COMP47460

Clustering - Part 1

Aonghus Lawlor
Deepak Anjwani

School of Computer Science
Autumn 2019

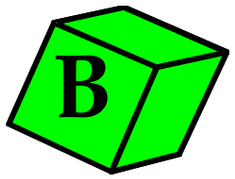
Overview

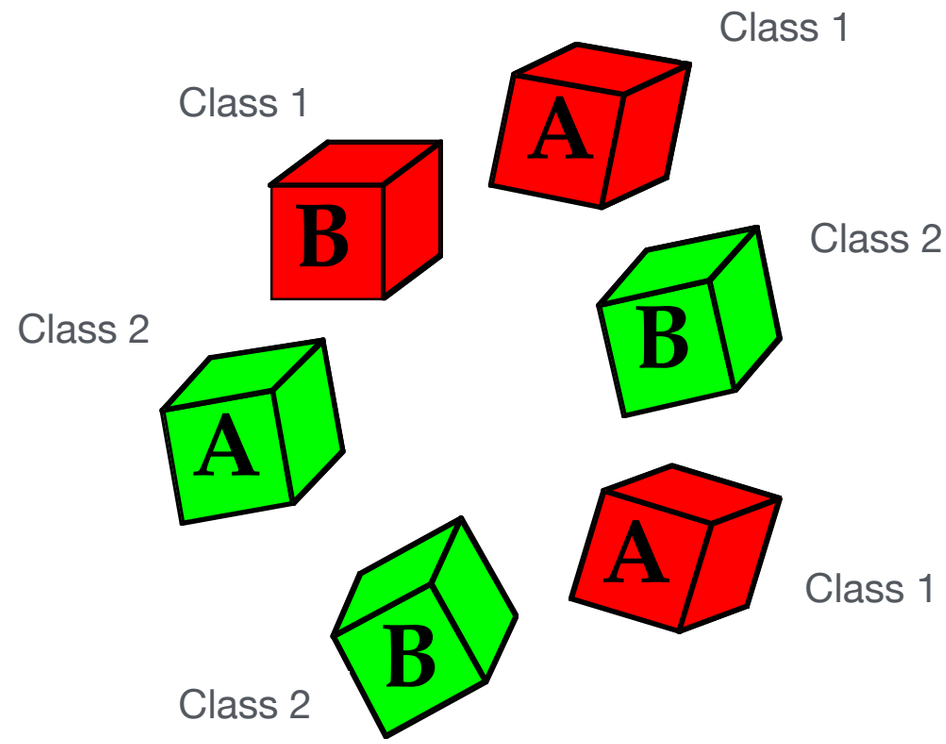
- Part 1
  - Supervised v Unsupervised Learning
  - Partitional Clustering
    - *k*-Means clustering
    - Cluster initialisation
- Part 2
  - Hierarchical Clustering
    - Agglomerative algorithms
    - Cluster metrics
    - Divisive algorithms
  - Cluster Validation

Supervised Learning

Supervised learning is based on a training set where labelling of instances with a fixed number of classes represents the target function.

To which class does this new training example belong?





Use a model built on training data to make a prediction for the new example.

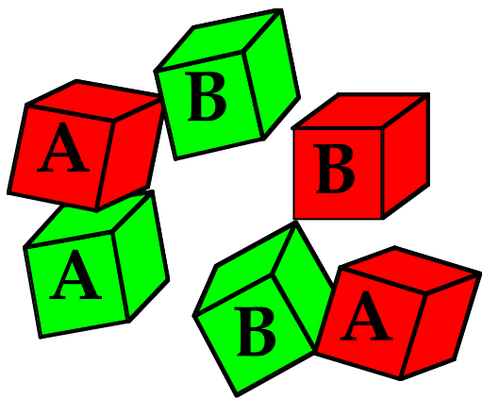
For many tasks, annotated class labels for data are not available - either unknown or too expensive to obtain.

Unsupervised Learning

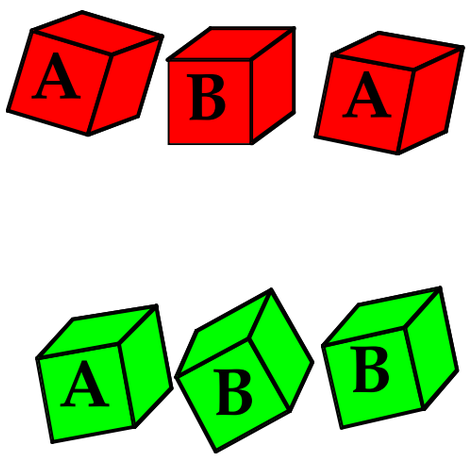
Unsupervised learning algorithms attempt to identify patterns by relying solely on the intrinsic characteristics of the data, without referring to any class information.

Important for knowledge discovery and data exploration tasks, also for summarisation, visualisation, compression, outlier detection…

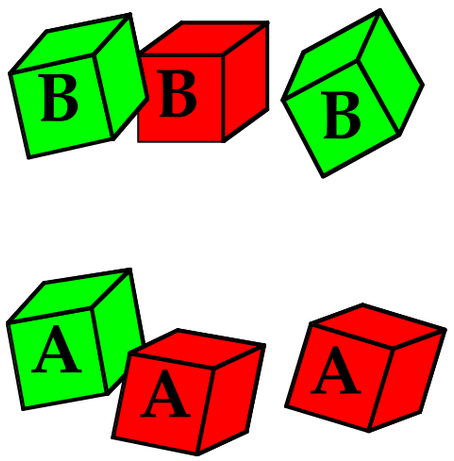
Organise these blocks into groups



Two possible groupings. No guidance on which is the "correct" grouping



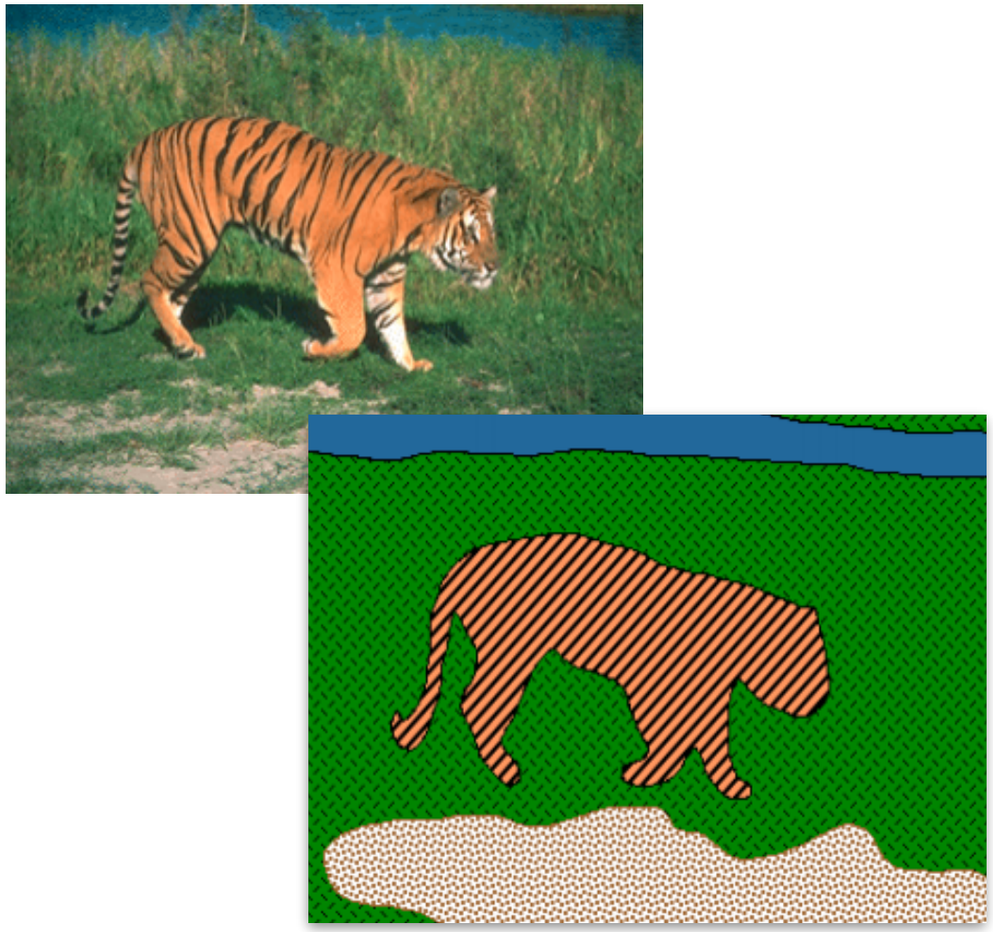
Grouping 1

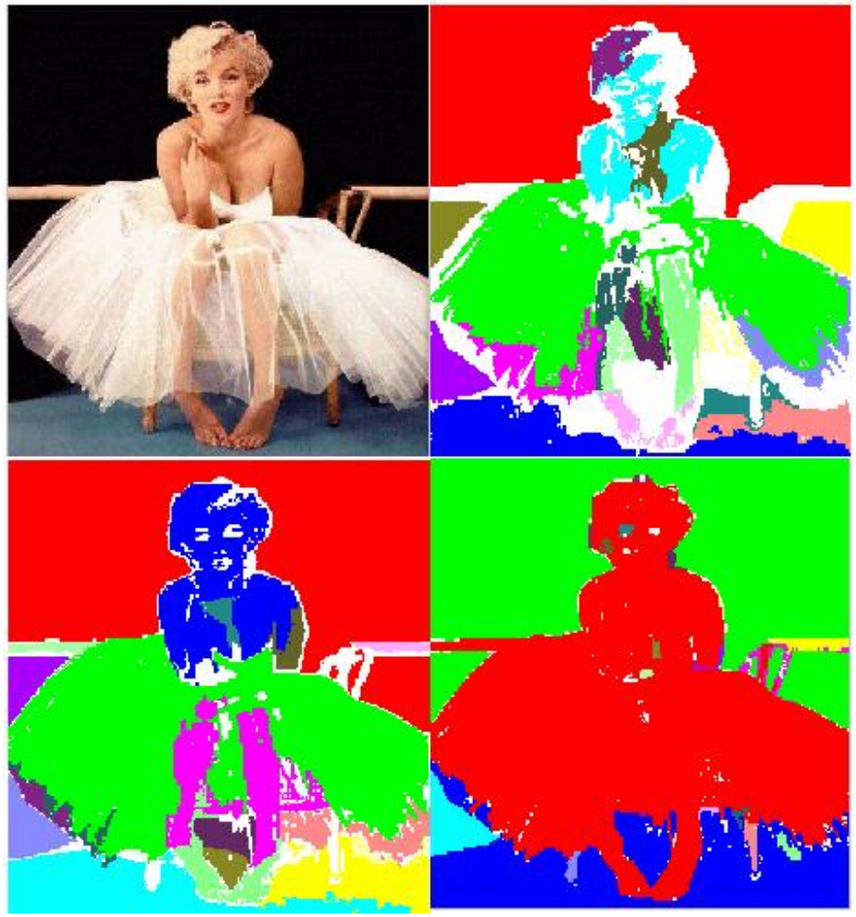


Grouping 2

Unsupervised Learning: Applications

Image segmentation: Unsupervised task in computer vision that attempts to automatically split an image into regions with similar colour or texture, or both. Aim is to partition the image into its constituent “objects”.

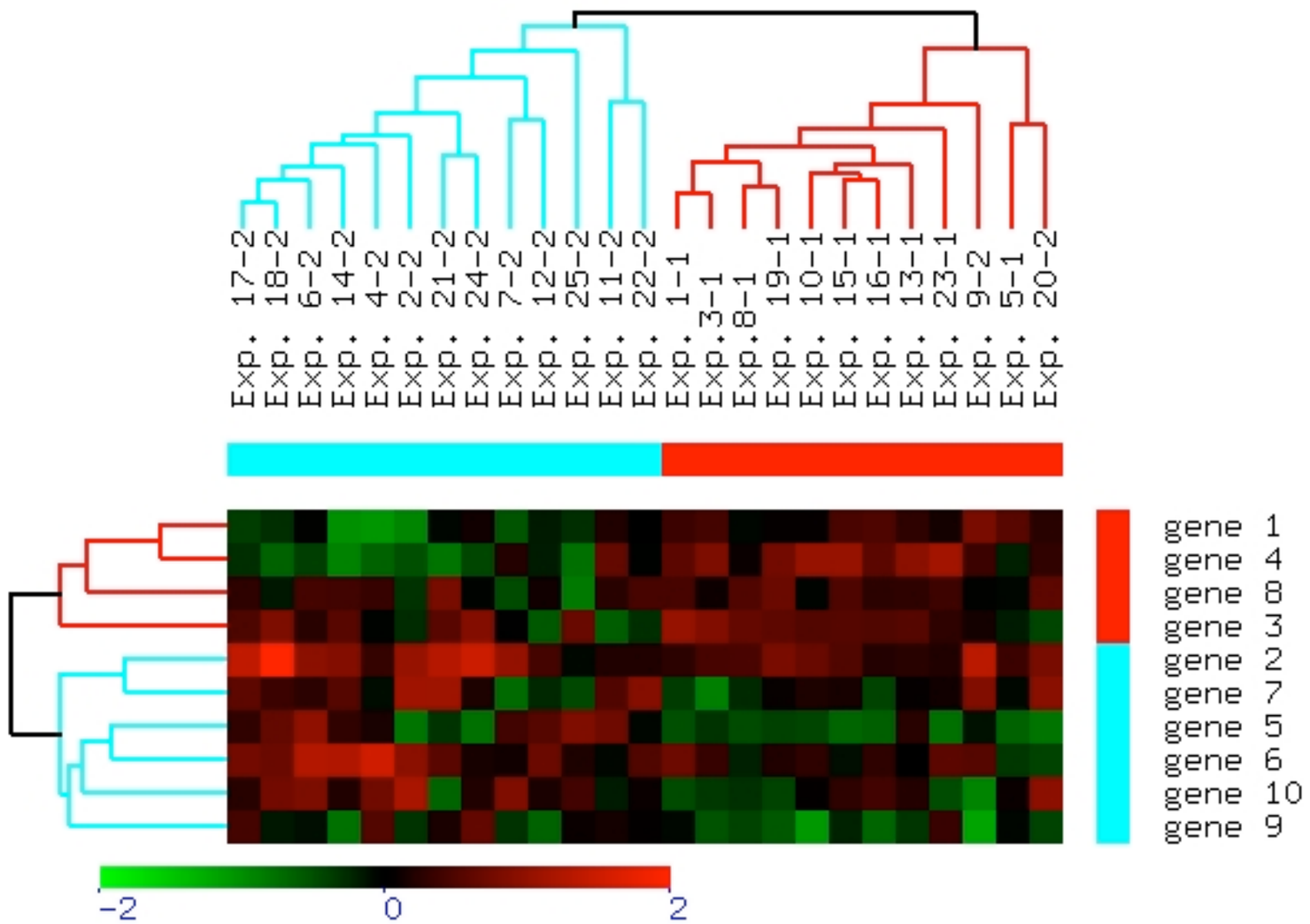




http://web.mit.edu/manoli/imagina/www/imagina.html

Unsupervised Learning: Applications

Hierarchical clustering is frequently applied in biology when studying gene expression data to infer biological function of unknown genes. Often want to cluster both genes and experiments (conditions).

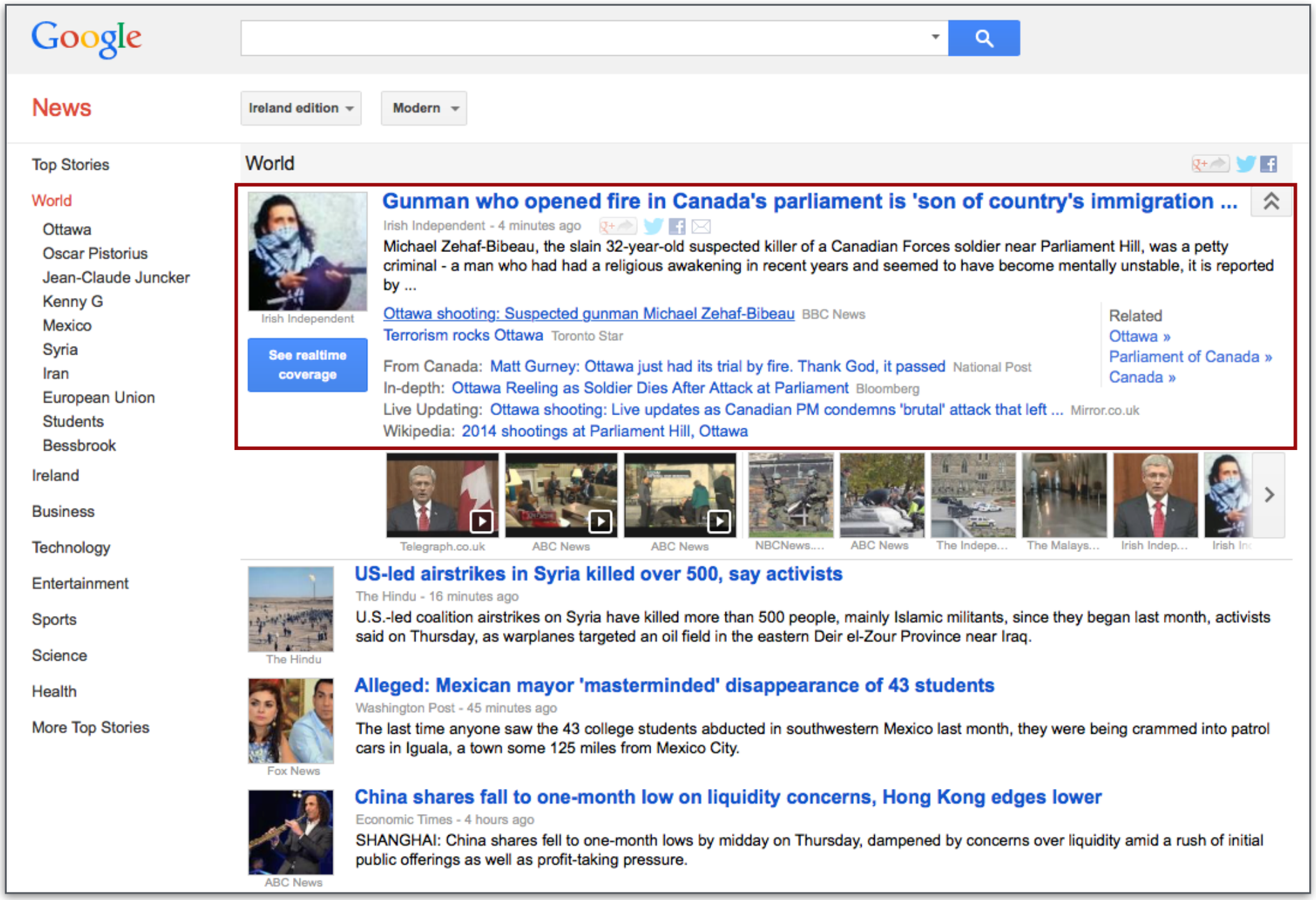


Unsupervised Learning: Applications

Topic modeling: Unsupervised task of discovering the underlying thematic structure in a text corpus - i.e. the key "topics" in the data.

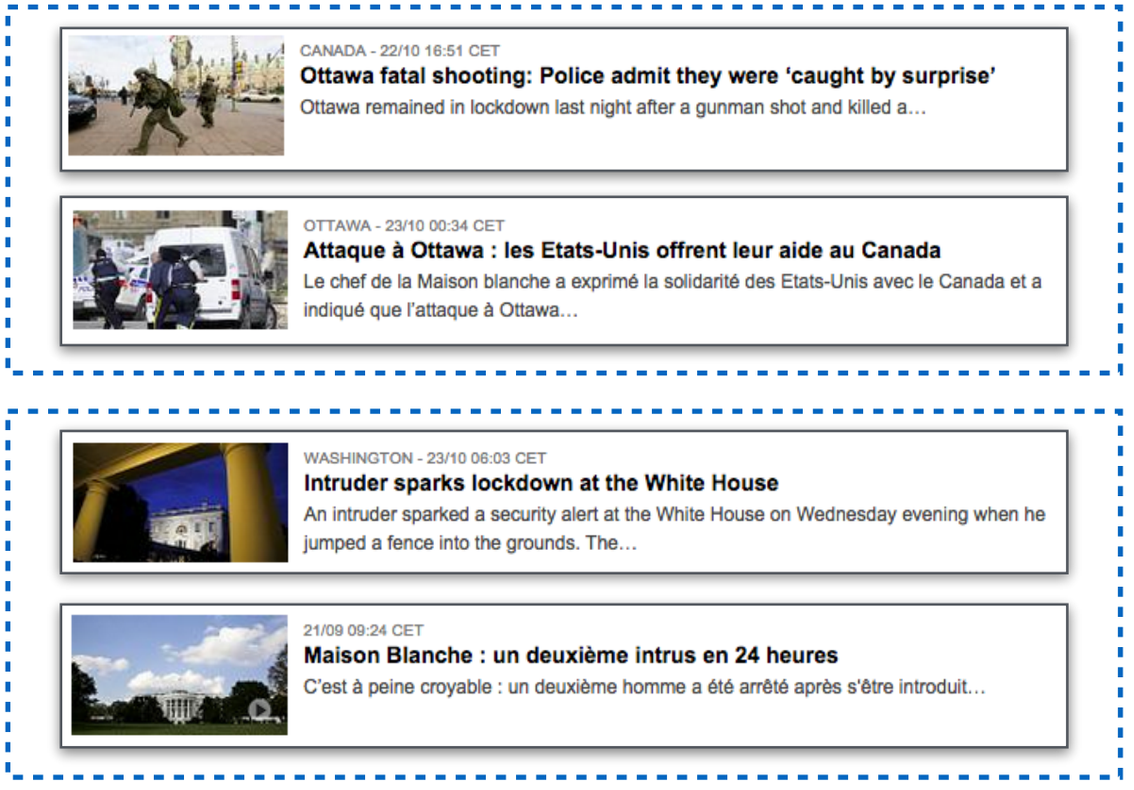
Unsupervised Learning: Applications

Document clustering: Automatically group related documents together based on similar content (e.g. related articles on Google News).



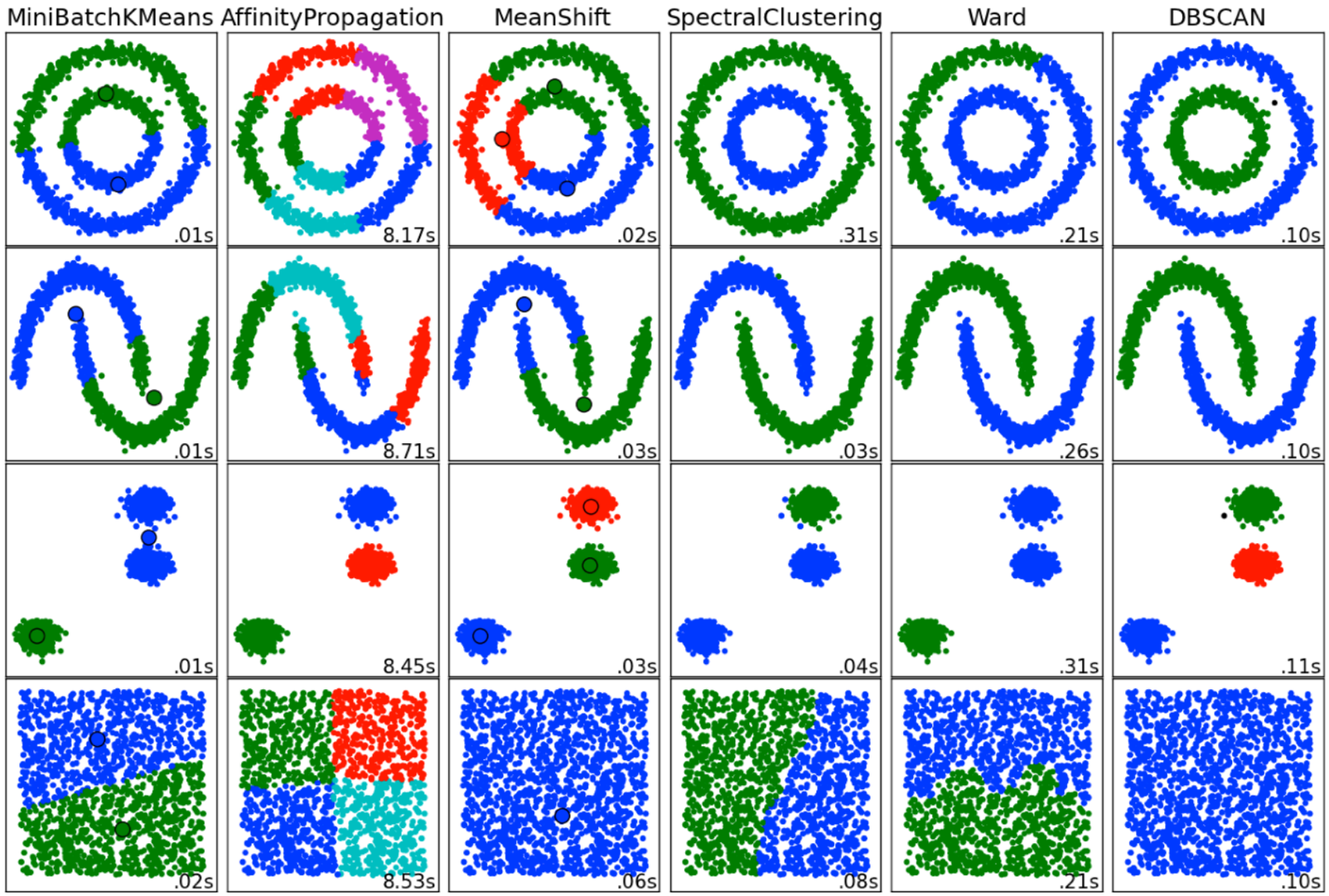
Clustering

- Clustering algorithms organise data into groups (“clusters”) in the absence of any external information.
  - ➡ No labelled training examples to learn from.
  - ➡ Generally won’t know in advance how many clusters are in the data.
- Different clusterings can reveal different things about the same data. Generally not “correct” or “incorrect”, but some clusterings will be more useful than others.



Clustering

Many different ways to cluster the same data set. Clustering algorithms differ significantly in their definition of what constitutes a cluster and how to efficiently find them.

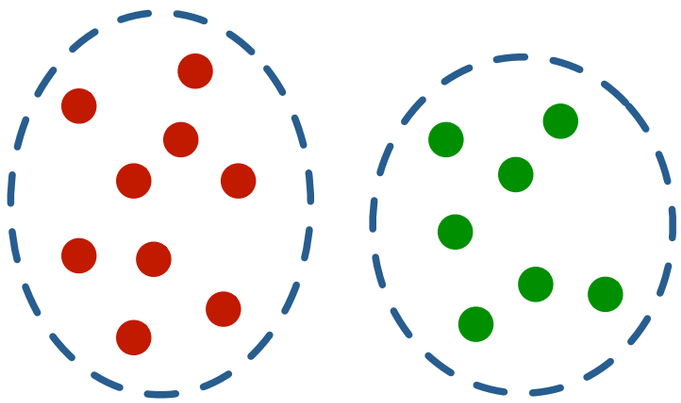


http://scikit-learn.org/stable/modules/clustering.html

Clustering

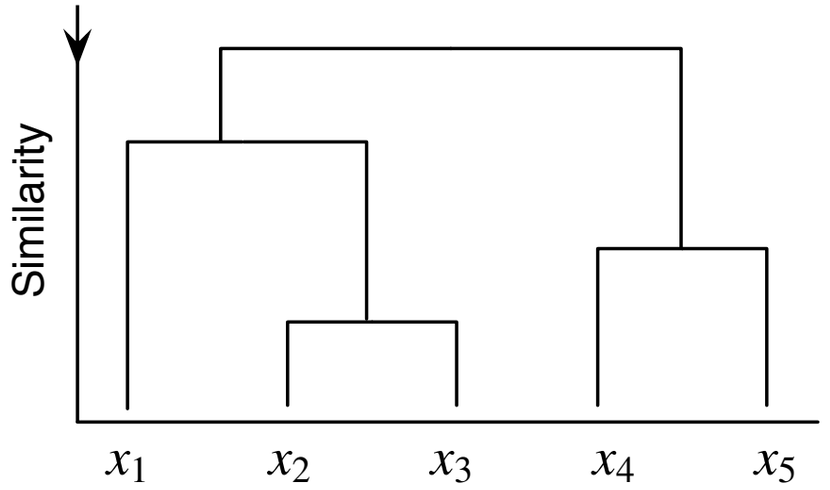
- **General goal**: Assign similar items to the same cluster, keep dissimilar items apart.
- Algorithms employ different definitions of similarity/dissimilarity and objective function for determining a “good” cluster.

Partitional Algorithms



Build a “flat” clustering of the data all at once

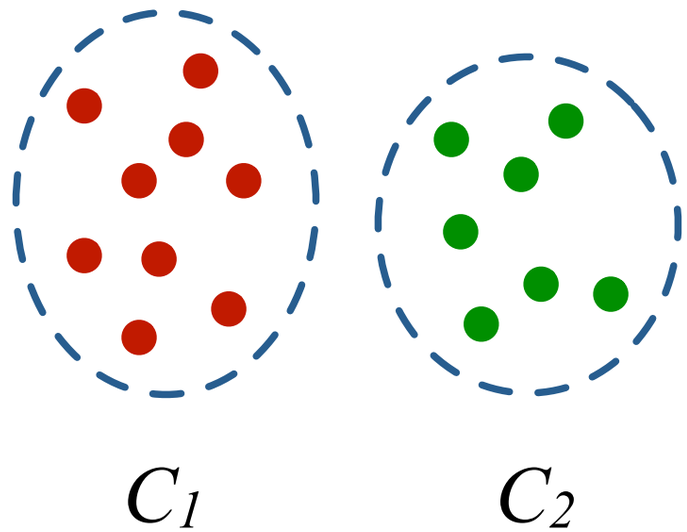
Hierarchical Algorithms



Gradually build a nested tree structure of clusters

Partitional Clustering

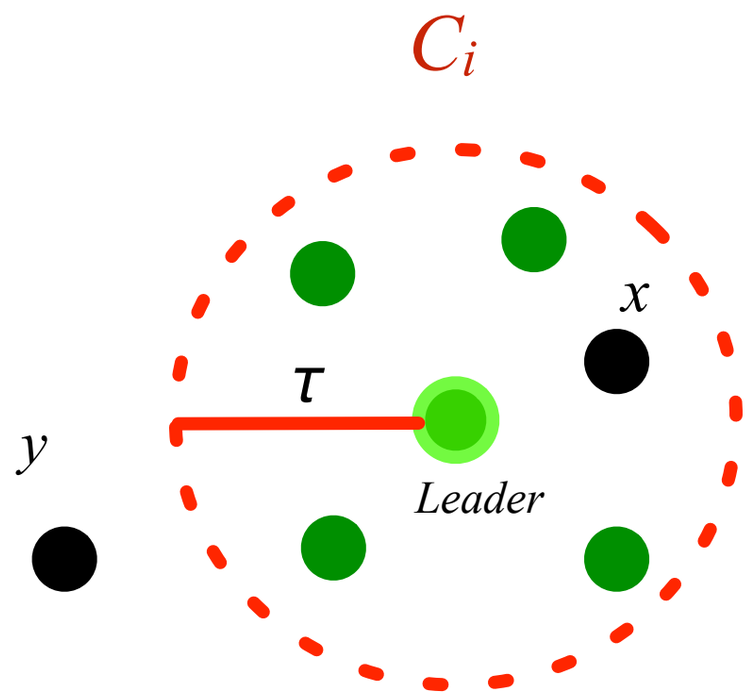
- Attempt to directly decompose a data set into a “flat” grouping consisting of a number of disjoint (non-overlapping) clusters.
- Usually pre-specify number of clusters *k*, although some methods adaptively add/remove clusters.
- Start with an initial set of *k* clusters, often chosen at random.
- Use a heuristic to find the best local solution for an objective function, identified by iteratively improving the initial solution.
- Examples:
  - Sequential leader clustering
  - *k*-Means
  - Partitioning Around Medoids (PAM)



Sequential Leader Clustering

- Simplest partitional algorithm, which incrementally builds clusters as each new item arrives. Useful in real-time streaming applications.
- Divide the data into *k* clusters. For each cluster, there is a "leader" item and all other items have distance ≤ τ to the leader.
- The value τ controls the radius around a cluster's centre (i.e. leader), into which items must fall to belong to that cluster.

- Read new input item x.
- Find "winning" cluster C_i with leader nearest to x.
- IF distance to winning leader ≤ τ THEN
  - Assign x to C_i
- ELSE
  - Create a new cluster with x as leader.



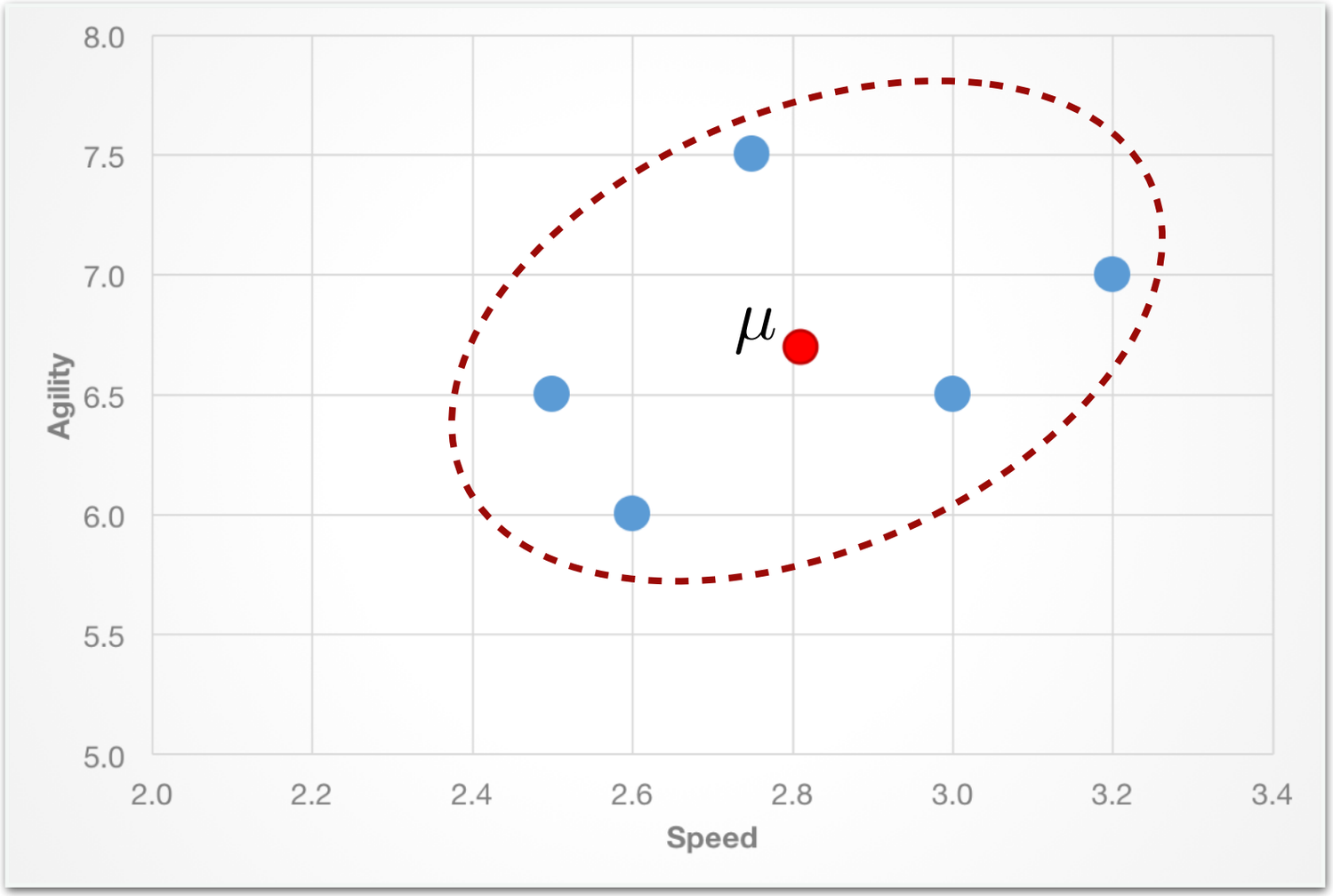
k-Means Clustering

- Centroid: The mean of all items assigned to a given cluster (i.e. the mean of their feature vectors).

Athlete	*Speed*	*Agility*
1	2.6	6.0
2	3.0	6.5
3	2.5	6.5
4	3.2	7.0
5	2.8	7.5
Centroid	2.82	6.7

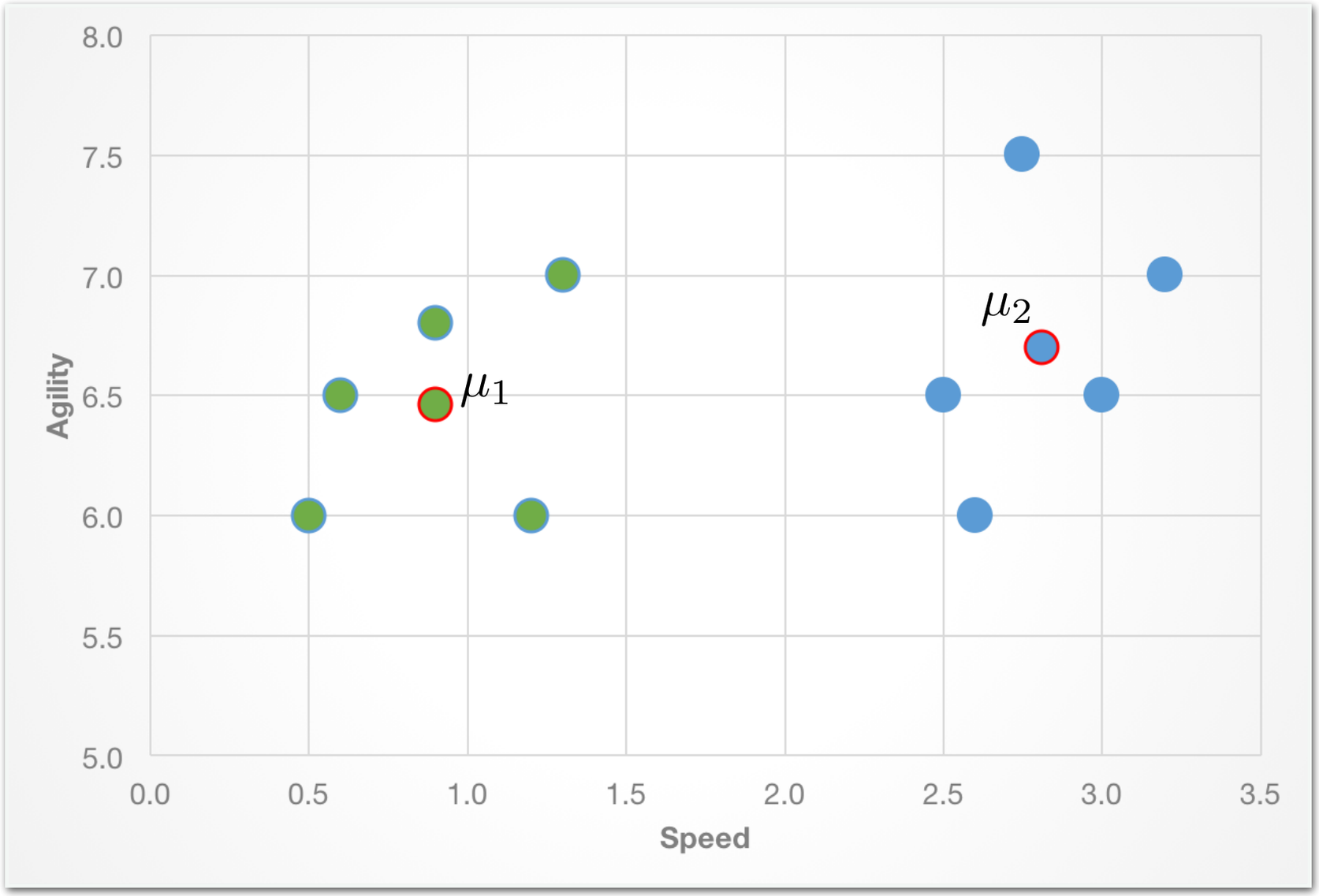
$(2.6 + 3.0 + 2.5 + 3.2 + 2.8)/5 = 2.82$

$(6.0 + 6.5 + 6.5 + 7.0 + 7.5)/5 = 6.7$



k-Means Clustering

- Each of the *k* clusters in a clustering can be represented by its own centroid μ_i



k-Means Clustering

- **Goal:** Minimise distances between the items and their nearest centroid - i.e. minimisation of *sum-of-squared error* (SSE):

$$SSE(\mathcal{C}) = \sum_{c=1}^{k} \sum_{x_i \in C_c} D(x_i, \mu_c)^2 \quad \text{where} \quad \mu_c = \frac{\sum_{x_i \in C_c} x_i}{|C_c|}$$

- In the standard algorithm, *D* is measured using Euclidean distance:

$$D(x, \mu) = \sqrt{\sum_{l=1}^{m} (x_l - \mu_l)^2}$$

sum of squared difference over all *m* feature values

- *k*-Means tries to reduce SSE via a two step iterative process:

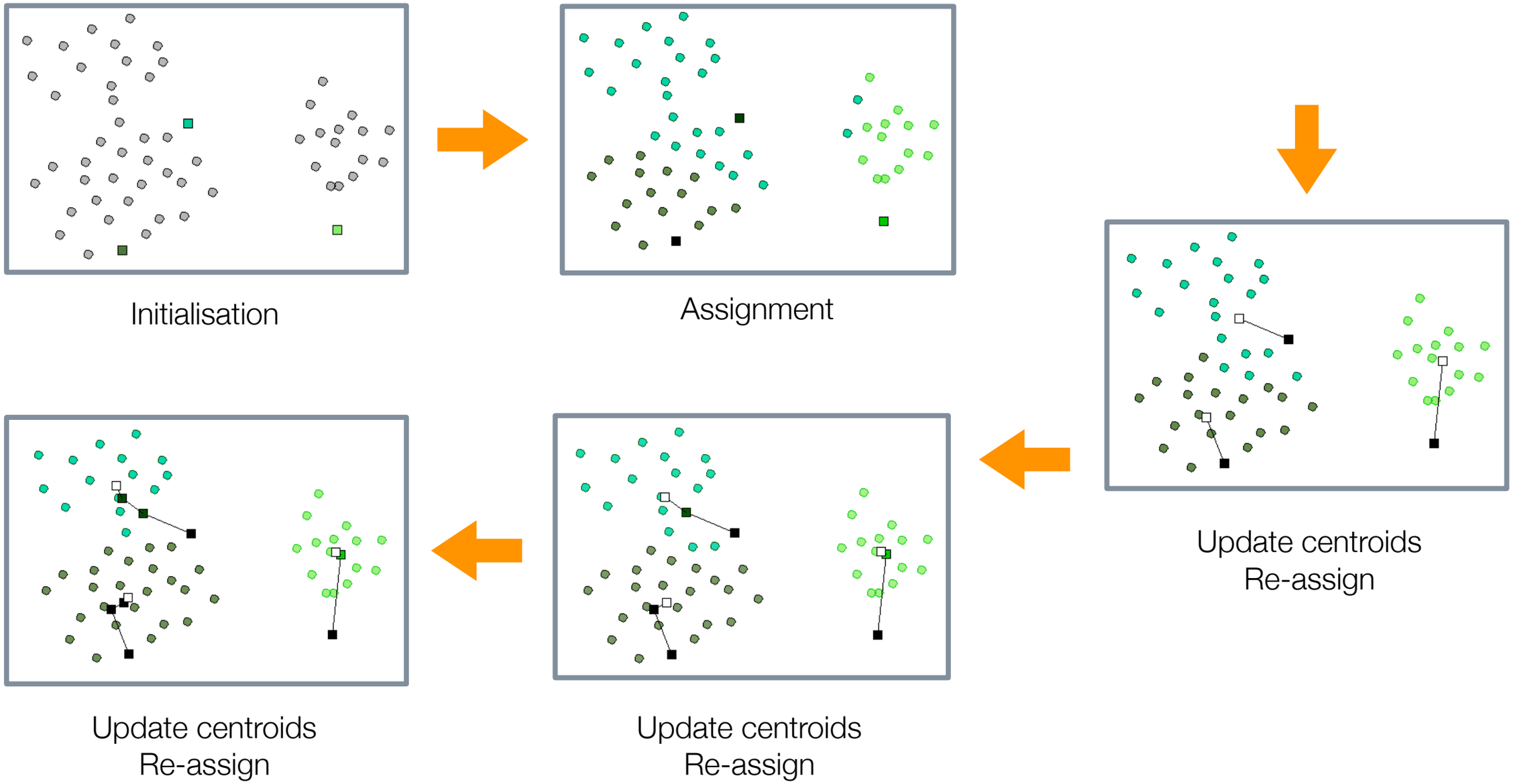
  1) Reassign items to their nearest cluster centroid

  2) Update the centroids based on the new assignments

- Repeatedly apply these two steps until the algorithm converges to a final result.

Example: *k*-Means Clustering

Simple example of several iterations of *k*-Means for *k*=3…



k-Means Clustering

Algorithm Summary:

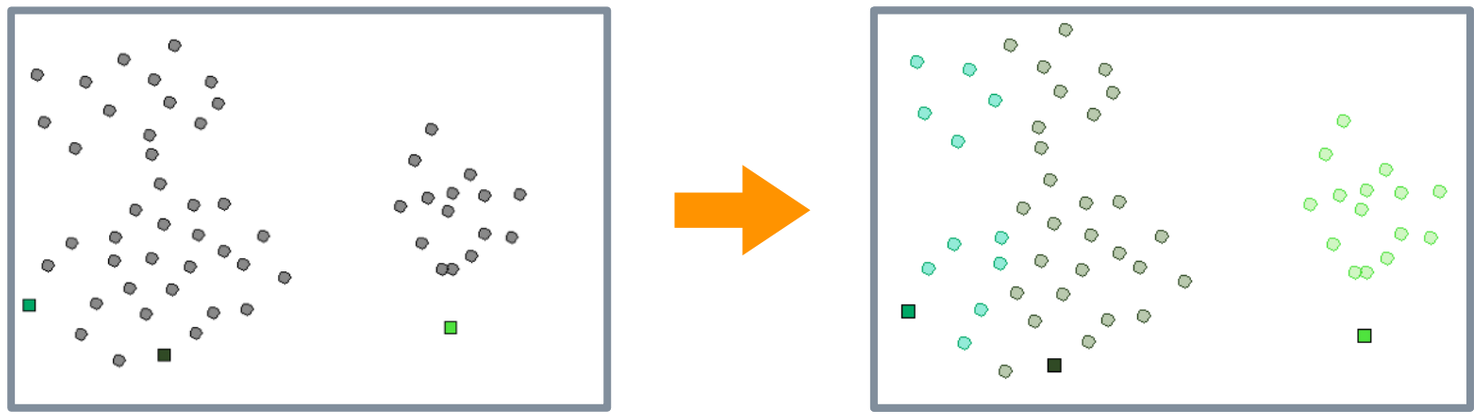
1. *Initialisation*: Select *k* initial cluster centroids (e.g. at random)
2. *Assignment step*: Assign every item to its nearest cluster centroid (e.g. using Euclidean distance).
3. *Update step*: Recompute the centroids of the clusters based on the new cluster assignments, where a centroid is the mean point of its cluster.
4. Go back to Step 2, until when no reassignments occur (or until a maximum number of iterations is reached).

- Key input parameter *k* - how many clusters?
  - *k* too low - “smearing” of clusters that should not be merged.
  - *k* too high - “over-clustering” of the data into many small, similar clusters.

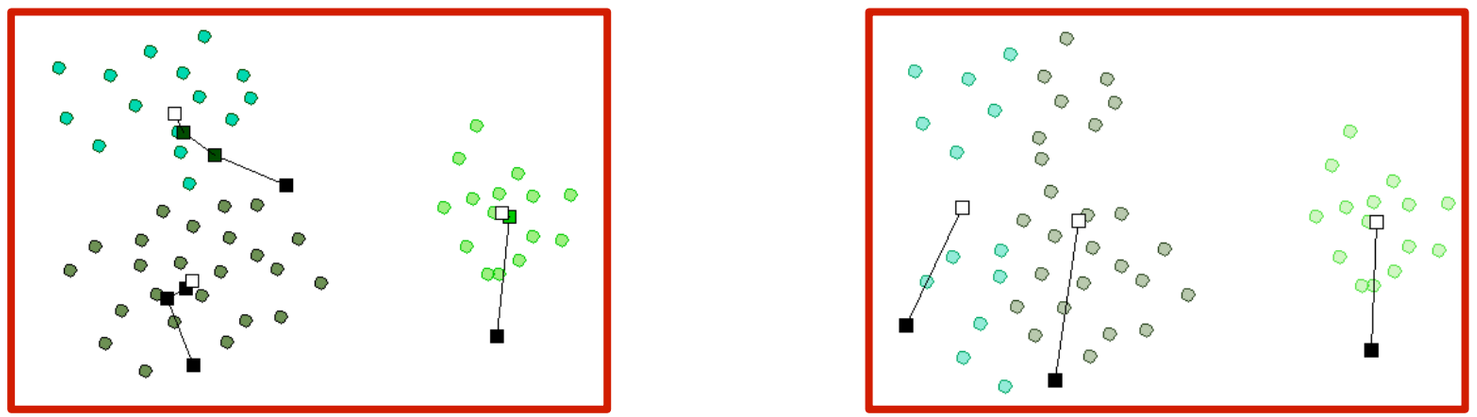
Cluster Initialisation

Results produced by *k*-Means are often highly dependent on the initial solution. Different starting positions can lead to different local minima - i.e. different clusterings of the same data.

New Initialisation for *k=3*

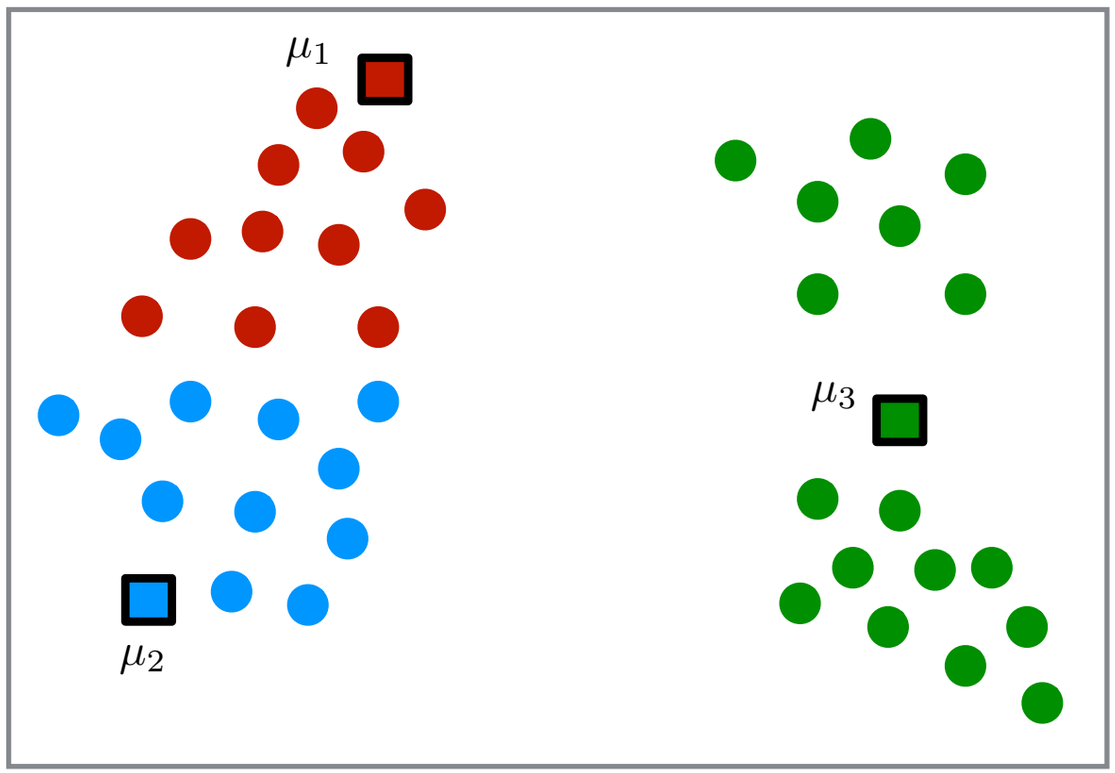


Previous Solution for *k=3*

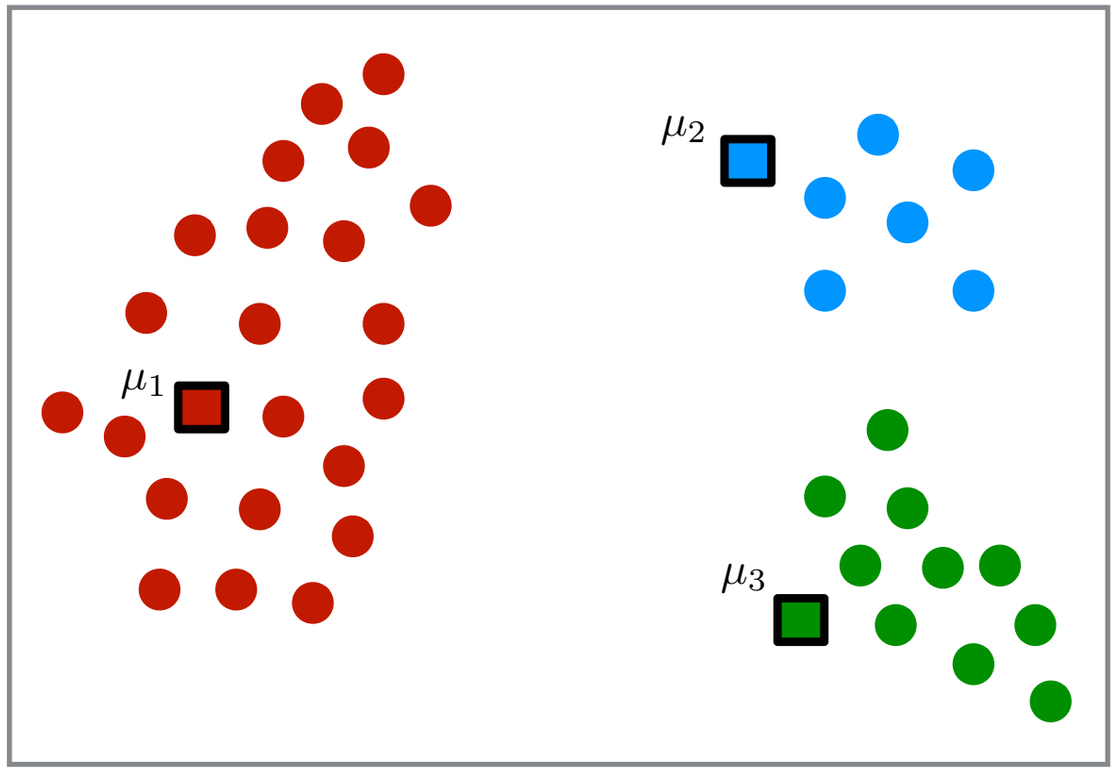


Cluster Initialisation

A poor choice of initial centroids will often lead to a poor clustering that is not useful. A better initialisation will lead to different clusters.



Initialisation 1



Initialisation 2

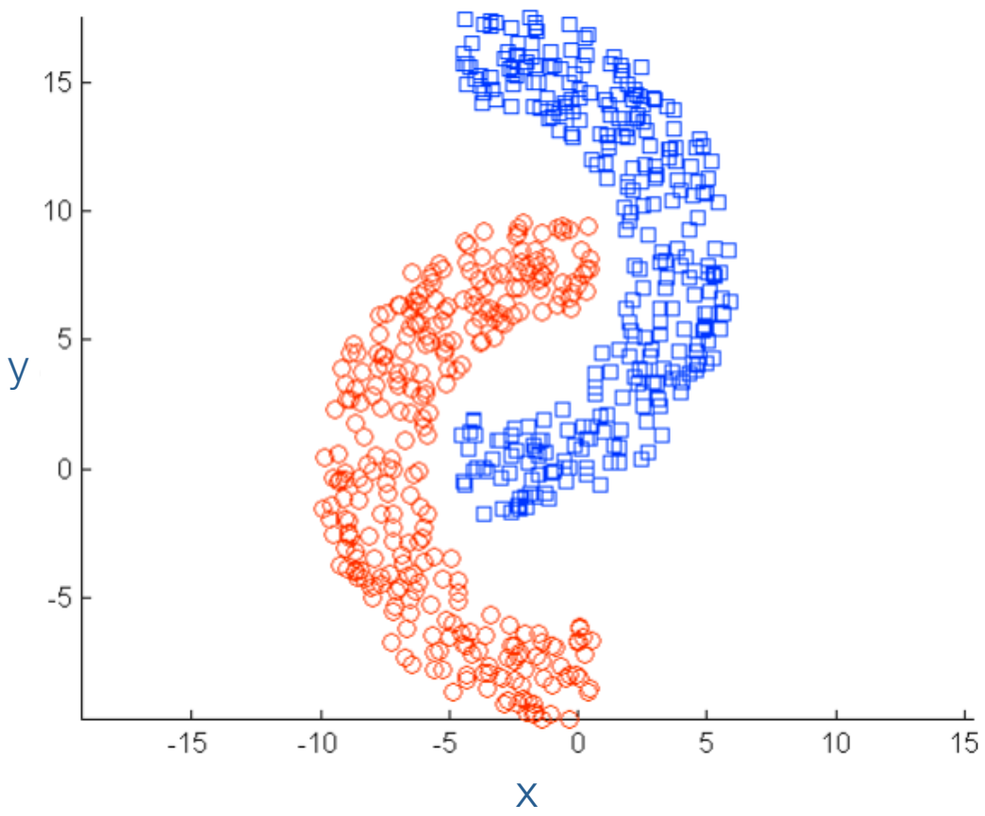
➡ Common strategy: Run the algorithm multiple times, select the solution(s) that scores well according to some validation measure.

Limitations of *k*-Means

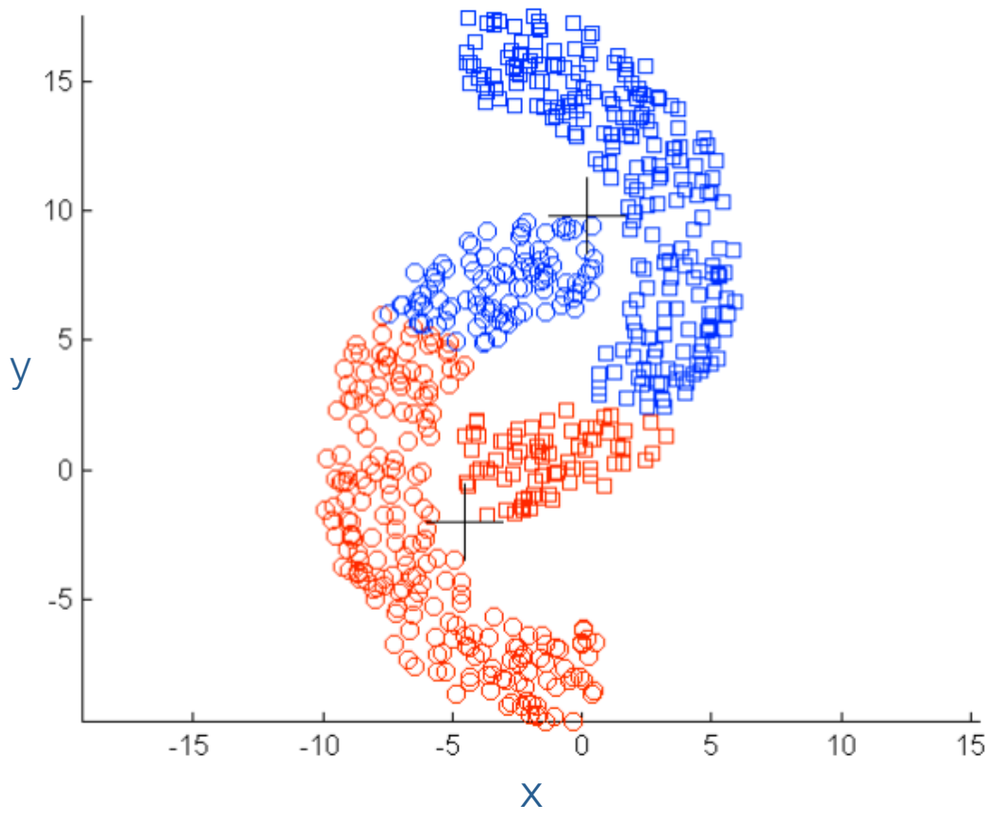
- **Advantages:**
  - Fast, easy to implement.
  - “Good enough” in a wide variety of tasks and domains.
- **Disadvantages:**
  - Must pre-specify number of clusters *k*.
  - Highly sensitive to choice of initial clusters.
  - Assumes that each cluster is spherical in shape and data examples are largely concentrated near its centroid.
  - Traditional objective can give undue influence to outliers.
  - Iterative process can lead to empty clusters, particularly for higher values of *k*.

Limitations of *k*-Means

Example: *k*-Means assumes that clusters are spherical in shape and data examples are largely concentrated near its centroid.



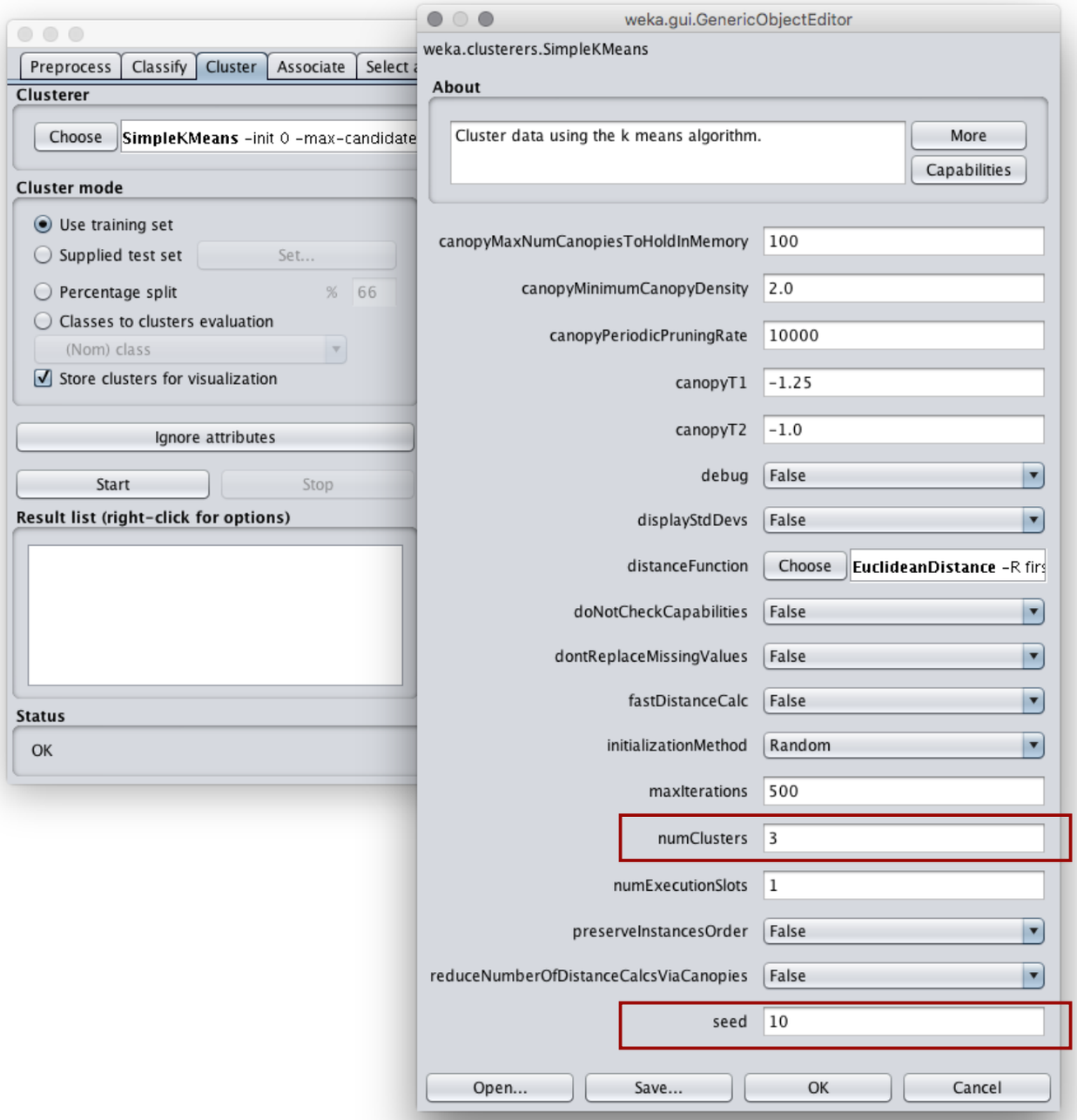
Original “correct” groups in the data



Clusters identified by *k*-means for *k=2*

k-Means Clustering in WEKA

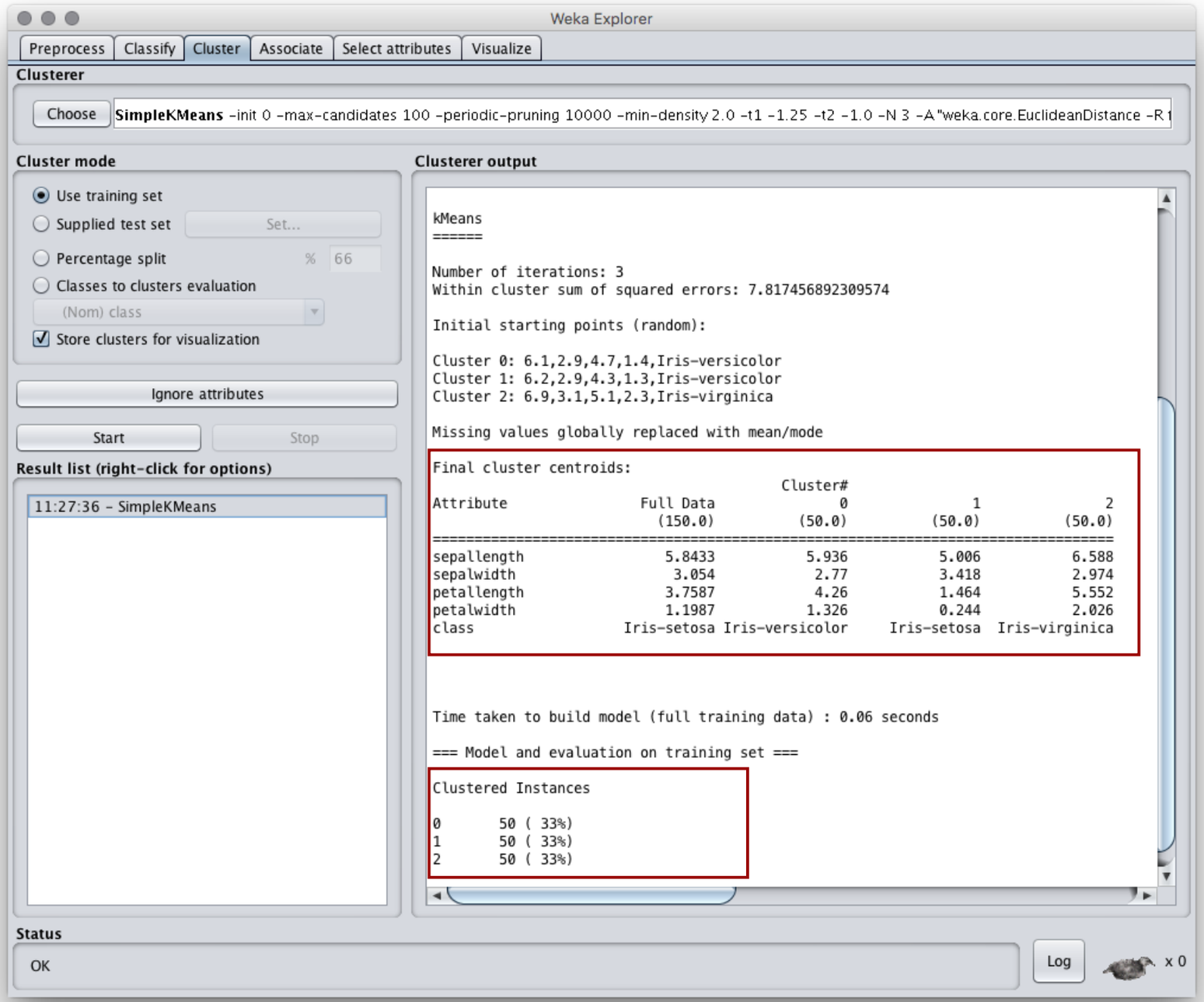
In the Weka *Cluster* tab, choose *SimpleKMeans* as the clusterer. Right-click on *SimpleKMeans* name to change parameters.



Example: iris.arff

k-Means Clustering in WEKA

Weka reports: number of iterations, SSE, cluster centroids, sizes and numbers of resulting clusters.



Summary

- Part 1
  - Supervised v Unsupervised Learning
  - Partitional Clustering
    - *k*-Means clustering
    - Cluster initialisation
- Part 2
  - Hierarchical Clustering
    - Agglomerative algorithms
    - Cluster metrics
    - Divisive algorithms
  - Cluster Validation