & 0 & 41 & 31 & 3 \\ 4 & 9 & 41 & 0 & 10 & 38 \\ 6 & 1 & 31 & 10 & 0 & 28 \\ 34 & 29 & 3 & 38 & 28 & 0 \end{pmatrix}$$

Merge classes with the smallest distance ⇒ Merge classes 2 and 5 (ages 37 and 38)

Example: agglomerative procedure

- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9

	$\{2,5\}$	$\{1\}$	$\{3\}$	$\{4\}$	$\{6\}$
$\{2,5\}$					
$\{1\}$					
$\{3\}$		37			
$\{4\}$		4	41		
$\{6\}$		34	3	38	

How is the distance between {2, 5} and the other classes determined?

- We use some type of *Linkage* to calculate the distance $D(C_r, C_s)$ with $C_r, C_s \subset \mathbb{C}^{(i)}$ for step i.

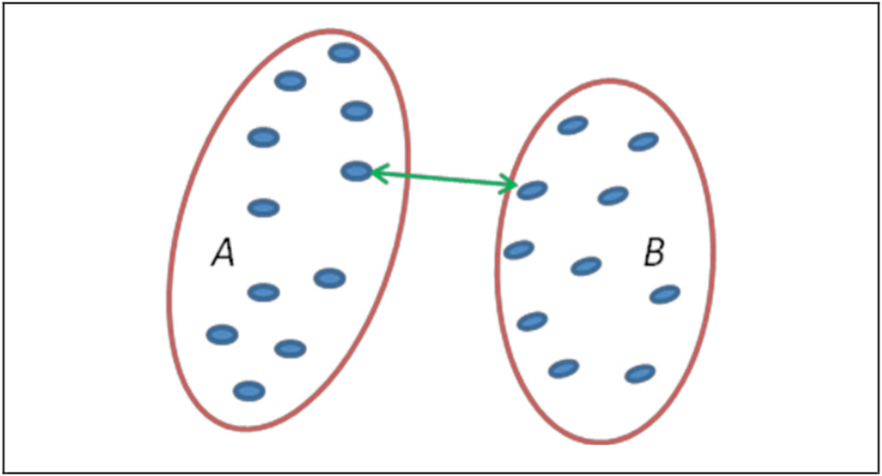
Single Linkage

For Single Linkage, we have

$$D_{SL}(C_r, C_s) = \min_{\substack{x_i \in C_r \\ x_j \in C_s}} \{d(x_i, x_j)\}$$

- "Nearest Neighbor"
- Robust against small changes to individual data points
- Risk of chain formation or bridge formation
- Application in taxonomy

$$\min\{d(a,b) : a \in A, b \in B\}$$



Example: Single Linkage

- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9
- $D_{SL}(\{2,5\},\{1\}) = \min\{d_{21}, d_{51}\} = \min\{5,6\} = 5$
- $D_{SL}(\{2,5\},\{3\}) = \min\{d_{23}, d_{53}\} = \min\{32,31\} = 31$
- $D_{SL}(\{2,5\},\{4\}) = \min\{d_{24}, d_{54}\} = \min\{9,10\} = 9$
- $D_{SL}(\{2,5\},\{6\}) = \min\{d_{26}, d_{56}\} = \min\{29,28\} = 28$

Example: Single Linkage

- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9

	$\{2,5\}$	$\{1\}$	$\{3\}$	$\{4\}$	$\{6\}$
$\{2,5\}$					
$\{1\}$	5				
$\{3\}$	31	37			
$\{4\}$	9	4	41		
$\{6\}$	28	34	3	38	

$\Rightarrow$ Merge classes 3 and 6 (age 6 and 9)

Example: Single Linkage

- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9

	$\{2,5\}$	$\{1\}$	$\{3,6\}$	$\{4\}$
$\{2,5\}$				
$\{1\}$	5			
$\{3,6\}$	28	34		
$\{4\}$	9	4	38	

$\Rightarrow$ Merge classes 1 and 4 (age 43 and 47)

Example: Single Linkage

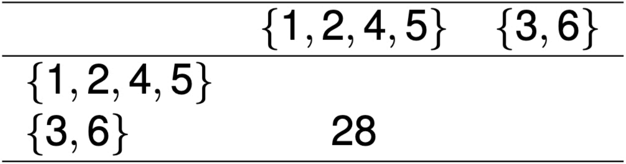
- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9

	$\{2,5\}$	$\{1,4\}$	$\{3,6\}$
$\{2,5\}$			
$\{1,4\}$	5		
$\{3,6\}$	28	34	

$\Rightarrow$ Merge {1,4} and {2,5}

Example: Single Linkage

- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9



	$\{1,2,4,5\}$	$\{3,6\}$
$\{1,2,4,5\}$		
$\{3,6\}$	28	

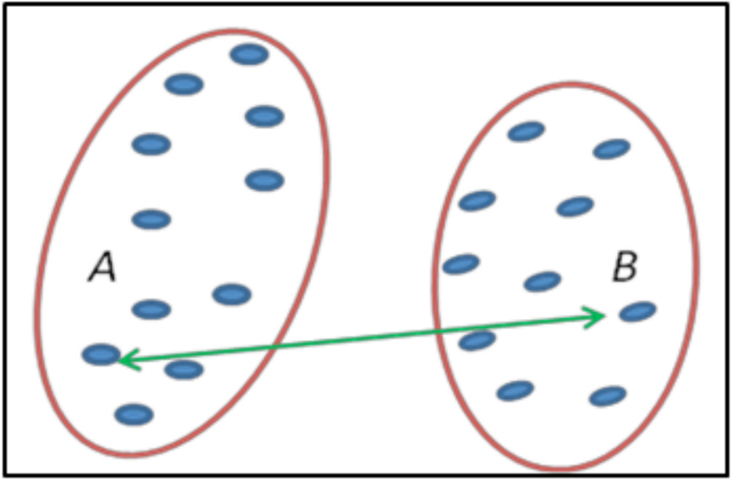
Complete Linkage

For Complete Linkage, we have

$$D_{CL}(C_r, C_s) = \max_{\substack{x_i \in C_r \\ x_j \in C_s}} d(x_i, x_j)$$

- "Furthest Neighbor"
- Large clusters grow slowly
- Instability with regard to small changes
- Suitable for splitting data without a clear structure

$$\max\{d(a,b) : a \in A, b \in B\}$$



Example: Complete Linkage

- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9
- $D_{SL}(\{2,5\},\{1\}) = \max\{d_{21}, d_{51}\} = \max\{5,6\} = 6$
- $D_{SL}(\{2,5\},\{3\}) = \max\{d_{23}, d_{53}\} = \max\{32,31\} = 32$
- $D_{SL}(\{2,5\},\{4\}) = \max\{d_{24}, d_{54}\} = \max\{9,10\} = 10$
- $D_{SL}(\{2,5\},\{6\}) = \max\{d_{26}, d_{56}\} = \max\{29,28\} = 29$

Example: Average Linkage

- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9

	$\{2,5\}$	$\{1\}$	$\{3\}$	$\{4\}$	$\{6\}$
$\{2,5\}$					
$\{1\}$	6				
$\{3\}$	32	37			
$\{4\}$	10	4	41		
$\{6\}$	29	34	3	38	

⇒ Merge classes 3 and 6 (age 6 and 9)

Example: Average Linkage

- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9

	$\{2,5\}$	$\{1\}$	$\{3,6\}$	$\{4\}$
$\{2,5\}$				
$\{1\}$	6			
$\{3,6\}$	32	37		
$\{4\}$	10	4	41	

$\Rightarrow$ Merge classes 1 and 4 (age 43 and 47)

Example: Complete Linkage

- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9

	$\{2,5\}$	$\{1,4\}$	$\{3,6\}$
$\{2,5\}$			
$\{1,4\}$	10		
$\{3,6\}$	32	41	

$\Rightarrow$ Merge {1,4} and {2,5}

Example: Complete Linkage

- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9

	$\{1,2,4,5\}$	$\{3,6\}$
$\{1,2,4,5\}$		
$\{3,6\}$	41	

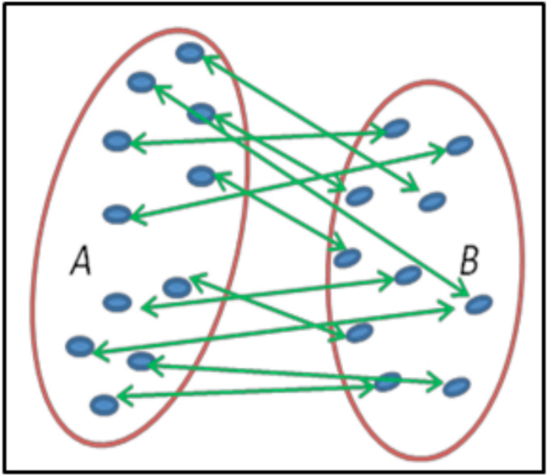
Average Linkage

For Average Linkage, we have

$$\begin{aligned} D_{AL}(C_r, C_s) &= \frac{1}{n_r n_s} \sum_{x_i \in C_r} \sum_{x_j \in C_s} d(x_i, x_j) \\ \text{with } n_i &= \mid C_i \mid \end{aligned}$$

- Compromise between complete linkage and single linkage
- Averaging should make sense

$$\frac{1}{|A||B|}\sum_{a\in A}\sum_{b\in B} d(a,b)$$

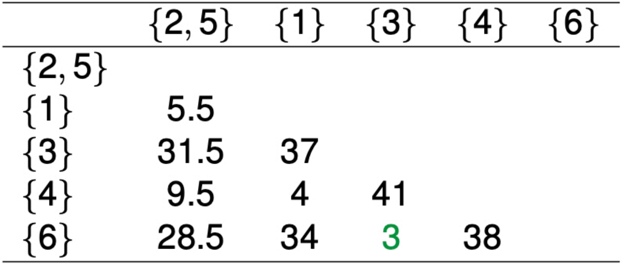


Example: Average Linkage

- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9
- $D_{AL}(\{2,5\},\{1\}) = \frac{d_{21}+d_{51}}{2} = \frac{5+6}{2} = 5.5$
- $D_{AL}(\{2,5\},\{3\}) = \frac{d_{23}+d_{53}}{2} = \frac{32+31}{2} = 31.5$
- $D_{AL}(\{2,5\},\{4\}) = \frac{d_{24}+d_{54}}{2} = \frac{9+10}{2} = 9.5$
- $D_{AL}(\{2,5\},\{6\}) = \frac{d_{26}+d_{56}}{2} = \frac{29+28}{2} = 28.5$

Example: Average Linkage

- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9



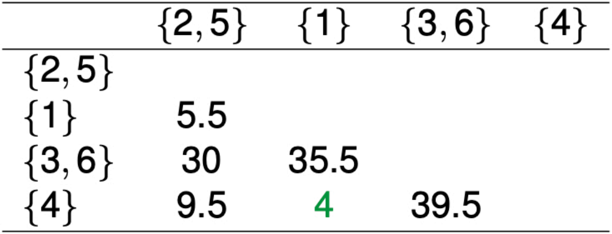
	$\{2,5\}$	$\{1\}$	$\{3\}$	$\{4\}$	$\{6\}$
$\{2,5\}$					
$\{1\}$	5.5				
$\{3\}$	31.5	37			
$\{4\}$	9.5	4	41		
$\{6\}$	28.5	34	3	38	

Example: Average Linkage

- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9
- $D_{AL}(\{3,6\},\{2,5\}) = \frac{d_{32}+d_{35}+d_{62}+d_{65}}{4} = \frac{32+31+29+28}{4} = 30$
- $D_{AL}(\{3,6\},\{1\}) = \frac{d_{31}+d_{61}}{2} = \frac{37+34}{2} = 35.5$
- $D_{AL}(\{3,6\},\{4\}) = \frac{d_{34}+d_{64}}{2} = \frac{41+38}{2} = 39.5$

Example: Average Linkage

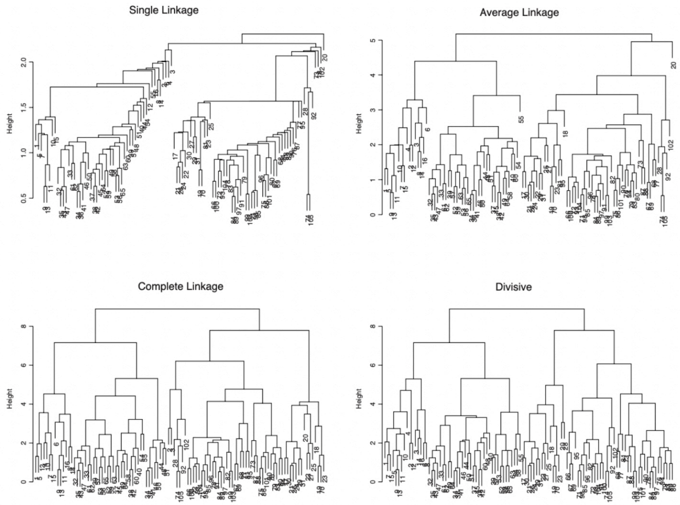
- We are still considering the age of 6 persons: 43, 38, 6, 47, 37, 9



	$\{2,5\}$	$\{1\}$	$\{3,6\}$	$\{4\}$
$\{2,5\}$				
$\{1\}$	5.5			
$\{3,6\}$	30	35.5		
$\{4\}$	9.5	4	39.5	

- And so on...

Comparison of Single, Average, and Complete Linkage



Zentroid-Procedure

For the Zentroid-Procedure, we have

$$\begin{aligned} D_Z(C_r, Cs) &= \| \bar{\mathbf{x}}_r - \bar{\mathbf{x}}_s \|^2 \\ \text{with } \bar{\mathbf{x}}_i &= \frac{1}{n_i} \sum_{j \in C_i} \mathbf{x_j} \end{aligned}$$

- Note: This procedure is only suited for metric data points.

Example: Zentroid-Procedure

- Next, let's consider the age of 4 persons: 19, 25, 20, 23

	{1}	{2}	{3}	{4}
{1}				
{2}	36			
{3}	1	25		
{4}	16	4	9	

Example: Zentroid-Procedure

- Again considering the age of 4 persons: 19, 25, 20, 23
- Mean of class $\{1,3\} = (19+20)/2 = 19.5$
- $D_Z(\{1,3\},\{2\}) = (19.5-25)^2 = 30.25$
- $D_Z(\{1,3\},\{4\}) = (19.5-23)^2 = 12.25$

Example: Zentroid-Procedure

- Again considering the age of 4 persons: 19, 25, 20, 23

	$\{1,3\}$	$\{2\}$	$\{4\}$
$\{1,3\}$			
$\{2\}$	30.25		
$\{4\}$	12.25	4	

Example: Zentroid-Procedure

- Again considering the age of 4 persons: 19, 25, 20, 23
- Mean of class $\{2,4\} = (25+23)/2 = 24$
- $D_Z(\{1,3\},\{2,4\}) = (19.5-24)^2 = 20.25$

Comparison of Zentroid and Average-Linkage when using the squared euclidean distance

$$\begin{aligned} D_{AL}(C_r, C_s) &= \frac{1}{n_r n_s} \sum_{x_i \in C_r} \sum_{x_\ell \in C_s} \{d(x_i, x_\ell)\} \\ &= \frac{1}{n_r n_s} \sum_{\mathbf{x}_i \in C_r} \sum_{\mathbf{x}_\ell \in C_s} \|\mathbf{x}_i - \mathbf{x}_\ell\|^2 \\ &= \|\bar{\mathbf{x}}_r - \bar{\mathbf{x}}_s\|^2 + \frac{1}{n_r} \sum_{\mathbf{x}_i \in C_r} \|\mathbf{x}_i - \bar{\mathbf{x}}_r\|^2 + \frac{1}{n_s} \sum_{\mathbf{x}_\ell \in C_s} \|\mathbf{x}_\ell - \bar{\mathbf{x}}_s\|^2 \\ &= D_Z(C_r, C_s) + s_r^2 + s_s^2 \end{aligned}$$

→ Average linkage takes into account the distance between the centers of gravity and the spread around it.

Ward method

Motivation: Merge the two clusters that generate the minimum increase in variance (heterogeneity) in the new cluster:

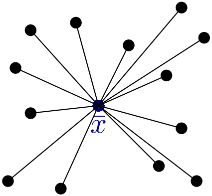
$$H(\mathbb{C}) = \sum_{r=1}^{k} \sum_{\mathbf{x}_i \in C_r} \left\| \mathbf{x}_i - \bar{\mathbf{x}}_r \right\|^2$$

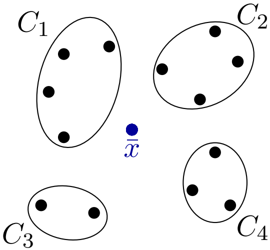
with

$$\bar{\mathbf{x}}_r = \frac{1}{n_r} \sum_{x_i \in C_r} \mathbf{x}_i$$

Ward method: properties of Inertia I

The overall inertia may be devided in:

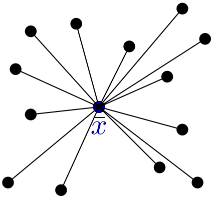


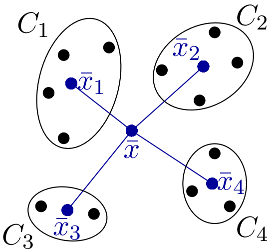


Ward method: properties of Inertia I

The overall inertia may be devided in:

- Inertia **between the K clusters** $C_k,\ k = 1, \dots, K$

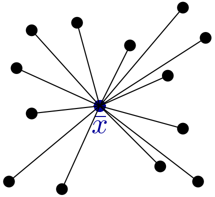


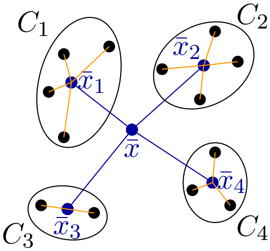


Ward method: properties of Inertia I

The overall inertia may be devided in:

- Inertia **between the K clusters** $C_k,\ k = 1, \ldots, K$
- Inertia **within the clusters** (sum of inertia in the K clusters)





Ward method: properties of Inertia II

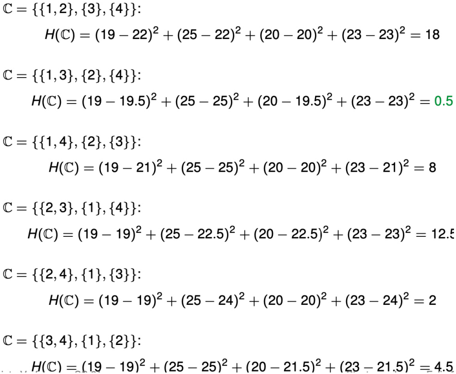
- total inertia = inertia between the clusters + inertia within the clusters

$$I_G = \frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}_i - \bar{\boldsymbol{x}}\|^2 = \sum_{k=1}^{K}\frac{n_k}{n}\|\bar{\boldsymbol{x}}_k - \bar{\boldsymbol{x}}\|^2 + \frac{1}{n}\sum_{k=1}^{K}\sum_{i\in C_k}\|\boldsymbol{x}_i - \bar{\boldsymbol{x}}_k\|^2$$

- The total inertia is constant, we aim **to minimize the within cluster inertia** (similar to maximizing the between cluster inertia)

Example: Ward method

- Again considering the age of 4 persons: 19, 25, 20, 23
- Consider all possibilities $\mathbb{C} = \{\{1\},\{2\},\{3\},\{4\}\}$ for merging



$\mathbb{C} = \{\{1,2\},\{3\},\{4\}\}$:

$$H(\mathbb{C}) = (19-22)^2 + (25-22)^2 + (20-20)^2 + (23-23)^2 = 18$$

$\mathbb{C} = \{\{1,3\},\{2\},\{4\}\}$:

$$H(\mathbb{C}) = (19-19.5)^2 + (25-25)^2 + (20-19.5)^2 + (23-23)^2 = 0.5$$

$\mathbb{C} = \{\{1,4\},\{2\},\{3\}\}$:

$$H(\mathbb{C}) = (19-21)^2 + (25-25)^2 + (20-20)^2 + (23-21)^2 = 8$$

$\mathbb{C} = \{\{2,3\},\{1\},\{4\}\}$:

$$H(\mathbb{C}) = (19-19)^2 + (25-22.5)^2 + (20-22.5)^2 + (23-23)^2 = 12.5$$

$\mathbb{C} = \{\{2,4\},\{1\},\{3\}\}$:

$$H(\mathbb{C}) = (19-19)^2 + (25-24)^2 + (20-20)^2 + (23-24)^2 = 2$$

$\mathbb{C} = \{\{3,4\},\{1\},\{2\}\}$:

$$H(\mathbb{C}) = (19-19)^2 + (25-25)^2 + (20-21.5)^2 + (23-21.5)^2 = 4.5$$

Example: Ward method

- Again considering the age of 4 persons: 19, 25, 20, 23
- Now, consider all possibilities for merging on $\mathbb{C} = \{\{1,3\},\{2\},\{4\}\}$

$\mathbb{C} = \{\{1,2,3\},\{4\}\}$:

$$H(\mathbb{C}) = (19-21.33)^2+(25-21.33)^2+(20-21.33)^2+(23-23)^2 = 20.67$$

$\mathbb{C} = \{\{1,3,4\},\{2\}\}$:

$$H(\mathbb{C}) = (19-20.67)^2+(25-25)^2+(20-20.67)^2+(23-20.67)^2 = 8.67$$

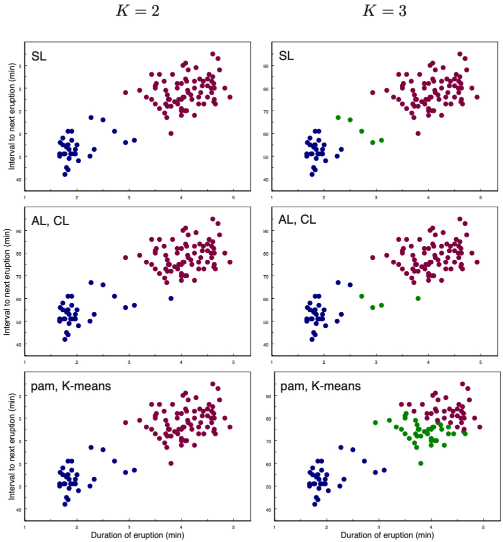
$\mathbb{C} = \{\{1,3\},\{2,4\}\}$:

$$H(\mathbb{C}) = (19-19.5)^2+(25-24)^2+(20-19.5)^2+(23-24)^2 = 2.5$$

Properties of agglomerative Clustering methods

- Single-Linkage tends to form chains (suitable for identifying outliers).
- Average-Linkage and Ward lead to very homogeneous clusters.
- Complete-Linkage is more sensitive to small changes in the data than Single-Linkage.
- Centroid and Ward are only applicable for metric features.
- Centroid and Ward can lead to Inversion (distance measure decreases compared to previous step)

Comparison of different Linkage-types



Distance-based partitioning

Distance-based partitioning

- Next, we will turn to Clustering methods based distance-based partitioning.

- Here, we

  1. Choose the desired number of clusters K.

  2. For a distance-based distortion function H, choose the "optimal" clustering partition $\mathbb{C}_{\text{opt}}$ so out of the set $\mathbb{C}$ of all possible partitions forming K clusters:

$$H(\mathbb{C}_{\text{opt}}) = \min_{\mathbb{C}^{(i)} \in \mathbb{C}} H(\mathbb{C}^{(i)}) \, .$$

- This may be repeated for different values of K or proceeded by a different algorithm to choose the best suited value of K - more later.

General numeric solution by substitution

1. Select an initial partition $\mathbb{C}^{(0)}$.
2. In the partition $\mathbb{C}^{(i)}$ $(i = 1, 2, \dots)$, check whether the assignment to another cluster improves the quality criterion H for each object .
3. Assign the object that results in the greatest improvement in H to the corresponding cluster. This results in the new partition $\mathbb{C}^{(i+1)}$.
4. Iterate steps 2 and 3 until there is no more improvement (i.e. the algorithm converges).

Common examples: k-Means and k-Medoids Algorithms I

- the k-means and k-medoids clustering methods are both instances of distance-based partitioning.
- They both define clusters around “centers”, with *k-means using centroids* (mean of points) and *k-medoid using medoids* (actual points).
- Specifically, these algorithms use the following distortion function, respectively:

(k-means), for $\bar{\boldsymbol{x}}_r := \frac{1}{|C_r|} \sum_{\mathbf{x}_i \in C_r} \mathbf{x}_i$

$$H(\mathbb{C}) = \sum_{r=1}^{k} \sum_{\mathbf{x}_i \in C_r} \|\mathbf{x}_i - \bar{\mathbf{x}}_r\|^2,$$

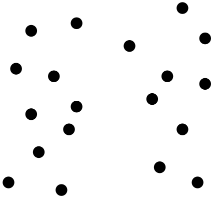
Common examples: k-Means and k-Medoids Algorithms II

k-medoids, for some distance function d and $\mathbf{m}_r = \underset{y \in C_r}{\arg\min} \sum_{\mathbf{x}_i \in C_r} d\left(y, x_i\right)$

$$H(\mathbb{C}) = \sum_{r=1}^{k} \sum_{\mathbf{x}_i \in C_r} d(\mathbf{x}_i, \mathbf{m}_r).$$

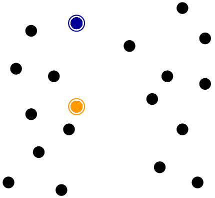
- For both these methods, we may use the “general numeric solution by substitution”, also called *Lloyd's algorithm*.
- However, this “naive” approach works much more reliably for k-means than k-medoids.

Lloyd's algorithm for k-means ("naive/standard k-means")



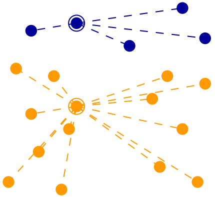
Example with $K = 2$ clusters

Lloyd's algorithm for k-means ("naive/standard k-means")



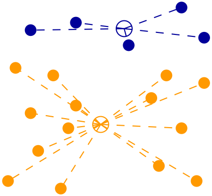
Iteration 1:
Randomly select two observations as initial mean values.

Lloyd's algorithm for k-means ("naive/standard k-means")



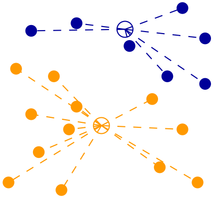
Iteration 1:
Assign each observation to its closest centroid, based on the Euclidean distance between the object and the centroid

Lloyd's algorithm for k-means ("naive/standard k-means")



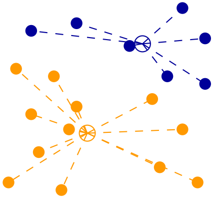
Iteration 2:
For each of the clusters, update the cluster centroid by calculating the new mean values of all the data points in the cluster

Lloyd's algorithm for k-means ("naive/standard k-means")



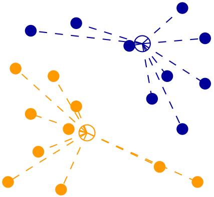
Iteration 2:
Allocate each observation to the cluster whose mean value has the smallest distance

Lloyd's algorithm for k-means ("naive/standard k-means")



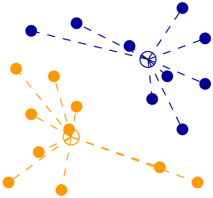
Iteration 3:
Update the cluster centroids

Lloyd's algorithm for k-means ("naive/standard k-means")



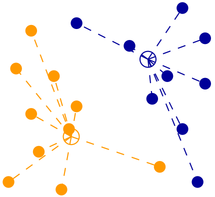
Iteration 3:
Again, allocate each observation to the cluster whose mean value has the smallest distance

Lloyd's algorithm for k-means ("naive/standard k-means")



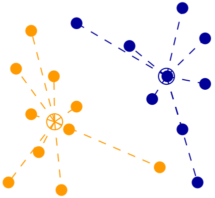
Iteration 4:
Update the cluster centroids

Lloyd's algorithm for k-means ("naive/standard k-means")



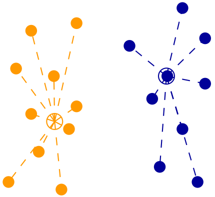
Iteration 4:
Again, allocate each observation to the cluster whose mean value has the smallest distance

Lloyd's algorithm for k-means ("naive/standard k-means")



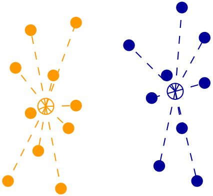
Iteration 5:
Update the cluster centroids

Lloyd's algorithm for k-means ("naive/standard k-means")



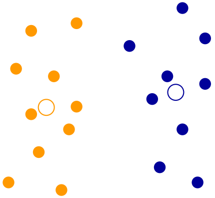
Iteration 5:
Again, allocate each observation to the cluster whose mean value has the smallest distance

Lloyd's algorithm for k-means ("naive/standard k-means")



Iteration 6:
Update the cluster centroids

Lloyd's algorithm for k-means ("naive/standard k-means")



Iteration 6:
No further changes in the allocation
→ algorithm is finished.

Standard k-means converges! I

It is relatively straightforward to show that the standard/naive k-means algorithm converges.

To do so, we first need the following Lemma by Shivaram Kalyanakrishnan.

Lemma

Consider the points $\boldsymbol{z}^1, \boldsymbol{z}^2, \ldots, \boldsymbol{z}^m$, *where* $m \geq 1$, *and for* $i \in \{1, 2, \ldots, m\}, \boldsymbol{z}^i \in \mathbb{R}^d$. *Let* $\overline{\boldsymbol{z}} = \frac{1}{m}\sum_{i=1}^m \boldsymbol{z}^i$ *be the mean of these points, and let* $\boldsymbol{z} \in \mathbb{R}^d$ *be an arbitrary point in the same (d-dimensional) space. Then*

$$\sum_{i=1}^m \left\|\boldsymbol{z}^i - \boldsymbol{z}\right\|^2 \geq \sum_{i=1}^m \left\|\boldsymbol{z}^i - \overline{\boldsymbol{z}}\right\|^2.$$

Proof

$$\sum_{i=1}^m \left\|\boldsymbol{z}^i - \boldsymbol{z}\right\|^2 = \sum_{i=1}^m \left\|\left(\boldsymbol{z}^i - \overline{\boldsymbol{z}}\right) + \left(\overline{\boldsymbol{z}} - \boldsymbol{z}\right)\right\|^2$$

Standard k-means converges! II

$$
\begin{aligned}
&= \sum_{i=1}^{m} \left(\left\| \boldsymbol{z}^i - \overline{\boldsymbol{z}} \right\|^2 + \| \overline{\boldsymbol{z}} - \boldsymbol{z} \|^2 + 2 \left(\boldsymbol{z}^i - \overline{\boldsymbol{z}} \right) \cdot \left(\overline{\boldsymbol{z}} - \boldsymbol{z} \right) \right) \\
&= \sum_{i=1}^{m} \left\| \boldsymbol{z}^i - \overline{\boldsymbol{z}} \right\|^2 + \sum_{i=1}^{m} \| \overline{\boldsymbol{z}} - \boldsymbol{z} \|^2 + 2 \sum_{i=1}^{m} \left(\boldsymbol{z}^i \cdot \overline{\boldsymbol{z}} - \boldsymbol{z}^i \cdot \boldsymbol{z} - \overline{\boldsymbol{z}} \cdot \overline{\boldsymbol{z}} + \overline{\boldsymbol{z}} \cdot \boldsymbol{z} \right) \\
&= \sum_{i=1}^{m} \left\| \boldsymbol{z}^i - \overline{\boldsymbol{z}} \right\|^2 + m \| \overline{\boldsymbol{z}} - \boldsymbol{z} \|^2 + 2 (m \overline{\boldsymbol{z}} \cdot \overline{\boldsymbol{z}} - m \overline{\boldsymbol{z}} \cdot \boldsymbol{z} - m \overline{\boldsymbol{z}} \cdot \overline{\boldsymbol{z}} + m \overline{\boldsymbol{z}} \cdot \boldsymbol{z}) \\
&= \sum_{i=1}^{m} \left\| \boldsymbol{z}^i - \overline{\boldsymbol{z}} \right\|^2 + m \| \overline{\boldsymbol{z}} - \boldsymbol{z} \|^2 \\
&\geq \sum_{i=1}^{m} \left\| \boldsymbol{z}^i - \overline{\boldsymbol{z}} \right\|^2
\end{aligned}
$$

☐

Standard k-means converges! III

- Let us denote by $t+1$ the iteration of the standard k-means algorithm that follows iteration t.

- If we can show that *given a specific initialization* $(\bar{\boldsymbol{x}}_1^{(0)}, \ldots, \bar{\boldsymbol{x}}_k^{(0)})$ the distortion function/cost $H(\mathbb{C})$ strictly decreases in every step until determination of the algorithm, i.e.

$$H\left(\mathbb{C}^{(t)}\right) = \sum_{r=1}^{k} \sum_{\mathbf{x}_i \in C_r^{(t)}} \|\mathbf{x}_i - \bar{\mathbf{x}}_r^{(t)}\|^2 > \sum_{r=1}^{k} \sum_{\mathbf{x}_i \in C_r^{(t+1)}} \|\mathbf{x}_i - \bar{\mathbf{x}}_r^{(t+1)}\|^2 = H\left(\mathbb{C}^{(t+1)}\right) \qquad (\star)$$

it follows that the algorithm will converge, since the number of possible partitions is finite and no clustering can be visited twice by $(\star)$.

Standard k-means converges! IV

Thereby, the following Theorem is equivalent to showing that the standard k-means algorithm converges:

Theorem

When applying the standard k-means algorithm to data $\mathcal{D}$*, the following holds given any specific initialization* $(\bar{\boldsymbol{x}}_1^{(0)}, \ldots, \bar{\boldsymbol{x}}_k^{(0)})$ *with* $\bar{\boldsymbol{x}}_1^{(0)} \in \mathcal{D}$ $\forall i \in \{1, \ldots, k\}$,

$$H\left(\mathbb{C}^{(t)}\right) > H\left(\mathbb{C}^{(t+1)}\right).$$

Proof

Hereafter, let $\boldsymbol{\mu}^t$ and $\mathcal{C}^t$ denote the set of cluster centroids and clusters in the tth iteration of the k-means algorithm, respectively; and define.

$$\mathrm{SSE}\left(\mathcal{C}^l, \boldsymbol{\mu}^j\right) := \sum_{r=1}^{k} \sum_{\mathbf{x}_i \in C_r^{(l)}} \|\mathbf{x}_i - \bar{\mathbf{x}}_r^{(j)}\|^2 .$$

Standard k-means converges! V

The proof is now completed in two steps:

Step 1: $\mathrm{SSE}\left(\mathcal{C}^{t+1}, \boldsymbol{\mu}^{t}\right) < \mathrm{SSE}\left(\mathcal{C}^{t}, \boldsymbol{\mu}^{t}\right)$

Step 2: $\mathrm{SSE}\left(\mathcal{C}^{t+1}, \boldsymbol{\mu}^{t+1}\right) \leq \mathrm{SSE}\left(\mathcal{C}^{t+1}, \boldsymbol{\mu}^{t}\right).$

The first step follows directly from the logic of the algorithm: $\mathcal{C}^{t}$ and $\mathcal{C}^{t+1}$ are different only if there is a point that finds a closer cluster centre in $\boldsymbol{\mu}^{t}$ than the one assigned to it by $\mathcal{C}^{t}$:

$$\mathrm{SSE}\left(\mathcal{C}^{t+1}, \boldsymbol{\mu}^{t}\right) = \sum_{i=1}^{n} \left\| \boldsymbol{x}^{i} - \boldsymbol{\mu}^{t}_{C^{t+1}(i)} \right\|^{2} < \sum_{i=1}^{n} \left\| \boldsymbol{x}^{i} - \boldsymbol{\mu}^{t}_{C^{t}(i)} \right\|^{2} = \mathrm{SSE}\left(\mathcal{C}^{t}, \boldsymbol{\mu}^{t}\right)$$

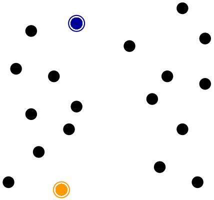
Standard k-means converges! VI

The second step follows from the Lemma by Shivaram Kalyanakrishnan:

$$\begin{aligned}
\mathrm{SSE}\left(\mathcal{C}^{t+1}, \boldsymbol{\mu}^{t+1}\right) &= \sum_{i=1}^{n} \left\| \boldsymbol{x}^{i} - \boldsymbol{\mu}^{t+1}_{C^{t+1}(i)} \right\|^{2} \\
&= \sum_{k'=1}^{k} \sum_{i \in \{1,2,\dots,n\}, \mathcal{C}^{t+1}(i)=k'} \left\| \boldsymbol{x}^{i} - \boldsymbol{\mu}^{t+1}_{C^{t+1}(i)} \right\|^{2} \\
&\leq \sum_{k'=1}^{k} \sum_{i \in \{1,2,\dots,n\}, \mathcal{C}^{t+1}(i)=k'} \left\| \boldsymbol{x}^{i} - \boldsymbol{\mu}^{t}_{C^{t+1}(i)} \right\|^{2} \\
&= \sum_{i=1}^{n} \left\| \boldsymbol{x}^{i} - \boldsymbol{\mu}^{t}_{C^{t+1}(i)} \right\|^{2} = \mathrm{SSE}\left(\mathcal{C}^{t+1}, \boldsymbol{\mu}^{t}\right)
\end{aligned}$$

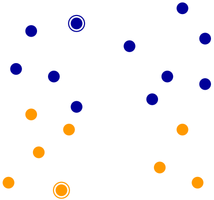
☐

Naive/standard k-means second example



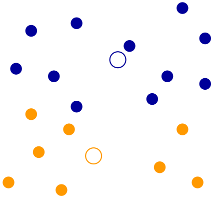
Iteration 1:
Randomly select two observations as initial mean values.

Naive/standard k-means second example



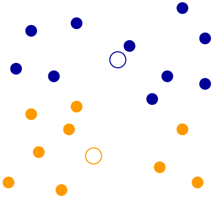
Iteration 1:
Allocate each observation to its closest centroid

Naive/standard k-means second example



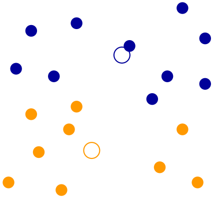
Iteration 2:
Update the cluster centroids

Naive/standard k-means second example



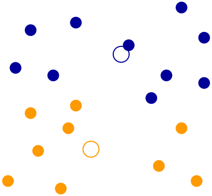
Iteration 2:
Allocate each observation to its closest centroid

Naive/standard k-means second example



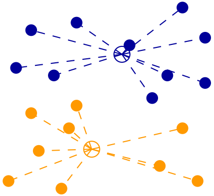
Iteration 3:
Update the cluster centroids

Naive/standard k-means second example



Iteration 3:
No further changes in the allocation
→ algorithm is finished

Naive/standard k-means second example



Within-cluster variance: 0.94

Naive/standard k-means second example

Within-cluster variance: 0.94

Within-cluster variance: 0.63

k-means++: Lessening the effect ob suboptimal random initialization

Algorithm 1: Initialization according to k-means++

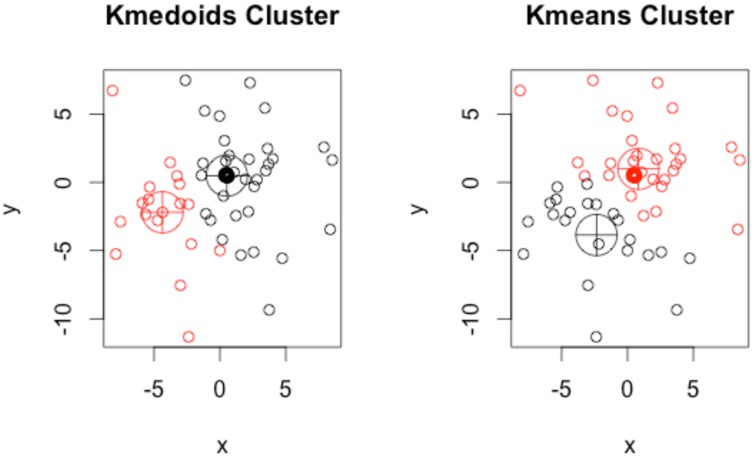
Data: Data $\mathcal{D}$ to be clustered into k cluster

Result: Not fully random initialization centroids for k-means

```
pick $x \in \mathcal{D}$ uniformly at random and set $T \leftarrow \{x\}$;
while $|T| < k$ do
    pick $x \in \mathcal{D}$ randomly with probability proportional to
     the cost, i.e. $p \propto \min_{z \in T} \|x - z\|^2$
    set $T \leftarrow T \cup \{x\}$
end
```

Arthur and Vassilvitskii (2007) actually suggest that this procedure in and of itself is a pretty solid clustering technique, but k-means is usually applied on top.

k-means and k-medoids will achieve different results!



K-Means vs. K-Medoids

K-Means

- Works with means of data points, i.e. computes centroids to use as cluster representatives.
- Sensitive to outliers.
- Lloyd's algorithm is reasonable (but can be approved upon).
- Uses Euclidean distance.

K-Medoids

- Works with medians of data points, i.e. chooses actual data points as cluster representatives.
- Robust to outliers.
- **Lloyd's algorithm gets stuck in local optima too easily.**
- Can use different distance metrics.

Suitable algorithms for k-medoids

- Since in k-medoids clustering only actual data points are chosen as cluster representatives, applying Lloyd's algorithm is much more prone to getting stuck in local optima than k-means.
- For this reason as well as computability etc., one usually employs other algorithms for k-medoids.
- Just as with k-means++, these algorithms should include some initialization step.
- A very common choice for k-medoids is *Partitioning Around Medoids (PAM)*, which will be detailed on the following slides.

Partitioning Around Medoids (PAM) I

We recall that with **k-medoids**

for some distance function d and $\mathbf{m}_r = \underset{y \in C_r}{\arg\min} \sum_{\mathbf{x}_i \in C_r} d\left(y, x_i\right)$, *we want to find the partition that minimizes* $H(\mathbb{C}) = \sum_{r=1}^{k} \sum_{\mathbf{x}_i \in C_r} d(\mathbf{x}_i, \mathbf{m}_r)$.

- The PAM algorithm consists of two steps:
  1. **PAM BULD** Initializes the clusters and
  2. **PAM SWAP** Improves the clustering.

Partitioning Around Medoids (PAM) II

- For both, we need to calculate the *change in* H from one cluster center to another - there are different methods to do this efficiently, but we will not discuss them in this lecture. Instead, we simply write ΔH for the *change of* $H(\cdot)$ in our algorithm.

Algorithm 2: PAM BUILD

Data: Data $\mathcal{D}$ to be clustered into k cluster

Result: Initial clusters for the PAM k-medoids algorithm

```
$\mathbf{m}_1 \leftarrow$ point $x \in \mathcal{D}$ that would minimize $H$ for $k = 1$;
for $i = 2, \ldots, k$ do
    $\mathbf{m}_i \leftarrow$ point $x \in \mathcal{D}$ that maximizes $-\Delta H$
end
```

Partitioning Around Medoids (PAM) III

Algorithm 3: PAM SWAP

Data: Initial clusters for the PAM k-medoids algorithm

Result: Clusters according to PAM k-medoids

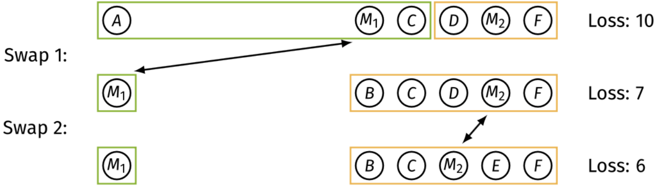
```
repeat
    for $\mathbf{m}_i \in \{\mathbf{m}_1, \dots, \mathbf{m}_k\}$ do
        for $x_j \in \mathcal{D} \setminus \{\mathbf{m}_1, \dots, \mathbf{m}_k\}$ do
            $\Delta H \leftarrow$ Change in $H$ with $x_j$ medoid instead of $\mathbf{m}_i$;
            Remember $(x_j, \mathbf{m}_i, \Delta H)$ for the best $\Delta H$
    Break if The best $\Delta H \geq 0$ (No improvement of the clustering);
    Swap $(\mathbf{m}_i, x_j)$ of the best $\Delta H$
until Clustering partition converges;
```

Visualization of why PAM helpts

The alternating optimization algorithm can easily get stuck:

Every object is assigned to the nearest medoid, the medoid is optimal for each partition.
⇝ the alternating optimization has converged

The swap-based approaches can further optimize and find a better solution:



Source: `https://dm.cs.tu-dortmund.de/en/mlbits/cluster-kmedoids-intro/#ref-DBLP:conf/sisap/SchubertR19`

Model-based Clustering

Model-based Clustering I

- **Definition:** Model-based clustering assumes that data is generated from a mixture of underlying probability distributions, where each distribution represents a cluster.

- **Key Concept:** Each cluster can be thought of as a distribution (often Gaussian) and the goal is to identify the parameters of these distributions.

Model-based Clustering II

Some advantages of Model-based clustering:

- *Probabilistic Framework*: Provides a probabilistic model of the data, which can handle noise and outliers better.
- *Flexibility through soft assignment:* Can model clusters of varying shapes and sizes by assigning each observations a probability of belonging to each cluster.
- *Automatic Determination of Number of Clusters:* Uses criteria such as the Bayesian Information Criterion (BIC) for model selection.

General approach of mixture-distributions I

We assume that the underlying population of our data is partitioned into k groups $C_1, \ldots, C_k$, with each group/component following a different distribution.
Underlying this are, for a random variable Z with realizations in $\{1, \ldots, k\}$, where

- $p(r) = P(Z = r)$: unknown probability of selecting component r, with $\sum_{r=1}^{k} p(r) = 1$.
- $f(x|\boldsymbol{\theta}_r)$: density of the random vector X in the r-th population with parameters $\boldsymbol{\theta}_r$.

One then assumes that observations $X_1, \ldots, X_n$ are drawn i.i.d. from the distribution defined via the following *mixed density*:

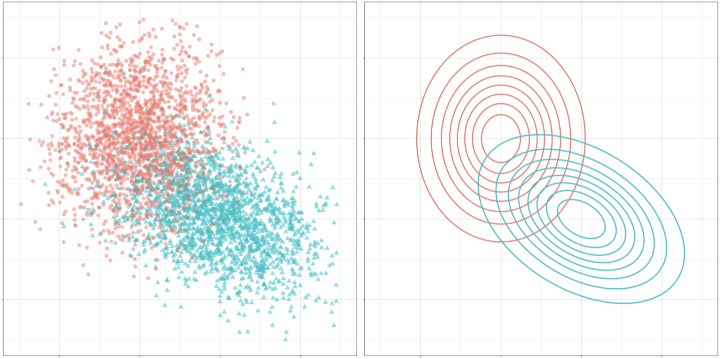
$$f(x_i) = \sum^{k} p(r) f(x_i|\boldsymbol{\theta}_r) .$$

General approach of mixture-distributions I

What does this remind you of?

General approach of mixture-distributions I

What does this remind you of? **Hopefully generative models!**



Left: *A sample from the feature distributions for the two-class case.* ***Right:*** *Their densities.*

In fact, model-based clustering methods such as GMMs may be used to generate new data!

In this lecture, we will specifically be looking at

- *Gaussian Mixture Models (GMMs)*
- How to "solve" them using the *Expectation-Maximization (EM)* algorithm
- A nice reference is the Stanford STATS 306B: Unsupervised Learning lecture

Gaussian Mixture Models (GMMs) I

- GMMs follow the general approach of mixture-distributions from before, with densities $f(x|\boldsymbol{\theta}_r)$ chosen as multivariate Gaussian density with unknown parameters (μ_j, Σ_j):

$$f(x|\boldsymbol{\theta}_r) = \phi\left(x; \mu_j, \Sigma_j\right) = \frac{1}{\left|(2\pi)^k \Sigma_j\right|^{1/2}} \exp \Bigg(-\frac{1}{2} \left(x - \mu_j\right)^T \Sigma_j^{-1} \left(x - \mu_j\right.$$

- Note that the actual component any X_i belongs to may be seen as a latent variable z_i with $X_i|z_i$ being *conditionally independently distributed* with density $\phi\left(x; \mu_{z_i}, \Sigma_{z_i}\right)$.

- Under the GMM, our clustering task amounts to inferring the latent component z_i responsible for each x_i.

Gaussian Mixture Models (GMMs) II

How we would carry out the clustering task if the parameter values were already known?

Since the GMM then defines a joint distribution over (x_i, z_i), it is natural to consider the conditional distribution of each z_i given x_i :

$$\begin{aligned} f\left(z_i = j \mid x_i\right) &= \frac{p\left(z_i = j\right) f\left(x_i \mid z_i = j\right)}{f\left(x_i\right)} \\ &= \frac{\pi_j \phi\left(x_i; \mu_j, \Sigma_j\right)}{\sum_{l=1}^{k} \pi_l \phi\left(x_i; \mu_l, \Sigma_l\right)}, \quad \text{with } \pi_k := p(z_i = k) = p(Y = k). \end{aligned}$$

$\Longrightarrow$ These conditionals reflect our updated beliefs concerning z_i after x_i is observed: before we observe x_i, we have the prior belief that it belongs to cluster j with probability π_j; after observing x_i, we can update this belief in accordance with the likelihood of x_i under each Gaussian component.

Gaussian Mixture Models (GMMs) III

$\Longrightarrow$ **the inference issue in GMMs is to find the parameters** $(\pi_{1:k}, \mu_{1:k}, \Sigma_{1:k})$

- Usually, this would mean optimizing the log-likelihood

$$\sum_{i=1}^{n} \log\left(p\left(x_i\right)\right) = \sum_{i=1}^{n} \log\left(\sum_{j=1}^{k} \pi_j \phi\left(x_i; \mu_j, \Sigma_j\right)\right) .$$

- However, for $k > 1$, which is what we are interested in here, this is not quite straightforward (Keywords *closed form solution*, *identifiability*).

$\Longrightarrow$ This is where the **Expectation-Maximization (EM)** algorithm comes in!

EM algorithm for GMMs I

- **Step 1:** Initialize parameter values $(\pi_{1:k}, \mu_{1:k}, \Sigma_{1:k})$ arbitrarily (or using some initialization algorithm)

Then, iterate the following two steps until convergence:

- **Step 2:** (Expectation) Compute soft class memberships, given the current parameters:

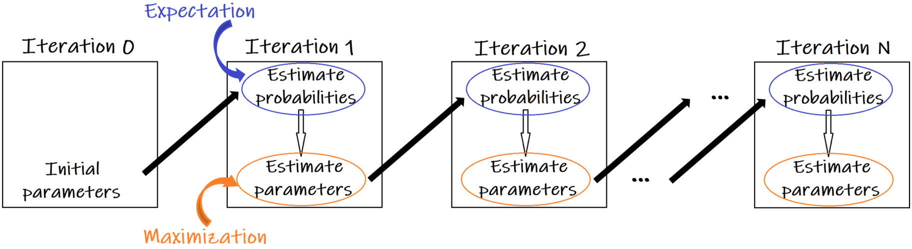
$$\tau_{ij} = P\left(z_i = j \mid x_{ij}, \pi, (\mu_\ell, \Sigma_\ell)\right)$$

- **Step 3:**(Maximization) Update parameters by plugging in τ_{ij} (our guess) for the unknown $\mathbb{I}_{\{z_i=j\}}$, which gives us:

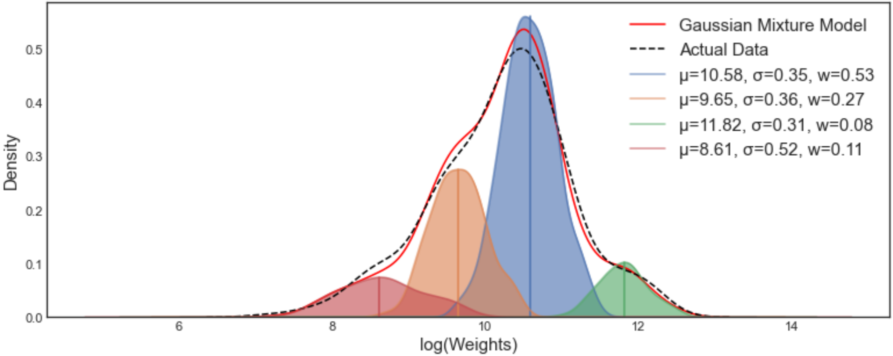
$$\pi_j = \frac{1}{n}\sum_{i=1}^{n} \tau_{ij}, \quad \mu_j = \frac{\sum_{i=1}^{n} \tau_{ij} x_i}{\sum_{i=1}^{n} \tau_{ij}},$$

$$\Sigma_j = \frac{\sum_{i=1}^{n} \tau_{ij} \left(x_i - \mu_j\right)\left(x_i - \mu_j\right)^T}{\sum_{i=1}^{n} \tau_{ij}}.$$

EM algorithm for GMMs II



Visualization of Gaussian Mixture Models



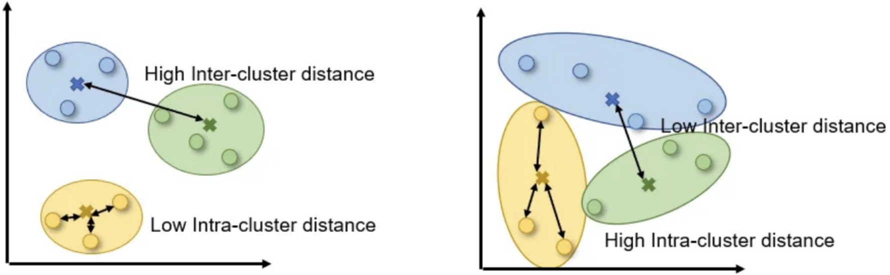
Source: `https://stats.stackexchange.com/questions/517652/how-to-evaluate-the-loss-on-a-gaussian-mixture-model`

Validating Clustering results

Validating Clustering results I

How can one measure the quality of clustering results?

A common approach is to aim for a **high** intra-cluster (within-cluster) similarity and a **low** inter-cluster (between-cluster) similarity.



Good (left) versus bad (right) clustering based on the Inter-cluster and Intra-cluster distance

Source: Medium

Validating Clustering results II

Generally, there are 3 different approaches for validating clustering results

- **internal cluster validation** uses internal information of the clustering process, e.g., the within-cluster sum of squares.
- **relative cluster validation** varies parameters of the clustering method, e.g., number of clusters
- **external cluster validation** compares results to externally known results, e.g., provided labels.

Example of **relative** cluster validation: Model selection for GMMs

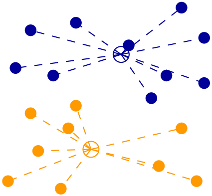
- The idea behind the "automatic" determination of number of clusters for GMMs is to run model selection over models resulting from a variety of choices for number of clusters K.

- For this, we need some measure for the *goodness of model fit*, calculated, e.g. on the fitted likelihood. (One suitable and often used option would be the Bayesian information criterion (BIC))

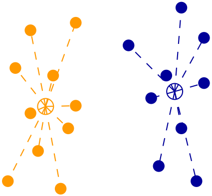
- **Note that measures for internal cluster validation usually are also suitable for relative validation!**

$\Longrightarrow$ More generally, we can always run a clustering algorithm with different parameters and choose the "best" version according to some validation technique.

Previous example: 2-means with different random initializations



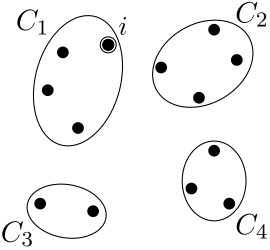
Within-cluster variance: 0.94



Within-cluster variance: 0.63

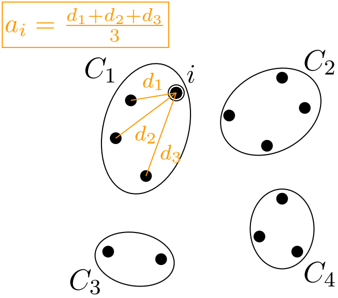
$\Longrightarrow$ Even with the same choice of cluster numbers K, one random initialization seems to produce much better results than the other.

Example of **internal** validation: Silhouette method I



Example of **internal** validation: Silhouette method I

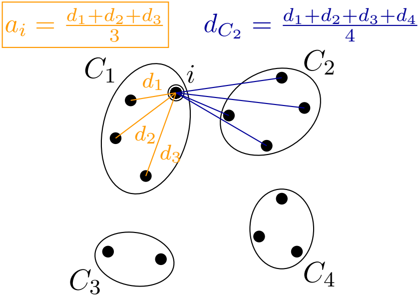
- Let a_i be the average distance between observation i and all other observations in the same cluster



Example of **internal** validation: Silhouette method I

- Let a_i be the average distance between observation i and all other observations in the same cluster

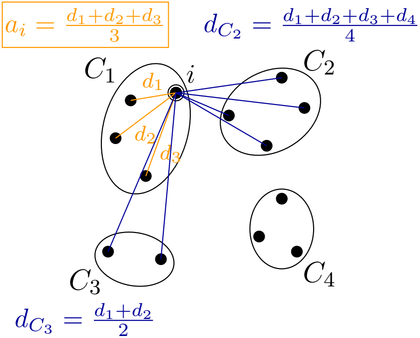
- Let b_i be the average distance between observation i and the observations in the nearest cluster



Example of **internal** validation: Silhouette method I

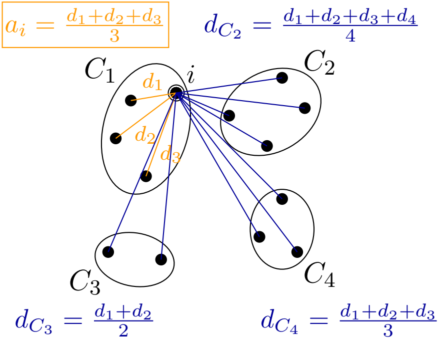
- Let a_i be the average distance between observation i and all other observations in the same cluster

- Let b_i be the average distance between observation i and the observations in the nearest cluster



Example of **internal** validation: Silhouette method I

- Let a_i be the average distance between observation i and all other observations in the same cluster
- Let b_i be the average distance between observation i and the observations in the nearest cluster

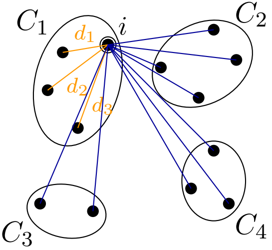


Example of **internal** validation: Silhouette method I

- Let a_i be the average distance between observation i and all other observations in the same cluster

- Let b_i be the average distance between observation i and the observations in the nearest cluster

$$\boxed{a_i = \frac{d_1+d_2+d_3}{3}} \qquad d_{C_2} = \frac{d_1+d_2+d_3+d_4}{4}$$



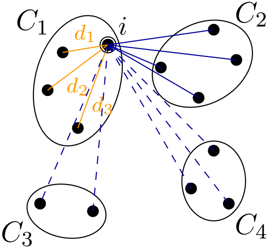
$$d_{C_3} = \frac{d_1+d_2}{2} \qquad d_{C_4} = \frac{d_1+d_2+d_3}{3}$$

$$\boxed{b_i = \min(d_{C_2}, d_{C_3}, d_{C_4})}$$

Example of **internal** validation: Silhouette method I

- Let a_i be the average distance between observation i and all other observations in the same cluster
- Let b_i be the average distance between observation i and the observations in the nearest cluster

$$\boxed{a_i = \frac{d_1+d_2+d_3}{3}} \qquad d_{C_2} = \frac{d_1+d_2+d_3+d_4}{4}$$

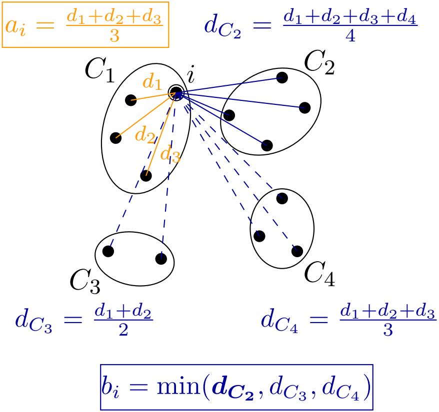


$$d_{C_3} = \frac{d_1+d_2}{2} \qquad d_{C_4} = \frac{d_1+d_2+d_3}{3}$$

$$\boxed{b_i = \min(\boldsymbol{d_{C_2}}, d_{C_3}, d_{C_4})}$$

Example of **internal** validation: Silhouette method I

- Let a_i be the average distance between observation i and all other observations in the same cluster
- Let b_i be the average distance between observation i and the observations in the nearest cluster
- $S_i = \frac{b_i - a_i}{\max(a_i, b_i)}$



$\boxed{a_i = \frac{d_1+d_2+d_3}{3}}$

$d_{C_2} = \frac{d_1+d_2+d_3+d_4}{4}$

$d_{C_3} = \frac{d_1+d_2}{2}$

$d_{C_4} = \frac{d_1+d_2+d_3}{3}$

$\boxed{b_i = \min(\boldsymbol{d_{C_2}}, d_{C_3}, d_{C_4})}$

Example of **internal** validation: Silhouette method II

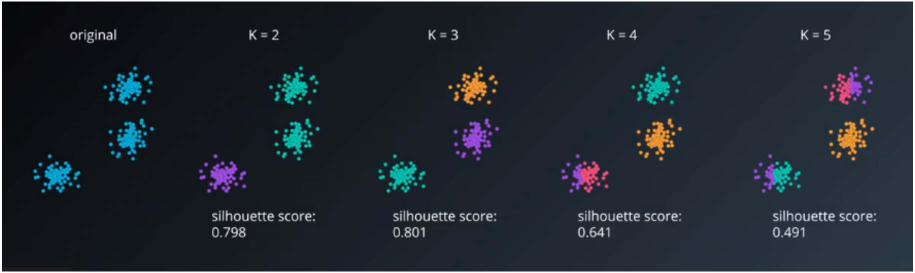
- Observations with $S_i < 0$ are probably in the wrong cluster.
- The Average Silhouette Width of a partition $(\mathbb{C})$ is defined as

$$\text{ASW}((\mathbb{C})) = \frac{1}{n}\sum_{i=1}^{n} S_i$$

- A high value of $\text{ASW}((\mathbb{C}))$ indicates a good cluster solution

⇒ The optimal number of clusters could be the one that maximizes the average silhouette width
Again, the silhouette method may also be used for relative validation.

Exemplary plot



Source: `https://ryanwingate.com/intro-to-machine-learning/unsupervised/gaussian-mixture-models-and-cluster-validation/`