An introduction to data clustering

Jean-Karim Hériché
EMBL

18 May 2012

What is clustering ?

The goal of clustering is to organise data by finding some 'sensible' grouping of the data items.

Clustering is unsupervised learning because it doesn't use predefined category labels associated with data items.

Clustering algorithms are engineered to find structure in the current data, not to categorise future data.

Why do clustering ?

- To get an idea of the nature or structure of the data:
  help with hypothesis generation or anomaly detection

- To categorise data when labelling is too costly

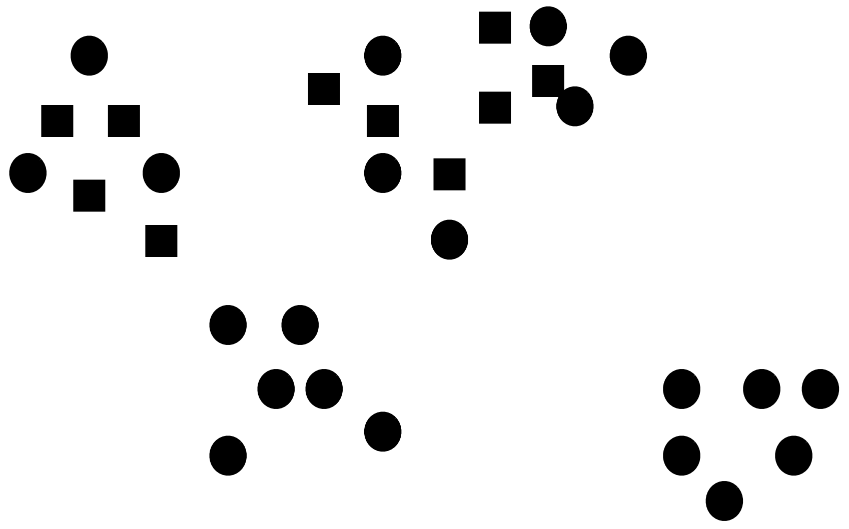
- To compress data:
  objects can be represented by a cluster prototype

What is a cluster ?

A cluster is a set of objects that are similar,
objects from different clusters are not similar.

What is similarity ?
How to measure it ?

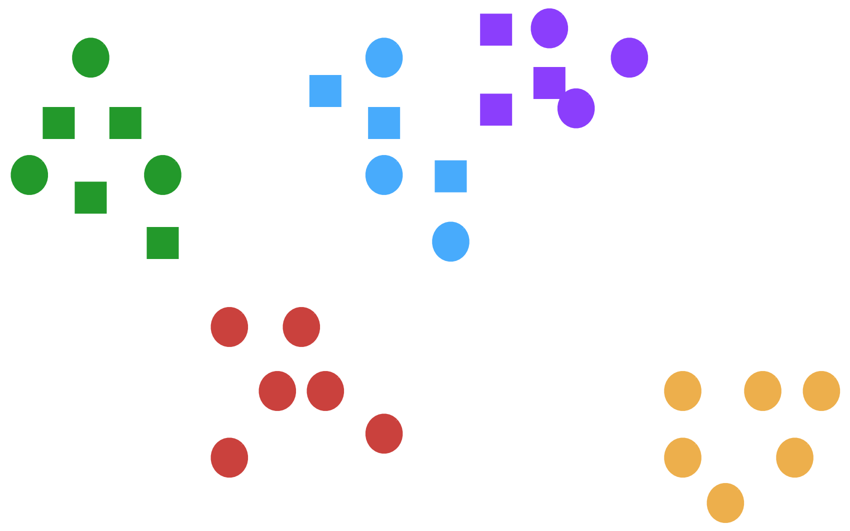
How many clusters ?



EMBL

How many clusters ?

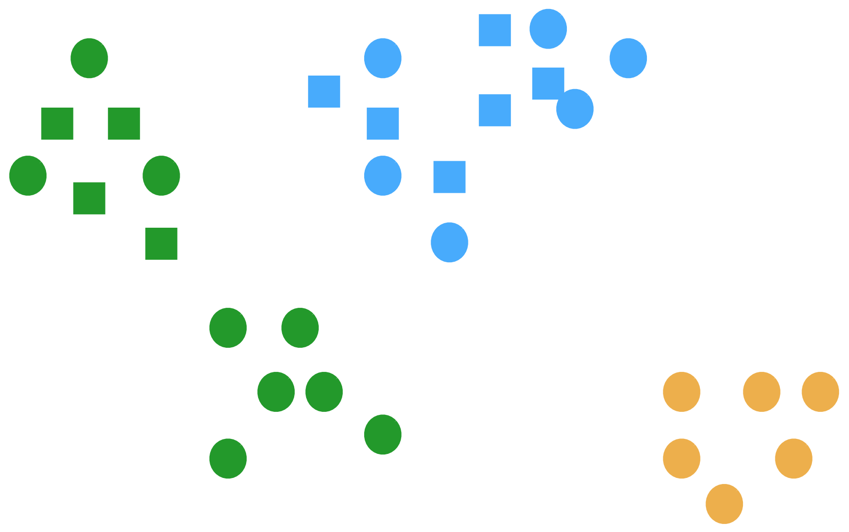
5 ?



EMBL

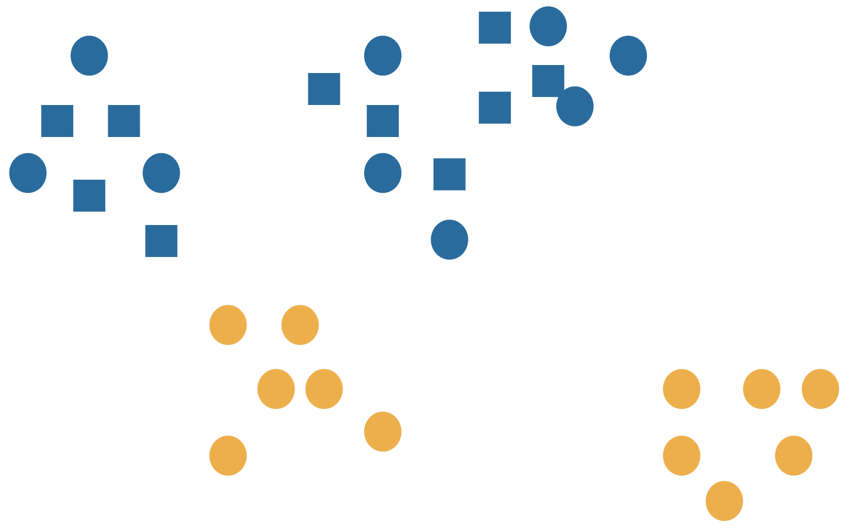
How many clusters ?

3 ?



How many clusters ?

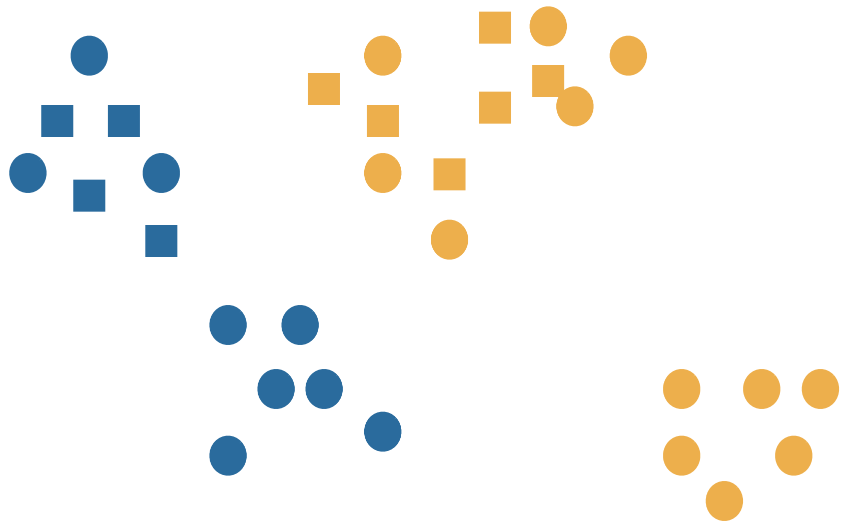
2 ?



EMBL

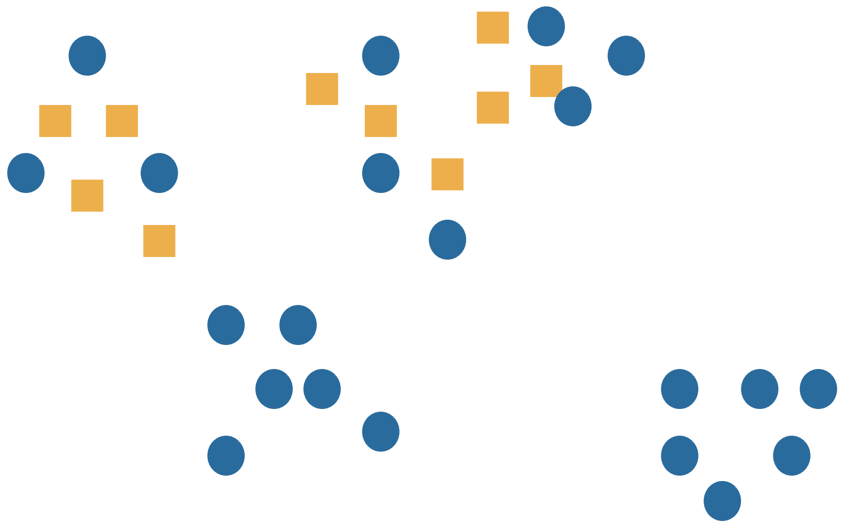
How many clusters ?

2 ?

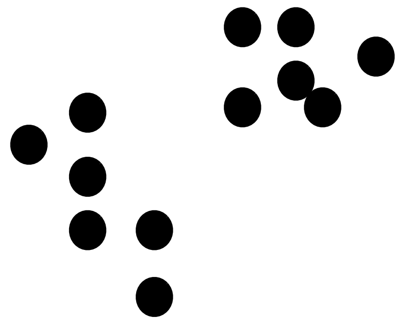


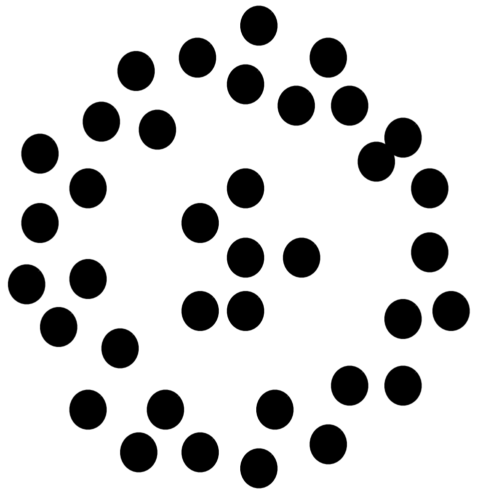
How many clusters ?

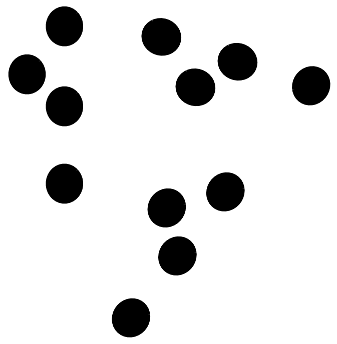
2 ?

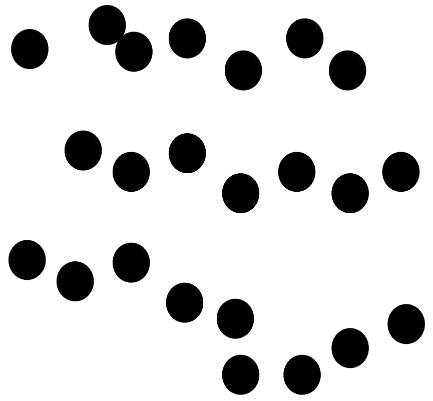


How many clusters ?



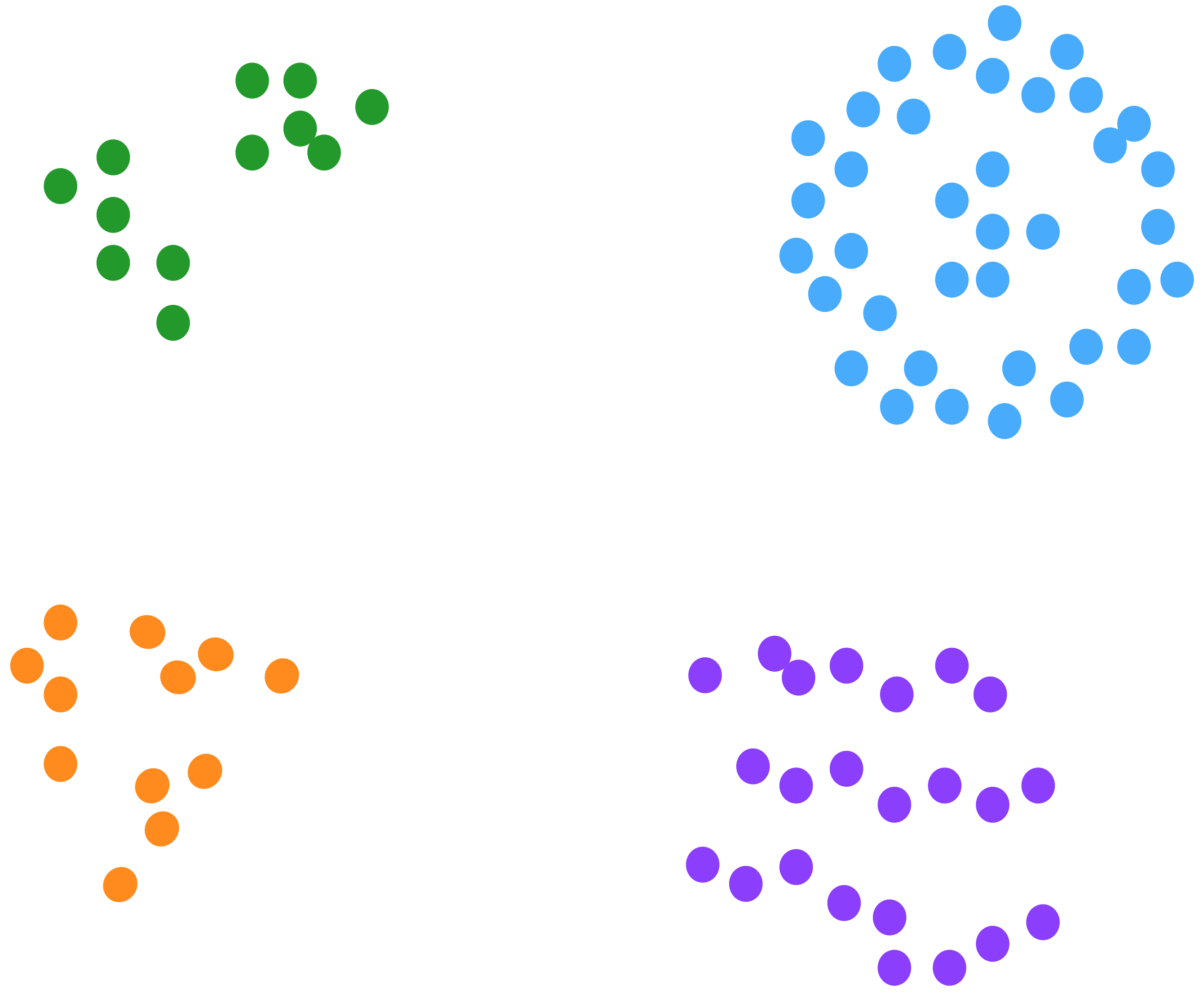






How many clusters ?

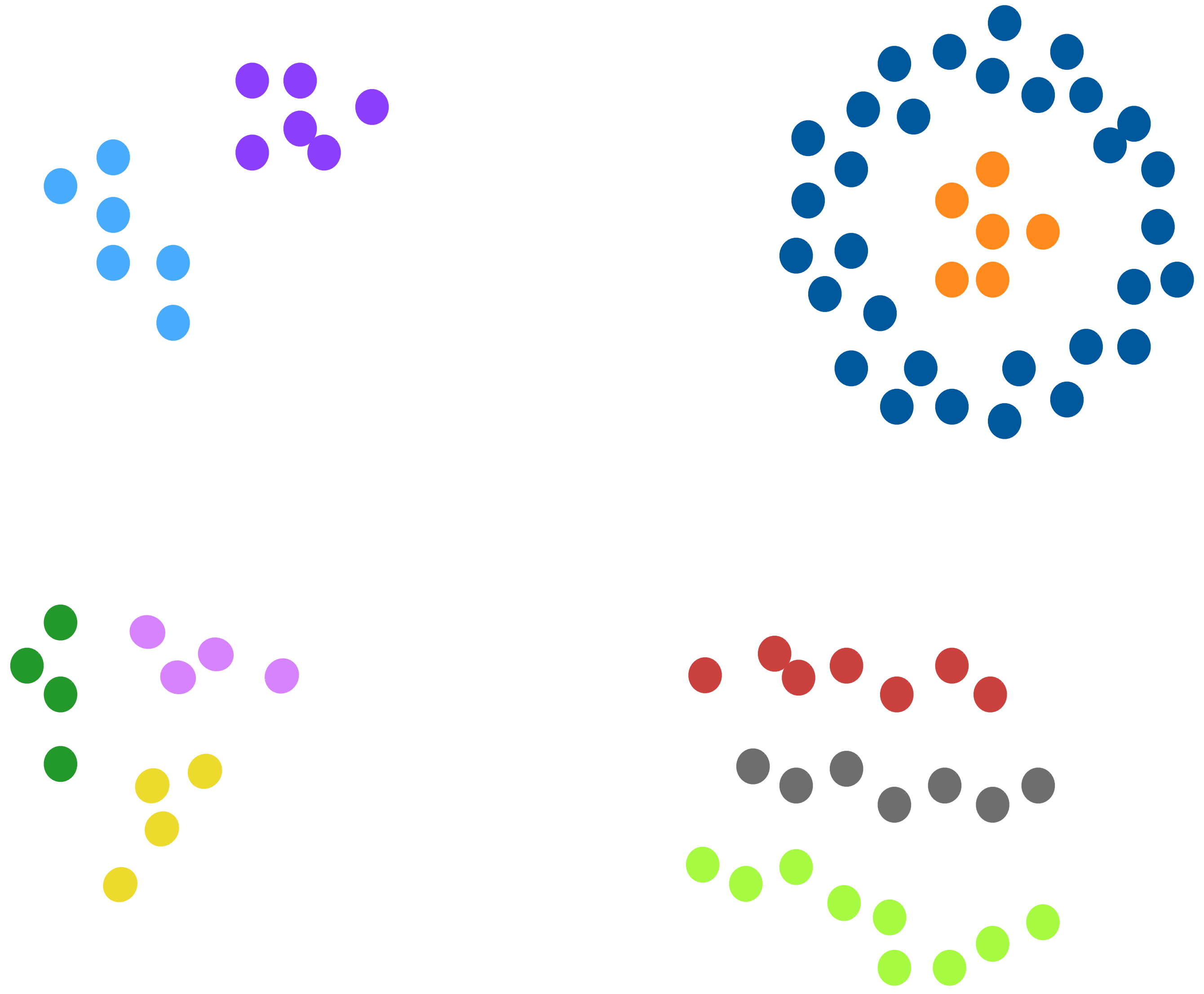
4 ?



EMBL

How many clusters ?

10 ?



Problems

- The result depends on the definition of similarity.

- There may not be an obvious measure of similarity.

- There is often no single correct result of clustering: results are difficult to evaluate

- Algorithms may find clusters even when there is no natural clusters in the data

Requirements

- scalability
- dealing with different types of attributes
- finding clusters with arbitrary shape
- minimal need for domain knowledge
- dealing with noise and outliers
- dealing with high dimensionality
- insensitivity to input order
- usefulness and interpretability

Clustering algorithms only satisfy these requirements to various degrees

Measures of similarity

Similarity is the converse of distance (i.e. a small distance means a high similarity and vice versa).

Some functions can transform the one into the other:

e.g. $s = 1 - d/\max(d)$ or $s = e^{-(d/\sigma)}$

Different measures exist for different data types.

Measures of similarity for real-valued vectors

$A = (x_1,x_2,\ldots,x_n)$
$B = (y_1,y_2,\ldots,y_n)$

- Euclidean distance:

$$d(A,B) = [(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots]^{½}$$

- Inner product:

$$s(A,B) = \langle A,B \rangle = x_1y_1 + x_2x_2 + \ldots$$

and their derivatives.

Measures of similarity for binary vectors

- Simple matching coefficient:
  fraction of bits that agree

- Tanimoto/Jaccard's/Dice’s coefficient:
  fraction of 1s that agree
  (different normalisations)

- Cosine similarity

Measures of similarity for non-vectorial data

- Strings:
  Alignment scores

- Time series:
  Dynamic time warping

- Graphs:
  Kernels

Metrics

Most algorithms assume use of a metric (or a similarity measure equivalent to a metric).

A distance is a metric if it has the following properties:

- identity of undiscernibles:
  $d(x,y) = 0$ if and only if $x=y$
- symmetry:
  $d(x,y) = d(y,x)$
- triangle inequality:
  $d(x,z) \leq d(x,y) + d(y,z)$

The curse of dimensionality

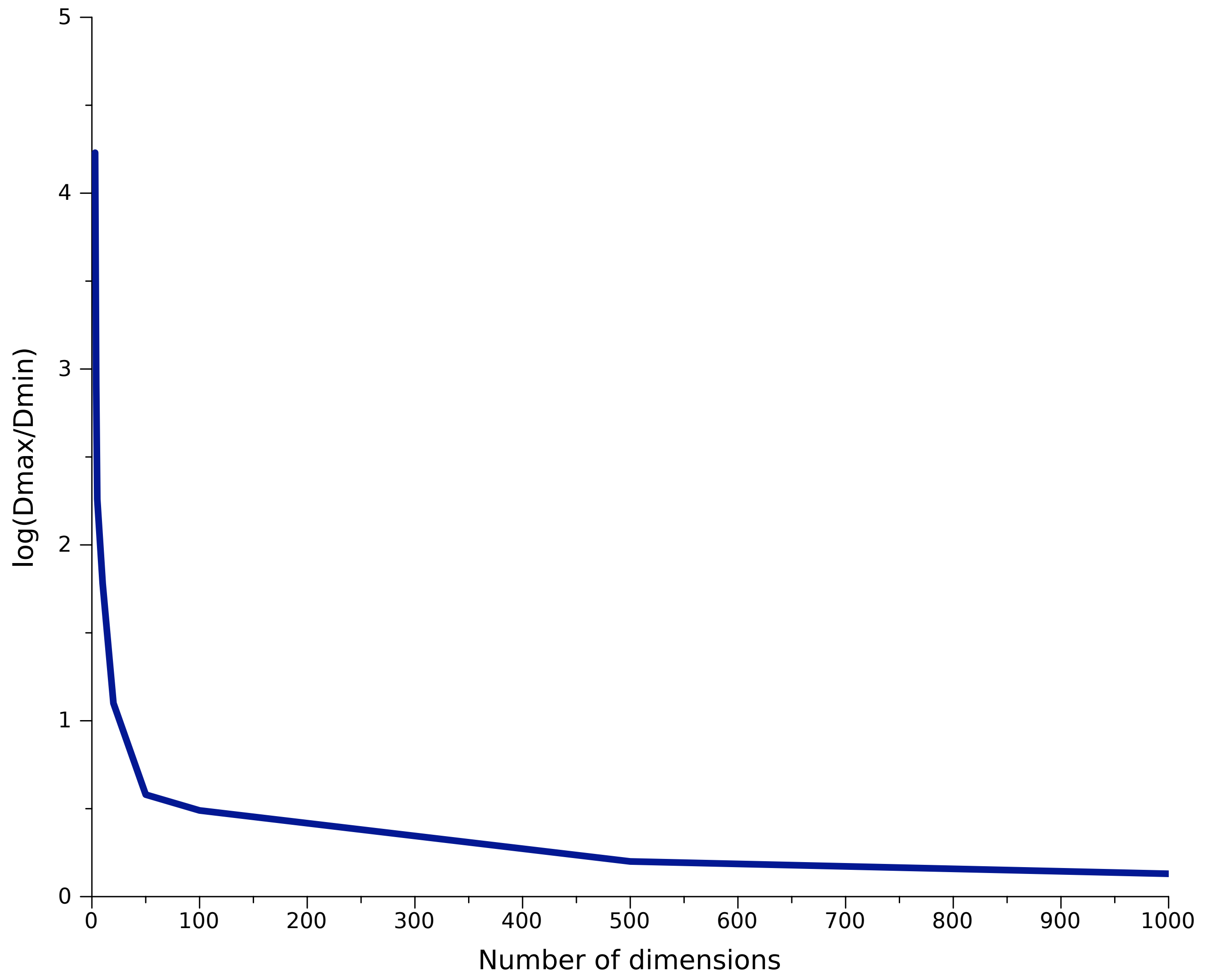
As the number of dimensions increases:

- the spatial density of points decreases leading to all points being equidistant (at $n \rightarrow \infty$).

- some points may tend to become the nearest neighbours of many points.

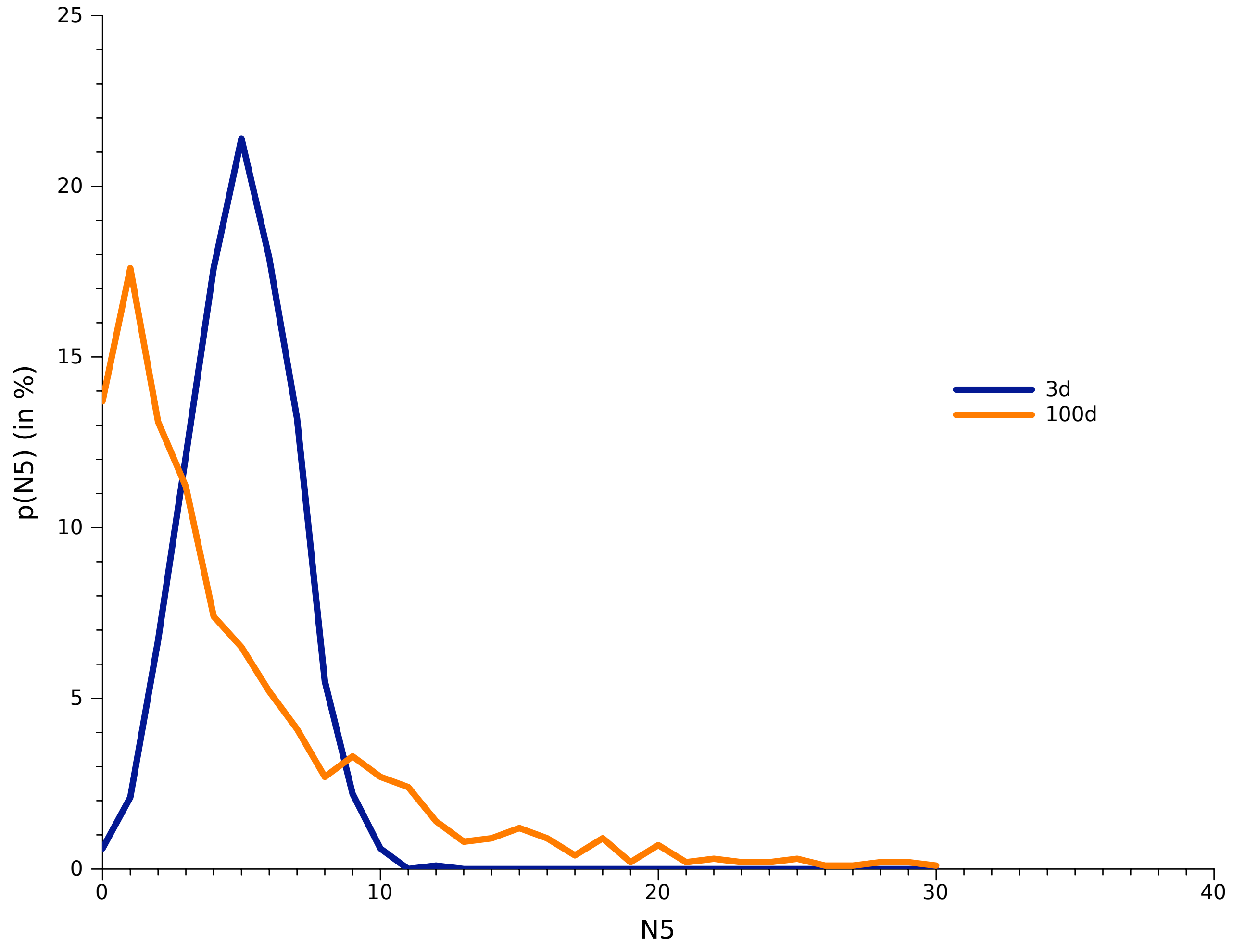
Concentration of Euclidean distance

Evolution of Dmax/Dmin for points with uniformly distributed independent features



Formation of hubs in high dimensions

Distribution of how many times a point is in the 5-nearest neighbours of other points (for 1000 points with uniformly distributed independent features and Euclidean distance)



Does it matter ?

In many real life cases, data have enough structure to minimise the problem of distance concentration.

However, noise (i.e. features close to i.i.d.) makes it worse.

Microarray data are known to be particularly prone to the problem

Strategies to avoid the problem:

- dimension reduction
- feature selection
- subspace clustering
- modification of the distance/similarity measure

Types of algorithms

- Partitioning:
  Split the data into groups and iteratively improve the repartition

- Hierarchical:
  Agglomerative:
    Start with each object in a separate cluster and iteratively merge
  Divisive:
    Start with one cluster and iteratively split

- Density-based:
  Require clusters to have a certain number of objects in a given space

- Model-based:
  Define a model to describe each cluster and try to fit data into the model

K-means

1- Select k points at random as centres of the k clusters

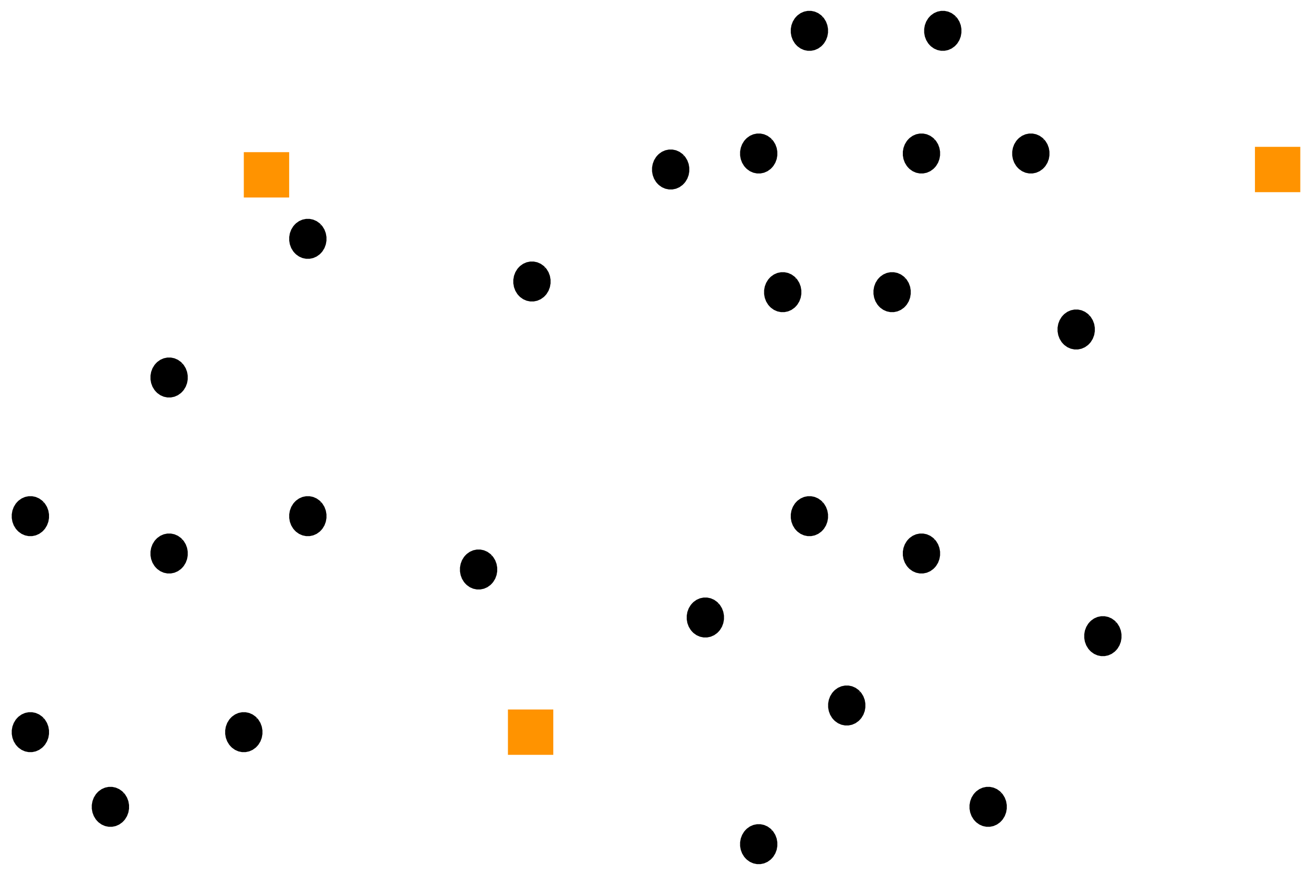
2- Assign each object to the closest cluster
(based on distance to the centre)

3- Recalculate the cluster centres

4 - Repeat from 2 until there is no more change
(or an infinite loop)

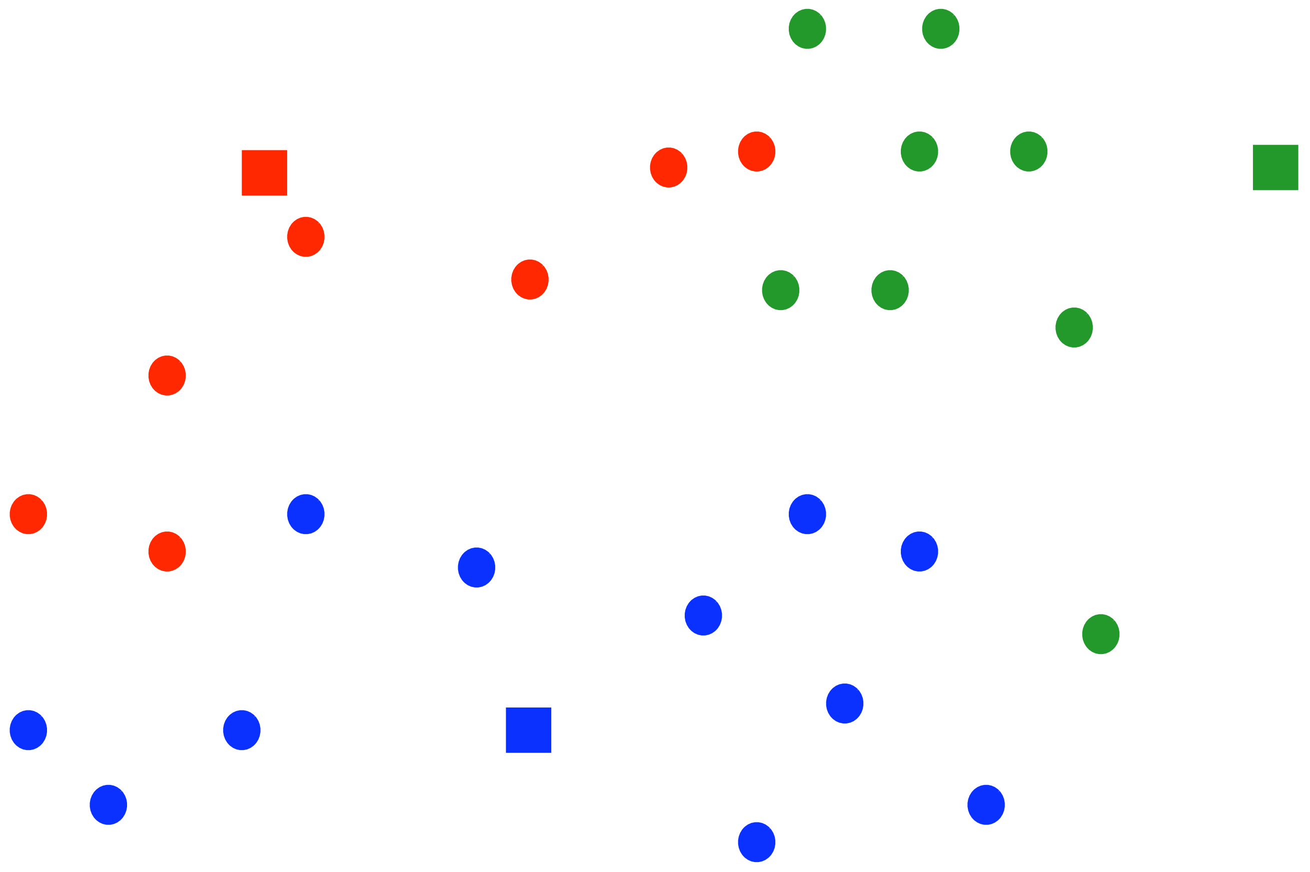
K-means example: step 1

Select 3 initial cluster centres at random



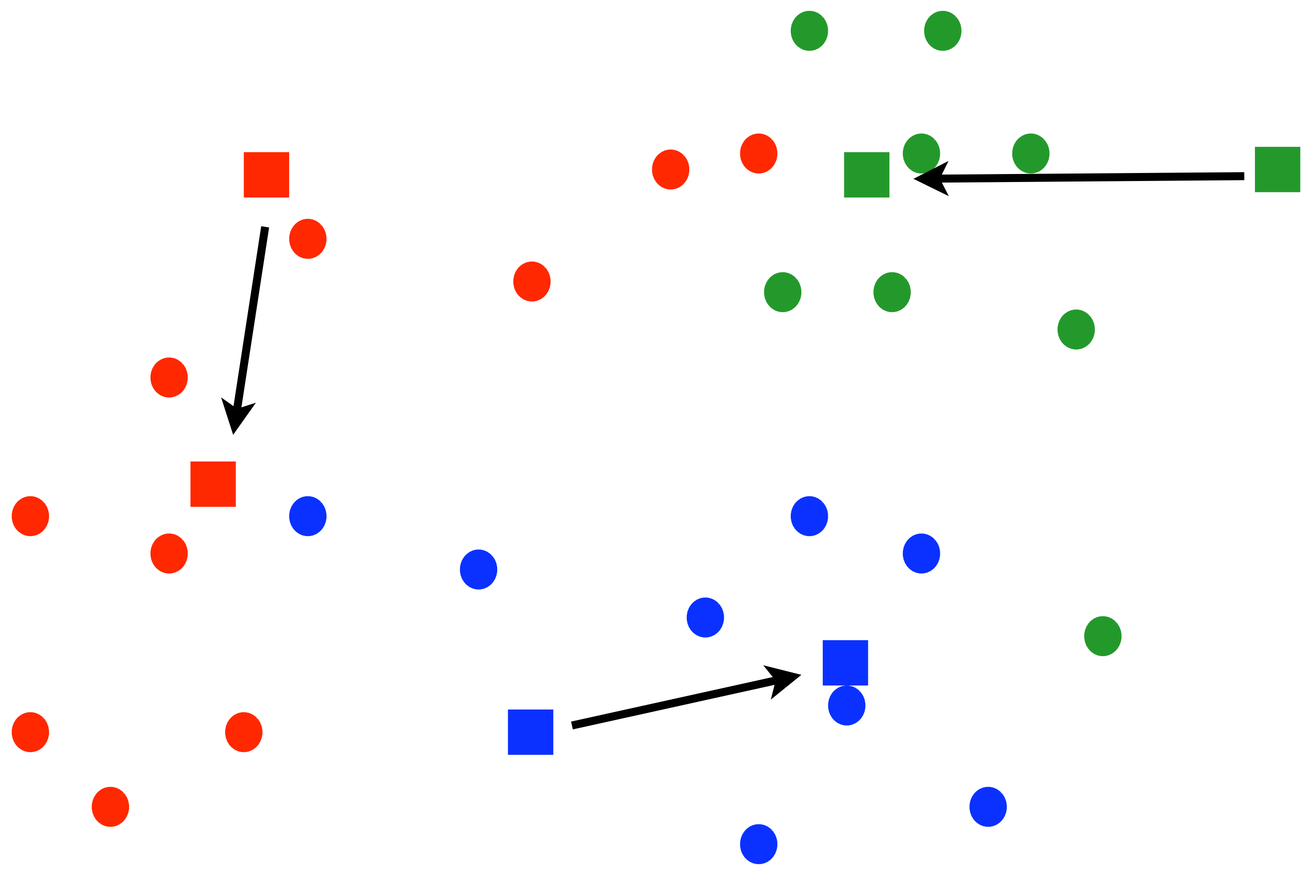
K-means example: step 2

Assign each point to the closest centre



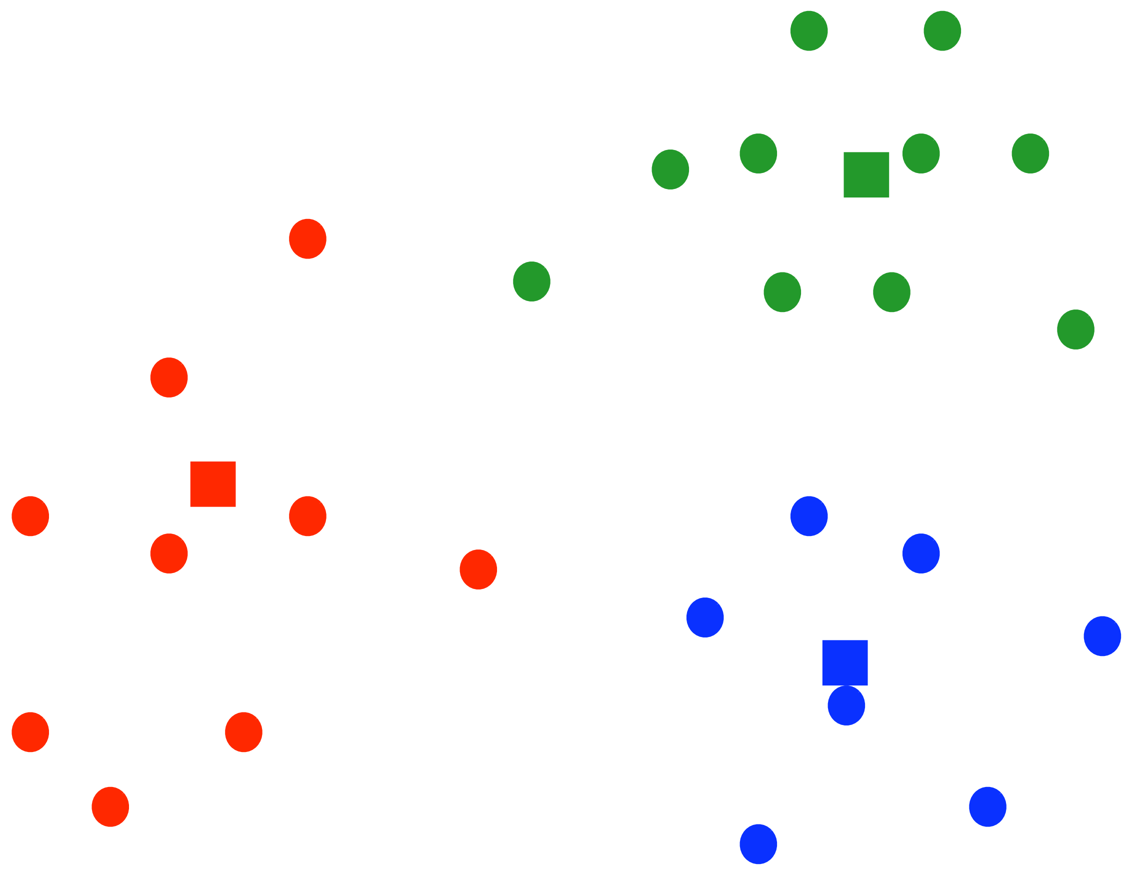
K-means example: step 3

Reposition the cluster centres



K-means example: step 3

Reassign each point to the closest centre



Weaknesses of K-means

1- Results can vary depending on the choice of starting points:

→ run several times with different choices

2- Sensitive to outliers:

→ use the median instead of the mean: K-medoids algorithm

3- Finds spherical clusters

4- Need to choose k in advance

Hierarchical clustering

1- Assign each object to its own cluster

2- Merge the 2 most similar clusters

3- Compute the similarity between the new cluster and each of the old ones

4- Repeat 2 and 3 until there is only one cluster

There are different ways of doing step 3:
single linkage, complete linkage, average linkage, Ward's method ...

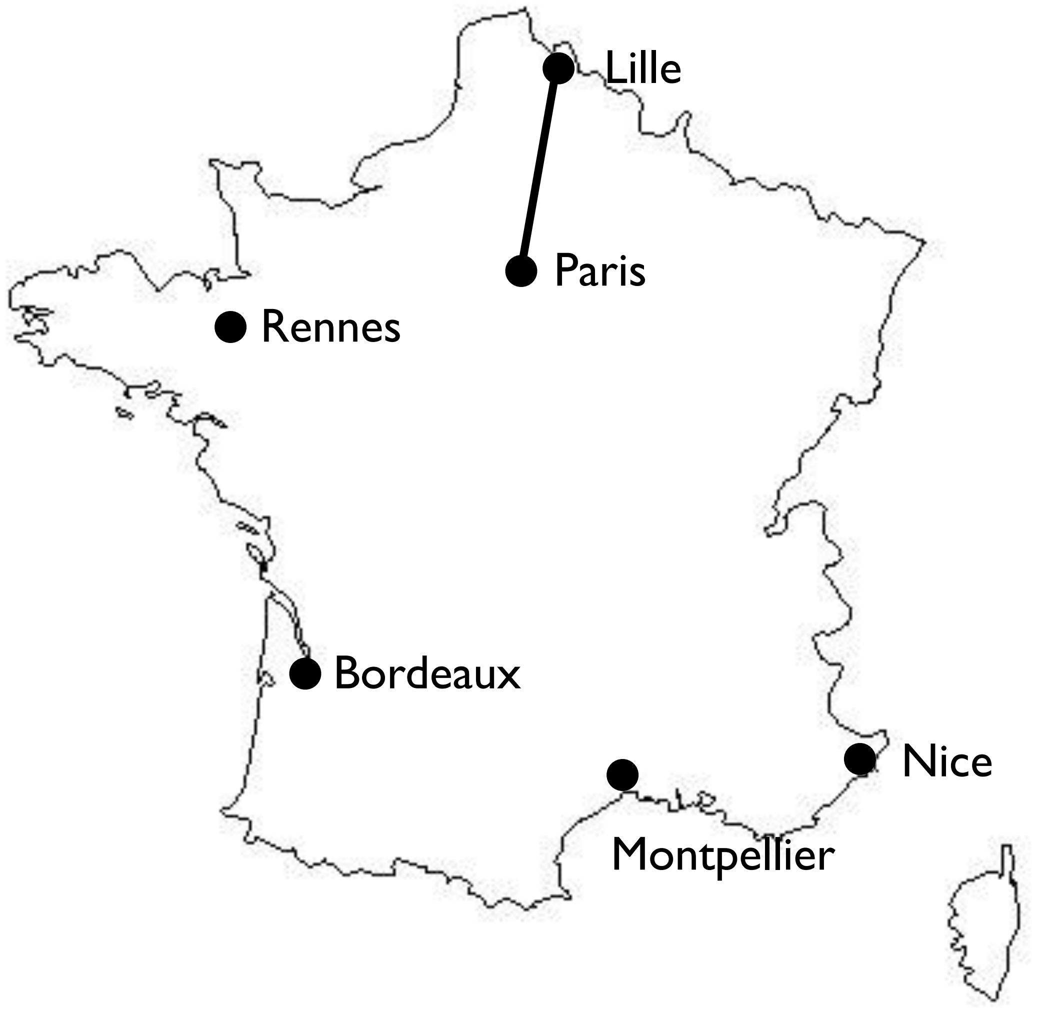
Single-linkage clustering example: input

Distances between French cities (as the crow flies)

	Paris	Nice	Bordeaux	Lille	Montpellier	Rennes
Paris	0	700	500	200	600	300
Nice	700	0	650	850	300	850
Bordeaux	500	650	0	700	400	400
Lille	200	850	700	0	800	450
Montpellier	600	300	400	800	0	650
Rennes	300	850	400	450	650	0

Single-linkage clustering example: step 2

Merge the 2 most similar clusters



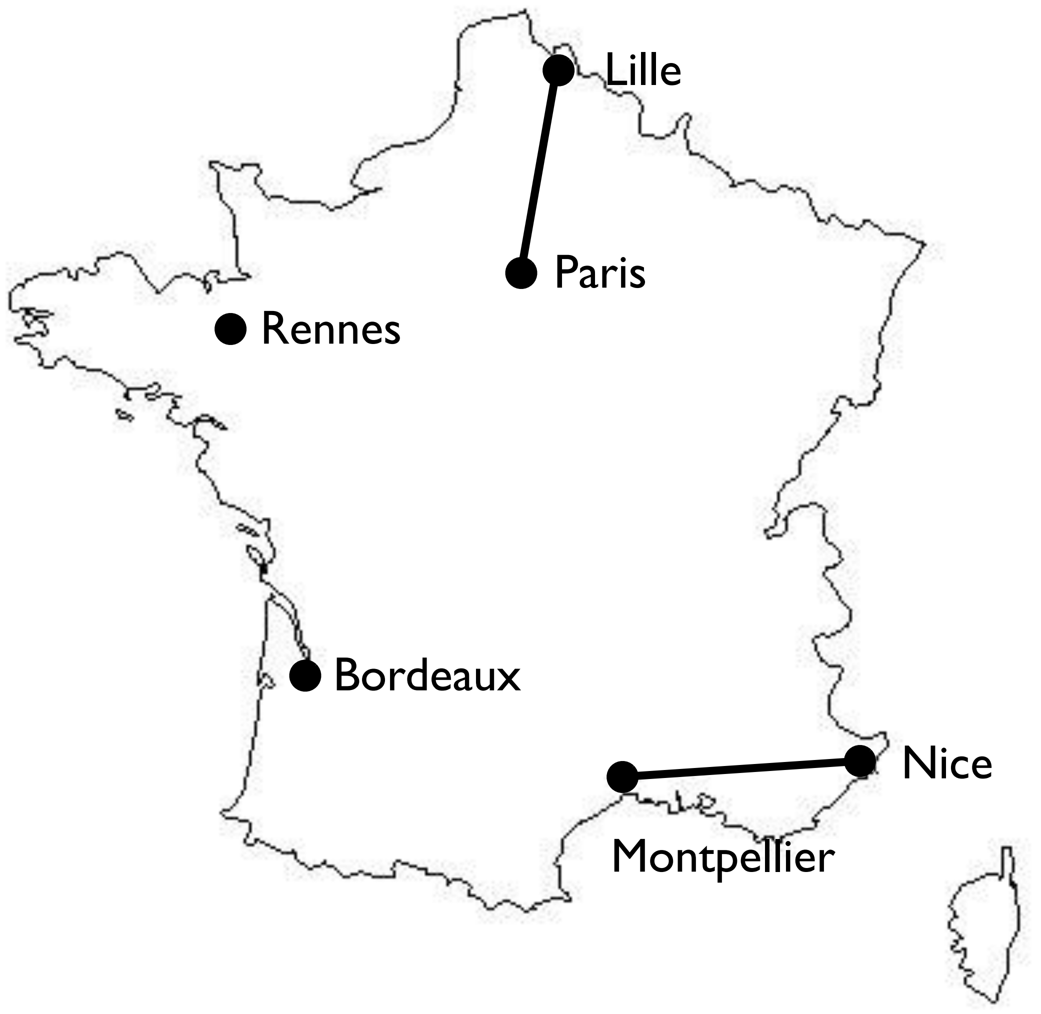
Single-linkage clustering example: step 3

Compute the similarity between the new cluster and each of the old ones

	Paris/Lille	Nice	Bordeaux	Montpellier	Rennes
Paris/Lille	0	700	500	600	300
Nice	700	0	650	300	850
Bordeaux	500	650	0	400	400
Montpellier	600	300	400	0	650
Rennes	300	850	400	650	0

Single-linkage clustering example: repeat step 2

Merge the 2 most similar clusters



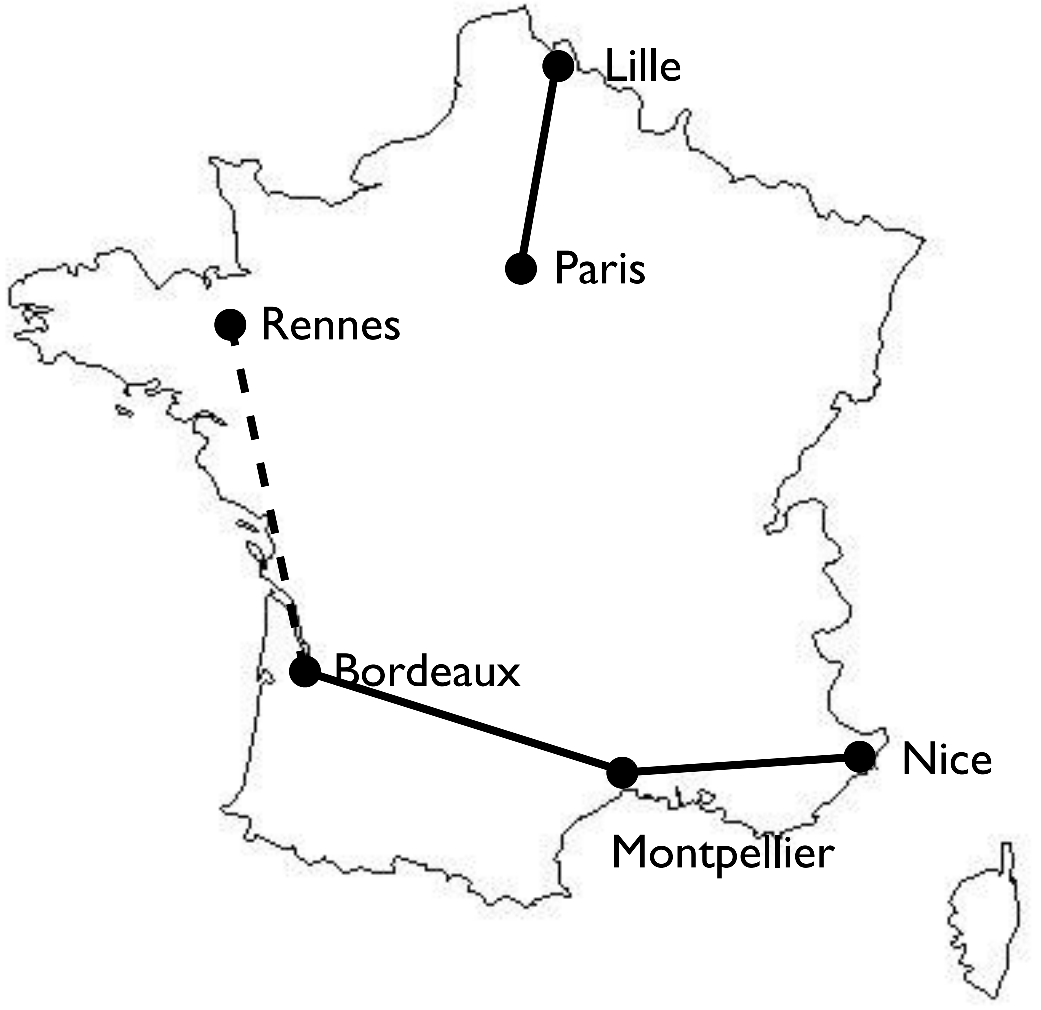
Single-linkage clustering example: repeat step 3

Compute the similarity between the new cluster and each of the old ones

	Paris/Lille	Bordeaux	Montpellier/Nice	Rennes
Paris/Lille	0	500	600	300
Bordeaux	500	0	400	400
Montpellier/Nice	600	400	0	650
Rennes	300	400	650	0

Single-linkage clustering example: repeat step 2

Merge the 2 most similar clusters



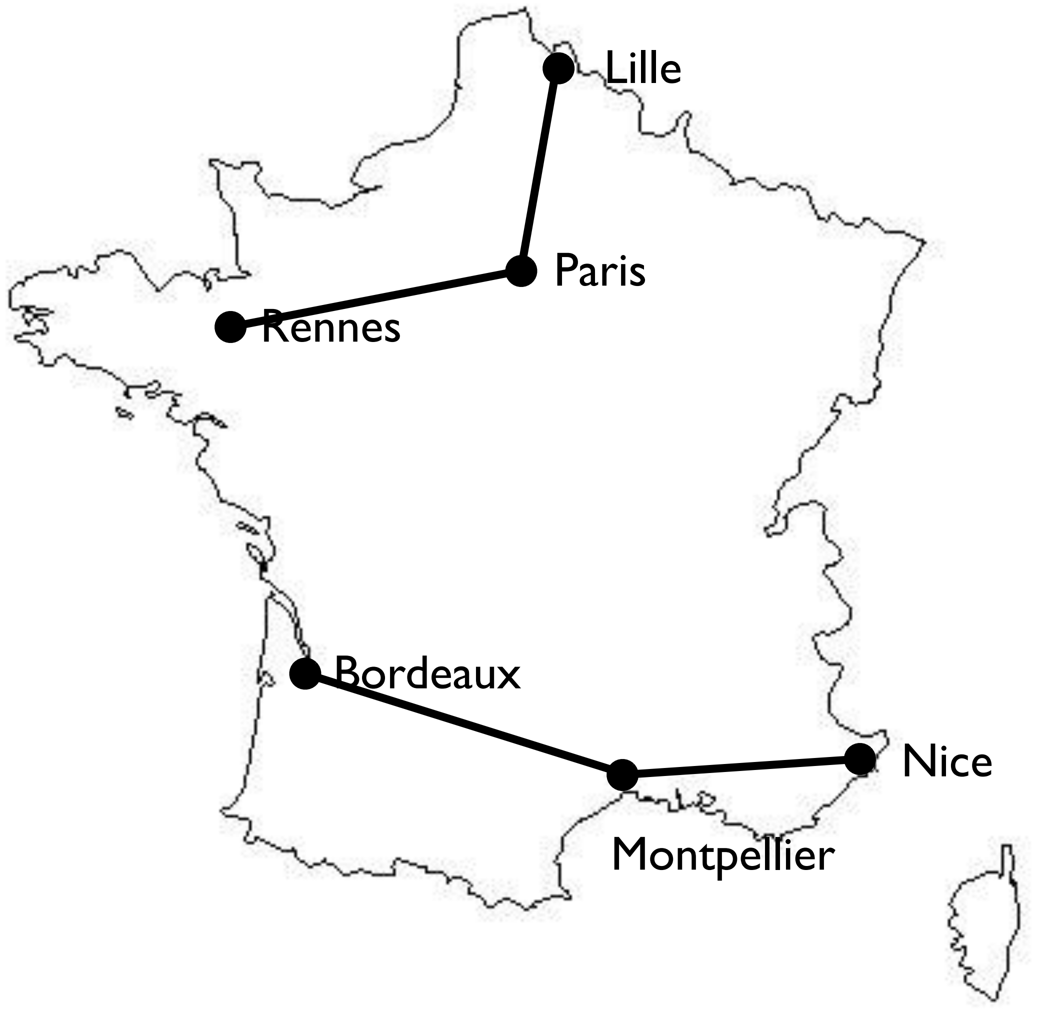
Single-linkage clustering example: repeat step 3

Compute the similarity between the new cluster and each of the old ones

	Paris/Lille	Montpellier/Nice/ Bordeaux	Rennes
Paris/Lille	0	500	300
Montpellier/Nice/ Bordeaux	500	0	400
Rennes	300	400	0

Single-linkage clustering example: repeat step 2

Merge the 2 most similar clusters



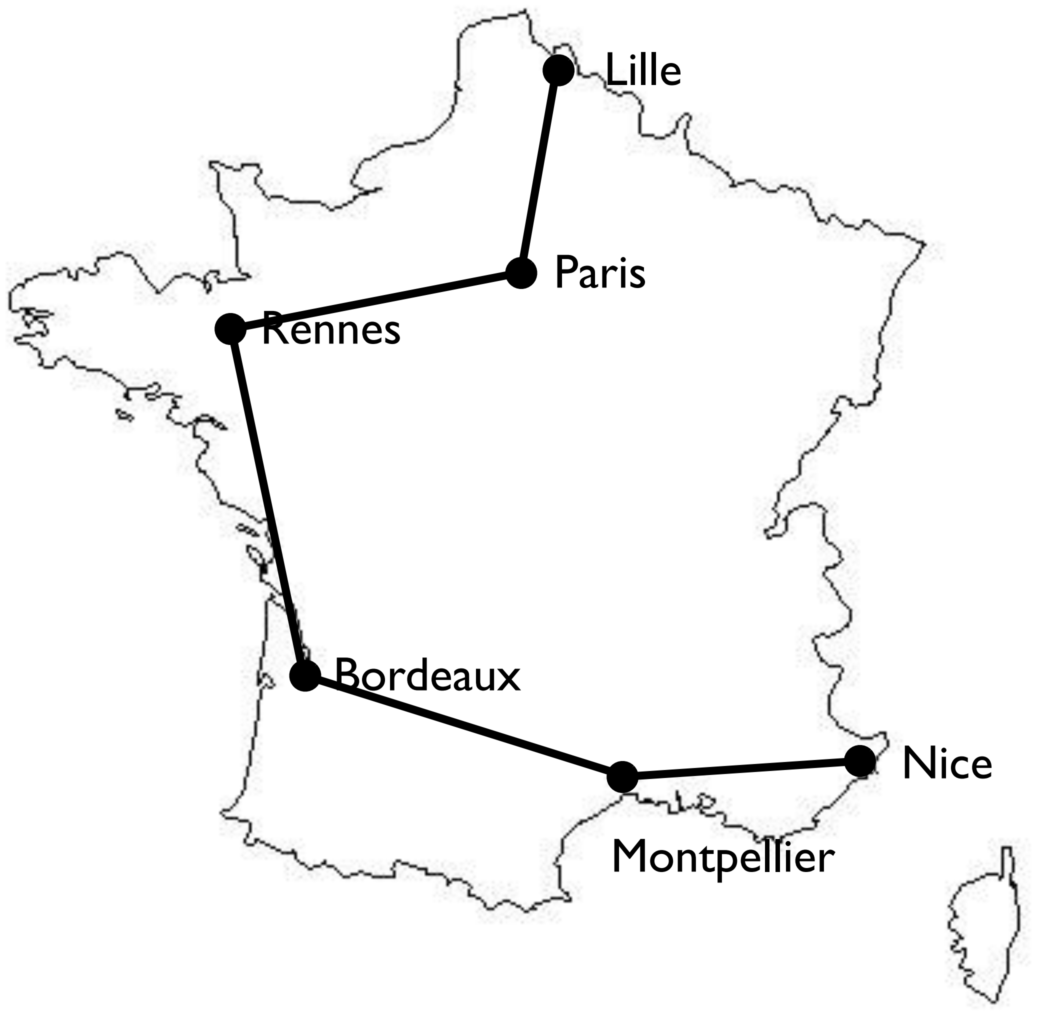
Single-linkage clustering example: repeat step 3

Compute the similarity between the new cluster and each of the old ones

	Paris/Lille/Rennes	Montpellier/Nice/Bordeaux
Paris/Lille/Rennes	0	400
Montpellier/Nice/Bordeaux	400	0

Single-linkage clustering example: repeat step 2

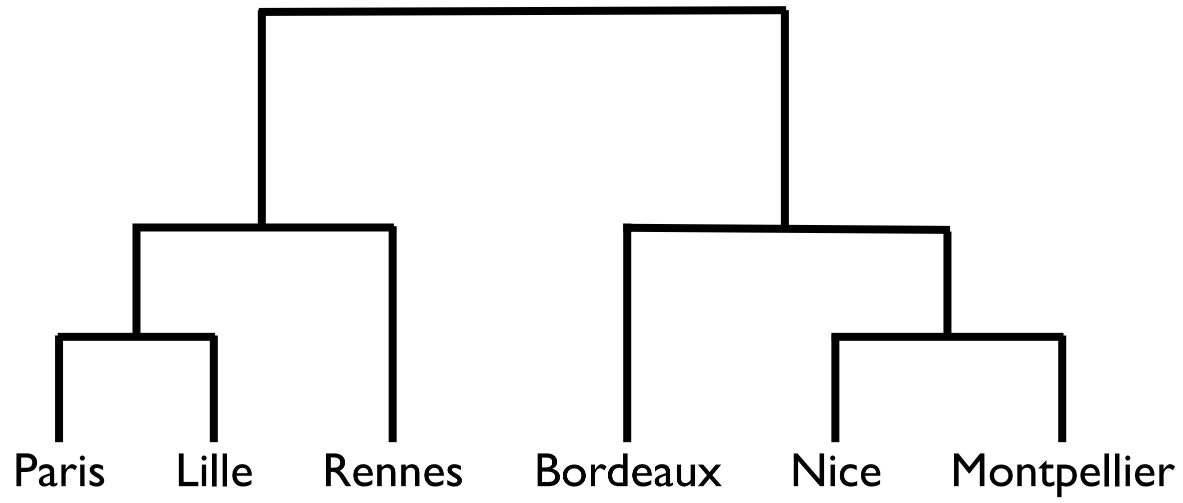
Merge the 2 most similar clusters



Single-linkage clustering example: Result

Dendrogram: tree of distances between objects

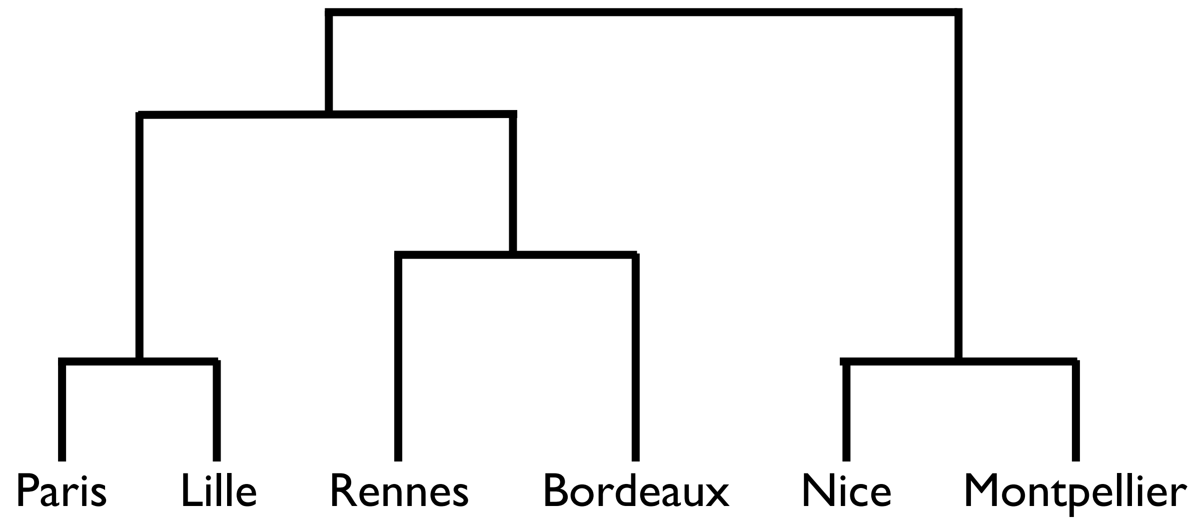
To obtain clusters, one has to cut the tree



Problems with agglomerative clustering

1- Can't undo previous merging:

e.g. alternative tree



2- Need to cut the tree into clusters

DBScan

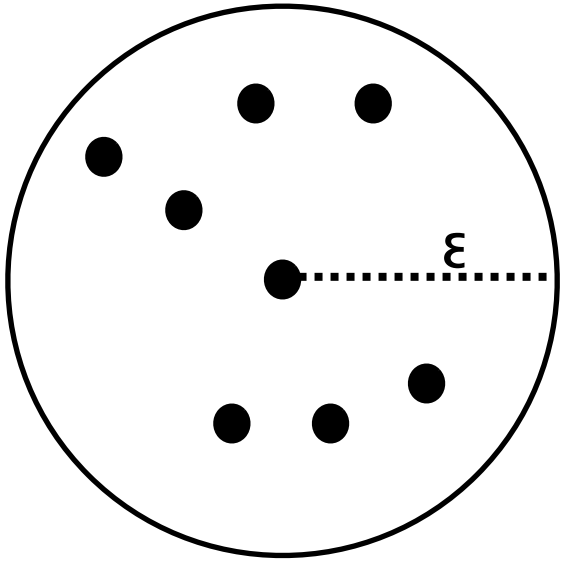
Clusters are defined as density connected groups:

- For each point, check if there are more than MinPts data points within a distance ε (density reachable points)

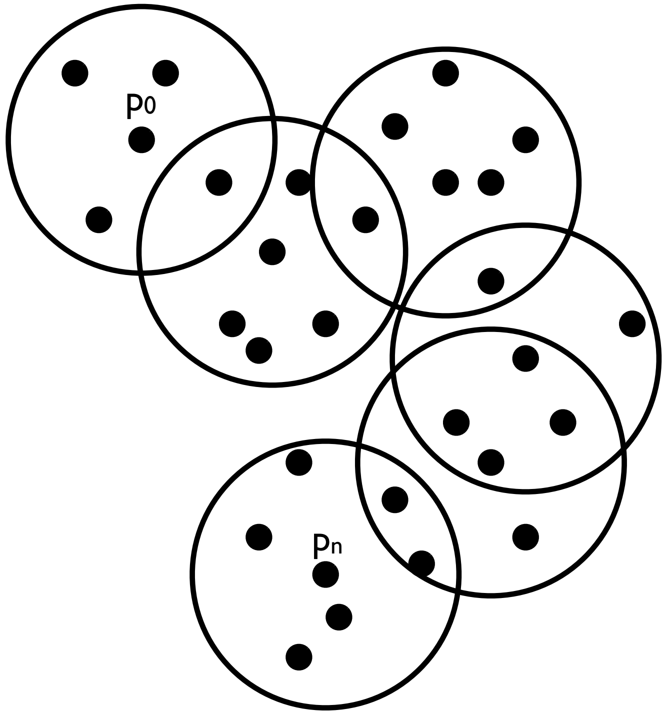
- p_0 and p_n are density connected if there is a sequence of density reachable points $p_1 \ldots p_{n-1}$ such that p_{i+1} is density reachable from p_i.

DBScan notions

Reachability



Connectivity



Advantages and issues of DBScan

Advantages:

- Can find clusters with arbitrary shapes.
- Has a notion of noise.
- Doesn't require the number of clusters.

Problems:

- Can't find clusters if their densities are too different.
- Doesn't scale well.
- May not work in high-dimensions.
- Need to find the right parameters.

Evaluation of clustering results

1- Is there non-random structure in the data ?

2- What's the correct number of clusters ?

3- How does the result fit the data (with no external information) ?

4- Which of two sets of clusters is better ?

5- How does the result fit labelled data ?

Many measures/coefficients/indexes
but often difficult to interpret in the absence of references:
e.g. what does a value of 5 represent ?

Silhouettes

a_i: average distance of i to other points in the same cluster
b_i: smallest average distance of i to points in another cluster

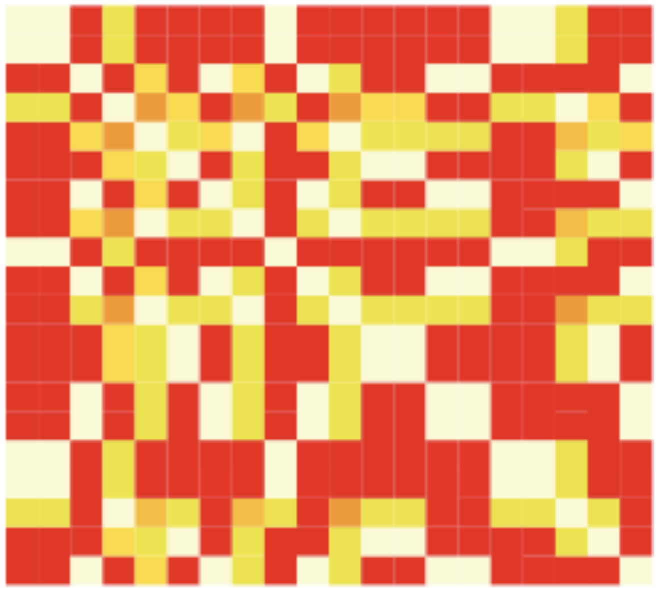
$S_i = (b_i - a_i) / \max(a_i, b_i)$

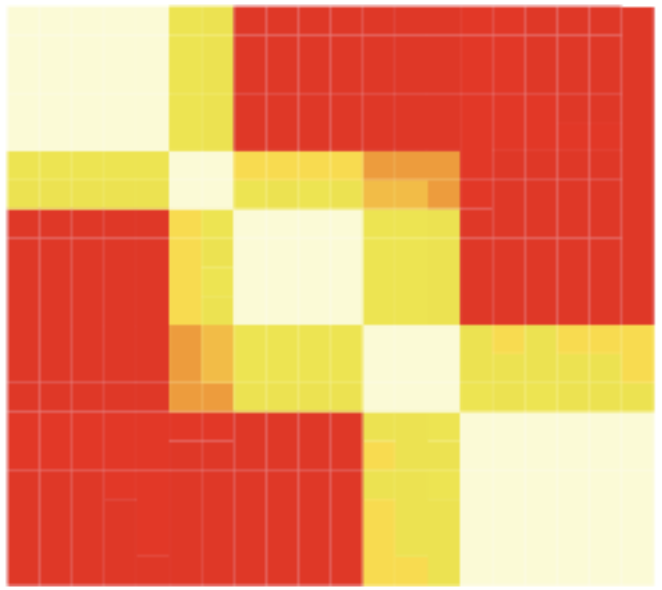
$-1 \leq S_i \leq 1$

Clustering quality = average S for all data points

Can be used to find the number of clusters

Relationship with visualisation


Unordered similarity matrix


Reordered similarity matrix

Conclusion

- There are thousands of algorithms and no best algorithm.

- Clustering is as much an art as a science.

- Despite many issues, simple algorithms generally perform well compared to more fancy algorithms

- The trick to get good results is to have good (i.e. structured) data.