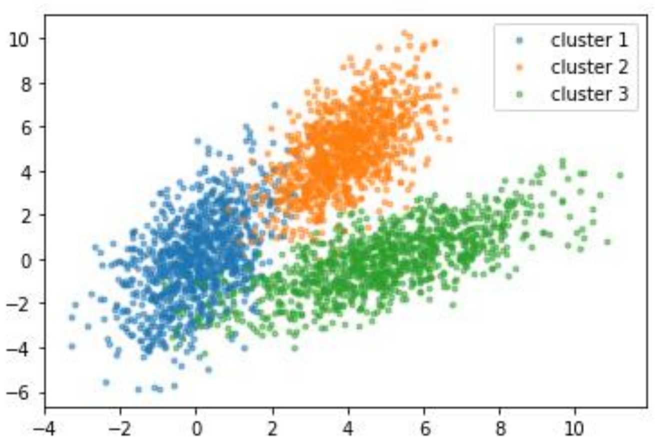
Clustering

Department of Computer Languages and Systems

General context

Unsupervised learning can be applied if …

- there are no labeled data
- possibly there are no predefined classes



- The objective is to divide the set of objects into a number of groups such that objects in the same group are more similar than those in other groups
- A cluster is, therefore, a collection of objects that are “similar” between them and are “dissimilar” to the objects belonging to other clusters

Introduction

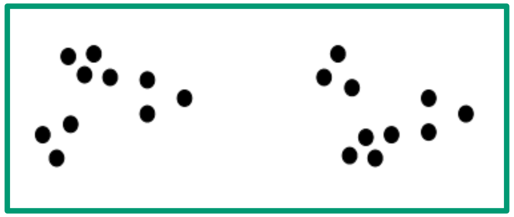
The results of clustering depend on:

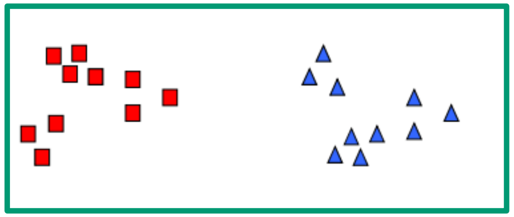
- the clustering algorithm used
- the available data set
- the similarity measure used to compare the objects

However, the clustering is, in many cases, quite subjective!

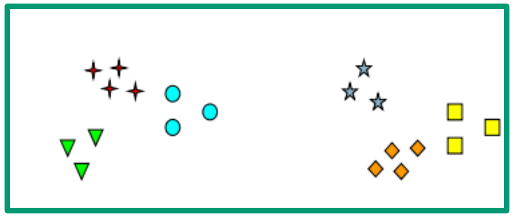
Introduction (ii): clustering is subjective

How many clusters?

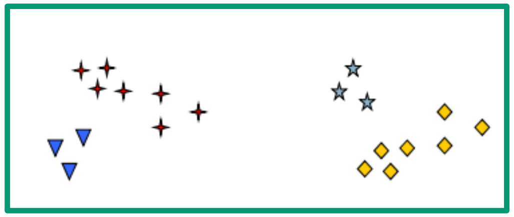




Two?



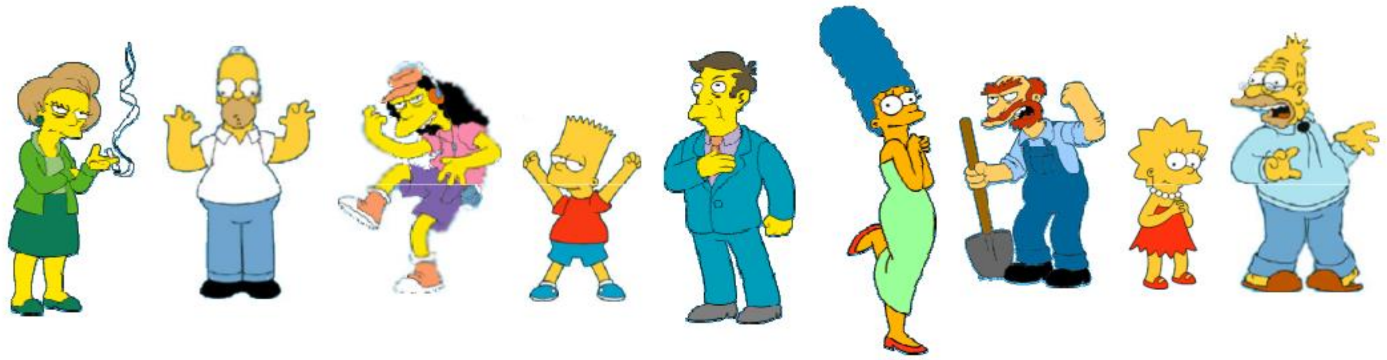
Six?

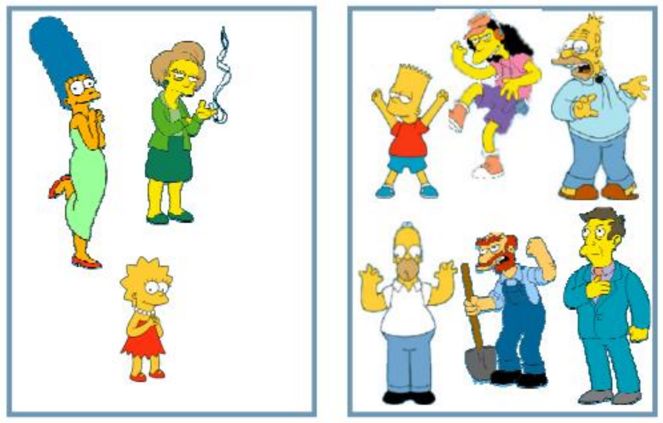


Four?

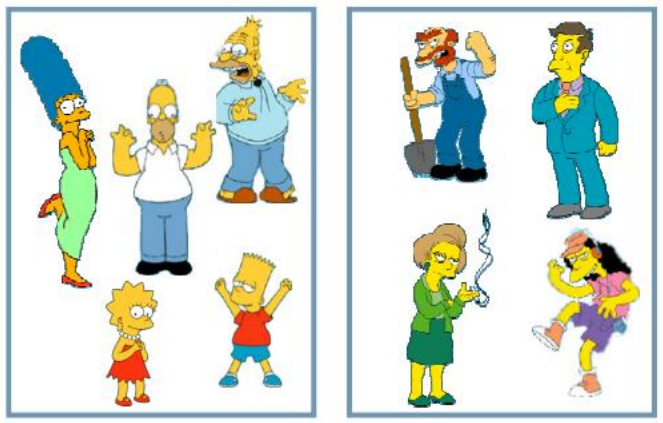
Introduction (iii): clustering is subjective

What is the natural way to group the characters?





Women vs. Men



Simpsons vs. Springfield School Employees

Applications

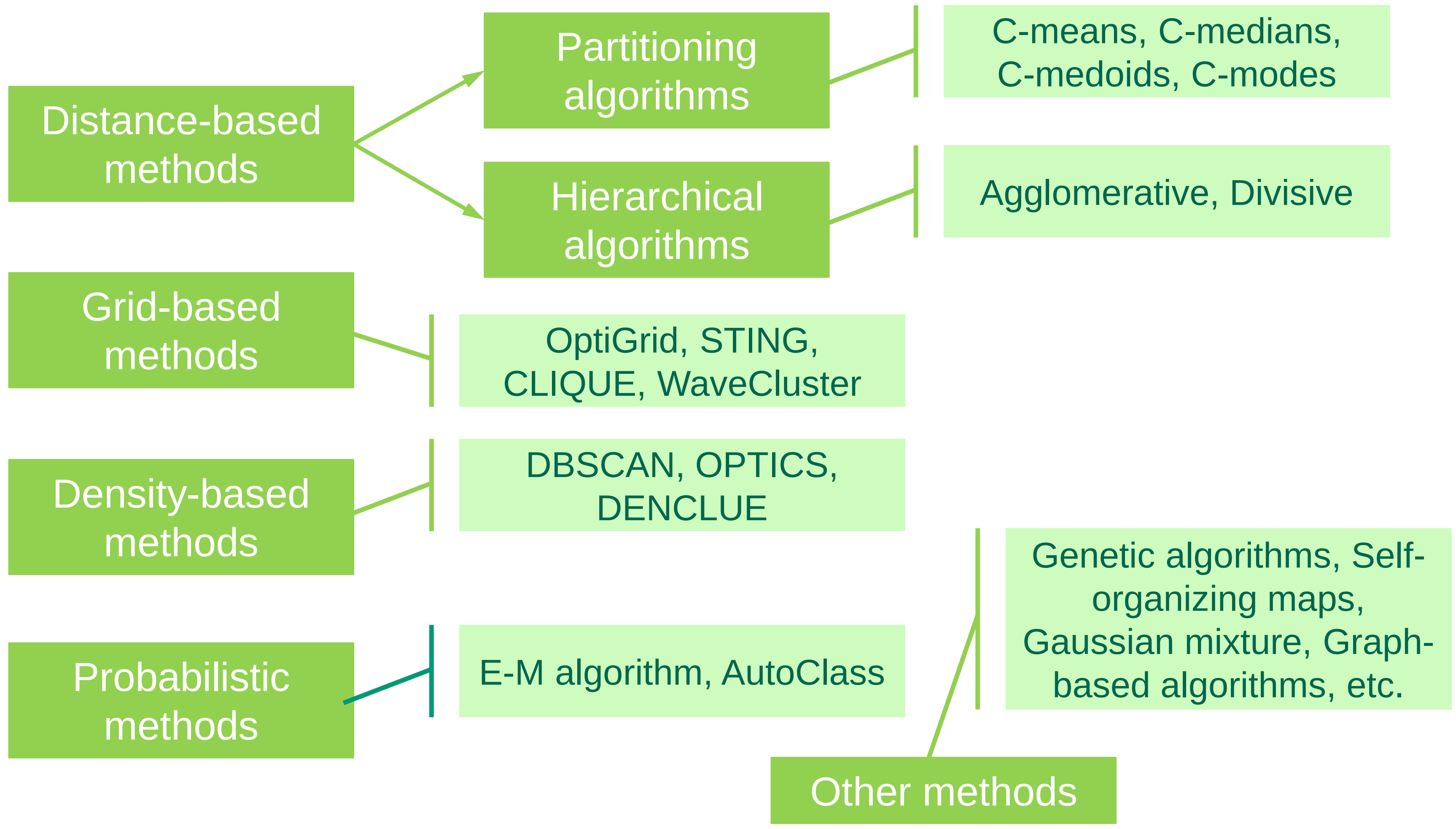
- Image processing: automatic segmentation of images, pixel classification in remote sensing imagery, MRI quantization, etc.
- Bioinformatics: gene-expression analysis to identify groups of genes, or to discover new subgroups of pathologies, etc.
- Marketing: market segmentation, customer segmentation, etc.
- Finance: to group either customer or company profiles, fraud detection, etc.
- Social networking: content understanding, language translation based on the location of the user, etc.
- Many others: document (or news) categorization, city planning, insurance, spam filter, identification of criminal behaviors, etc.

Similarity measures

See the slides in Unit 4 (Distance-based Classifiers)

In 2D:
- Manhattan (x_distance + y_distance)
- Euclidian -> Trigonometry
- Chebishev -> "*Maximum Norm*" -> MAX(x_distance, y_distance)
- Hamming distance: comparing binary vectors
- Cosine similarity: Cosine of the angle between two vectors to compute similarity

Types of clustering algorithms



Partitioning algorithms

- these algorithms use prototypes (central points or centroids) to characterize clusters and assign each object to the cluster whose centroid is the closest
- they require prior knowledge about the number of clusters (classes, *C*) to divide the data
- the best known algorithm and one of the most widely used within this group is C-means (some authors call it K-means)

The C-means algorithm

- it partitions the data points into *C* clusters based upon a similarity measure
- the object that is closest to the cluster centroid is assigned to that cluster
- after an iteration, it recalculates the centroids of those clusters, and the process continues until a predefined number of iterations is reached or the cluster centroids do not change after an iteration
- hyperparameters of the algorithm:
  - initial values of clusters
  - distance measures
  - number of clusters (*C*)

The C-means algorithm (ii)

- the centroid of a cluster corresponds to the mean (or the central point) of the cluster
- random initialization of centroids at the start of the algorithm may lead to different clusters when running the algorithm multiple times
- it computes the dissimilarity between data points using the Euclidean distance metric
- time complexity is linear O(n)

The C-means algorithm (ii)

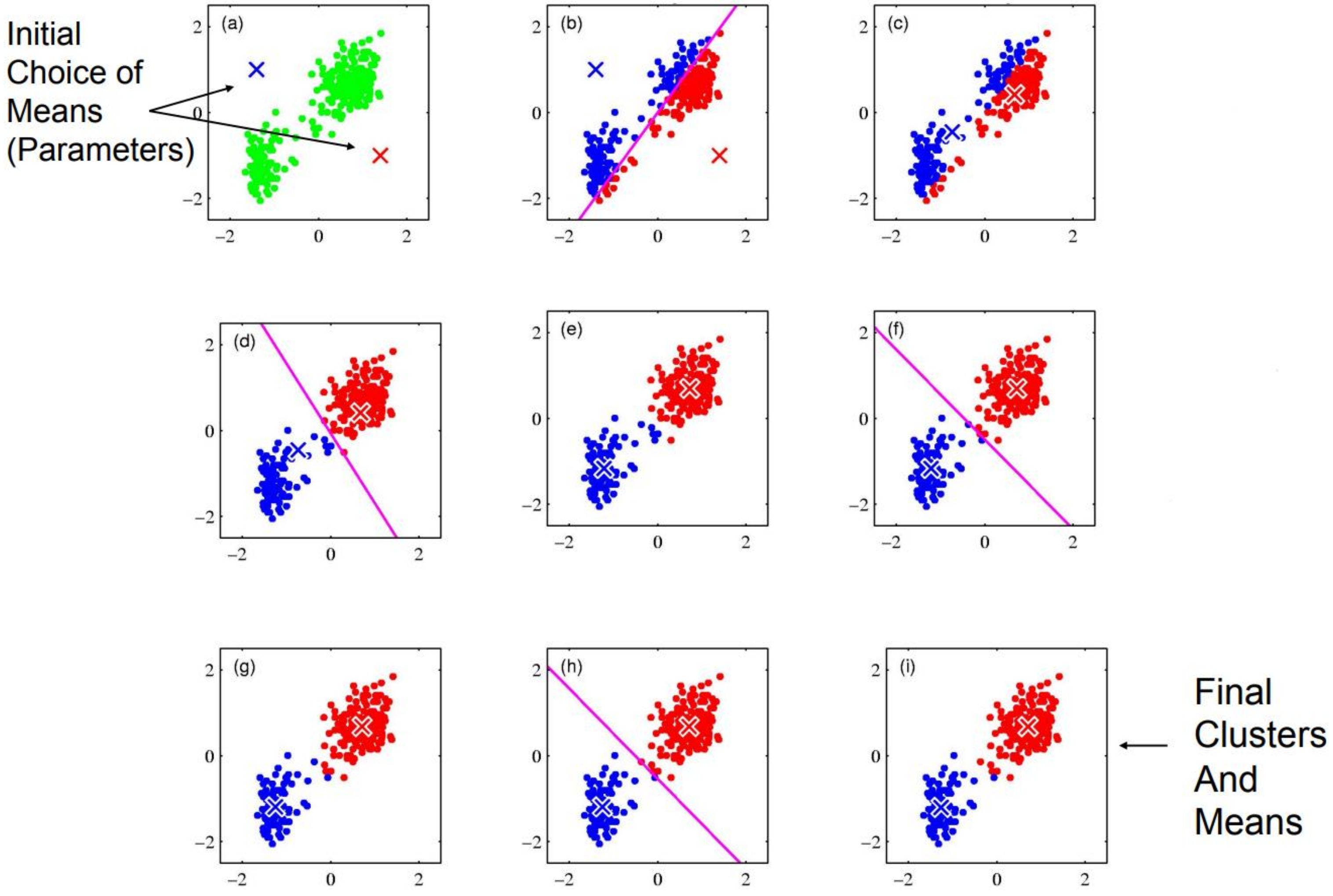
Variations

- C-medians calculates the median of each cluster to determine the centroids
  - this computes the dissimilarity between data points using the Manhattan distance
  - it is less sensitive to outliers in the data set
- C-medoids (or PAM, Partitioning Around Medoids), the medoid of a cluster has to be an input object of the data set
  - CLARA (Clustering Large Applications) is an extension to PAM to perform better for large data sets by applying PAM to multiple random samples of the original data and computing the best medoids in each sample
- C-modes is used with categorical data

The C-means algorithm (iv)

```
Randomly select $C$ cluster centroids
**repeat**
    Calculate the distance between each object and each cluster centroid
    Assign each object to the nearest cluster centroid
    Calculate the new centroid of each cluster
**until** the centroids stop moving (i.e. they do not change their
positions)
```

The C-means algorithm (v)



The optimal value of *C*

- a fundamental step for any unsupervised algorithm is to determine the optimal number of clusters *C* into which the data should be clustered
- the Elbow method is one of the most popular heuristics to determine this optimal value of *C*
- a better method than Elbow is the Silhouette method, which can be used to study the separation distance between the resulting clusters

The Elbow method

- it consists of plotting the sum of squared errors (SSE) as a function of the number of clusters *C* and picking the elbow (inflection point) of the curve as the optimal value of *C*
- Steps*:*

1. Run the clustering for a range of values for *C*
2. After each clustering, compute the sum of squared error (SSE) of each cluster
3. Create a line plot of the SSE for each value of *C*. This plot is called the elbow plot
4. Look at the elbow plot and find the value of *C* where the SSE begins to decrease linearly. That value is the optimal number of clusters

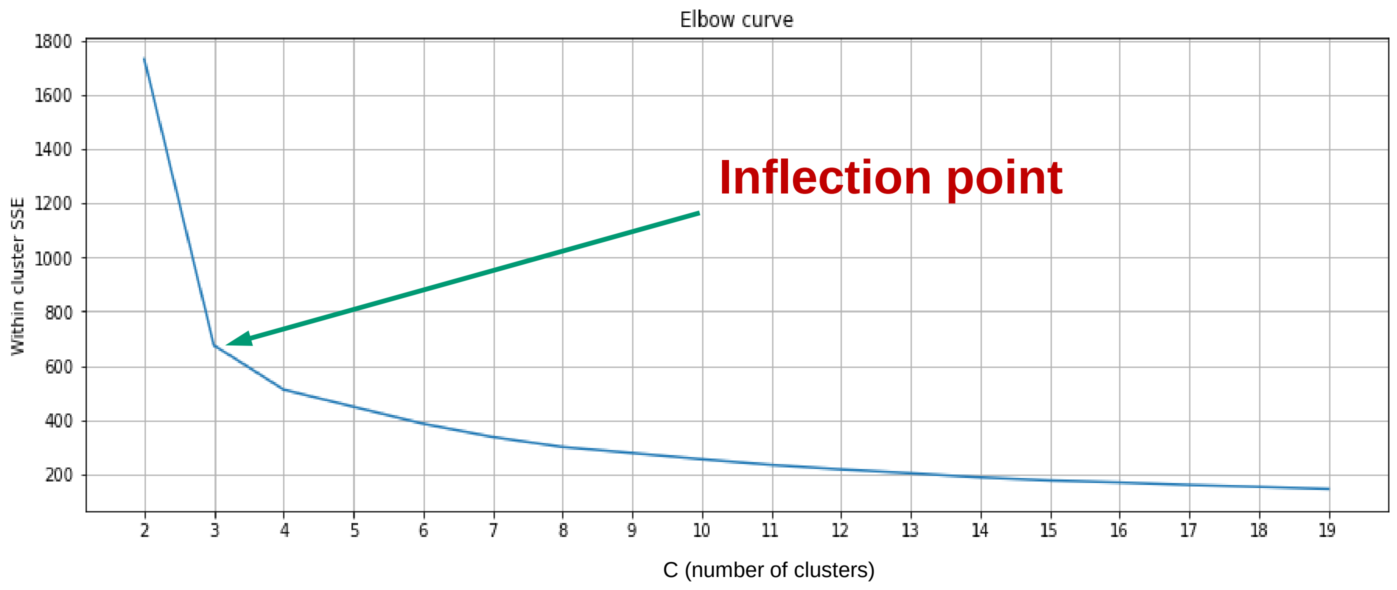
The Elbow method (ii)

- the SSE is the average distance between each object and the centroid
- Steps*:*

1. For each cluster
    1. For each object in the cluster
        1. Calculate the difference between each object and its cluster centroid
        2. Square the difference
    2. Compute the sum of all the differences
2. Add up the total distances computed for each cluster to give the sum of squared error (SSE)

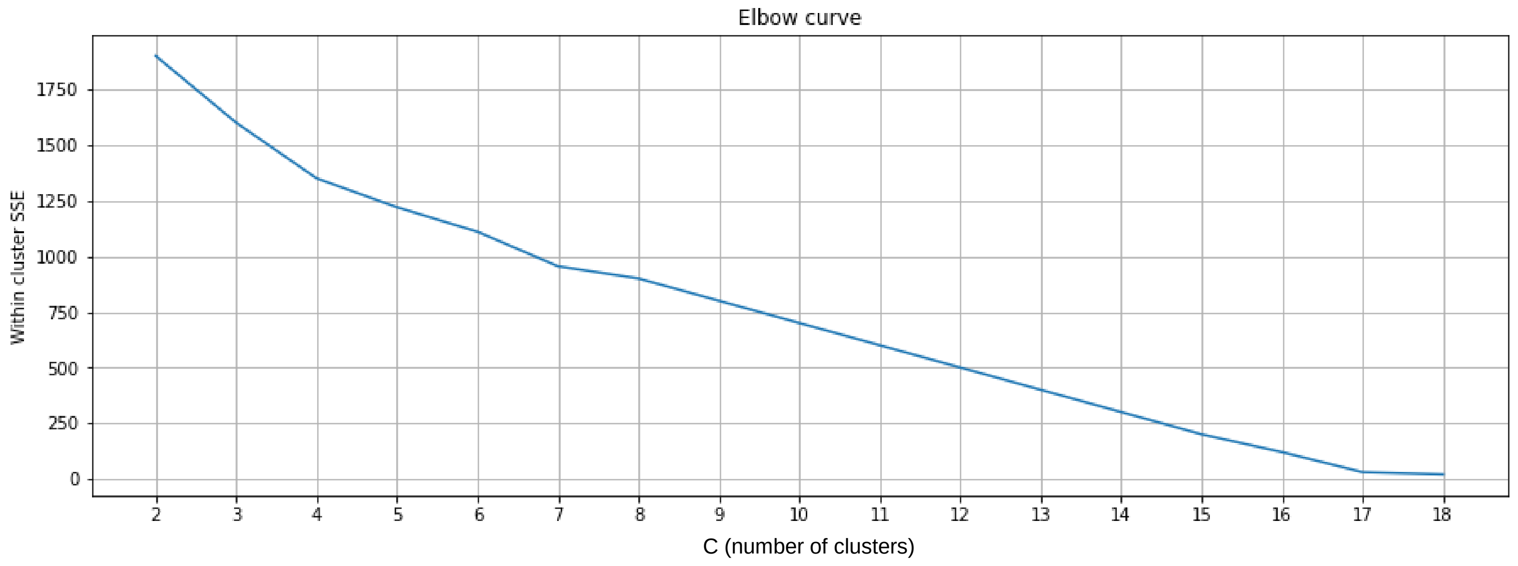
The Elbow method (iii)

- with an increase in *C*, the SSE decreases
- the inflection point is the point from where the decrease in SSE starts looking linear (there is a sharp and steep fall of the distance) → optimal value of *C*



The Elbow method (iv)

- what happens when you have a plot with a fairly smooth or linear curve and no obvious elbow point?



The Silhouette method

- a method of interpretation and validation of consistency within clusters of data
  - it consists of computing coefficients for each object
  - the silhouette coefficients measure how similar an object is to its own cluster (cohesion) compared to other clusters (separation)
  - the average of the silhouette coefficients for all the objects gives the silhouette score
  - the silhouette value ranges between [-1,1], where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters

The Silhouette method (ii)

- if most objects have a high value, then the clustering is appropriate. If many points have a low or negative value, then the clustering may have too many or too few clusters
  - the silhouette coefficient of +1 indicates that the object is far away from the neighboring clusters
  - the silhouette coefficient of 0 indicates overlapping clusters
  - the silhouette coefficient of <0 indicates that those objects might have been assigned to the wrong cluster or are outliers

The Silhouette method (iii)

Steps:

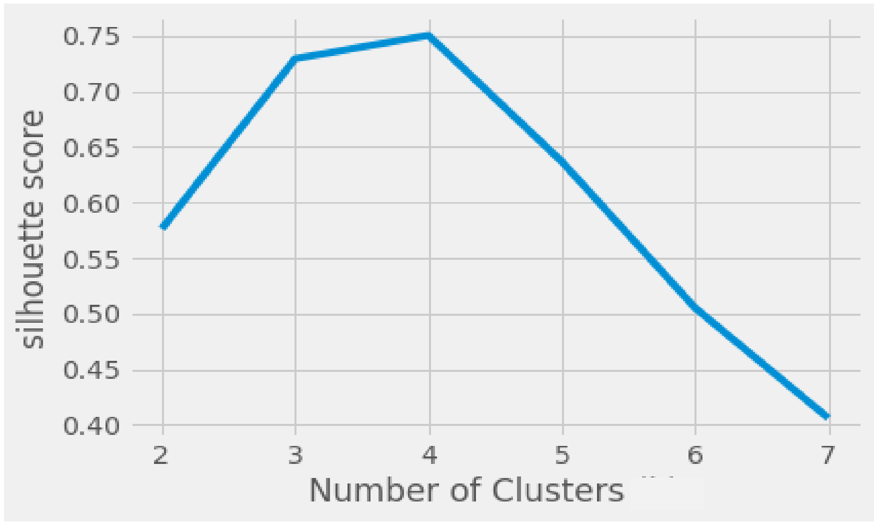
1. For each *i*'th object in the data set
    1. Compute the average distance of *i*'th object from all other objects in the same cluster, *a*(*i*)
    2. Compute the average distance of *i*'th object from all the objects in the closest cluster to its own cluster, *b*(*i*)
    3. Compute the silhouette coefficient of *i*'th object, *s*(*i*)

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

2. Average the silhouette coefficients to get the silhouette score

The Silhouette method (iv)

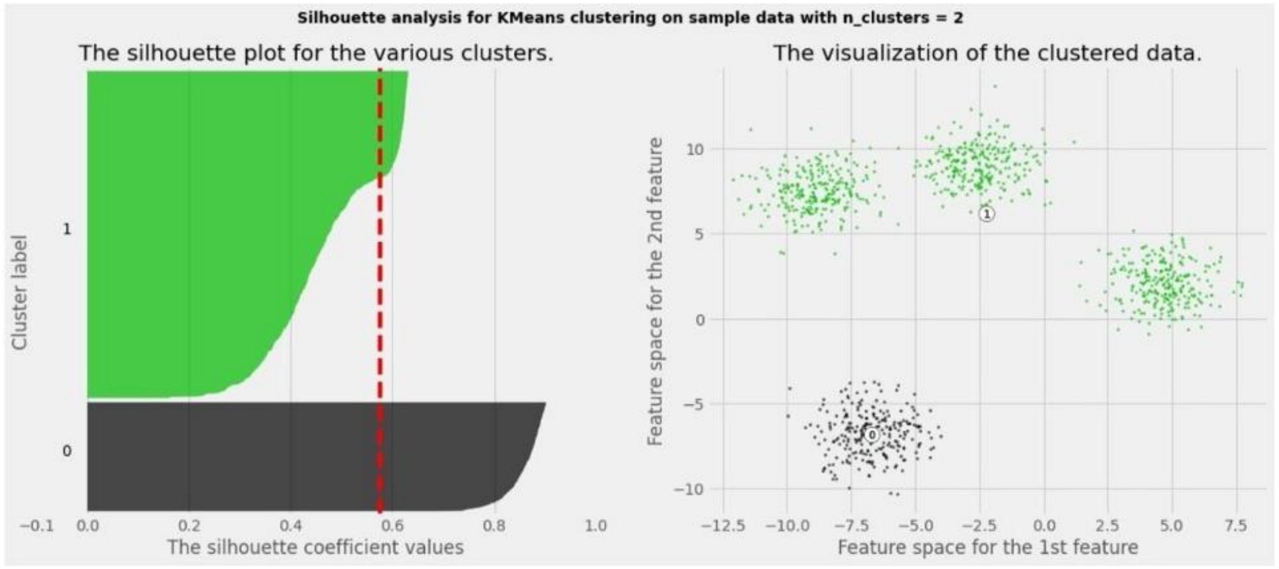
- similar to the Elbow method, we can pick a range of candidate values of *C*, run the clustering algorithm for each *C*, plot the silhouette score vs the number of clusters and take the clustering with the highest silhouette score as the optimal value of *C*



The Silhouette method (v)

- another way to determine the optimal value of C consists of analyzing a silhouette plot:
  - the x-axis displays the silhouette coefficients (sorted)
  - the y-axis displays the labels of each cluster
- if all the cluster plots are beyond the average silhouette score, have mostly uniform thickness, and do not have wide fluctuation in size, the number of clusters used is optimal
- the number of clusters is suboptimal if:
  - the plot for a cluster falls below the average coefficient, or
  - there are wide fluctuations in the size and thickness of the cluster plots

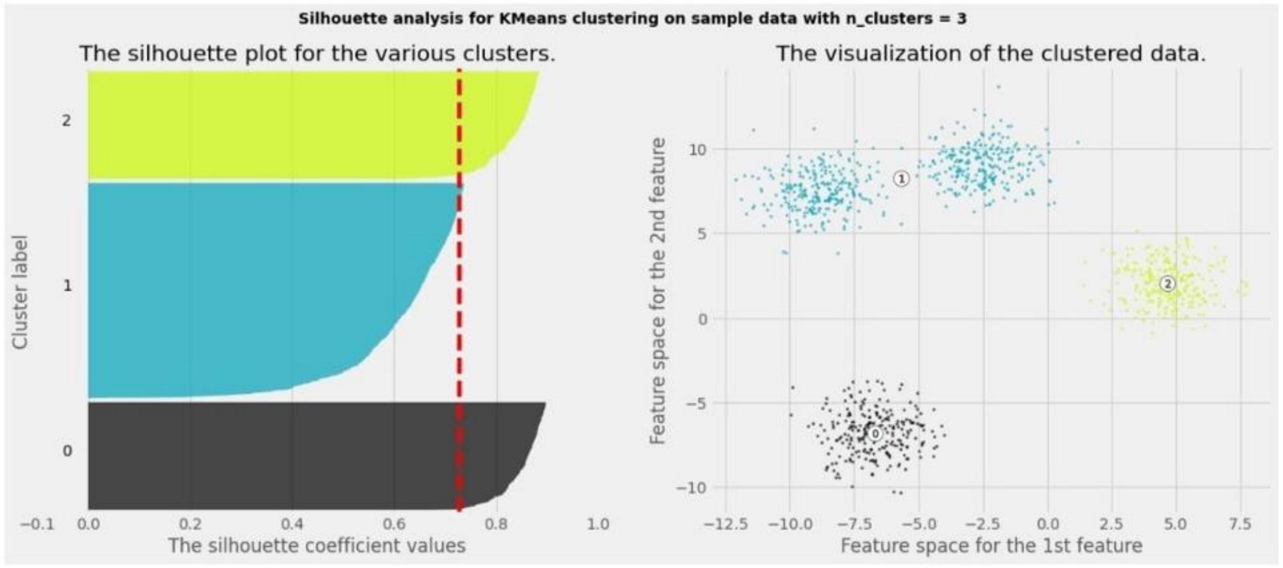
The Silhouette method (vi)



The silhouette plot for C = 2 shows that the cluster with label 1 is bigger in size due to the grouping of the 3 sub-clusters into one big cluster

Average silhouette score = 0.704978749608

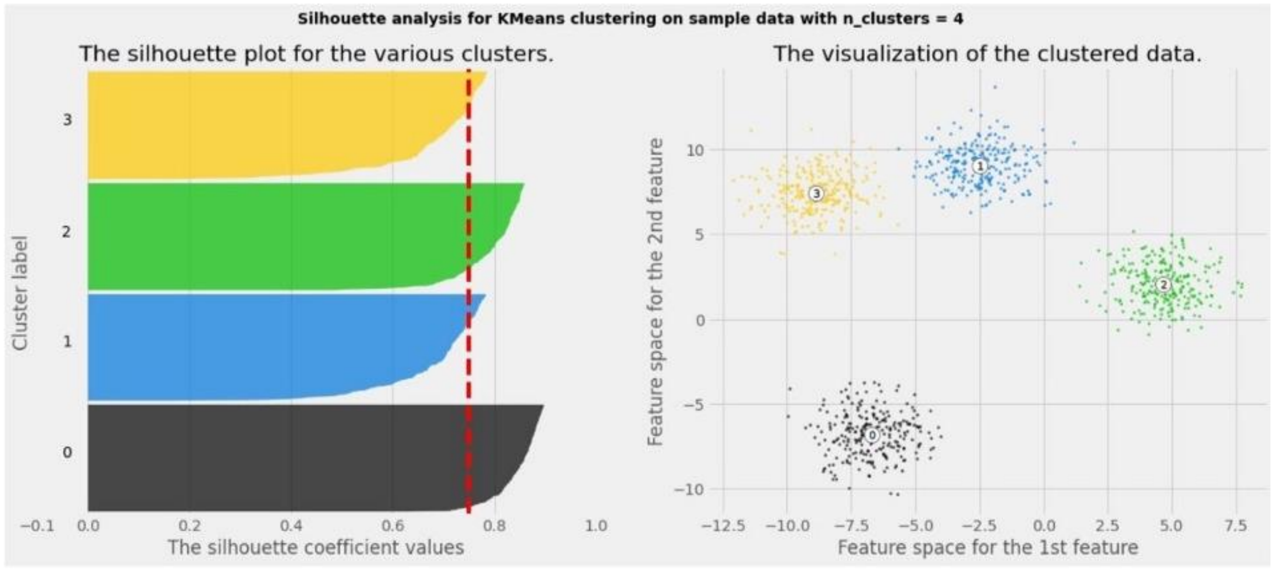
The Silhouette method (vii)



The silhouette plot shows that *C* = 3 is bad, as all the points in the cluster with label 1 are below-average silhouette scores

Average silhouette score = 0.588200401213

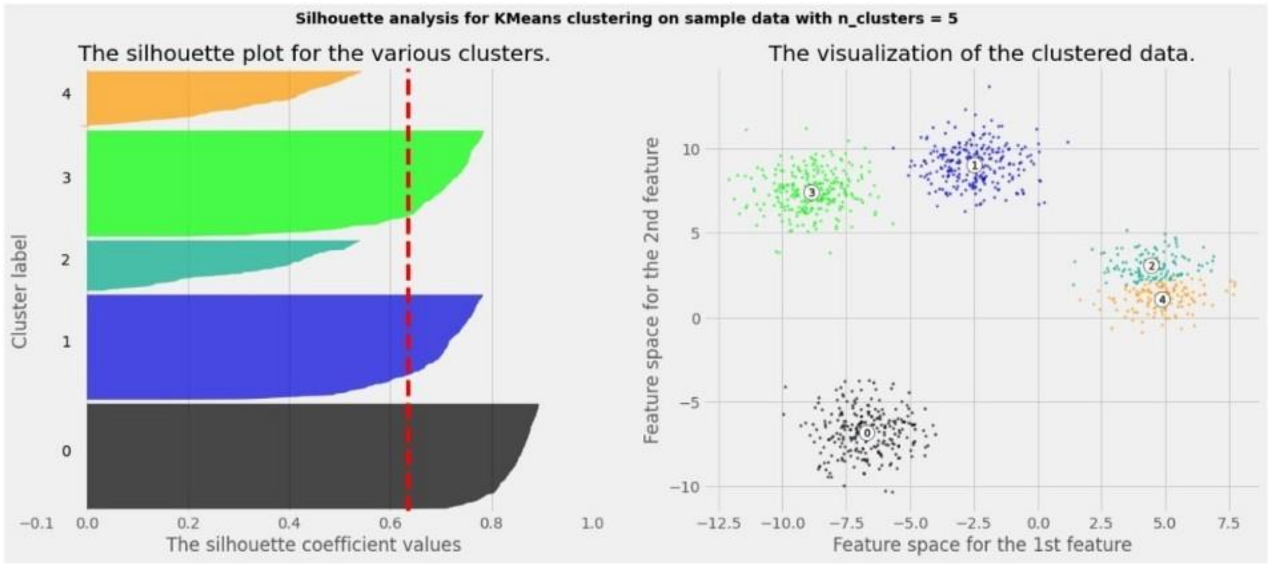
The Silhouette method (viii)



For *C* = 4, all the plots are more or less of similar thickness and hence are of similar sizes, as can be considered as the **best *C***

Average silhouette score = 0.650518663273

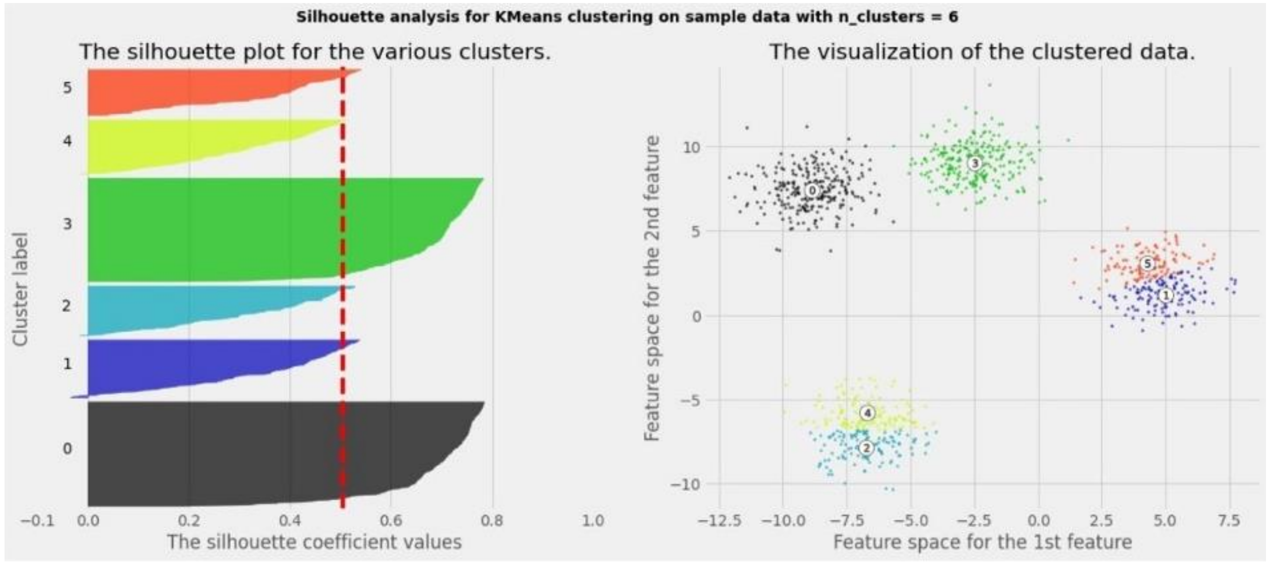
The Silhouette method (ix)



- The silhouette plot shows that *C* = 5 is bad, as all the points in the clusters 2 and 4 are below-average silhouette scores

Average silhouette score = 0.563764690262

The Silhouette method (x)



- The silhouette plot shows that *C* = 6 is bad, as all the points in the clusters 1, 2, 4 and 5 are below-average silhouette scores

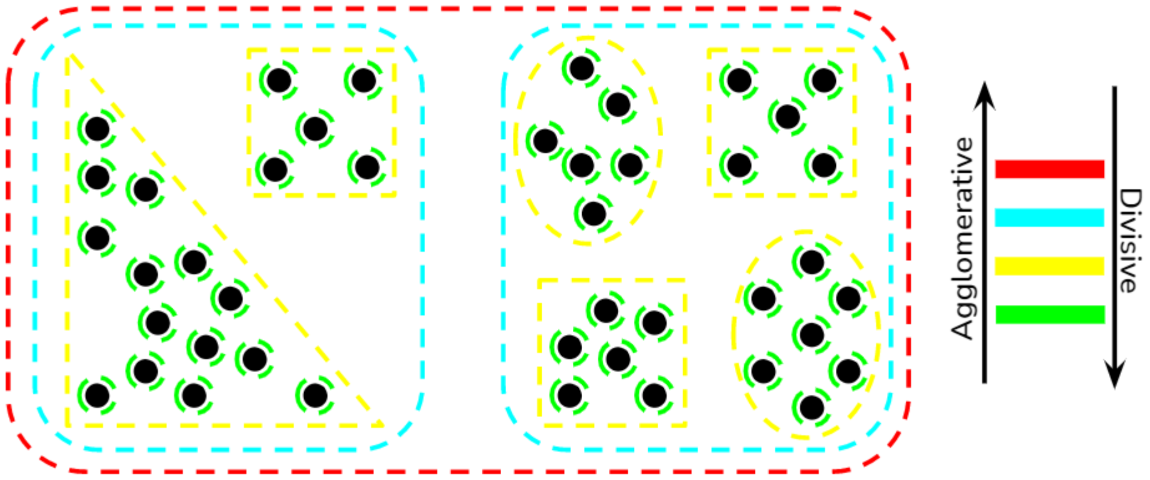
Average silhouette score = 0.450466629437

Hierarchical algorithms

- unsupervised learning algorithms that seek to build a hierarchy between objects while clustering them
- these are incremental algorithms
- they do not require the number of clusters to be specified initially
- it can create very complex shaped clusters
- time complexity is quadratic $O(n^2)$

Hierarchical algorithms (ii)

- they can be agglomerative or divisive
  - the agglomerative methods are bottom-up methods. Starting from a cluster for each object, they merge the most similar clusters into a new cluster until some stopping criterion is met
  - the divisive methods are top-down. Starting with a single cluster containing all the objects, this cluster is divided until the stopping criterion of the algorithm is verified

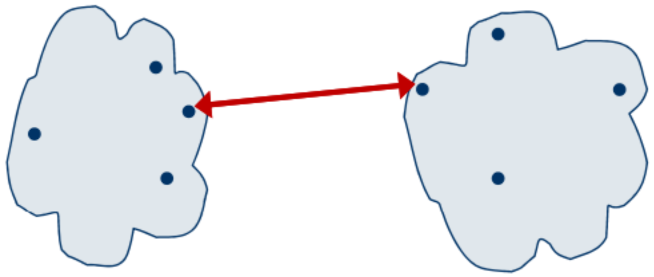


Hierarchical algorithms (iii)

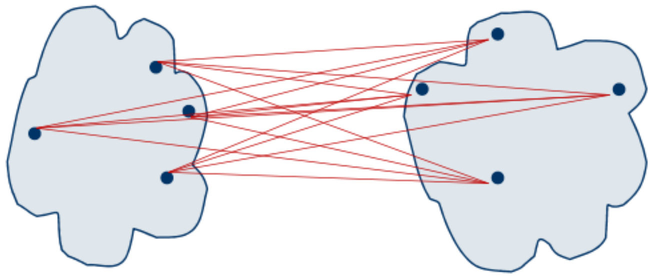
- these algorithms create a distance matrix of all the existing clusters and perform the linkage between the clusters depending on the criteria of the linkage:
  - Single Linkage: the distance between the two clusters is the shortest distance between objects in those two clusters
  - Complete Linkage: the distance between the two clusters is the farthest distance (diameter) between objects in those two clusters
  - Average Linkage: the distance between the two clusters is the average distance of every object in the cluster with every object in the other cluster
  - Centroid linkage: the distance between the two clusters is the distance between their centroids

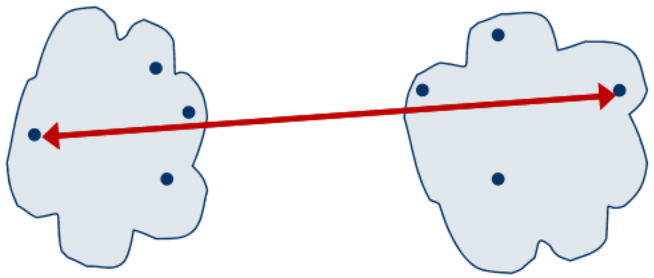
Hierarchical algorithms (iv)

Single linkage

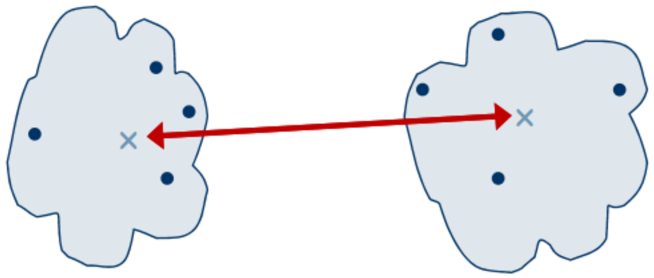


Average linkage





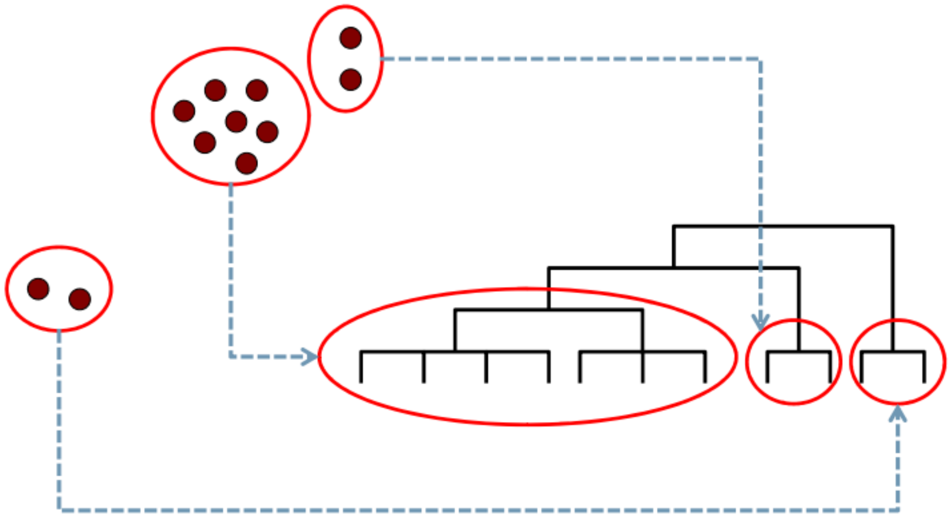
Complete linkage



Centroid linkage

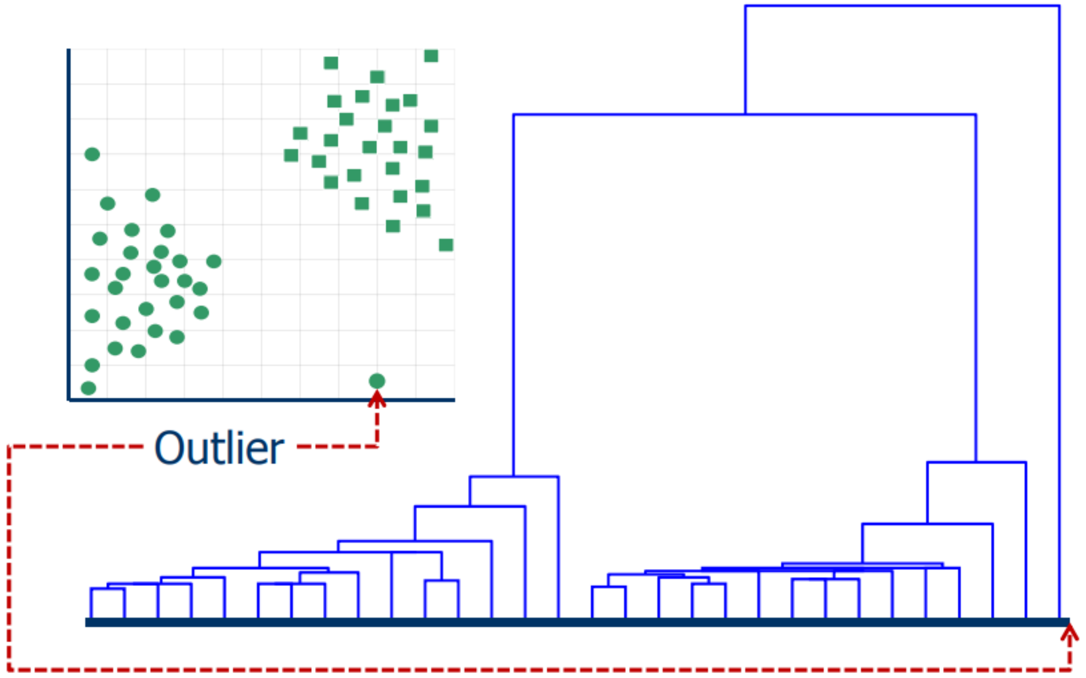
Hierarchical algorithms (v)

- the output is represented by a dendrogram, which shows the hierarchical relationship between the clusters
- the similarity between two objects is represented by the height of the node in the dendrogram



Hierarchical algorithms (vi)

- it is not sensitive to outliers



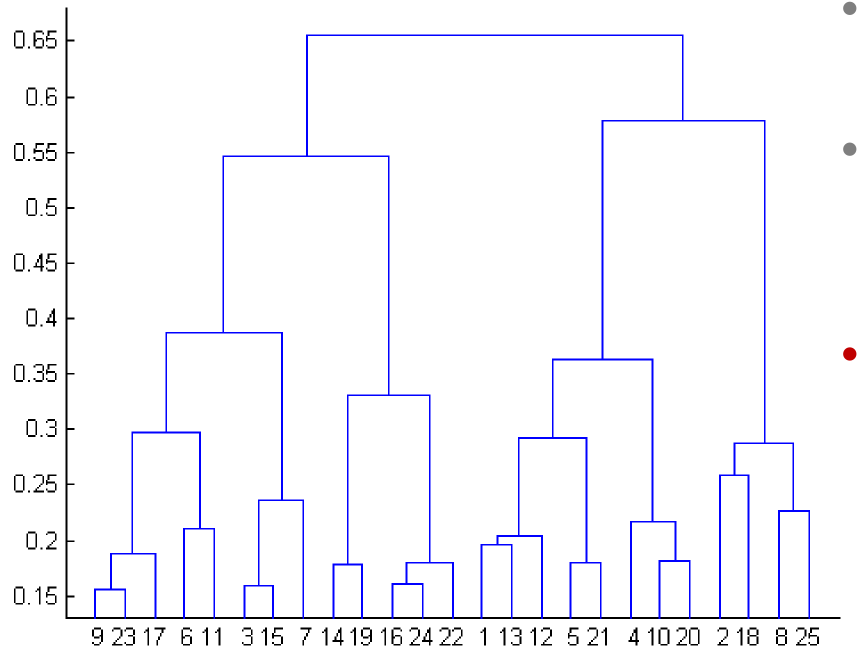
Hierarchical algorithms (vii)

The agglomerative hierarchical algorithm

```
Compute the distance (similarity) matrix
Initialization: each object is a separate cluster
repeat
    Merge the two closest clusters
    Update the distance matrix
until reaching only one cluster
```

Hierarchical algorithms (viii)

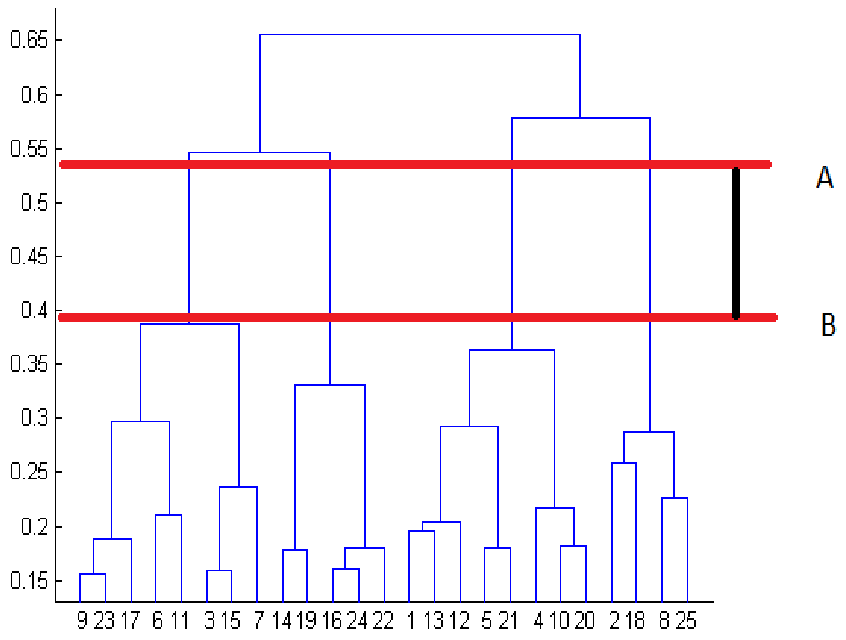
An example with 25 objects:



- we start by assigning each object to separate clusters
- two closest clusters are then merged till we have just one cluster at the top
- the height in the dendrogram at which two clusters are merged represents the similarity between two clusters

Hierarchical algorithms (ix)

An example with 25 objects:



- the decision of the optimal number of clusters can be chosen by observing the dendrogram
- the optimal number is the number of vertical lines in the dendrogram cut by a horizontal line that covers the maximum vertical distance (AB) without intersecting a cluster

Here the best choice will be 4 clusters

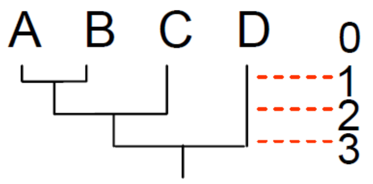
Hierarchical algorithms (x)

- An example with 4 objects described by the following distance matrix:

	A	B	C	D
A	0	1	4	5
B		0	2	6
C			0	3
D				0

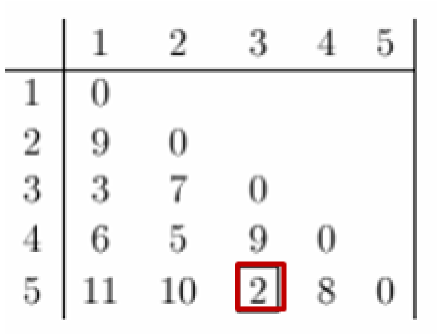
Single linkage

d	**k**	**K**	**Comments**
0	4	{A}, {B}, {C}, {D}	We start with each point = cluster
1	3	{A, B}, {C}, {D}	Merge {A} and {B} since A & B are the closest: d(A, B)=1
2	2	{A, B, C}, {D}	Merge {A, B} and {C} since B & C are the closest: d(B, C)=2
3	1	{A, B, C, D}	Merge D

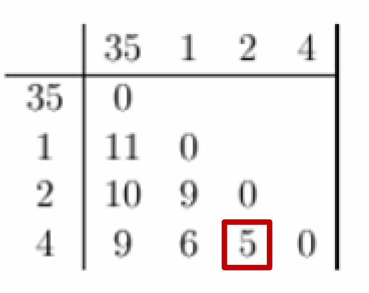


Hierarchical algorithms (xi)

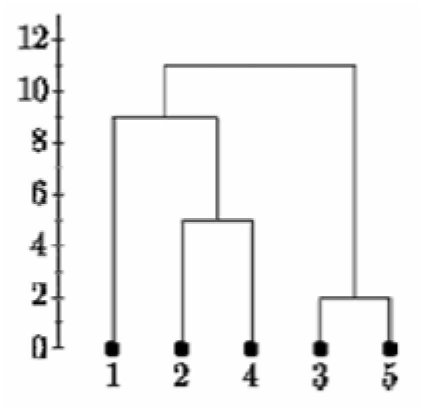
Complete linkage



	1	2	3	4	5
1	0				
2	9	0			
3	3	7	0		
4	6	5	9	0	
5	11	10	$\boxed{2}$	8	0



	35	1	2	4
35	0			
1	11	0		
2	10	9	0	
4	9	6	$\boxed{5}$	0



The smallest distance is between 3 and 5 and they get merged into a first cluster '35'

Now we replace the 3 and 5 entries by an entry "35“: the distance between "35" and every other item is the maximum of the distance between this item and 3 and this item and 5. The items with the smallest distance get clustered next ’24’

Density-based clustering

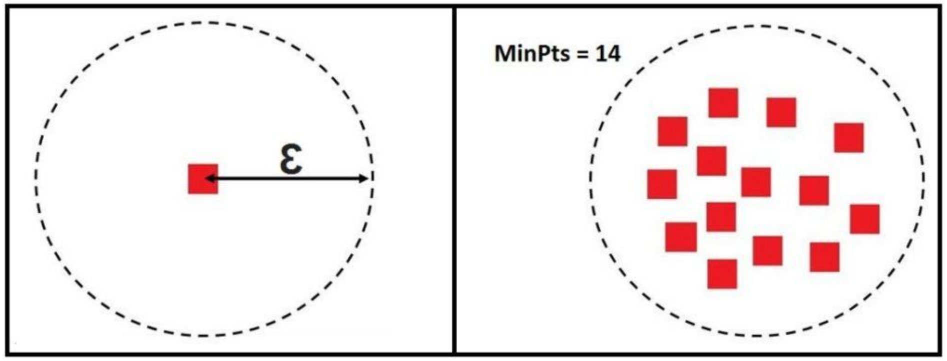
- they assume that a cluster is a dense region of points, separated by sparse regions of other dense regions
- useful when clusters can be of arbitrary shapes, are interleaved or there are noise/outliers in the data
- the best known and most widely used algorithm within this group is DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN

- clusters are formed by connecting the objects that are densely located in a region
- it uses concepts such as core point, border point and noise point to come up with clusters
- time complexity is logarithmic O(n·log n)

DBSCAN (ii)

- Two important parameters:
  - Epsilon (ε): the maximum distance between a pair of objects
  - Minimum points (MinPts): minimum number of objects required to form a dense region (cluster)
- ε decides the size of a circle and MinPts decides the minimum number of objects that require being in that circle to consider it a cluster



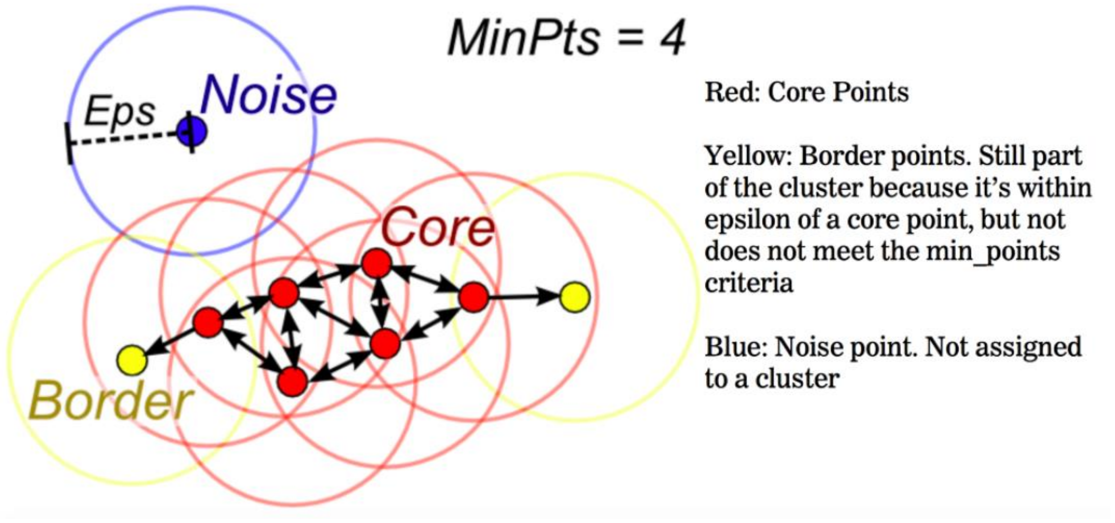
DBSCAN (iii)

Meaning of ε:

- two points are considered as neighbors if and only if they are separated by a distance less than or equal to epsilon

DBSCAN (iv)

- objects are categorized into three types:
  - an object is a core point if it has at least MinPts including itself within its ε-neighbourhood
  - an object that is within a neighborhood of a core point but it itself cannot be a core point is a border point
  - an object is a noise point if it is neither the a core point nor a border point



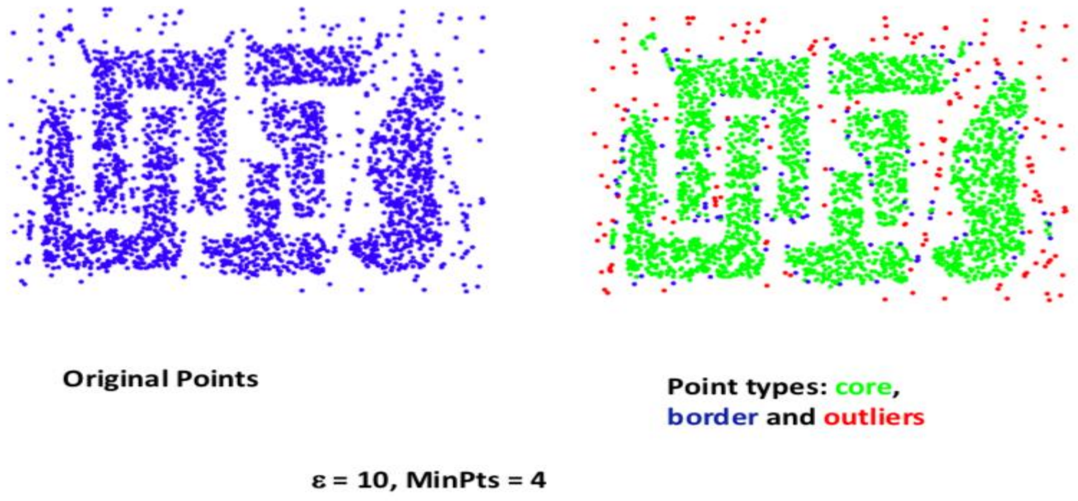
Red: Core Points

Yellow: Border points. Still part of the cluster because it's within epsilon of a core point, but not does not meet the min_points criteria

Blue: Noise point. Not assigned to a cluster

DBSCAN (v)

- Initially, the algorithm begins by selecting an object randomly from the data set, and checks if the selected point is a core point
- For each core point, find all the connected objects
- Assign each non-core object to the nearest cluster if the cluster is its ε-neighbor. Otherwise, assign it to noise
- The algorithm stops when it explores all the objects one by one and classifies them as either core, border or noise point



Summary: Pros

- C-means
  - can handle large amounts of data
  - simple to implement
- Hierarchical clustering
  - does not need to specify the initial value
  - easy to implement, scalable and easy to understand
- DBSCAN
  - does not need to specify the initial value
  - is robust to outliers (noise points)

Summary: Cons

- C-means
  - needs to manually choose the *C* value
  - sensitive to outliers in the data set
  - dependent on starting point
- Hierarchical clustering
  - difficulty to handle a large amount of data
  - no backtracking
  - more space and time complexity
- DBSCAN
  - dependent on the values of ε and MinPts
  - struggles to work with high dimensionality data