

Clustering

Paulo J Azevedo

DI - Universidade do Minho
2010-2022

Computational methods to derive *natural groups* from data. Non-supervised learning, segmentation, etc.

Definition

- Organize data in groups so that exists:
 - High similarity between the elements of a group
 - Low similarity between groups.
- Contrast to classification: considering the data, find the “labels” for the class attribute for each case in the training set and the number of classes.
- Informally, find the *natural grouping* among the elements of a dataset.
- Other names for this process:
 - Marketing, *segmentation*,
 - In psychology, *sorting*,
 - Statistics, *classification*,
 - In AI, *unsupervised learning*.

Natural groups in these data?

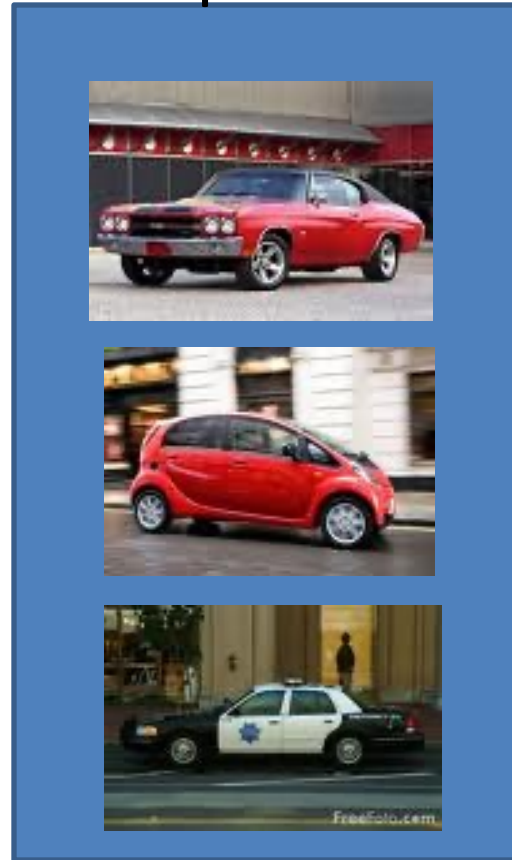


Natural groups in these data?

Sport Cars



Non Sport cars



Natural groups in these data?

Europeans



Asian

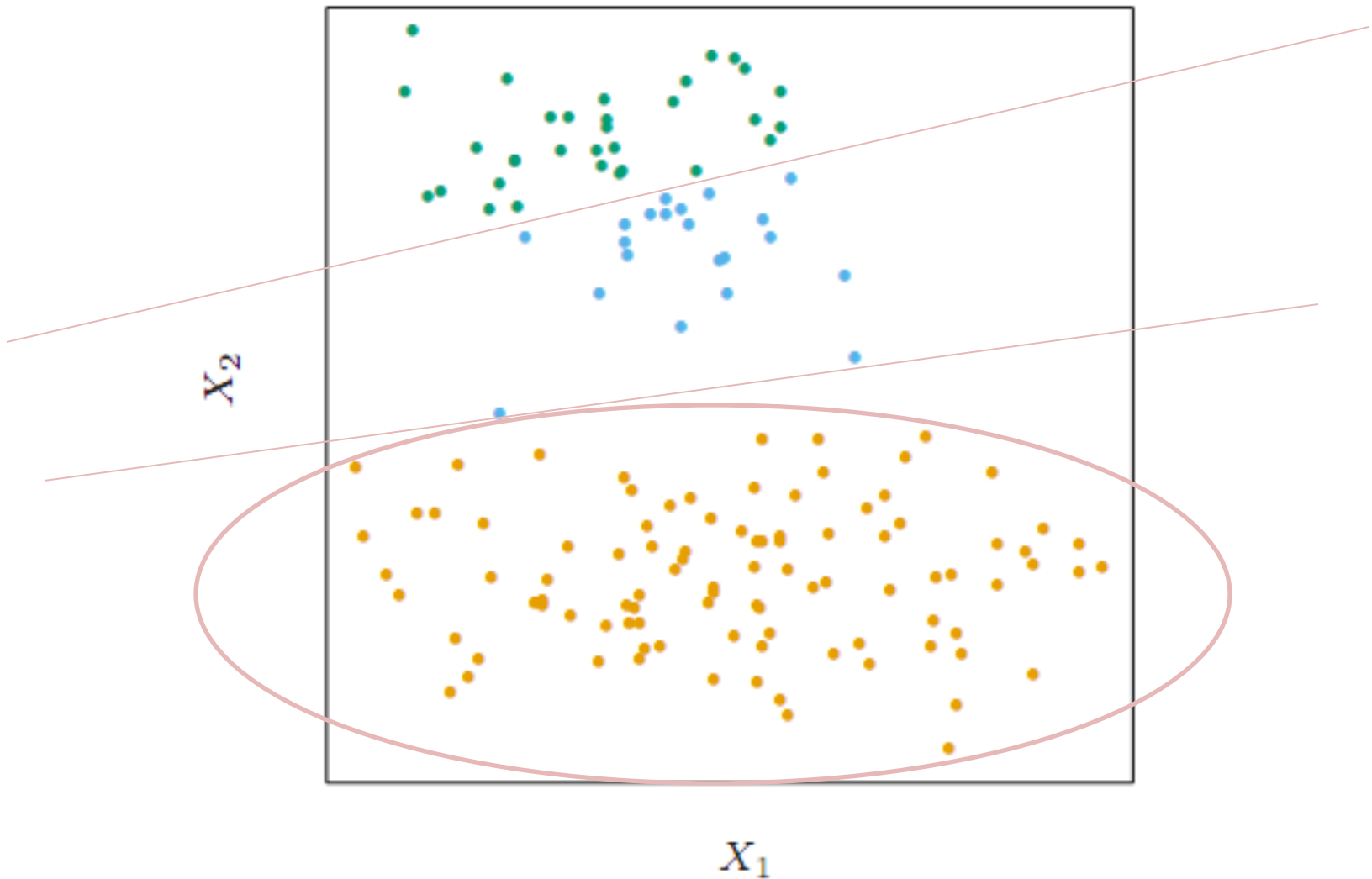


American



Clearly, a subjective process...

An example...



Clustering using k-means algorithm. Generated synthetic data to derive three partitions (blue, green and orange).

Similarity

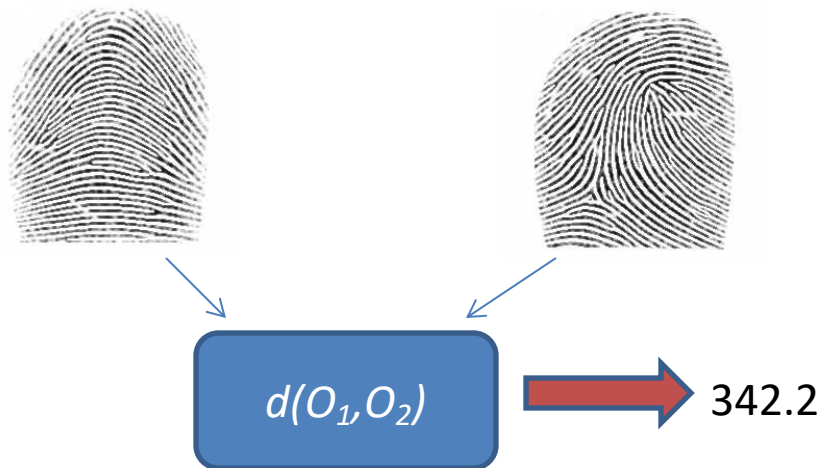
- Important concept:
 - the state of being similar; likeness; resemblance.
 - an aspect, trait, or feature like or resembling another or another's: - (dictionary.com)
- Question?



Similarity measure

- Defined as a measure of distance between two objects:

Definition: Let O_1 and O_2 be two objects (cases). The distance (dissimilarity) between O_1 and O_2 is a real number denoted as $d(O_1, O_2)$



Similarity Measures

- Real number attributes:
 - Euclidian distance

$$d(O_1, O_2) = \sqrt{\sum_i (x_i - y_i)^2}$$

- Manhattan distance

$$d(O_1, O_2) = \sum_i |x_i - y_i|$$

- Minkowski distance

$$d(O_1, O_2) = \left(\sum_i |x_i - y_i|^q \right)^{\frac{1}{q}}, \quad q > 0$$

Similarity Measures

- Binary Attributes

$$d(O_1, O_2) = \frac{r + s}{q + r + s + t}$$

	1	0	sum
1	q	r	q+r
0	s	t	s+t
sum	q+s	r+t	

- Nominal Attributes ($m = \# \text{matches}$, $p = \# \text{vars}$)

$$d(O_1, O_2) = \frac{p - m}{p}$$

Similarity between Objects

- How to compute distance between objects with vars of different types?

For p vars, the comparison is:

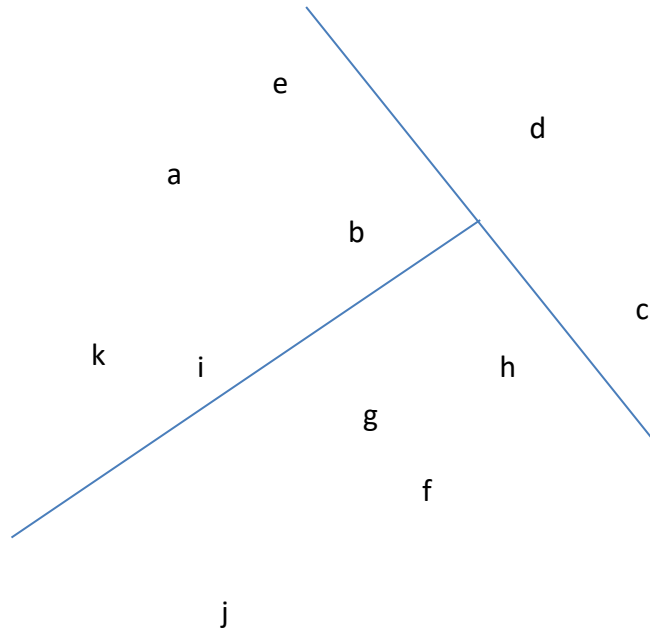
$$d(O_1, O_2) = \frac{\sum_{f=1}^p (\delta^{(f)} \times d^{(f)})}{\sum_{f=1}^p \delta^{(f)}}$$

0, if x_1 or y_1 do not exist. 1, otherwise.

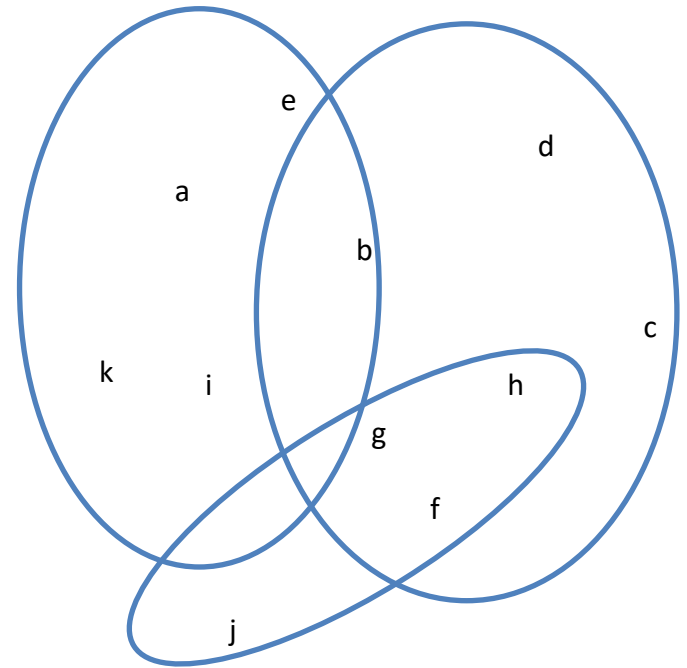
Distance between objs obtained using var f .

Representing Clusters

Disjoint sets



Intersection Sets

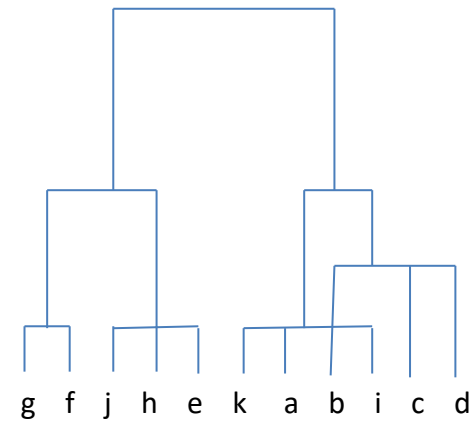


Representing Clusters

Probabilistic Clusters

	1	2	3
a	0.5	0.1	0.4
b	0.7	0.2	0.1
c	0.1	0.1	0.8
d	0.1	0.2	0.7
e	0.5	0.1	0.4
f	0.1	0.7	0.2
g	0.4	0.5	0.1
h	0.2	0.8	0.0
i	0.3	0.5	0.2
j	0.1	0.7	0.2
k	0.55	0.15	0.3

Hierarchical Clusters



Clustering

- Partitioning
 - Define several partitions of objects and then evaluate using a criterion. e.g. *k*-means, where *k* is given.
- Hierarchical
 - Build a hierarchical composition of objects according to some criteria. e.g. BIRCH , weka HClustering e Cluto.
- Density-based
 - Use the notion of cluster *density* (#objs in a cluster). Enables to find non spherical clusters (which does not happens in partition methods where a distance measure is used). It also permits to filter *outliers*. e.g. weka DBSCAN e OPTICS.
- Model-based
 - Define a model for each cluster. For each model, find the best fit for the data. Enables to find the ideal *k* value (#clusters) using standard statistics. e.g. weka CobWeb, EM.

Partition Algorithms

- Problem: Given n data objects and one k parameter defining the number of partitions, gather the n objects into k partitions. Each partition represents a cluster. The *clusters* are defined in such a way that optimizes a objective partition criterion i.e. similarity function. Thus, objects of the same partition are similar. On the other hand, objects from different clusters are different (in terms of attribute values).

$$E = \sum_{i=1}^k \sum_{p \in C_i} |p - m_i|^2$$

K-means Algorithm

k-modes for
categorical data

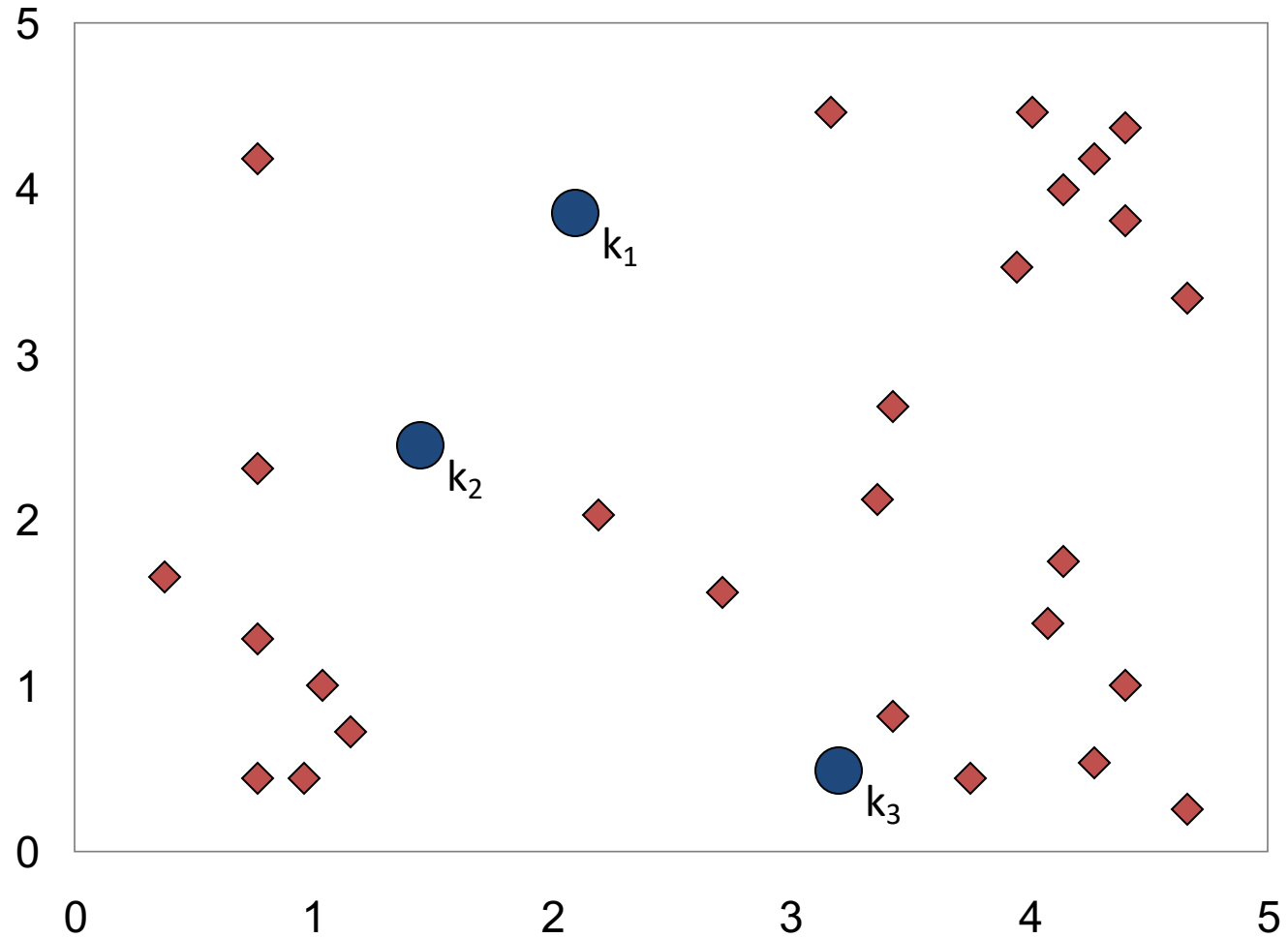
K-means

- Input: n objects and a k parameter defined by the user.
 - Output: a set of k clusters that minimizes the square error criterion.
1. Randomly choose k objs to play the role of the k clusters
centroids (m_i)
 2. Repeat
 1. (Re)Assign each obj to the most similar cluster (considering the mean value computed using the objs already assigned to the cluster),
 2. (Re)compute the mean for each cluster (obtain a new centroid).
 3. Until there is no more change (obj cluster stability)

Using the distance between
each obj and the centroid
(e.g. Euclidian distance)

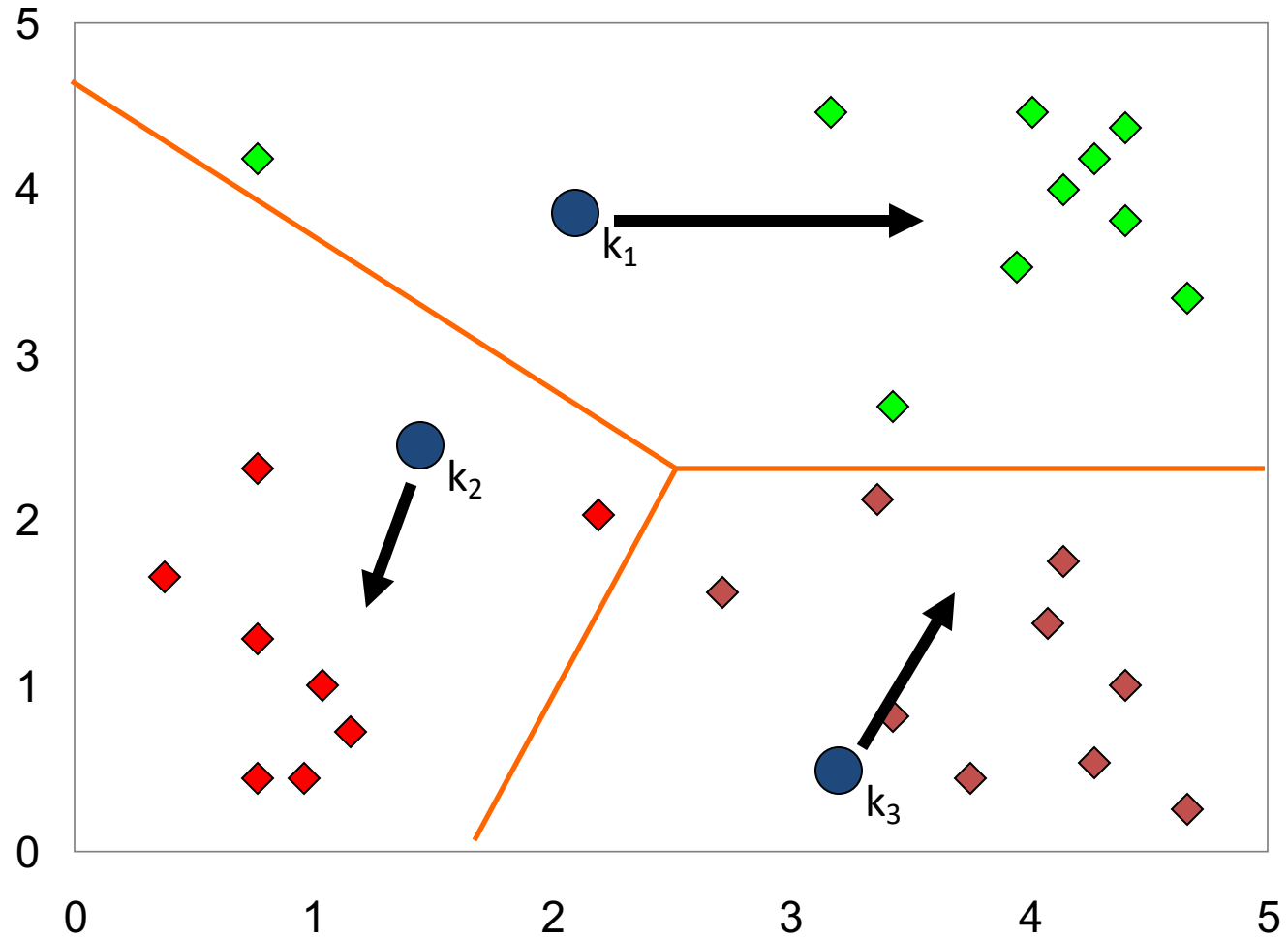
K-means Clustering: Initial state

Algorithm: k-means, distance measure : Euclidian distance, k_i : centroid of partition i



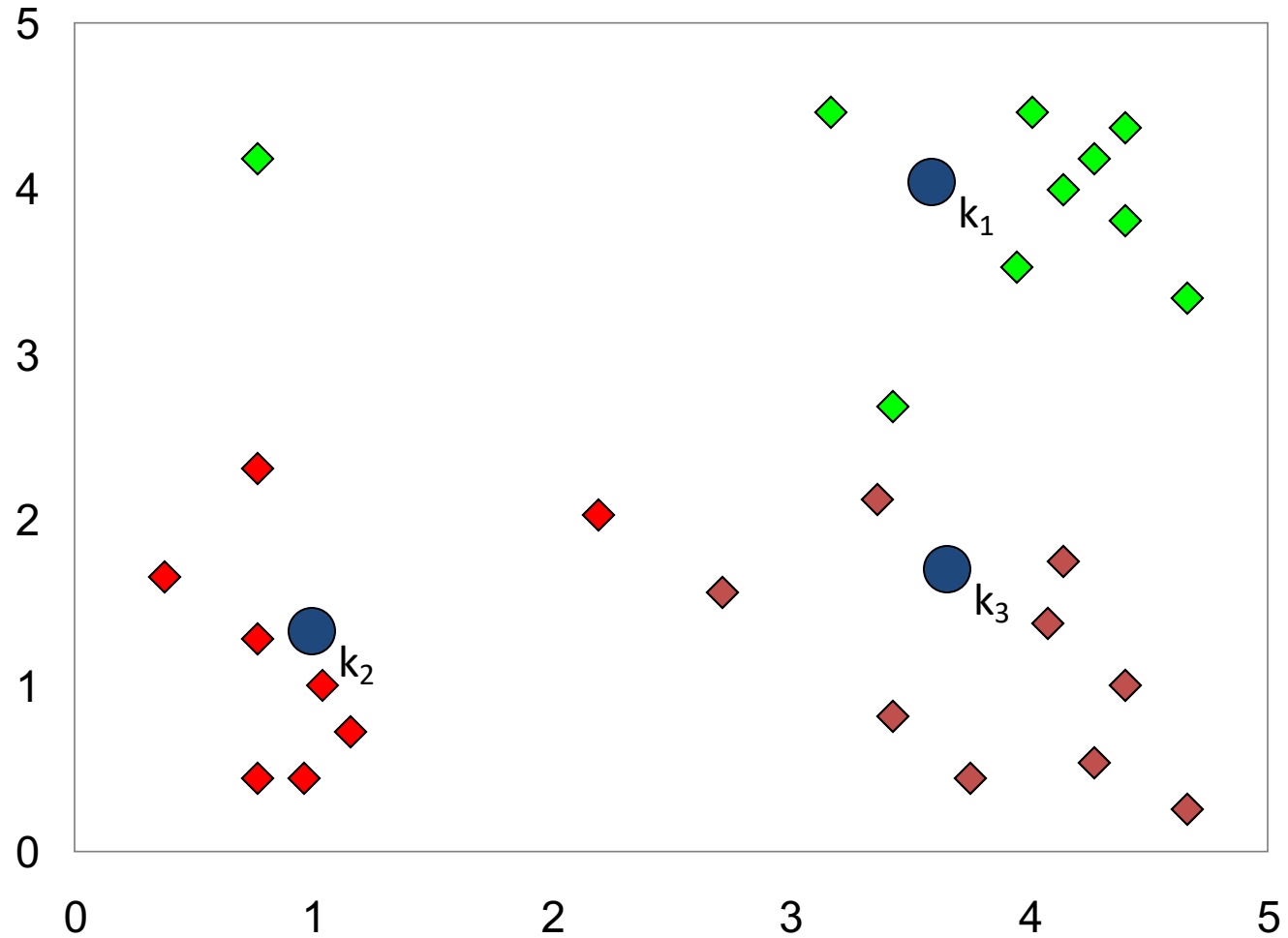
K-means Clustering: Step 2

Algorithm: k-means, distance measure : Euclidian distance, k_i : centroid of partition i



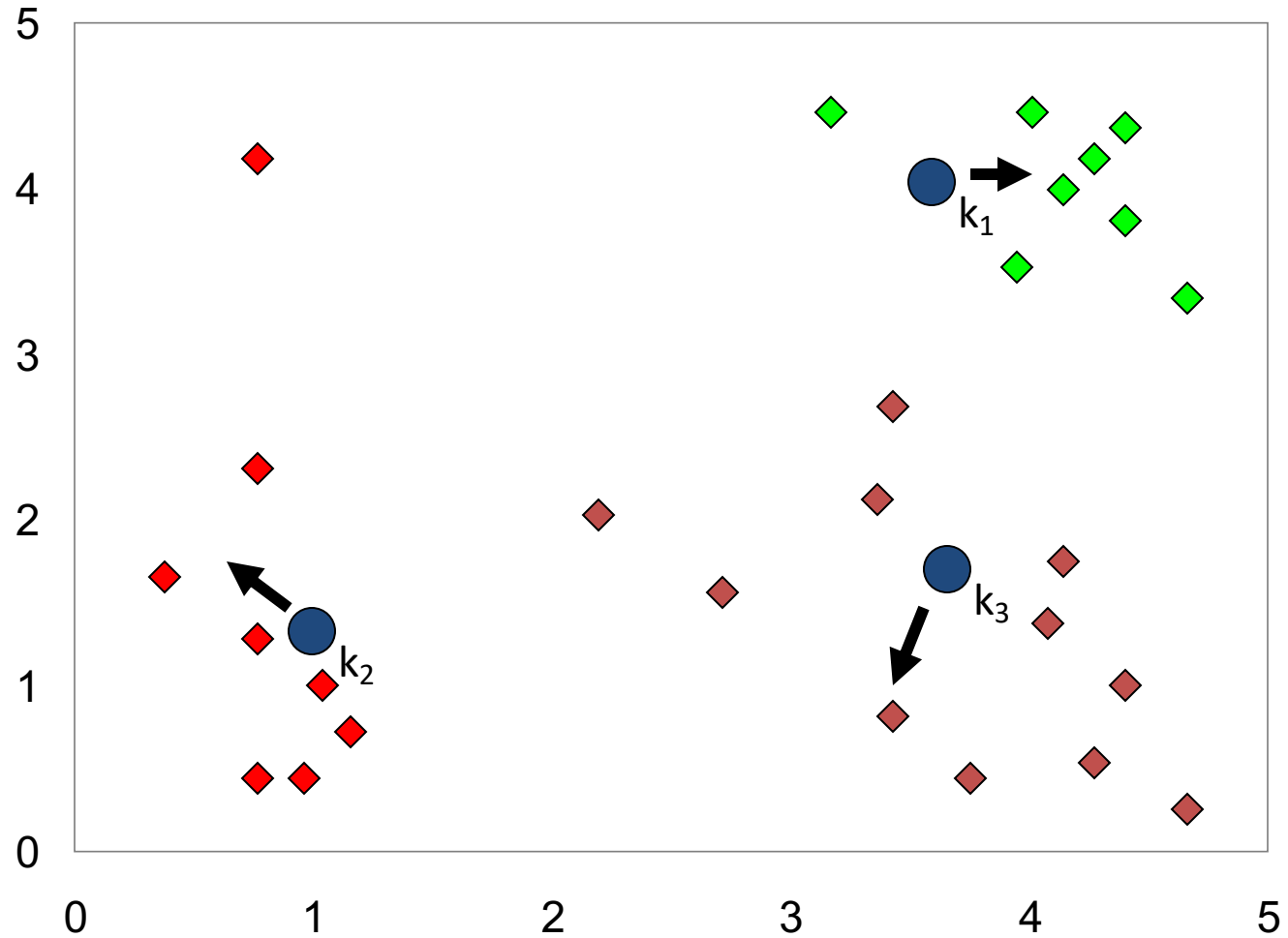
K-means Clustering: Step 3

Algorithm: k-means, distance measure : Euclidian distance, k_i : centroid of partition i



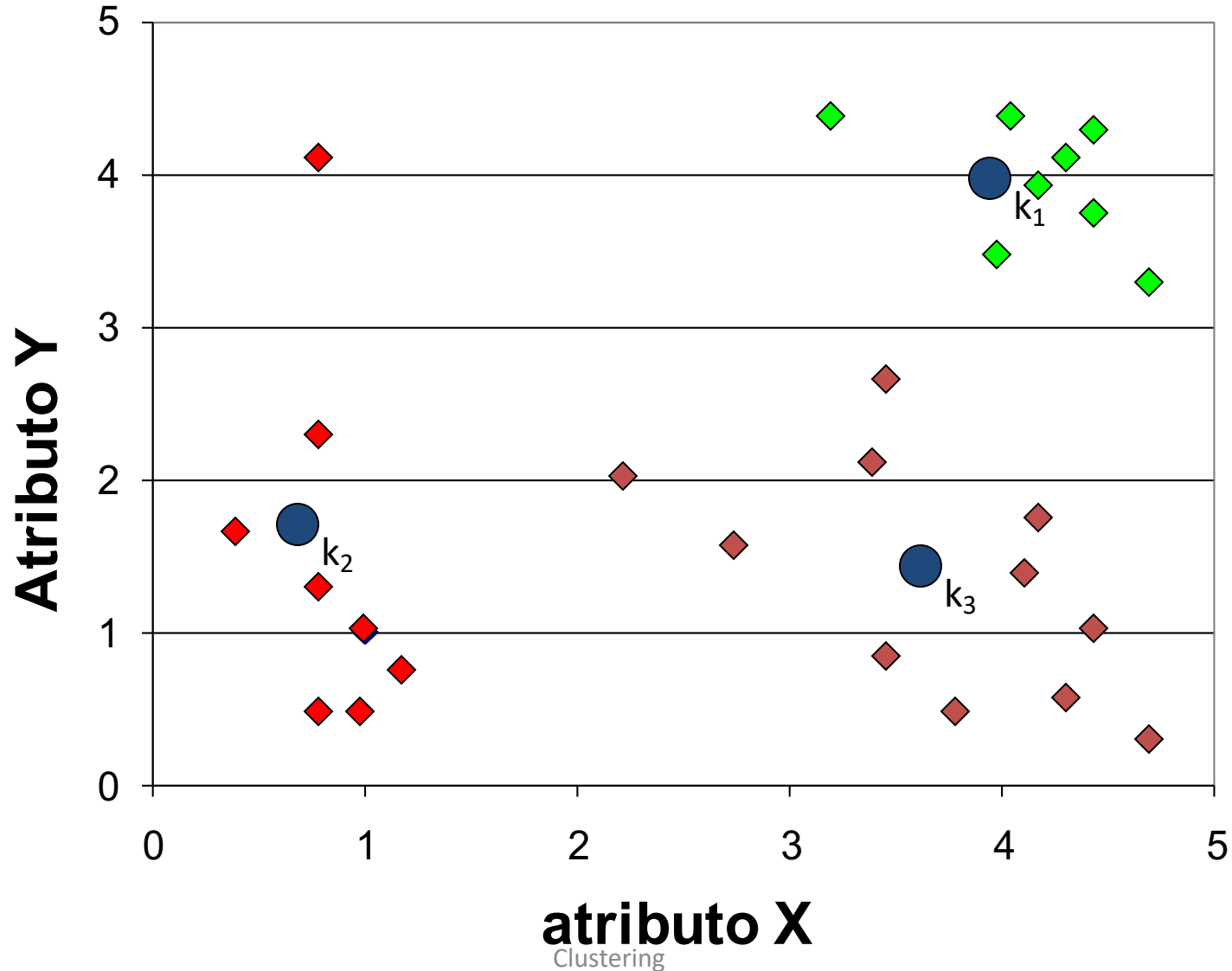
K-means Clustering: Step 4

Algorithm: k-means, distance measure : Euclidian distance, k_i : centroid of partition i



K-means Clustering: final state

Algorithm: k-means, distance measure : Euclidian distance, k_i : centroid of partition i



Comments on *k-Means*

- Pros

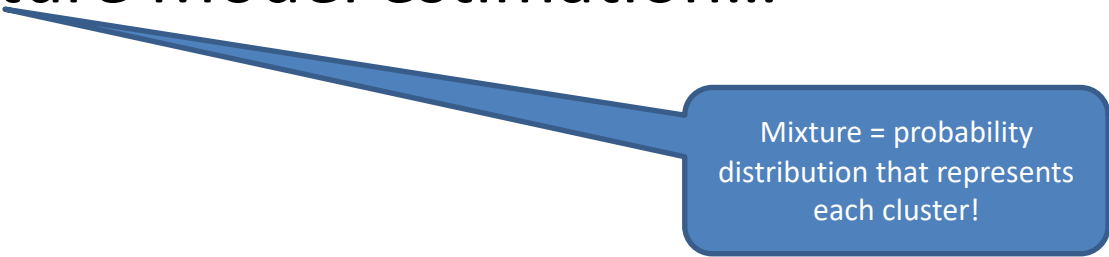
- *Efficient*: complexity $O(tkn)$, where n is the #objects, k is #clusters, and t the number of iterations. Typically, $k, t \ll n$.
- Frequently finish in a local optima. Global optima can be found using other techniques like: : *simulating annealing* and genetic algorithms.

- Weakness

- What about categorical data? No mean measure
- Algorithm requires a predefined parameter k (*number of clusters*),
- Cannot deal with noise and *outliers*,
- Has difficulty in identifying clusters with non convex shapes (*typically finds spherical shape clusters*).

Expectation-Maximization

- Extends k-means by considering the possibility of non rigid frontiers between clusters,
- The association between an object and a cluster is defined using a measure (weight) of membership probability ($O_i \in c_k$),
- Centroids are computed using these weight measures
- Gaussian Mixture Model estimation...



Mixture = probability distribution that represents each cluster!

Mixture Model

- Consider a dataset with a single attribute (numerical). Assume that clusters have a Gaussian distribution (with different parameters μ e σ).
- In this context, the clustering problem resumes to consider the training cases (without class), a parameter k and to compute the mean, standard deviation and probability distribution between clusters for each cluster i.e. for cluster A define μ_A , σ_A and $p(A)$.
- For $k = 2$: suppose that for a dataset where one has to compute the parameters μ_A , σ_A , μ_B , σ_B e $p(A)$ and where $p(A) + p(B) = 1$. This is the Mixture Model!
- Knowing each cluster distribution, it is easy to calculate the parameters (where $p(A)$ is obtained by the proportion in the dataset).

$$\mu = \frac{x_1 + x_2 + \dots + x_n}{n}$$

$$\sigma^2 = \frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + \dots + (x_n - \mu)^2}{n - 1}$$

Mixture Model

- Given a new case, computing the membership probability to cluster A is:

$$Pr(A/x) = \frac{Pr(x|A) \times Pr(A)}{Pr(x)} = \frac{f(x; \mu_A, \sigma_A) \times Pr(A)}{Pr(x)}$$

where $f(x; \sigma, \mu) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x - \mu)^2}{2\sigma^2}}$ represents the Gaussian distribution function.

- The final results is a not a set of clusters, rather it is the set of membership probabilities of each case x belonging to each cluster!
- The problem gets difficult because we do know neither the distribution of the training cases nor the 5 parameters of the *Mixture Model*.

Mixture Model

- One approach:
Use k-means and iterate. Assign random values to each parameters. Use them to compute the membership probabilities of each case to each cluster. Finally, use these probabilities to (re)estimate the parameters and then repeat. This is the essence of the EM clustering algorithm!!

- Being $w_i = \Pr(i \in A)$ for case i and cluster A :

$$\mu_A = \frac{w_1 x_1 + w_2 x_2 + \dots + w_n x_n}{w_1 + w_2 + \dots + w_n} \quad \sigma_A^2 = \frac{w_1 (x_1 - \mu_A)^2 + w_2 (x_2 - \mu_A)^2 + \dots + w_n (x_n - \mu_A)^2}{w_1 + w_2 + \dots + w_n}$$

- Compute *likelihood* through the probability of each case on each cluster to decide when to stop!
- In practice use *log-likelihood* of

$$\prod_i (\Pr(A) \Pr(x_i|A) + \Pr(B) \Pr(x_i|B))$$

EM algorithm (cont)

$$E = \sum_i \log\left(\sum_k \Pr(O_i \in C_k)\right) < \varepsilon$$

Stopping criterion:

The *log-likelihood* converges to an ε value.

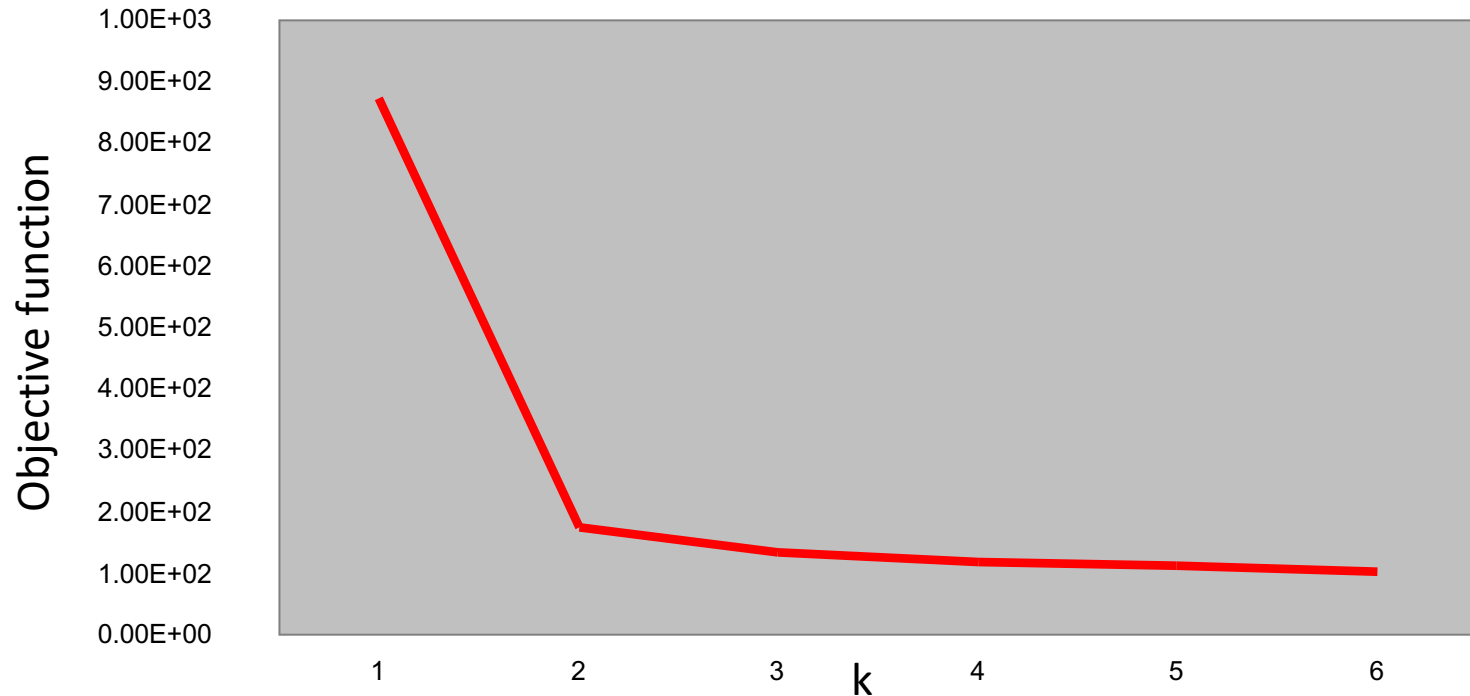
An *overall likelihood* measures the clustering quality.

Tends to increase along the number of iterations.

The process stops when the increment between iterations (ε) is negligible.

Using cross validation one can automatically derive the number k of clusters.

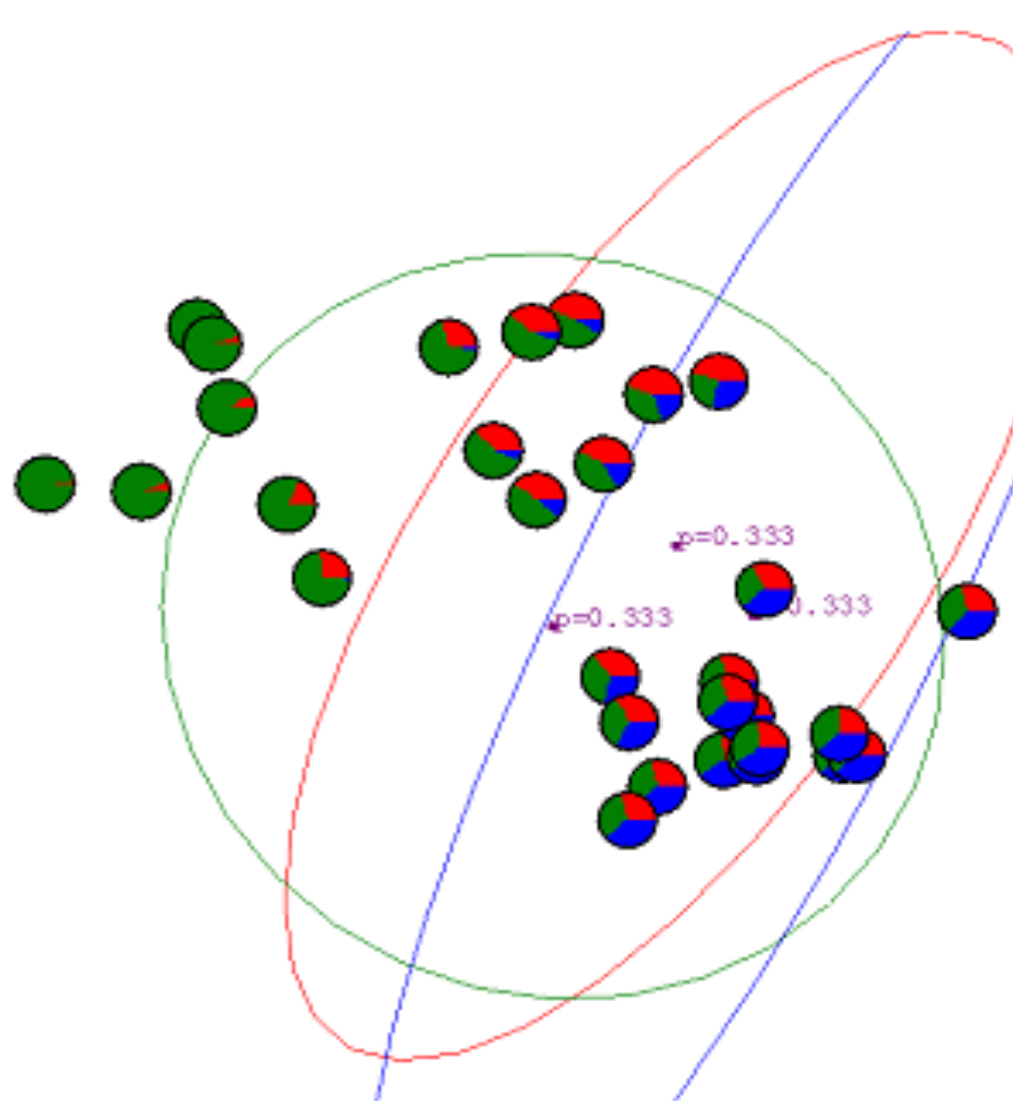
Find the number of clusters



The optimal number of cluster can be obtained by iterating over a objective function e.g. mean distance inside each cluster.

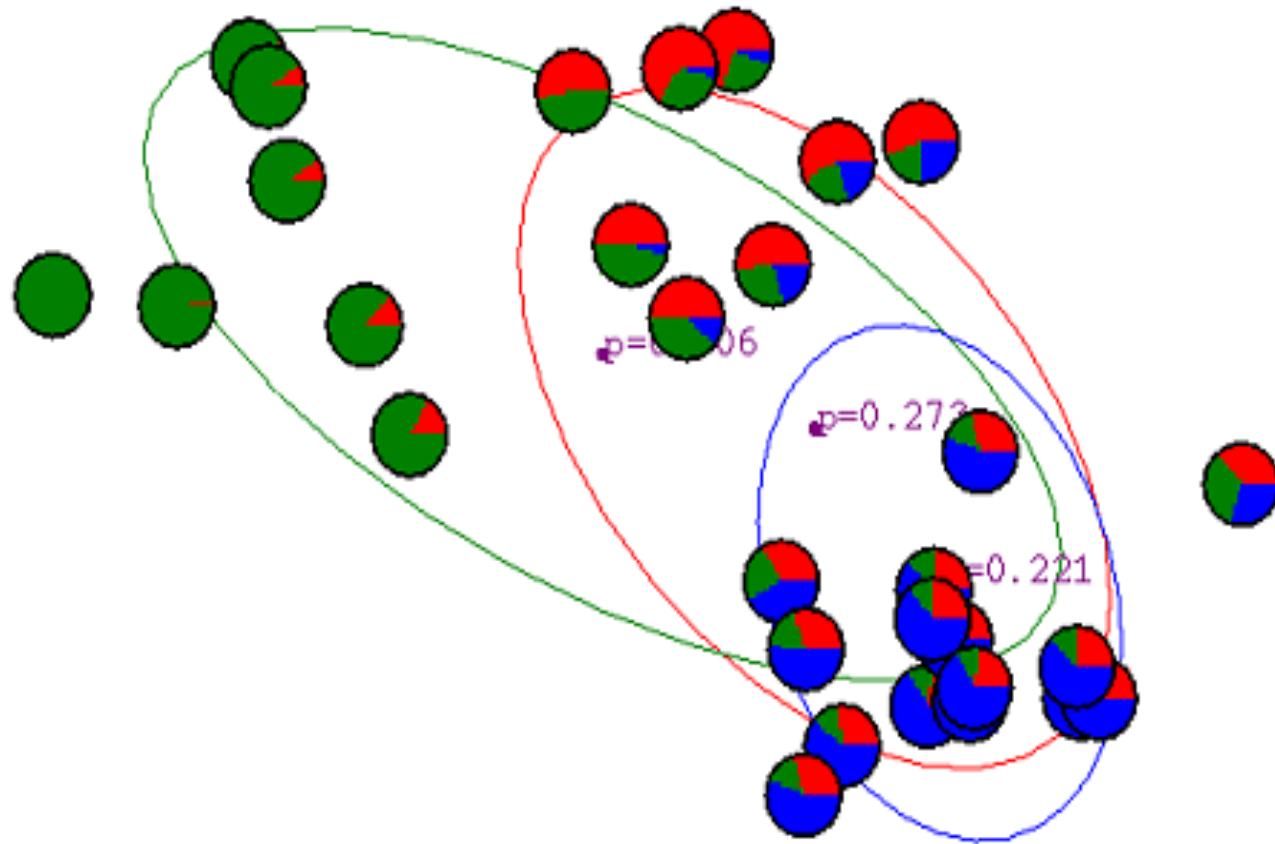
An abrupt change of values when $k = 2$ suggests two clusters in the data. This technique is known as “*knee finding*” or “*elbow finding*”.

Executing EM clustering



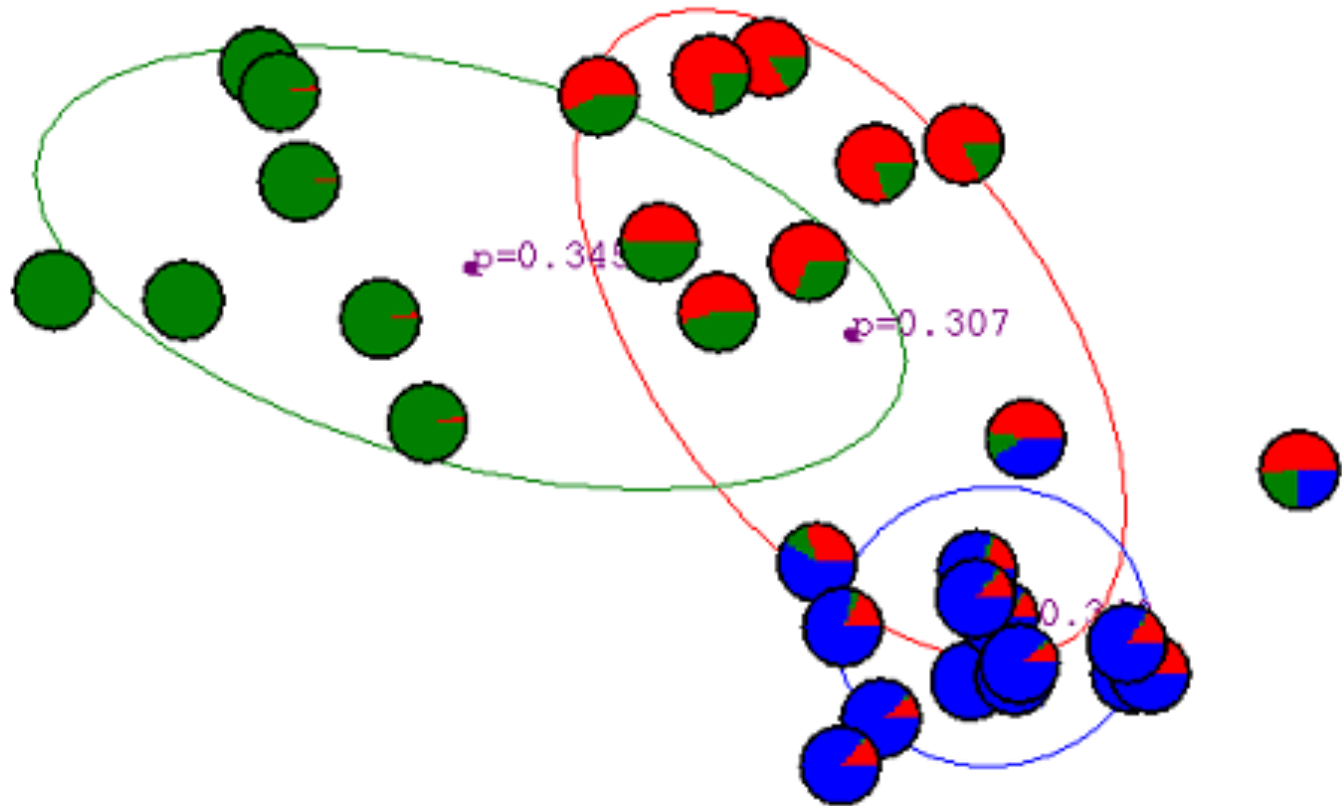
Initial model parameters.

Executing EM clustering



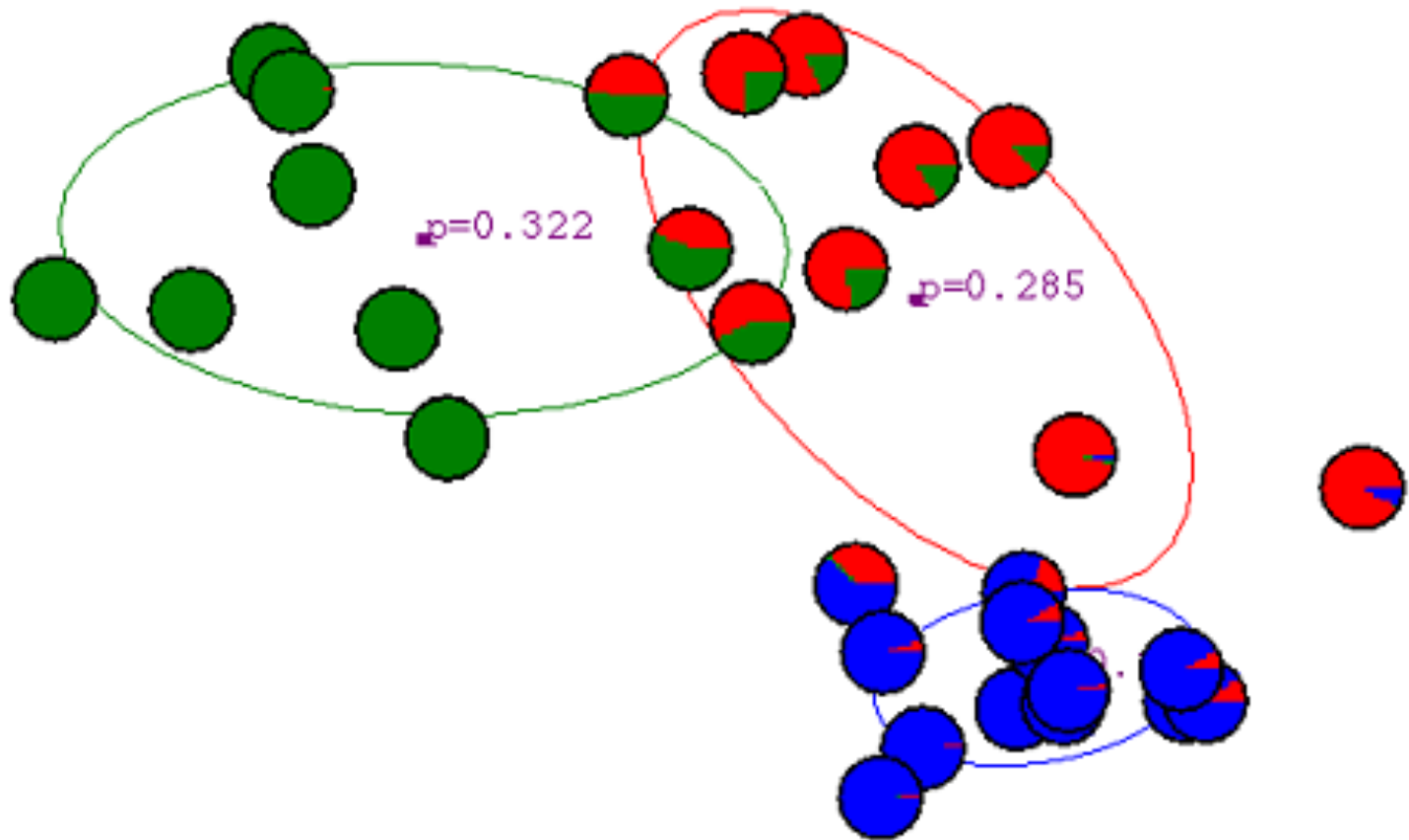
After first iteration

Executing EM clustering



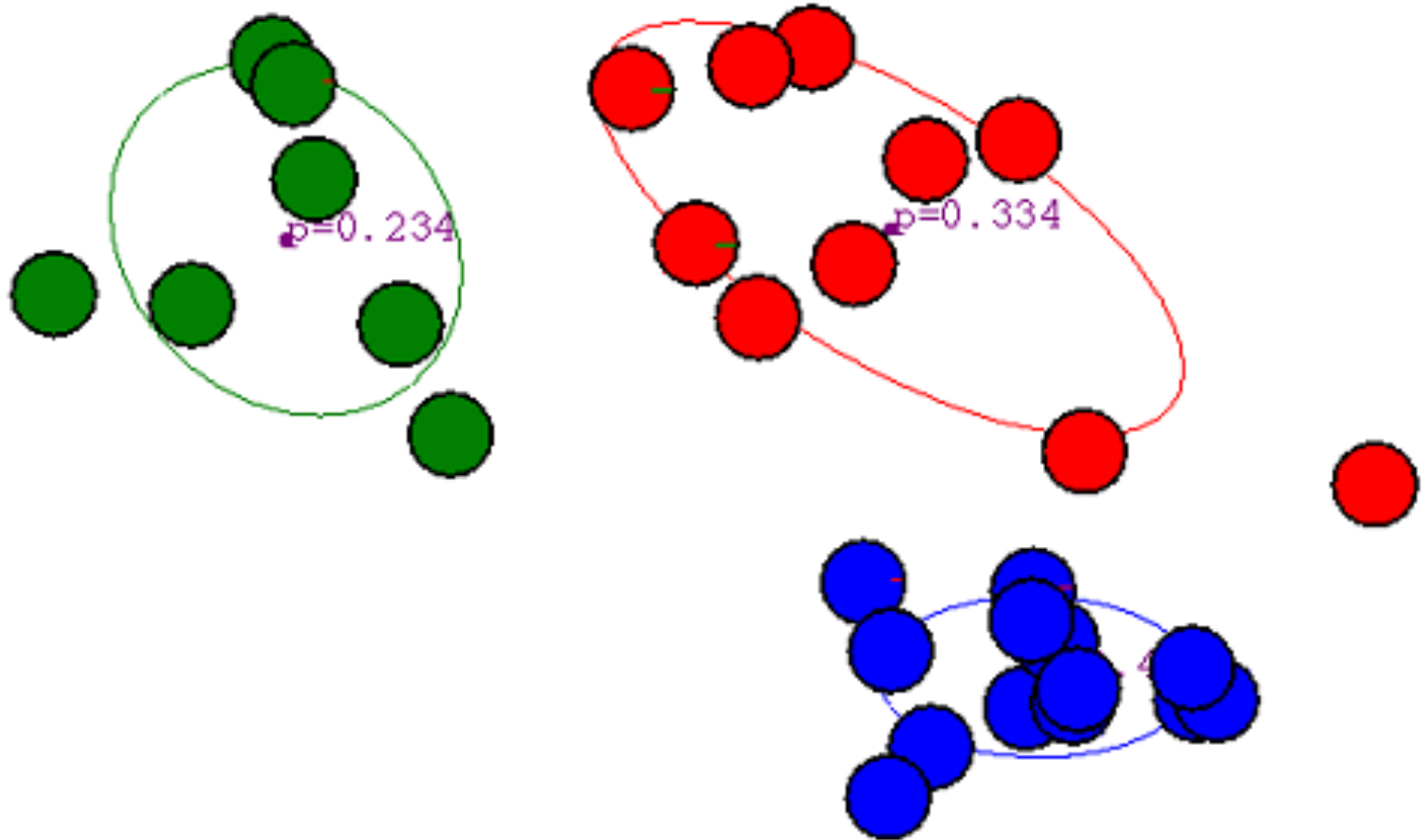
After third iteration

Executing EM clustering



After fifth iteration

Executing EM clustering



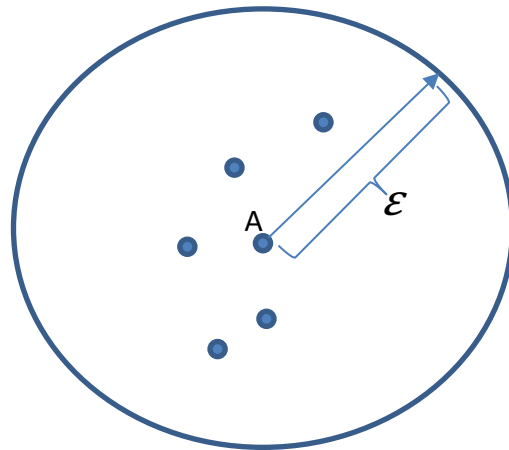
After convergence

Density Based Methods

- Clustering using density
 - The idea is to drive the process using density and neighbourhood. The process evolves through the localization of high density regions (clusters) that are well defined and separated from low density ones;
 - There is no need to define the number of clusters to be found!
 - DBSCAN is the best known algorithm.

Density: center-based approach

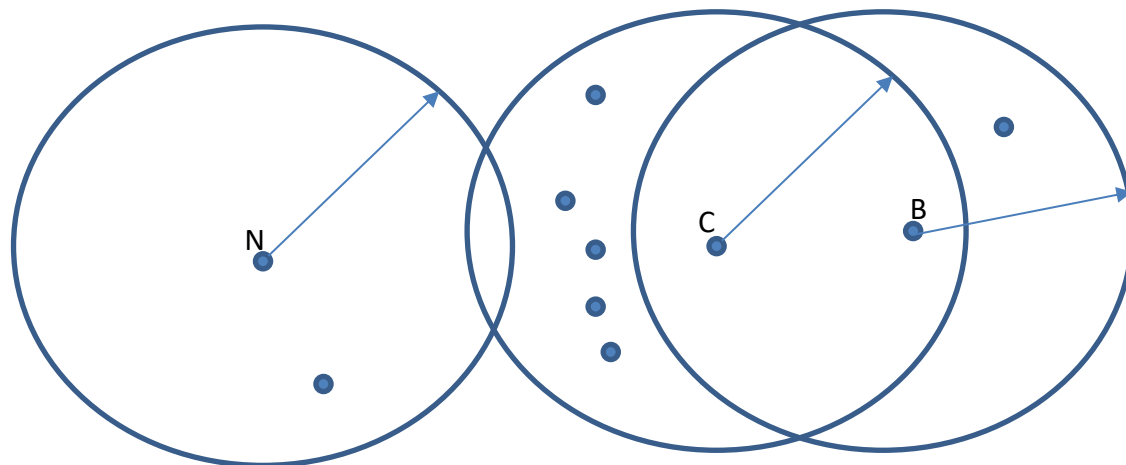
- Density for a given point (case) in the dataset is measured as the counting of points that distance a given range (radius) ϵ from that point.



- In this case density of A = 6 (5 neighbours + A)

Core, Border and Noise points

- The algorithm gets two input parameters: ε and *minpts*. That is, radius and minimal count.
- Classify the dataset points as:
 - Core: points that are part of a “density based cluster” i.e. where count (density) is $\geq \textit{minpts}$.
 - Border: non “core” points but that are in the neighbour of a “core”.
 - Noise: all the other points ...



C = Core
 B = Border
 N = Noise

DBSCAN algorithm (simplified)

Input($\epsilon, minpts$)

1. Classify all point in the dataset as Core, Border or Noise.
2. Delete all Noise points.
3. Establish a connection between Core points that distance less than ϵ between them.
4. Build a separated cluster out of the group of all interconnected points. (that are connected to a Core point).
5. Assign each Border point to a cluster of one of the associated Core points.

DBSCAN algorithm

Input: dataset, distfunct, ϵ and *minpts*

```
C = 0 // cluster counter
for each point p in dataset
{  If label(p) == undefined // just for "unlabeled" points!
  {  Neighbours = findN(distfunct, p,  $\epsilon$ ) // find neighbours
    if |Neighbours| < minpts
      label(p) = Noise
    else
      {  C++
        label(p) = C
        SeedSet = Neighbours \ {p}
        for each point q in SeedSet
          {  if label(q) == Noise
              label(q) = C // change Noise to Border point!
            if label(q) == undefined
              {  label(q) = C // label neighbour
                Neighbours = findN(distfunct, q,  $\epsilon$ )
                if |Neighbours|  $\geq$  minpts // density check
                  SeedSet = SeedSet U Neighbours
              }
            }
          }
        }
      }
    }
  }
}
```

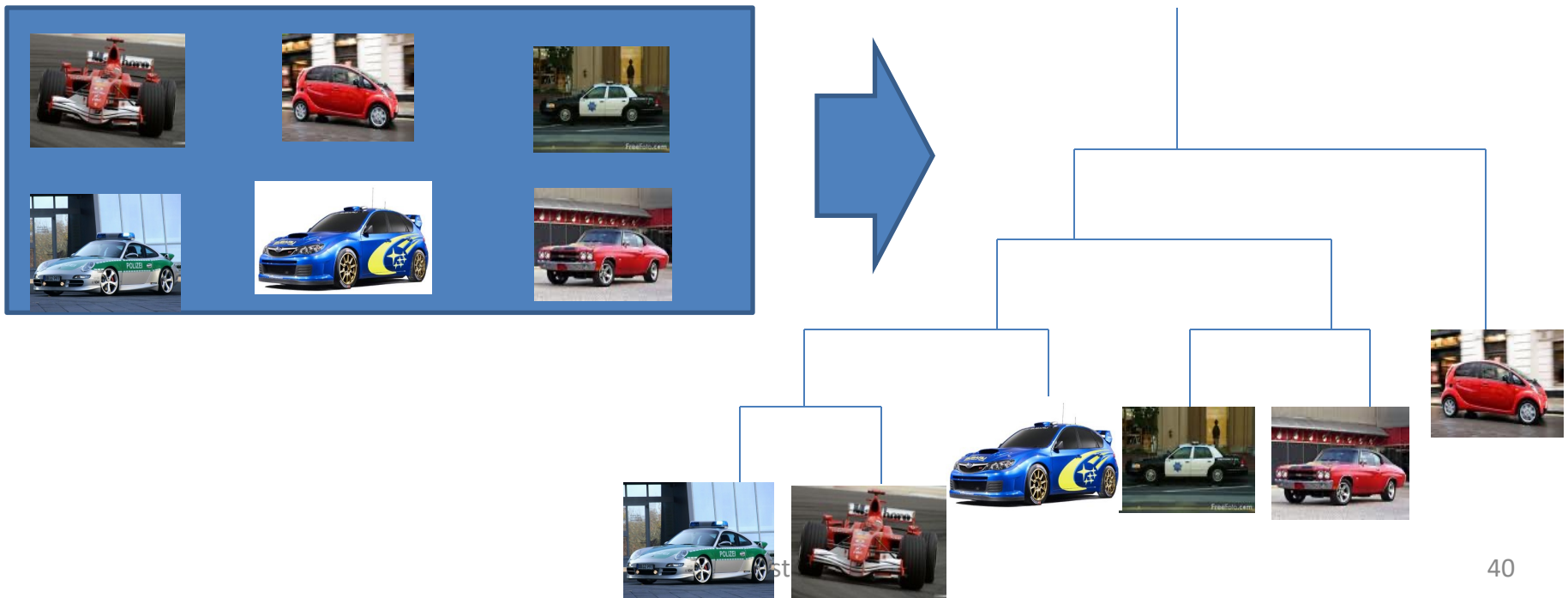
```
function findN(df,q,eps)
{ N = { }
  for each p in dataset
    if df(q,p)  $\leq$  eps
      N = N U {p}
  return N
}
```

Comments

- The algorithm is quadratic – in the worst case is $O(m^2)$, where $m = \#dataset$. However, there are data structures e.g. kd-trees, that yield implementation with $O(m \times \log(m))$.
- It is incomplete! Noise are ignored...
- The choice of ε is crucial. Very high values imply a number of clusters as the number of cases in the dataset. Low values imply density = 1 for all points.
- Similar situations for *minpts*...
- There are strategies to automatically derive these parameters.

Hierarchical Clustering

- To create an hierarchical decomposition of objects in a dataset following a specific criterion (similarity measure)

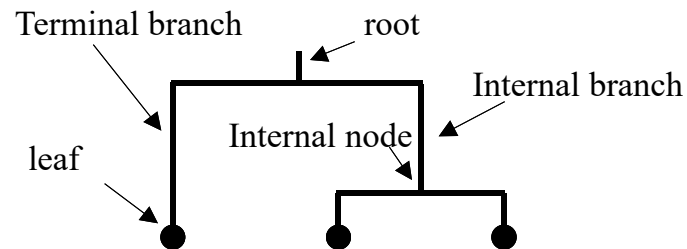
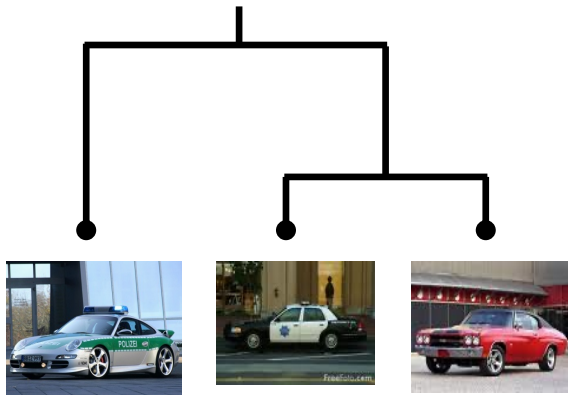


A structure to summarize similarity measures

Definition of a *dendrogram*:

A structure to evaluate/discriminate cases from a dataset.

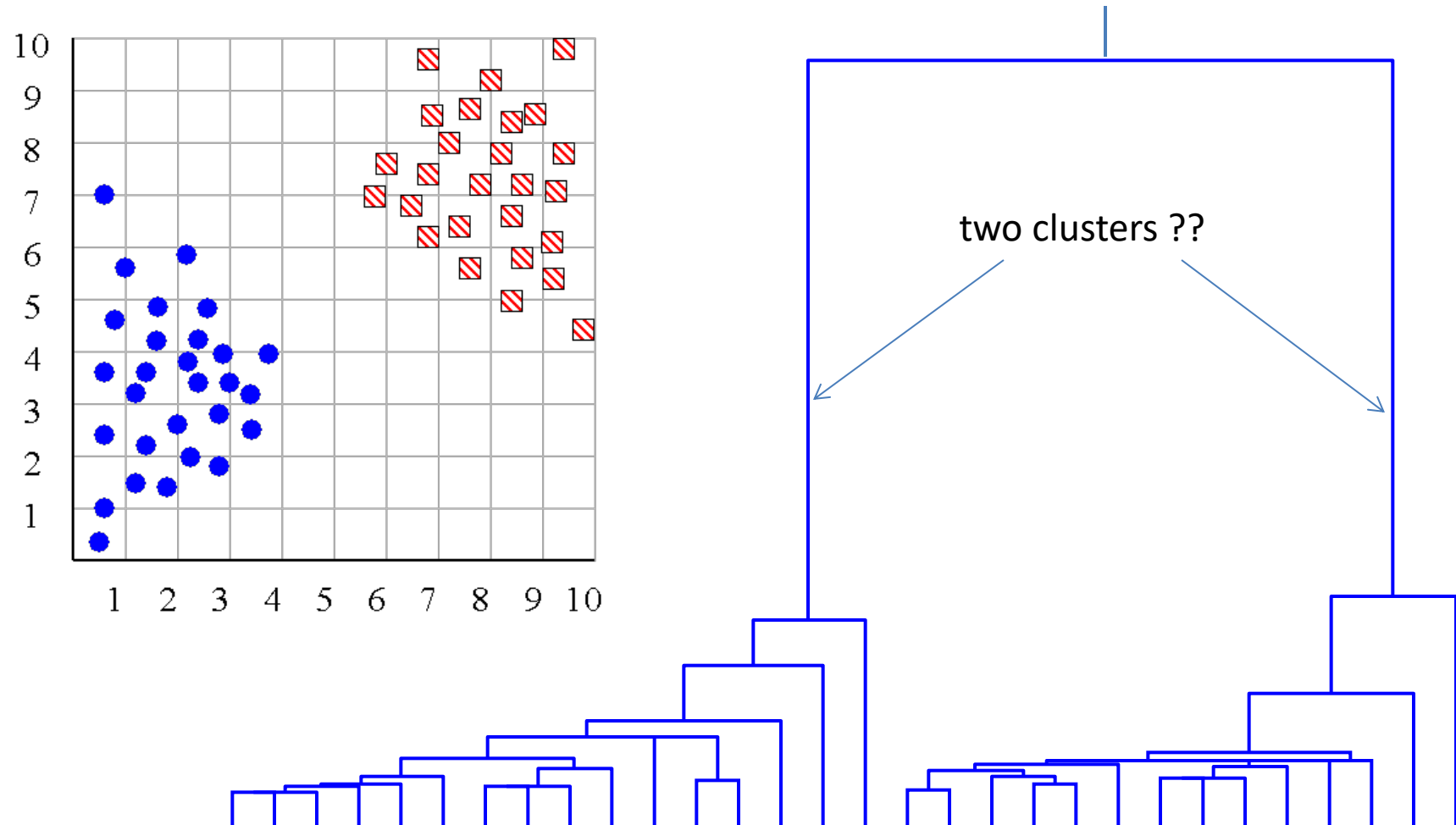
Known as phylogenetic trees in biology



Similarity between two objects in a *dendrogram* is represented by the height of the lowest internal node that both objects share.

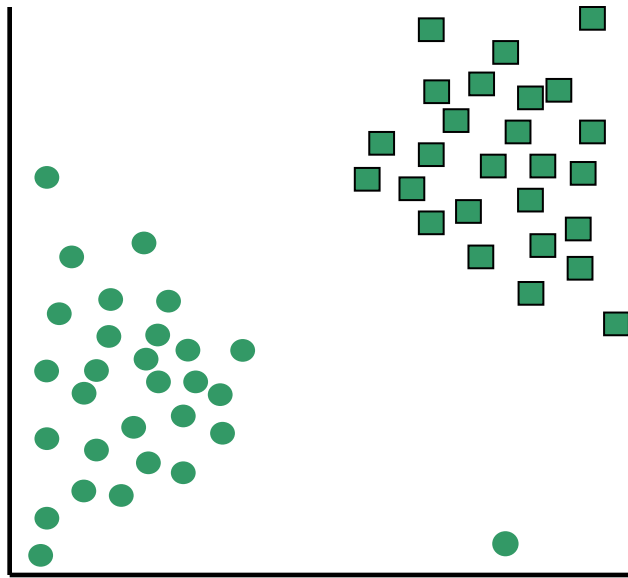
HC for defining the number of partitions

The dendrogram gives ways to find the “correct” number of clusters. As happens in this example, two sub-trees highly separated suggest two clusters.

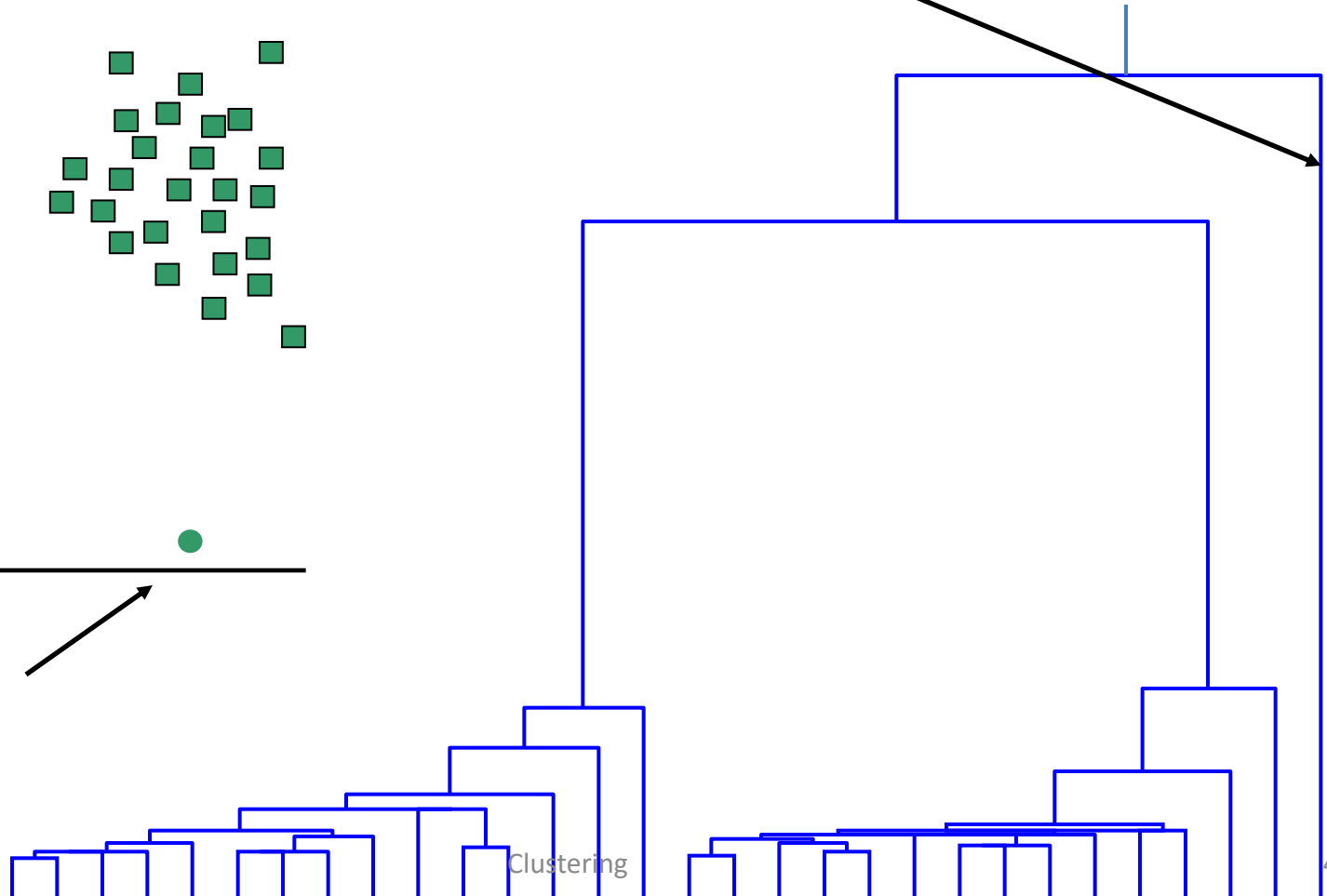


Outliers detection

This isolated branch suggest an object from the dataset that is very different from all the other objects



Outlier



Clustering

Types of Hierarchical Clustering

Like in decision tree building, it is not feasible to test all dendrograms. Heuristics have to be used to find the “best” structure.

It can be done using:

Number of possible dendrograms with n leaves = $(2n - 3)! / [(2^{(n-2)}) (n - 2)!]$

Leaves	Dendrograms
2	1
3	3
4	15
5	105
...	...
10	34,459,425

Bottom-Up (agglomeration): begin with n clusters (1 obj = 1 cluster). Find the best pair of objects to join in a new cluster. Repeat until all clusters are agglomerated into a single cluster.











Top-Down (division): begin with all objects in the same cluster. Consider all partitions that divide the cluster in two. Pick the best partition and recursively apply the same procedure to both sub-clusters yield by that partition.

Similarity Matrix

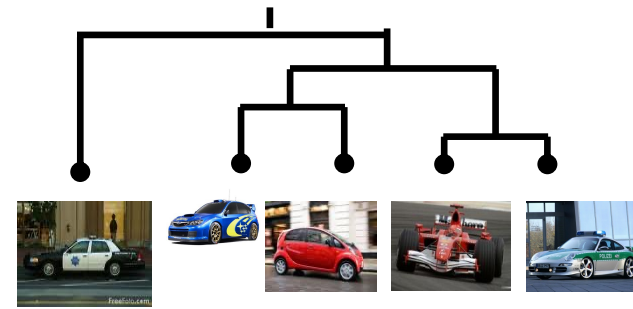
The process begins by defining a similarity matrix with distances between all objects of the dataset. Along the execution, this matrix is rebuilt with the introduction of new clusters and the deletion of already joint ones.

$$d(\text{red car}, \text{red car}) = 8$$

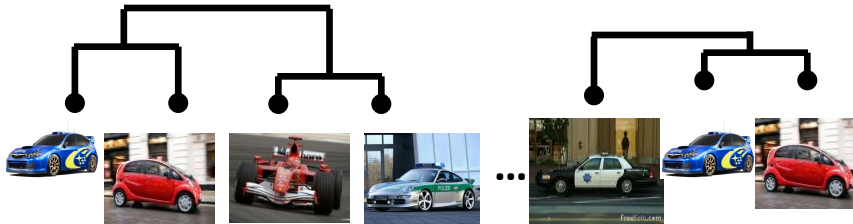
$$d(\text{green car}, \text{blue car}) = 1$$

				
0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0
				

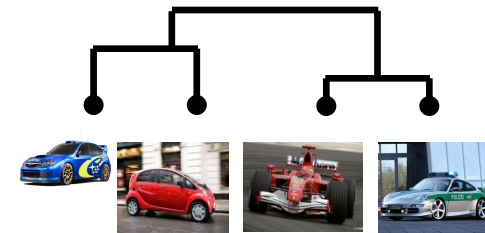
Bottom-Up (agglomeration):



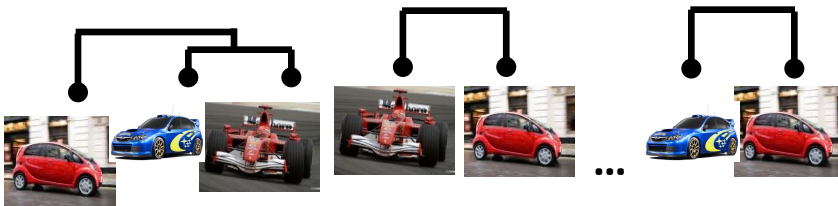
All possible joins...



Choose the best



All possible joins...



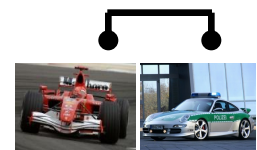
Choose the best



All possible pairs...



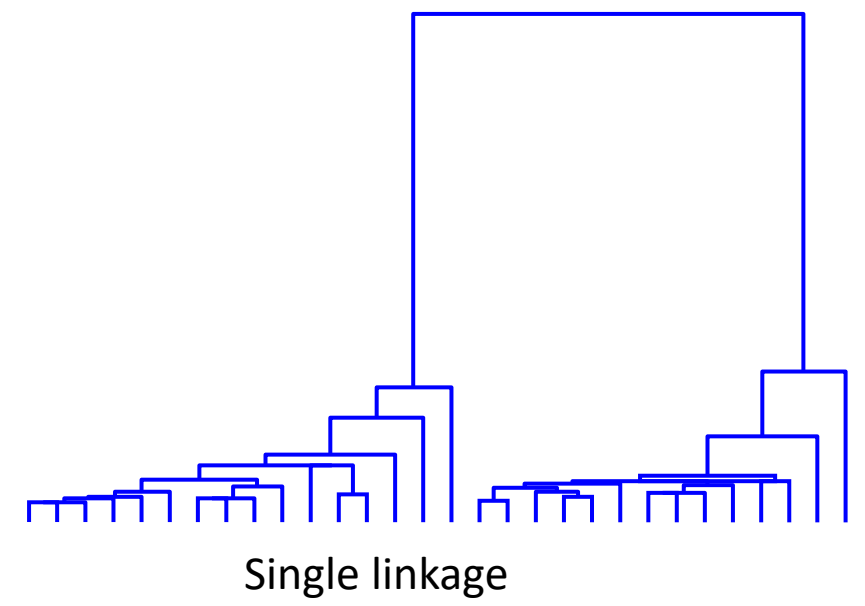
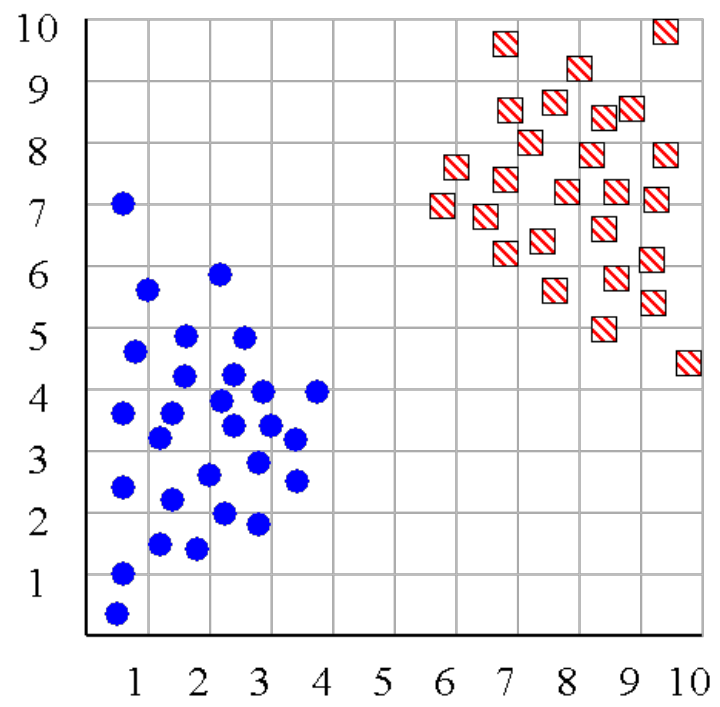
Choose the best



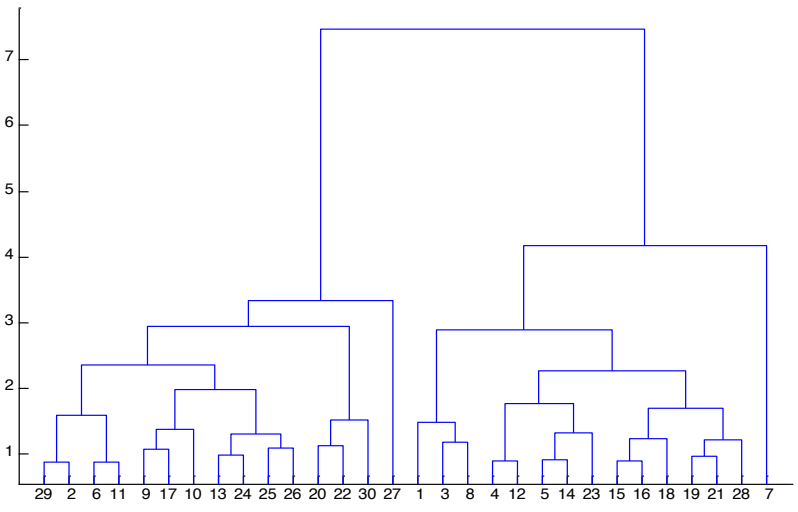
How to define the distance between an object and a cluster or the distance between clusters ?

- **Methods:**

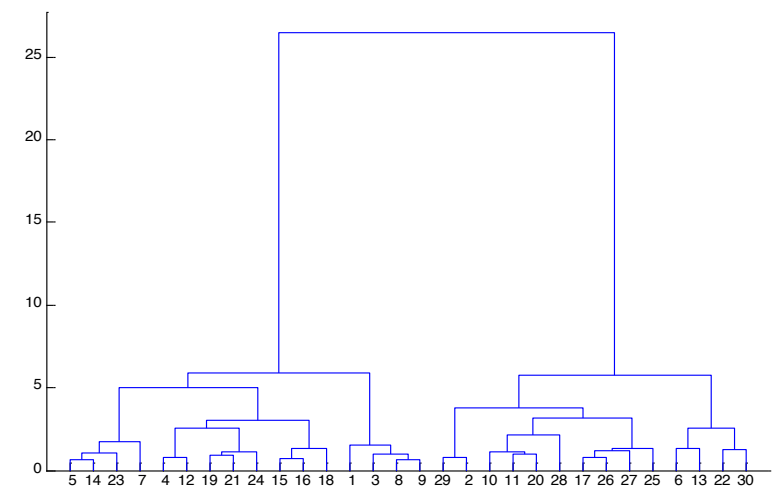
- **Single linkage (nearest neighbour):** distance between two clusters is measure through the distance between the two closest objects in both clusters (nearest neighbours).
- **Complete linkage (furthest neighbour):** distance obtained using the two furthest objects in both clusters (i.e., by the "furthest neighbours").
- **Group average linkage:** distance obtained using the mean distance between all pairs of objects from the two clusters.
- **Wards Linkage:** distance is computed by minimizing the variance from the cluster obtained from joining two clusters



Single linkage

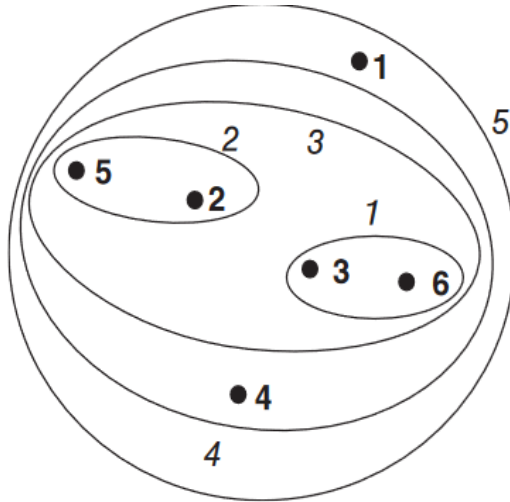


Average linkage

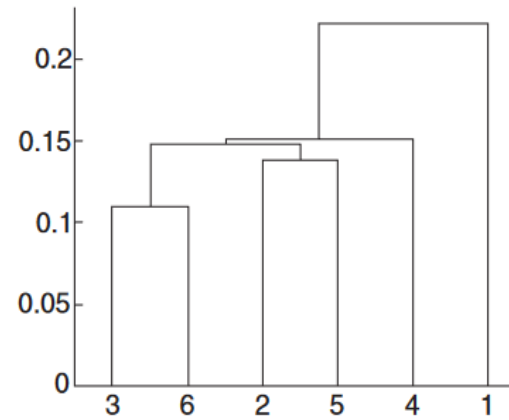


Wards linkage

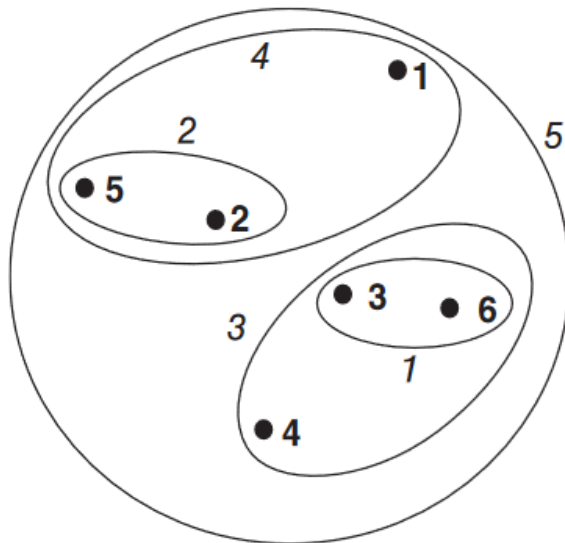
Distance and Dendrograms



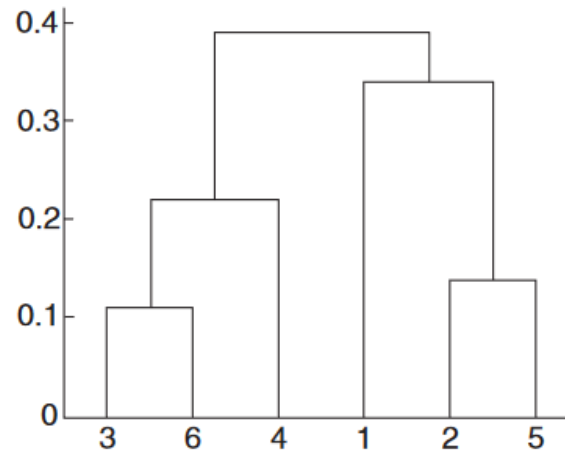
(a) Single link clustering.



(b) Single link dendrogram.

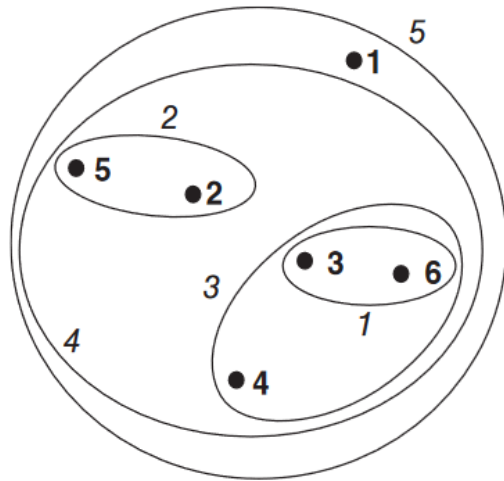


(a) Complete link clustering.

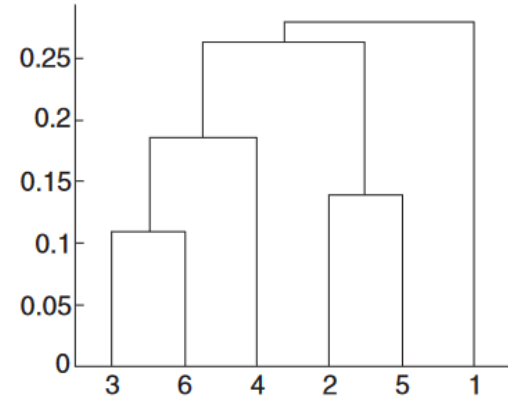


(b) Complete link dendrogram.

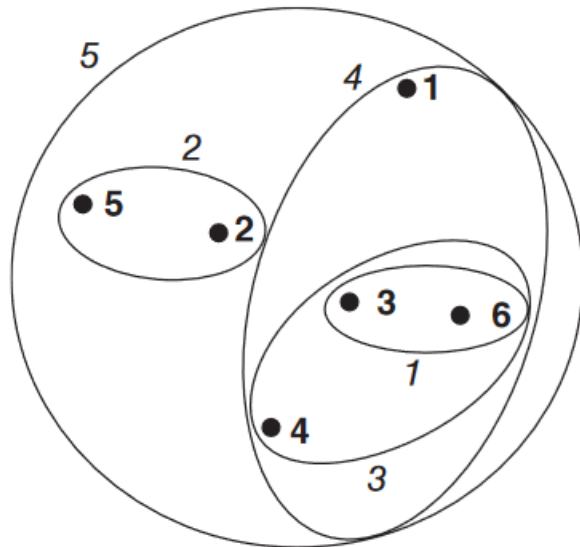
Distance and Dendrograms



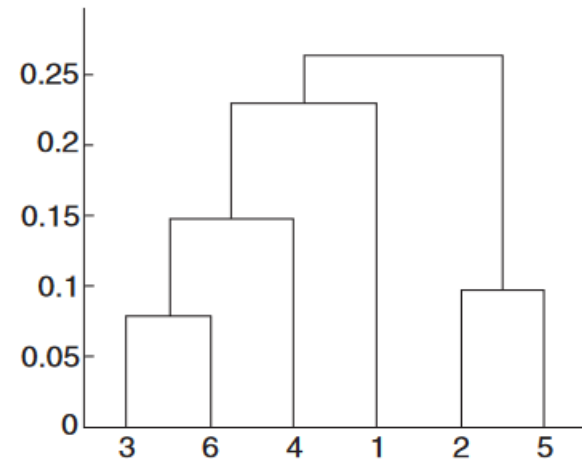
(a) Group average clustering.



(b) Group average dendrogram.



(a) Ward's clustering.



(b) Ward's dendrogram.

Hierarchical Methods: Summary

- No need to define the number k of clusters.
- There is a natural intuition to interpret hierarchies (at least in some contexts)
- Scales bad: complexity (time) is at least $O(n^2)$, being n the number of objects.
- As all heuristics methods, there are local maximum problems!
- Interpreting the results can be quite subjective...

Summary

- Natural groups in the data,
- Types of Clustering,
- Partitions and hierarchies,
- k-means and EM,
- Similarity measures,
- Density based methods
- Dendrograms
- Building dendrograms for hierarchical clustering.