Data Science for Business

Unsupervised Data Mining, Clustering and Cluster Validation

Asst. Prof. Teerapong Leelanupab (Ph.D.)
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang (KMITL)

Week 13

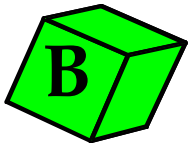
Overview

- 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

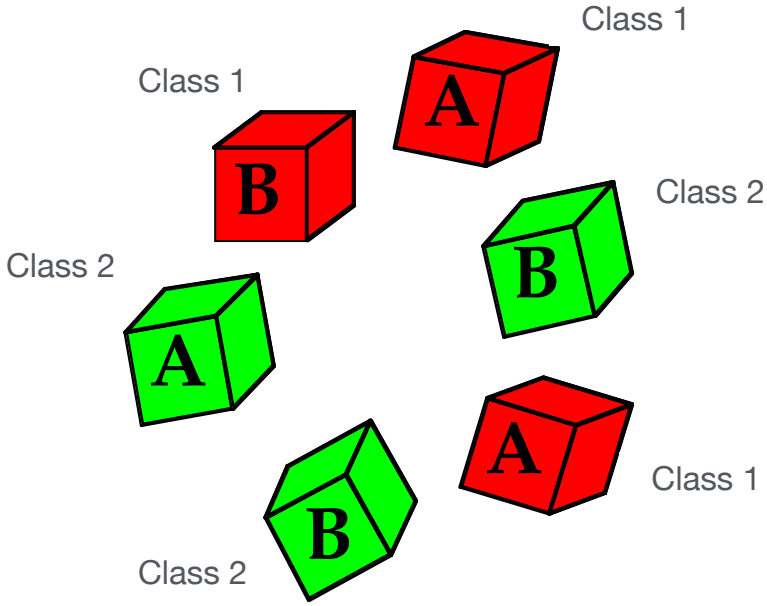
Supervised Learning

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.



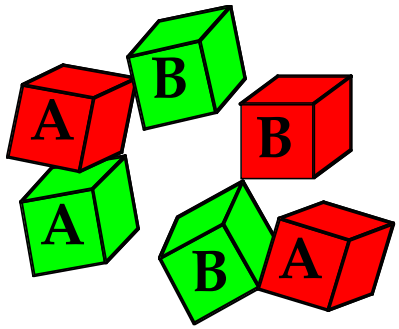
For many tasks, annotated class labels for data are not available - either unknown or too expensive to obtain.

Unsupervised Learning

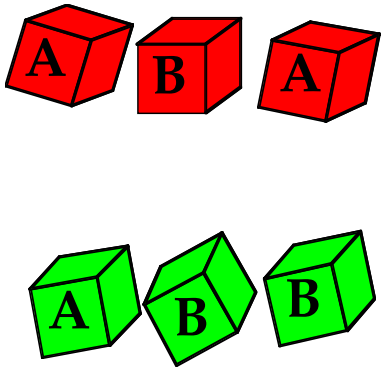
Unsupervised learning algorithms attempt to identify patterns by relying solely on the intrinsic characteristics of the data, without referring to any class information.

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

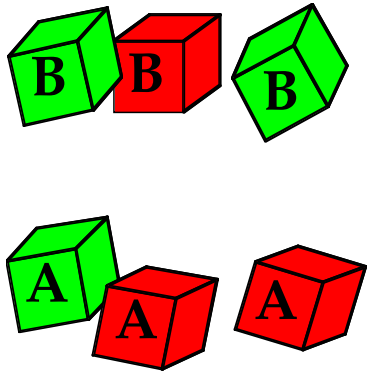
Organise these blocks into groups



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



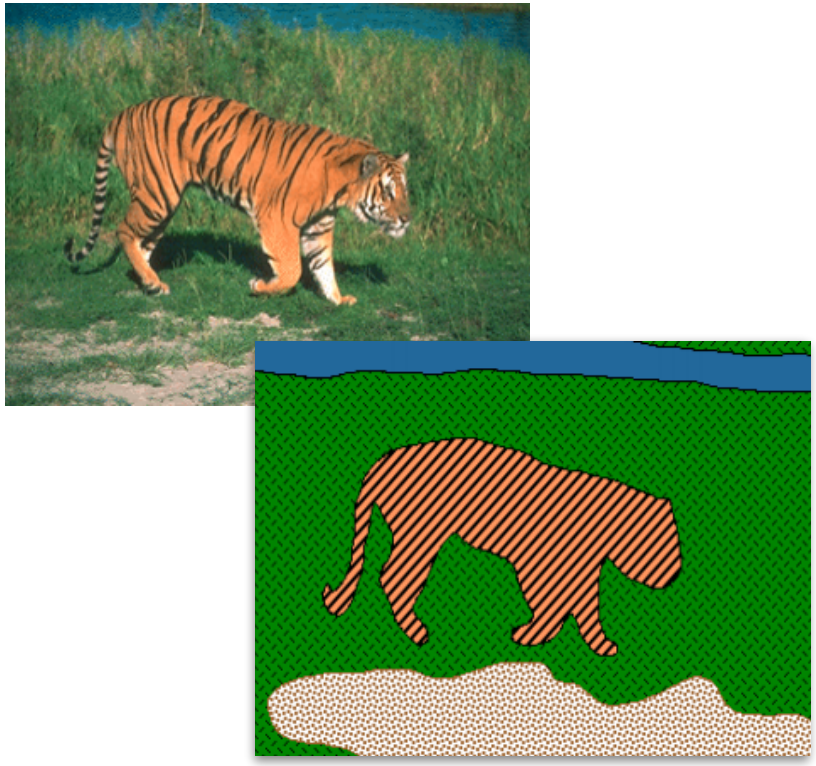
Grouping 1



Grouping 2

Unsupervised Learning: Applications

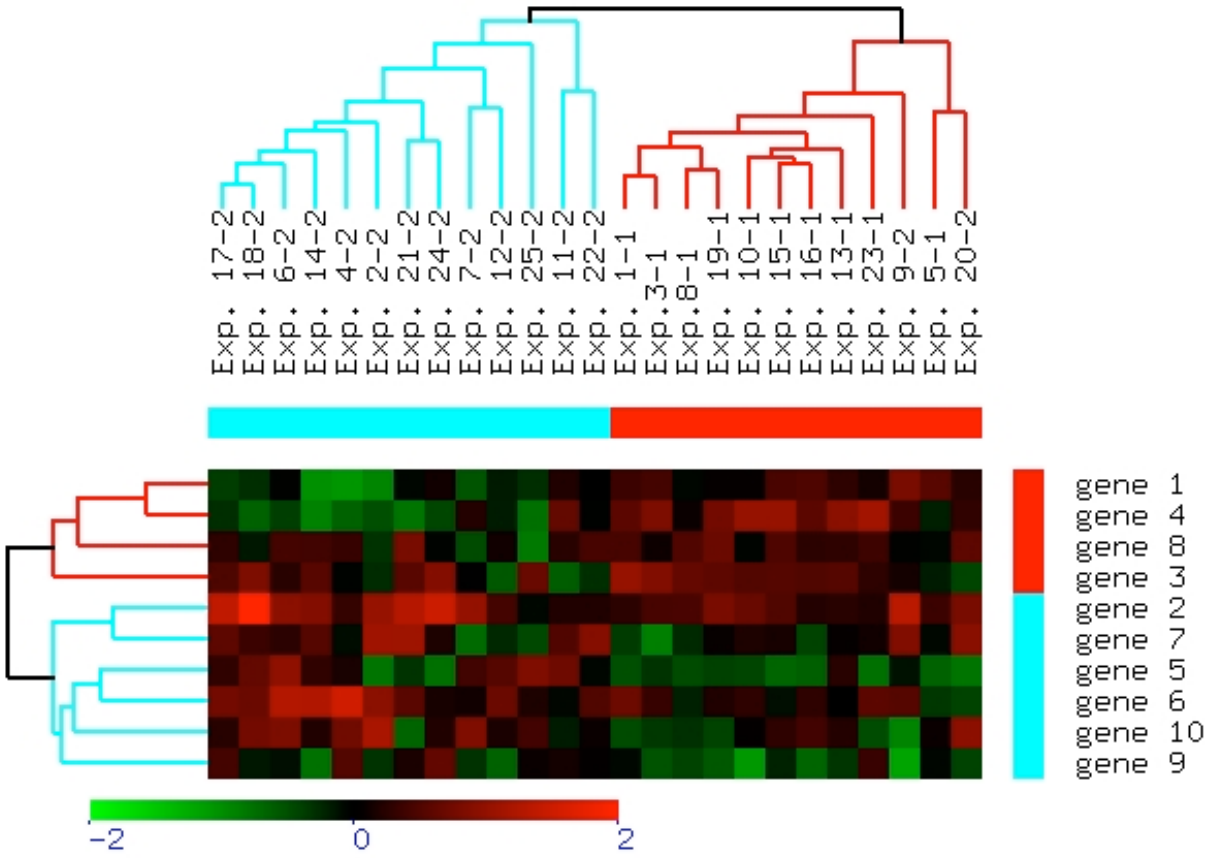
Image segmentation: Unsupervised task in computer vision that attempts to automatically split an image into regions with similar colour or texture, or both. Aim is to partition the image into its constituent “objects”.



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).

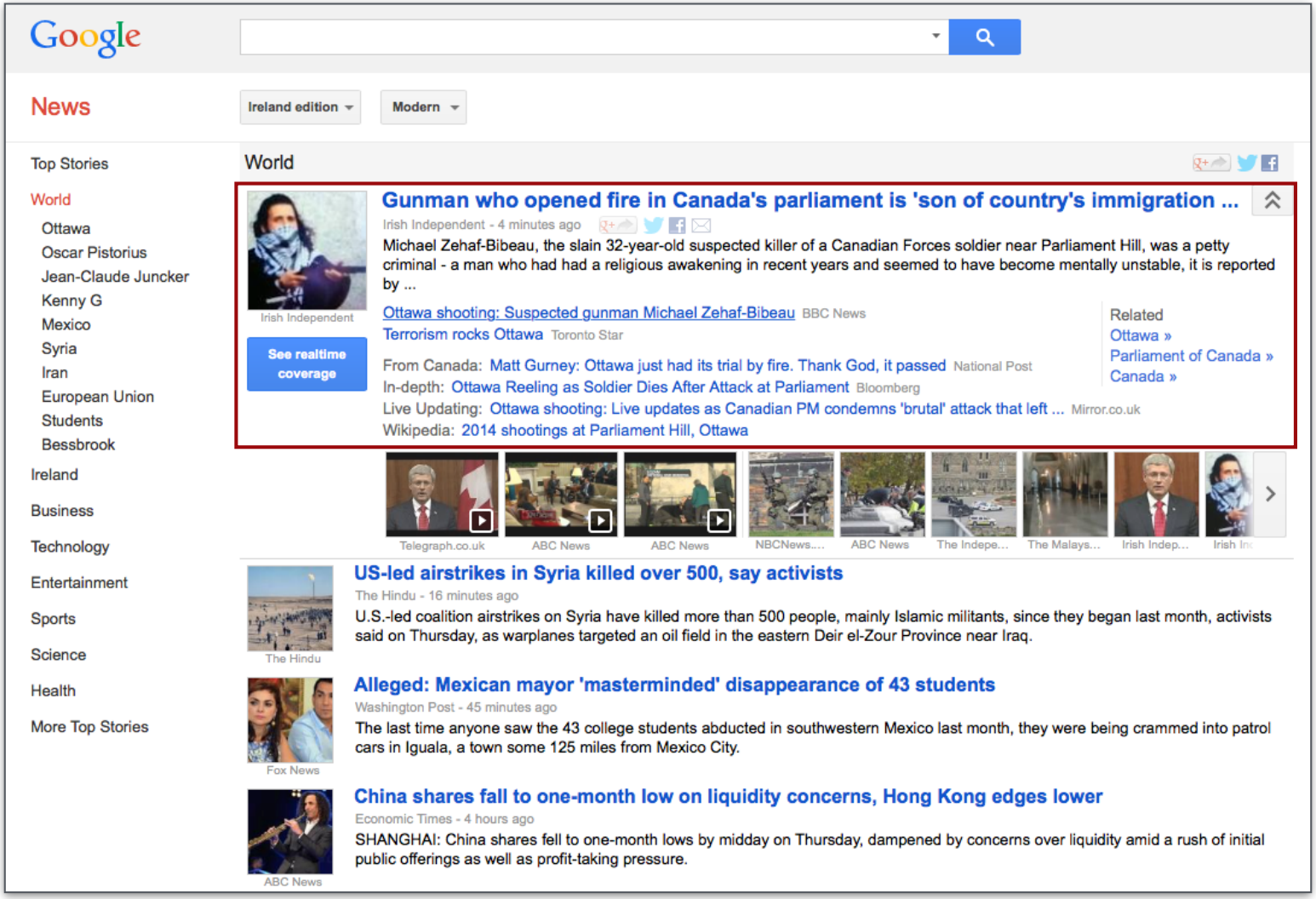


Unsupervised Learning: Applications

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).

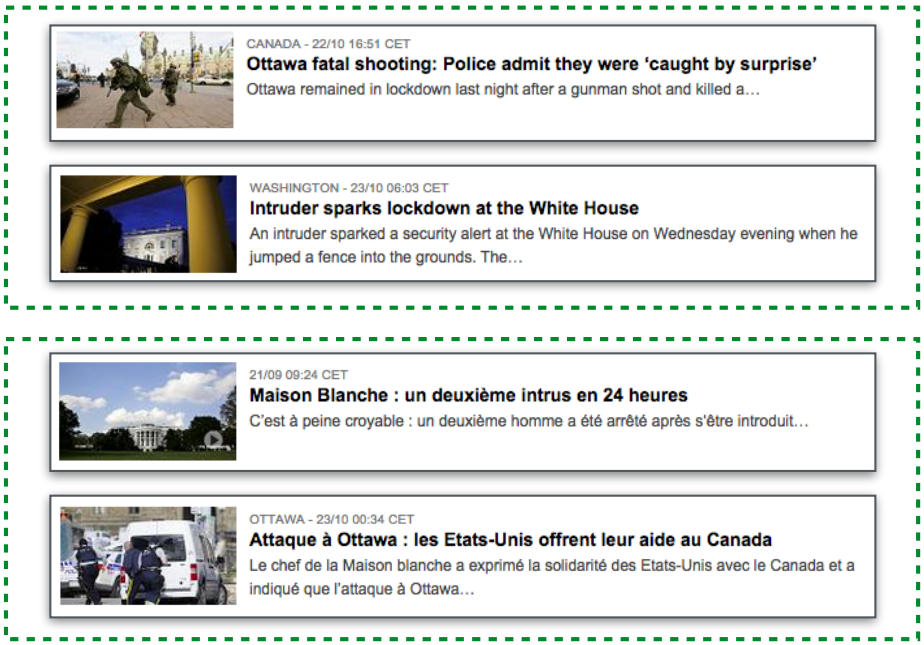


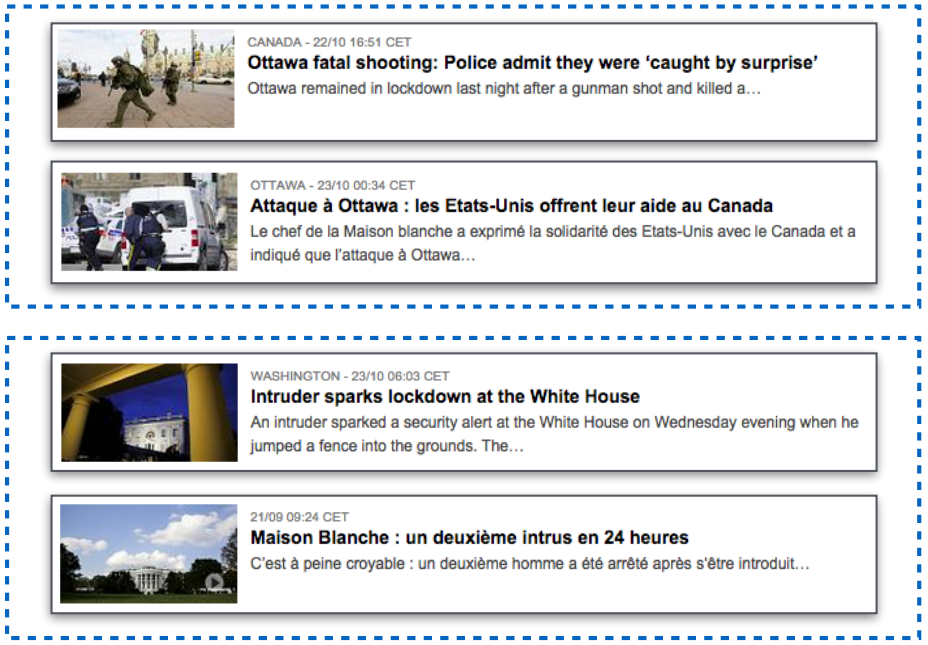
Definition: Clustering

- Clustering is another application of our fundamental notion of similarity
- The basic idea is that we want to find groups of objects (consumers, businesses, etc.), where the objects within groups are similar,
    - but the objects in different groups are not so similar

Clustering

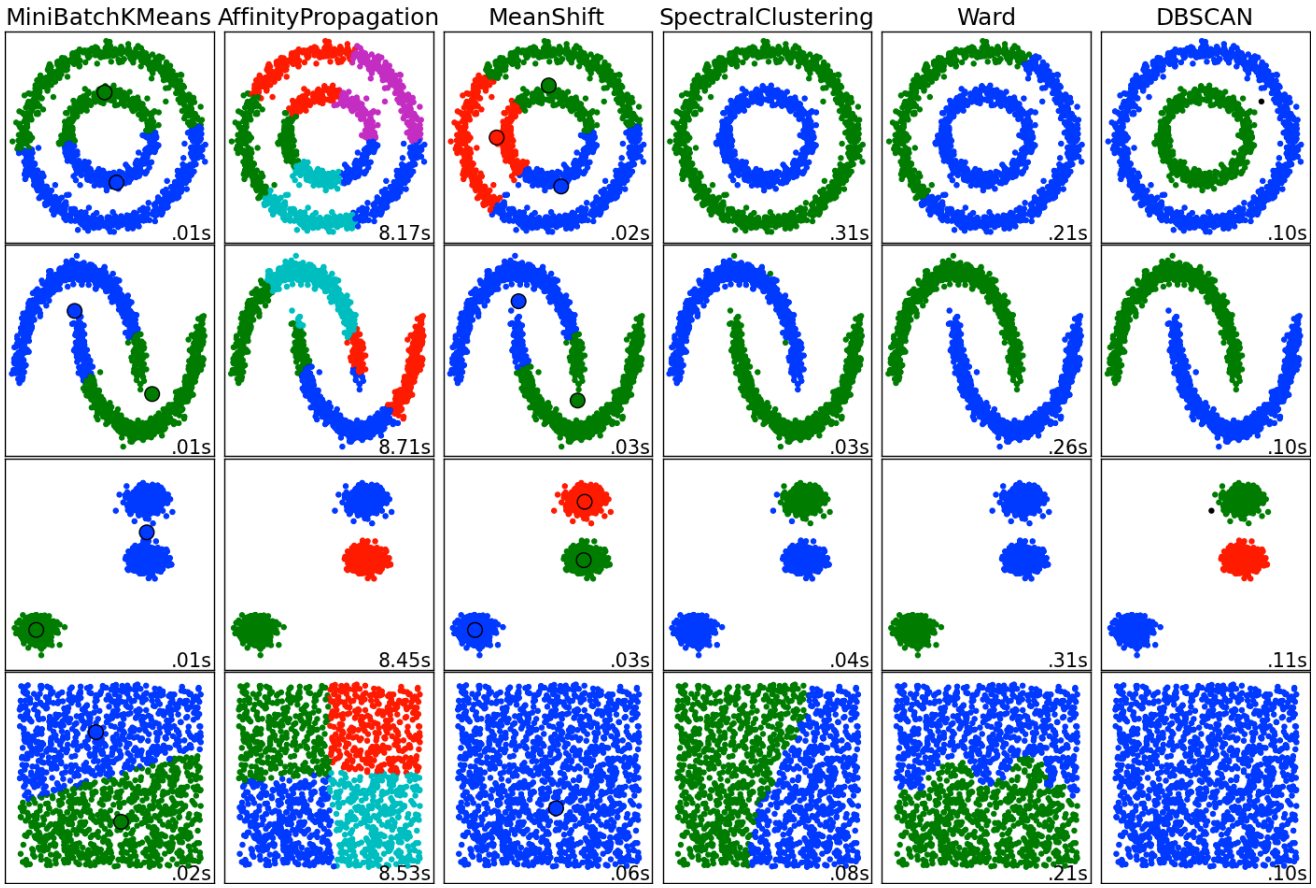
- Clustering algorithms organise data into groups (“clusters”) in the absence of any external information.
  - ➡ No labelled training examples to learn from.
  - ➡ Generally won’t know in advance how many clusters are in the data.
- Different clusterings can reveal different things about the same data. 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.

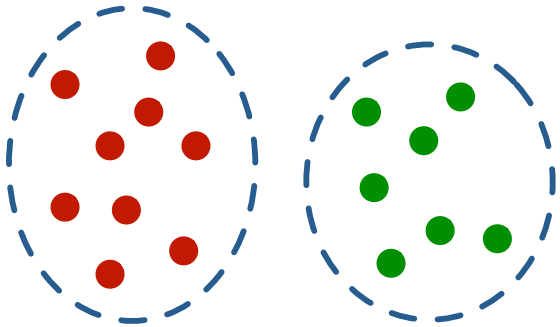


http://scikit-learn.org/stable/modules/clustering.html

Clustering

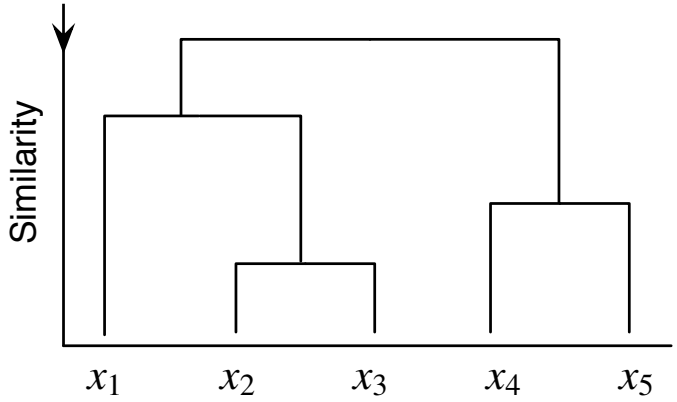
- **General goal**: Assign similar items to the same cluster, keep dissimilar items apart.
- Algorithms employ different definitions of similarity/dissimilarity and objective function for determining a "good" cluster.

Partitional Algorithms



Build a "flat" clustering of the data all at once

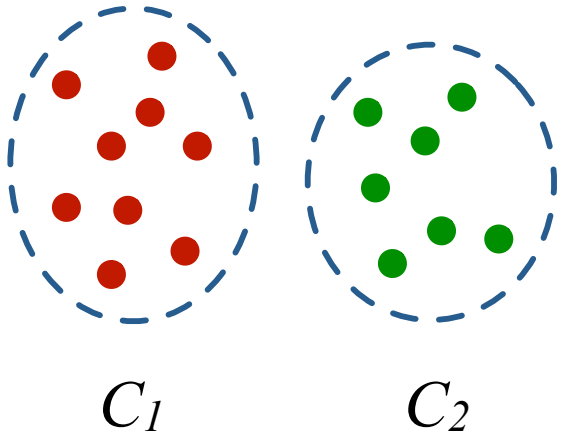
Hierarchical Algorithms



Gradually build a nested tree structure of clusters

Partitional Clustering

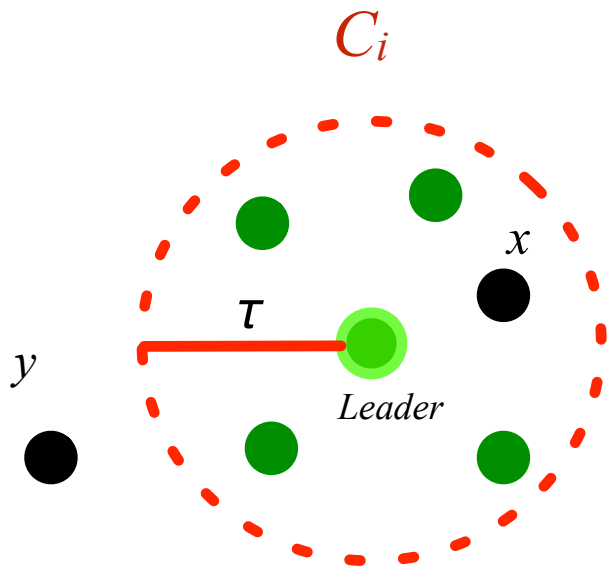
- Attempt to directly decompose a data set into a “flat” grouping consisting of a number of disjoint (non-overlapping) clusters.
- Usually pre-specify number of clusters *k*, although some methods adaptively add/remove clusters.
- Start with an initial set of *k* clusters, often chosen at random.
- Use 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.



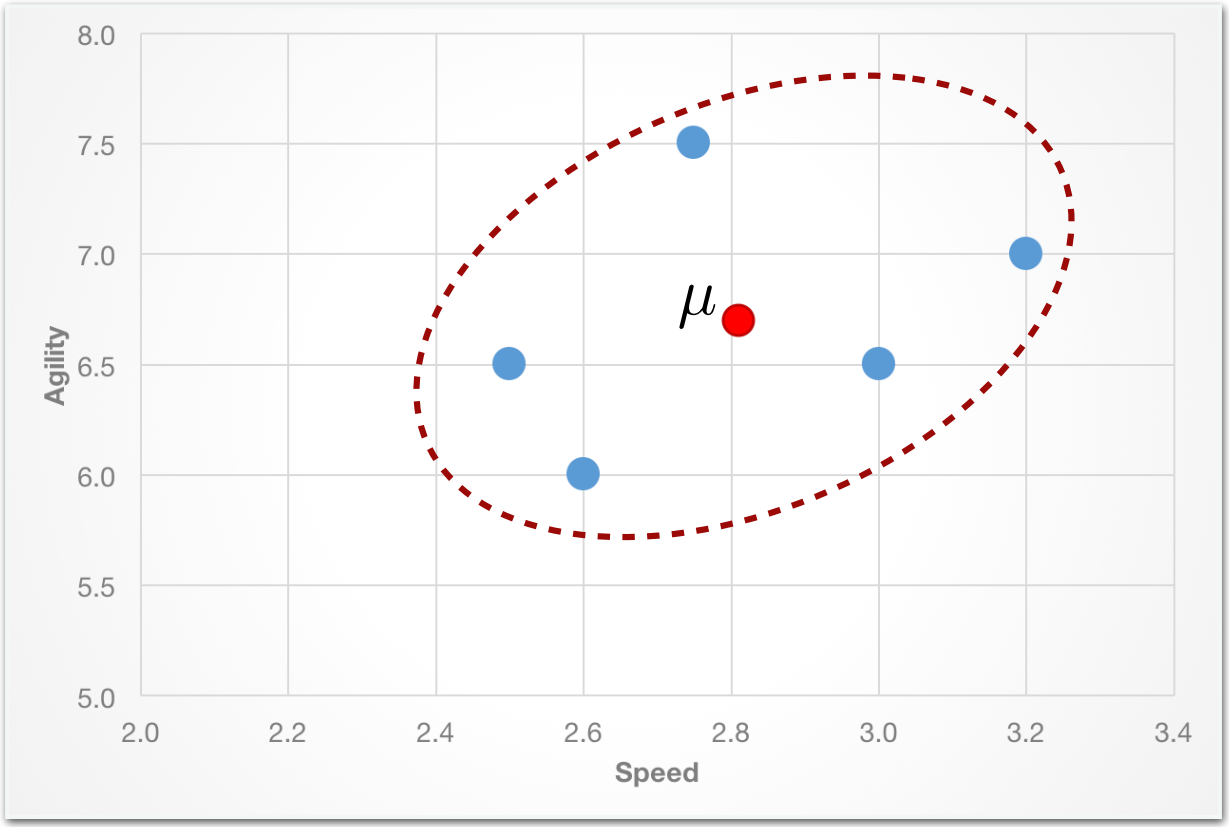
k-Means Clustering

- 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

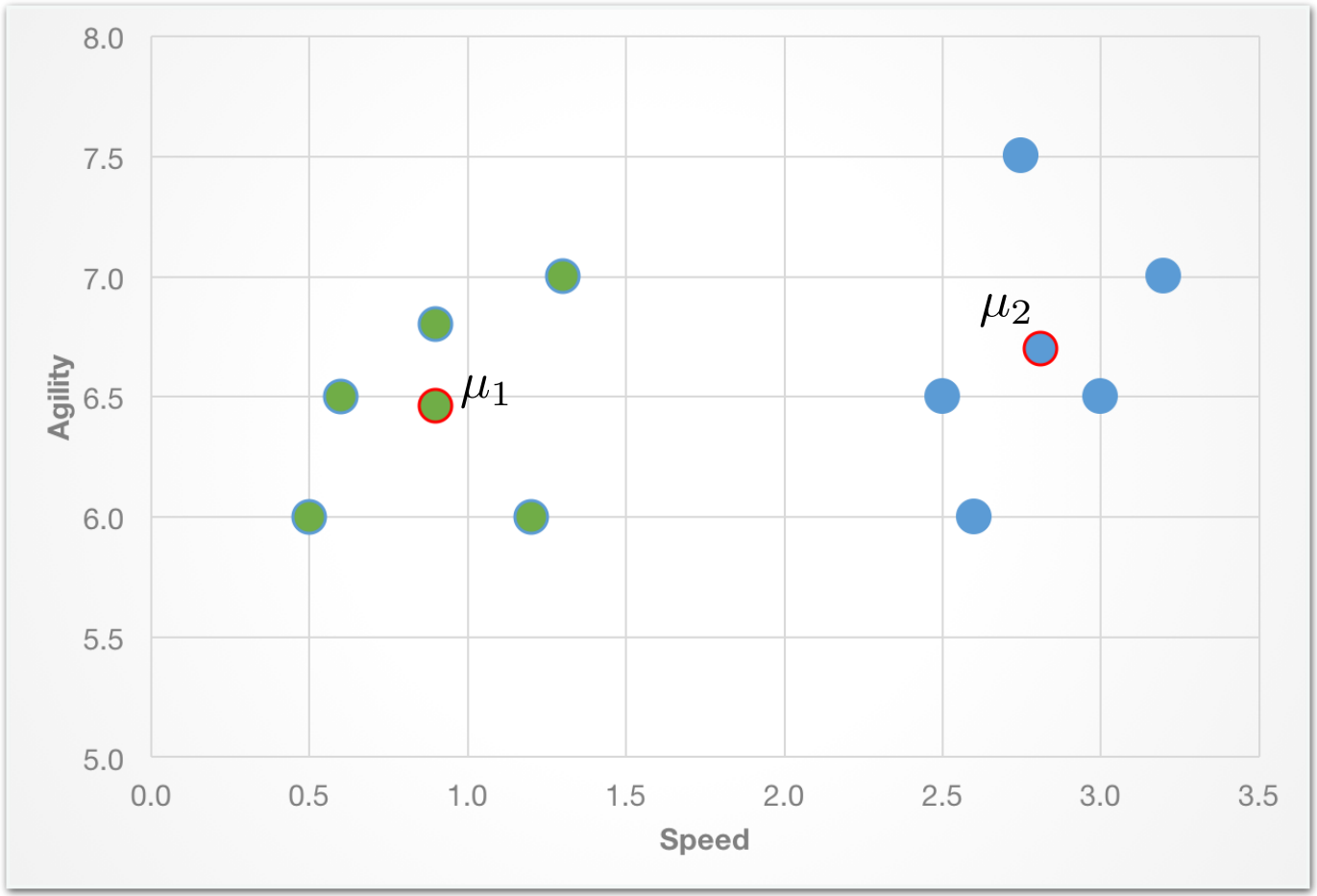
$(2.6 + 3.0 + 2.5 + 3.2 + 2.8)/5 = 2.82$

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



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

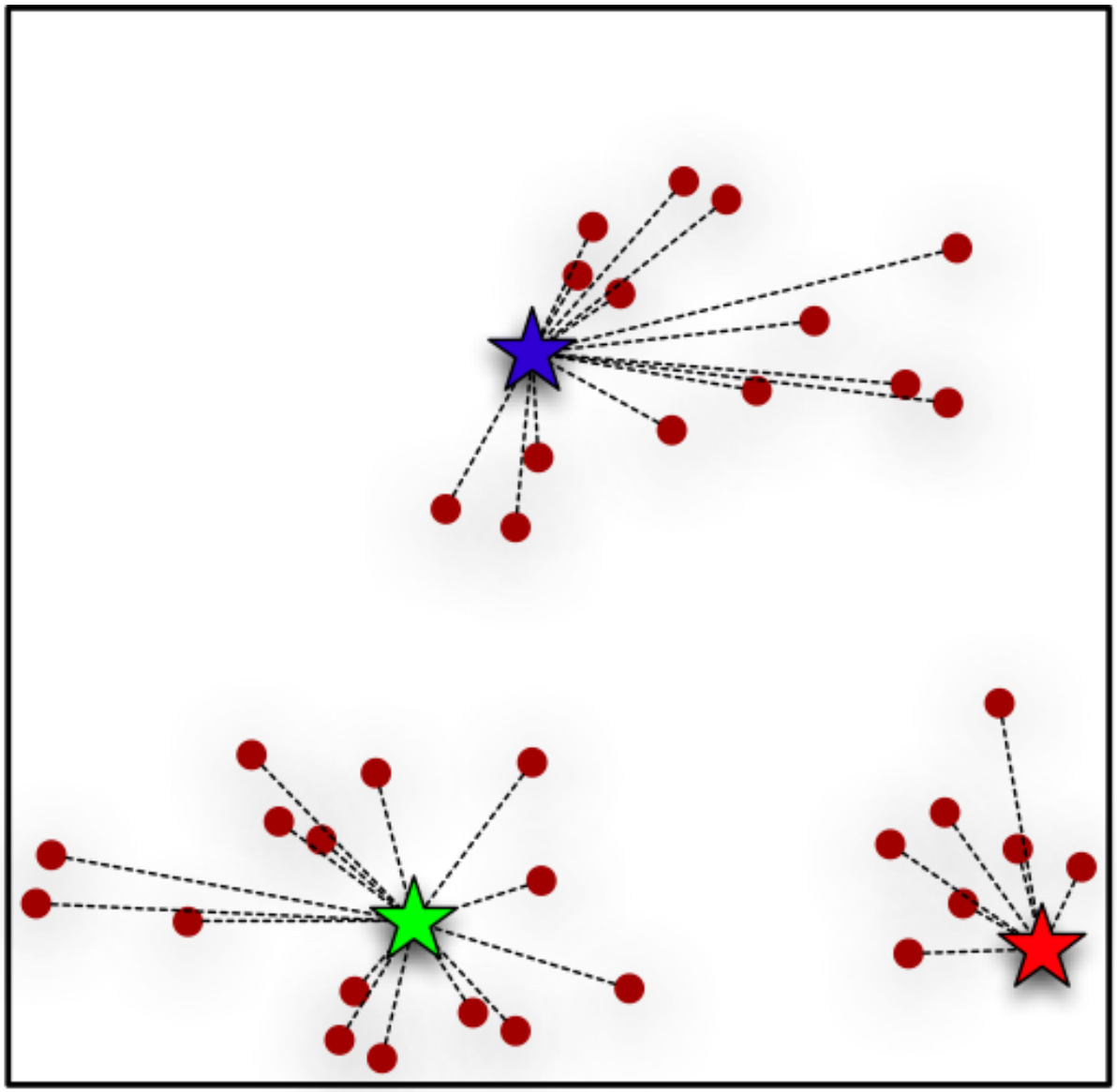
- *k*-Means tries to reduce SSE via a two step iterative process:

  1) Reassign items to their nearest cluster centroid

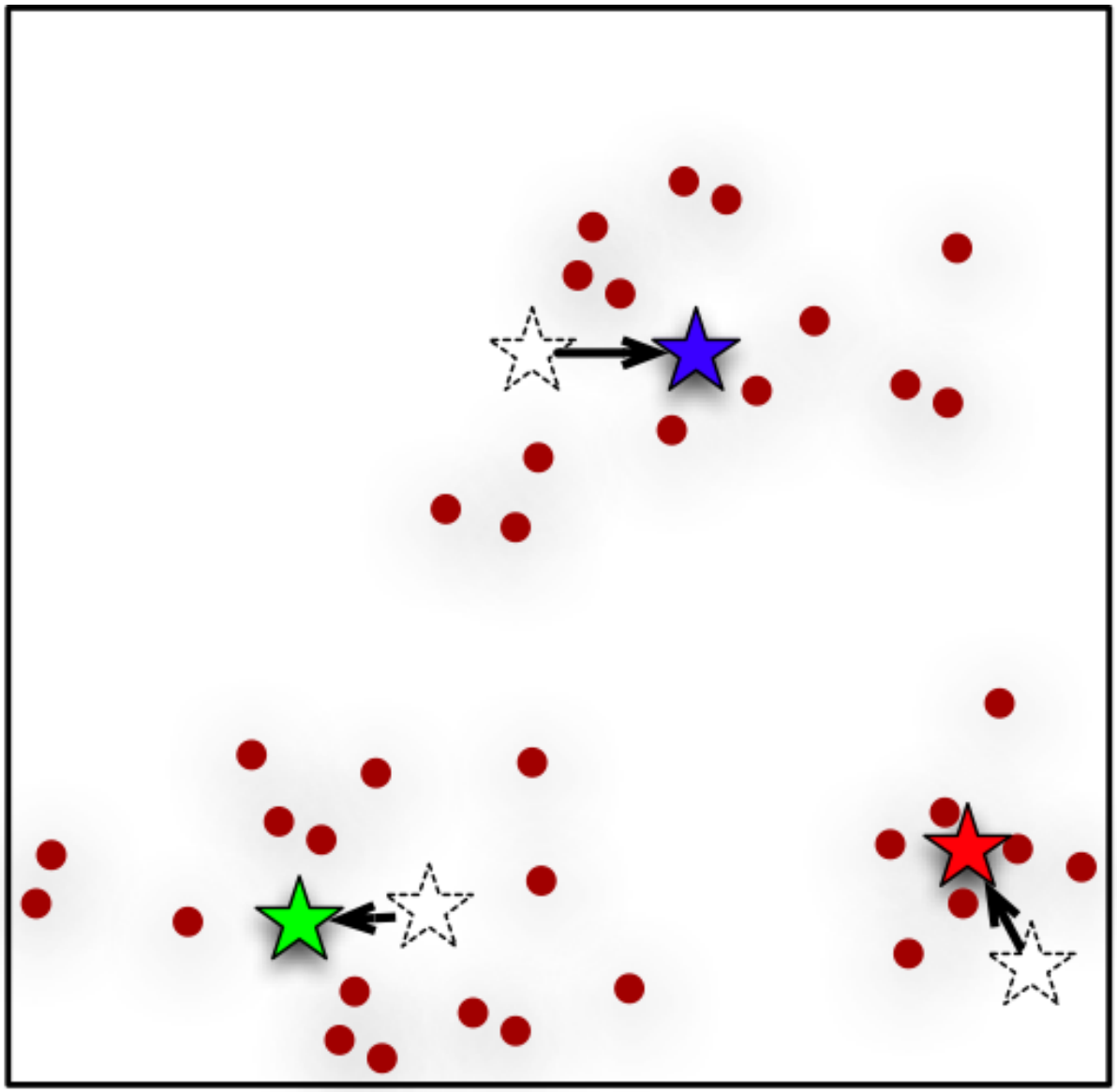
  2) Update the centroids based on the new assignments

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

Clustering Around Centroids

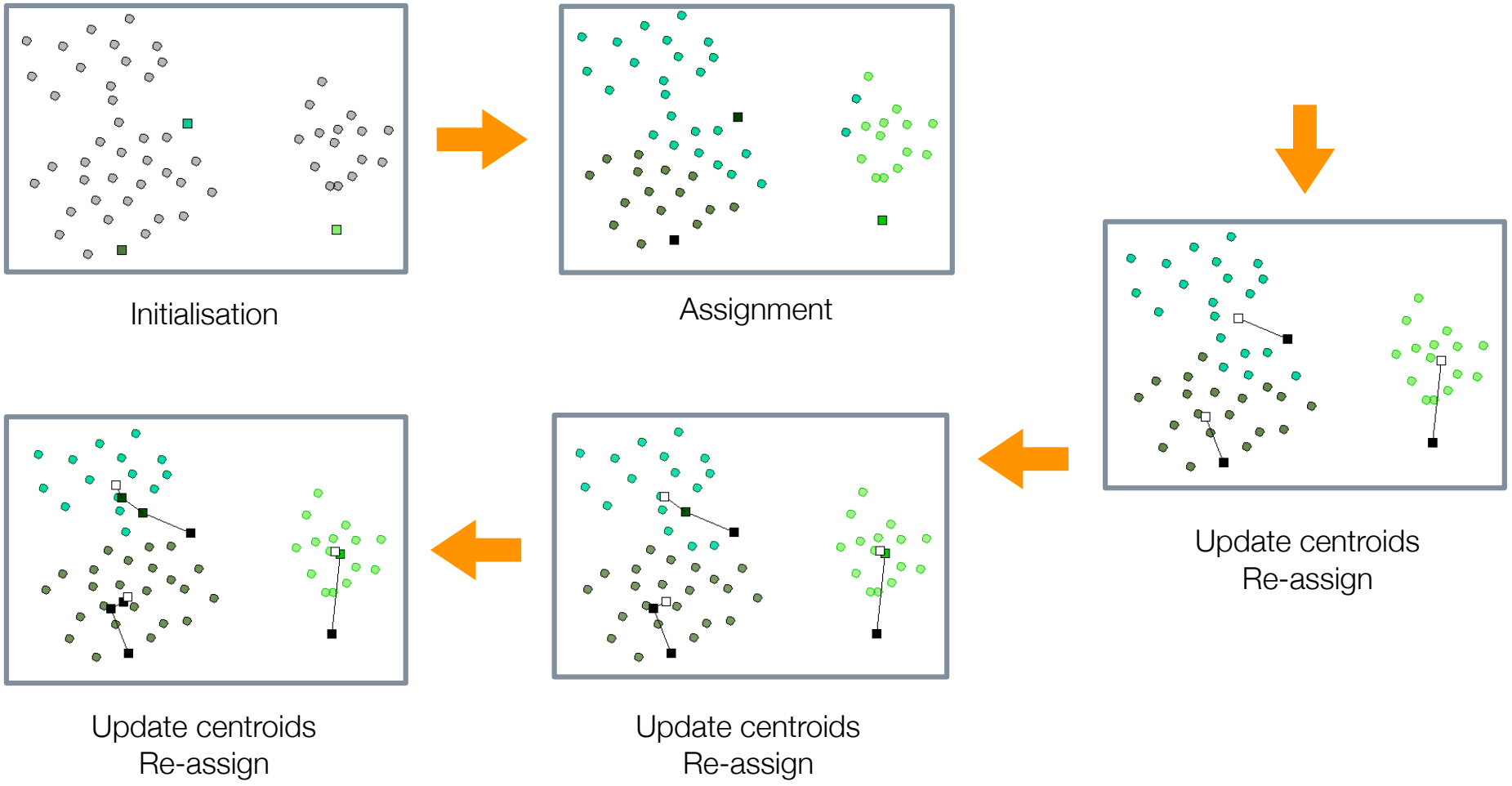


Clustering Around Centroids

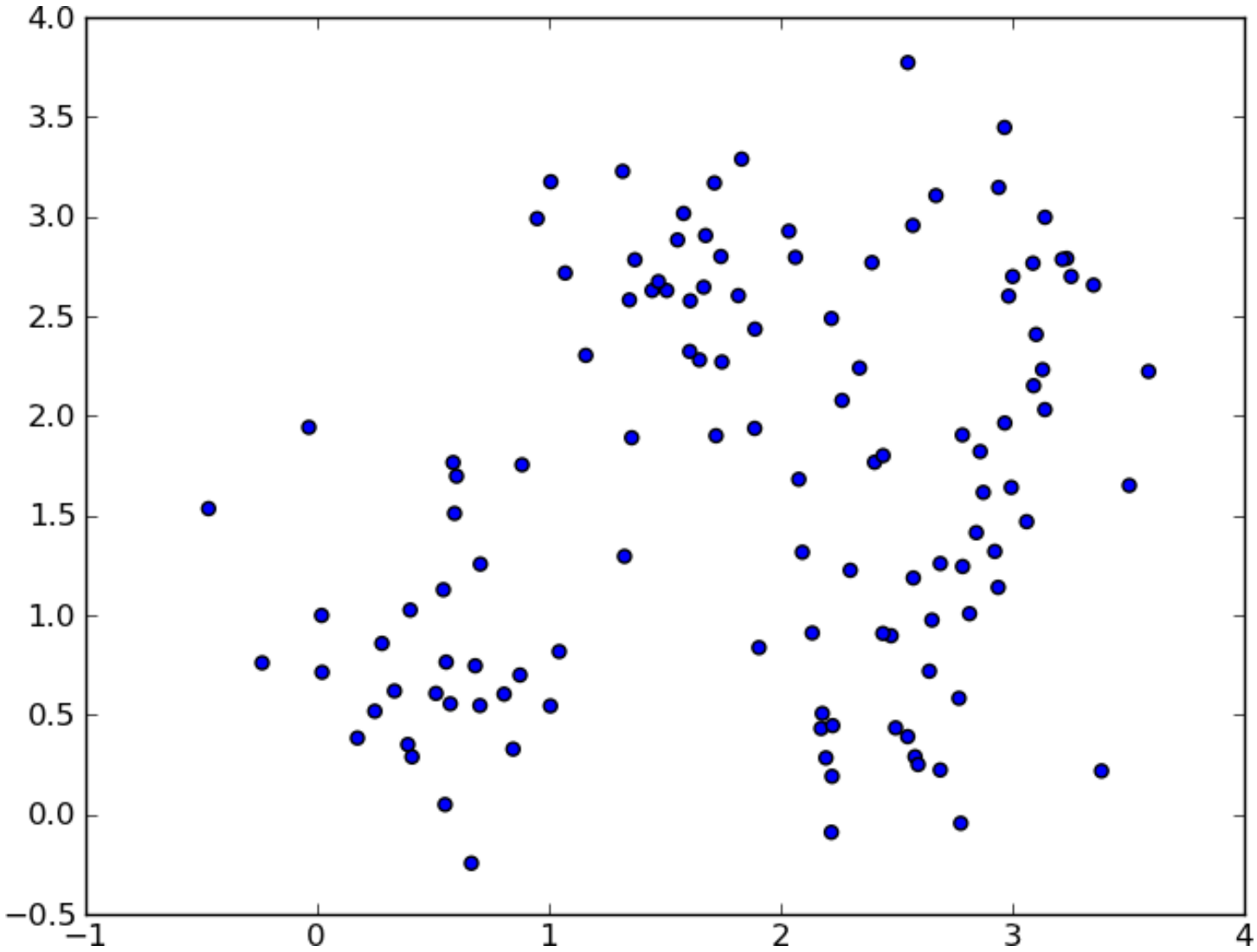


Example: *k*-Means Clustering

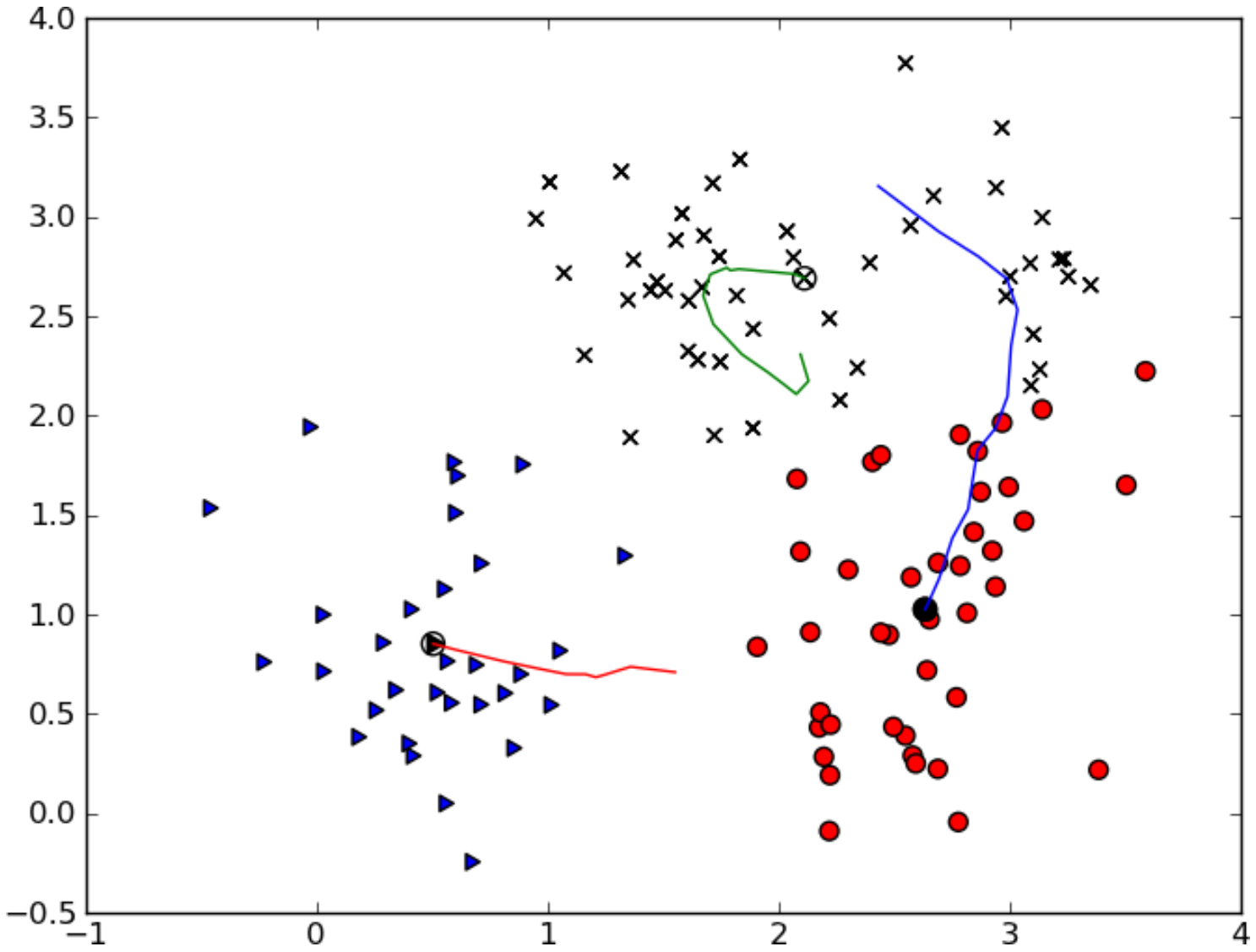
Simple example of several iterations of *k*-Means for *k*=3…



Example: k-means Clustering



Example: k-means Clustering



Clustering

- k-means clustering is efficient
- $k = ?$

k-Means Clustering

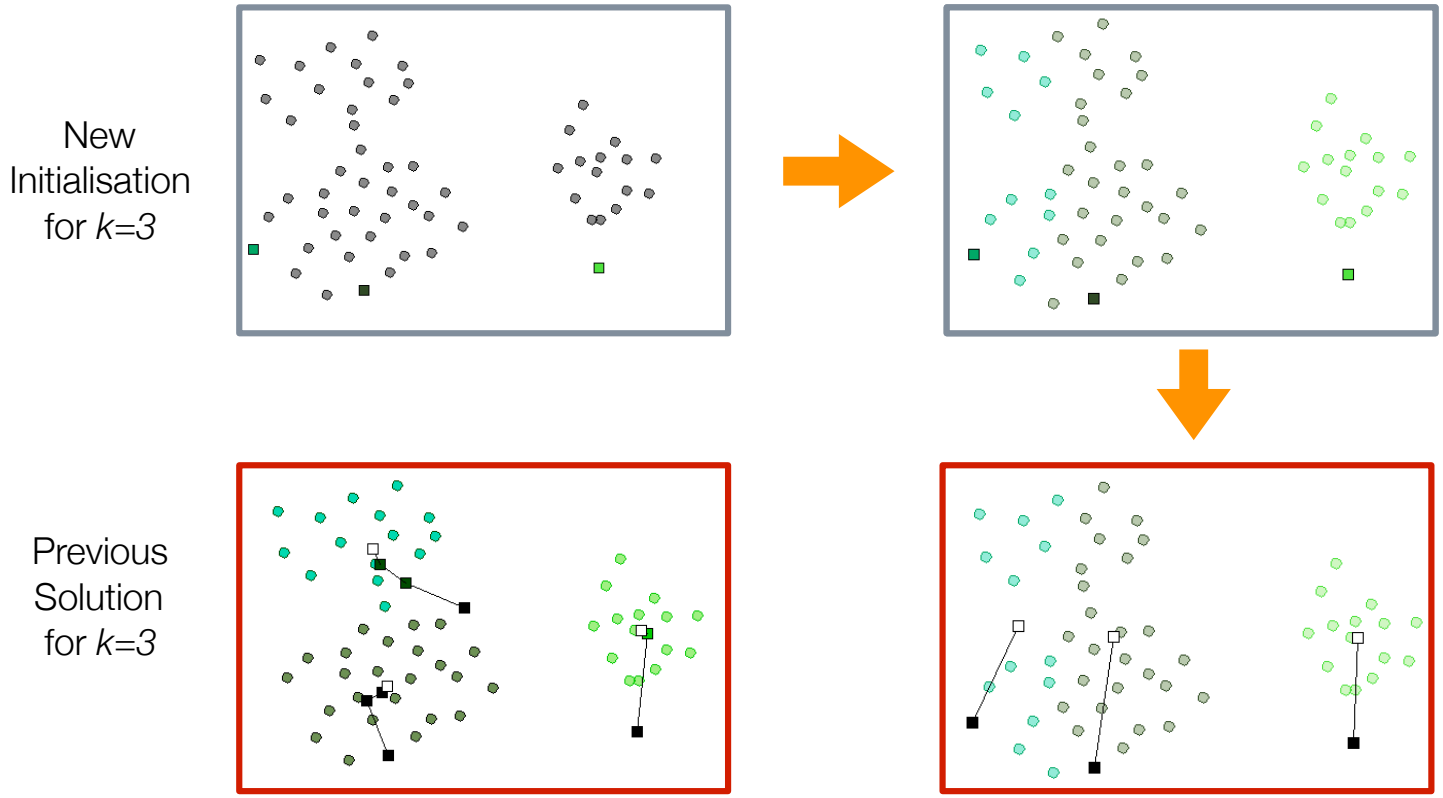
Algorithm Summary:

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.

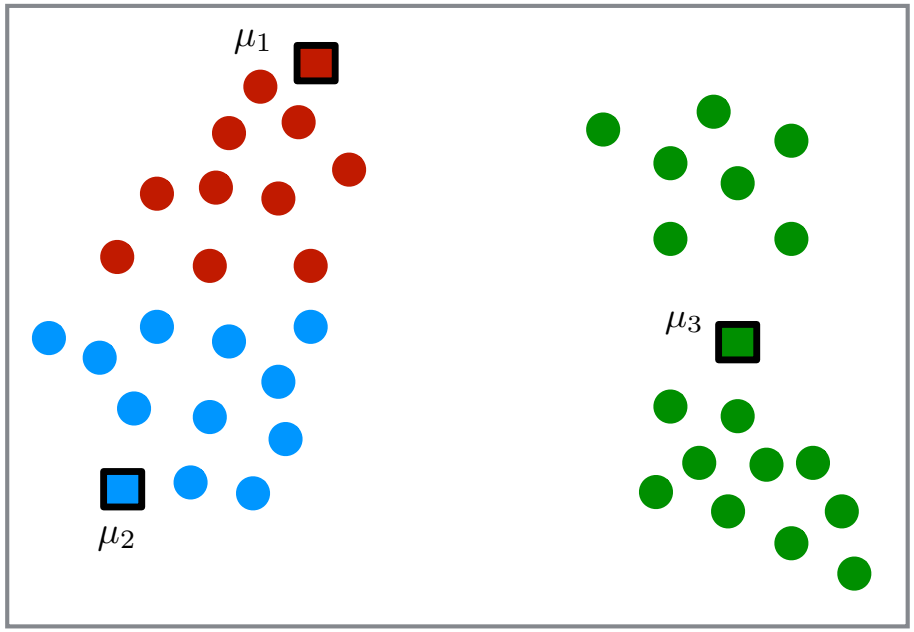
Cluster Initialisation

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.

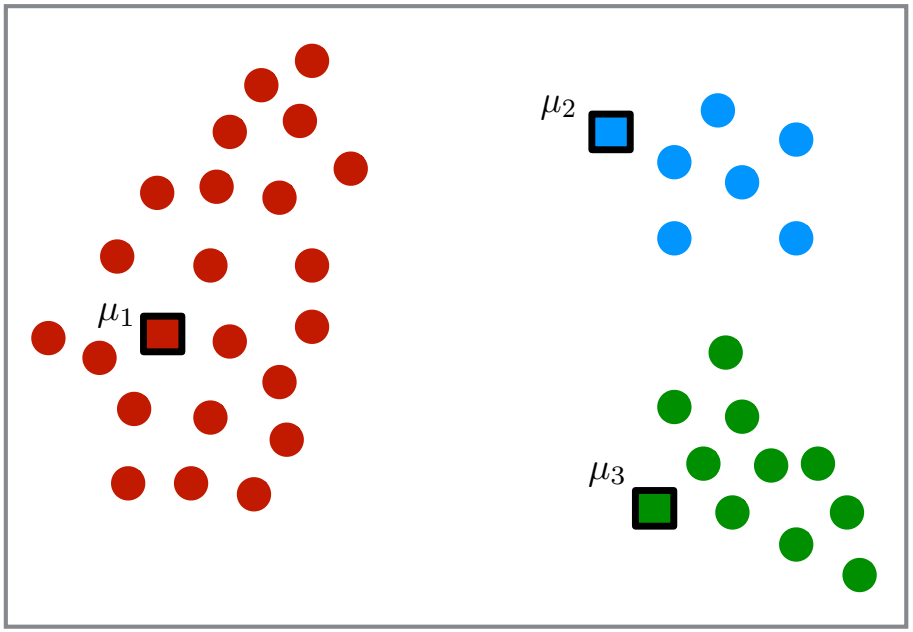


Cluster Initialisation

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

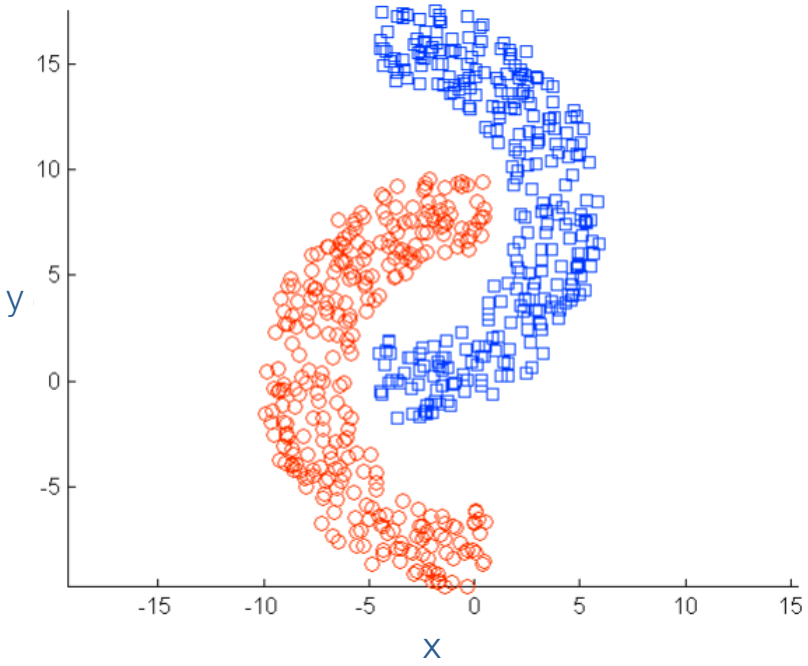
➡ Common strategy: Run the algorithm multiple times, select the solution(s) that scores well according to some validation measure.

Limitations of *k*-Means

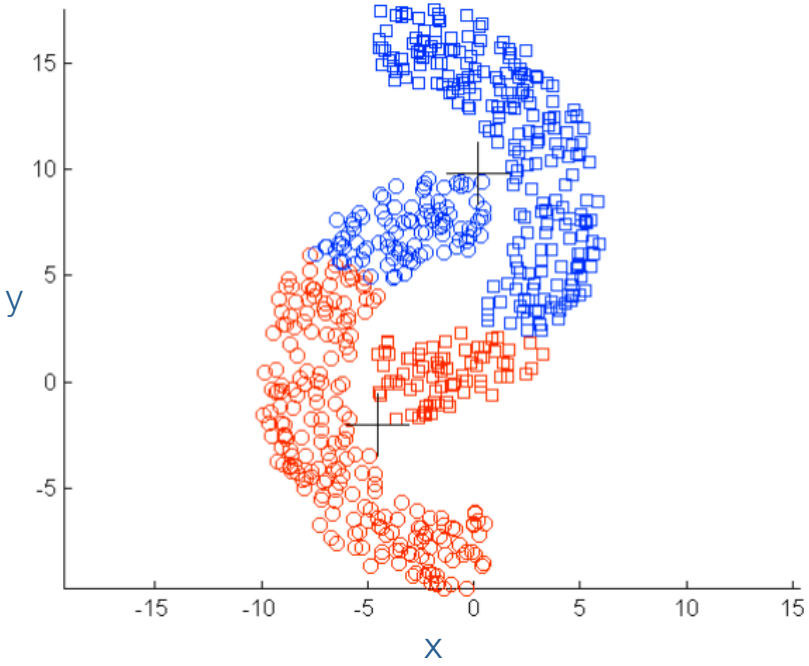
- **Advantages:**
  - Fast, easy to implement.
  - “Good enough” in a wide variety of tasks and domains.
- **Disadvantages:**
  - Must pre-specify number of clusters *k*.
  - Highly sensitive to choice of initial clusters.
  - Assumes that each cluster is spherical in shape and data examples are largely 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*

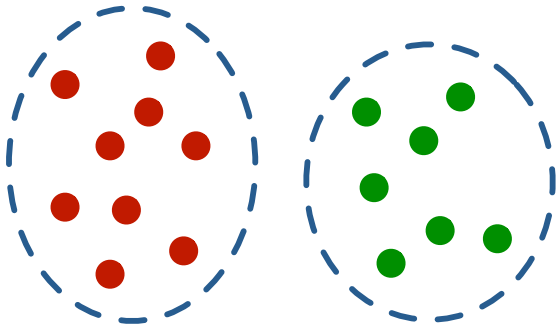
Overview

- 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

Clustering

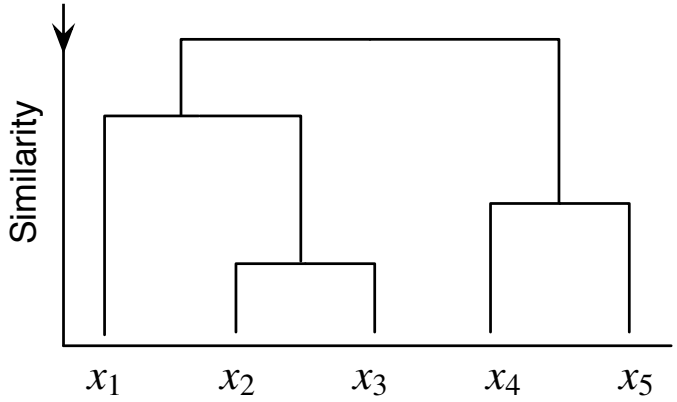
- **General goal**: Assign similar items to the same cluster, keep dissimilar items apart.
- Algorithms employ different definitions of similarity/dissimilarity and objective function for determining a "good" cluster.

Partitional Algorithms



Build a "flat" clustering of the data all at once

Hierarchical Algorithms



Gradually build a nested tree structure of clusters

Hierarchical Clustering

- Instead of generating a flat partition of data, it can be useful to construct a hierarchy of items by producing a set of nested clusters that are arranged to form a tree structure.
- Hierarchical structure allows for multiple levels of granularity…

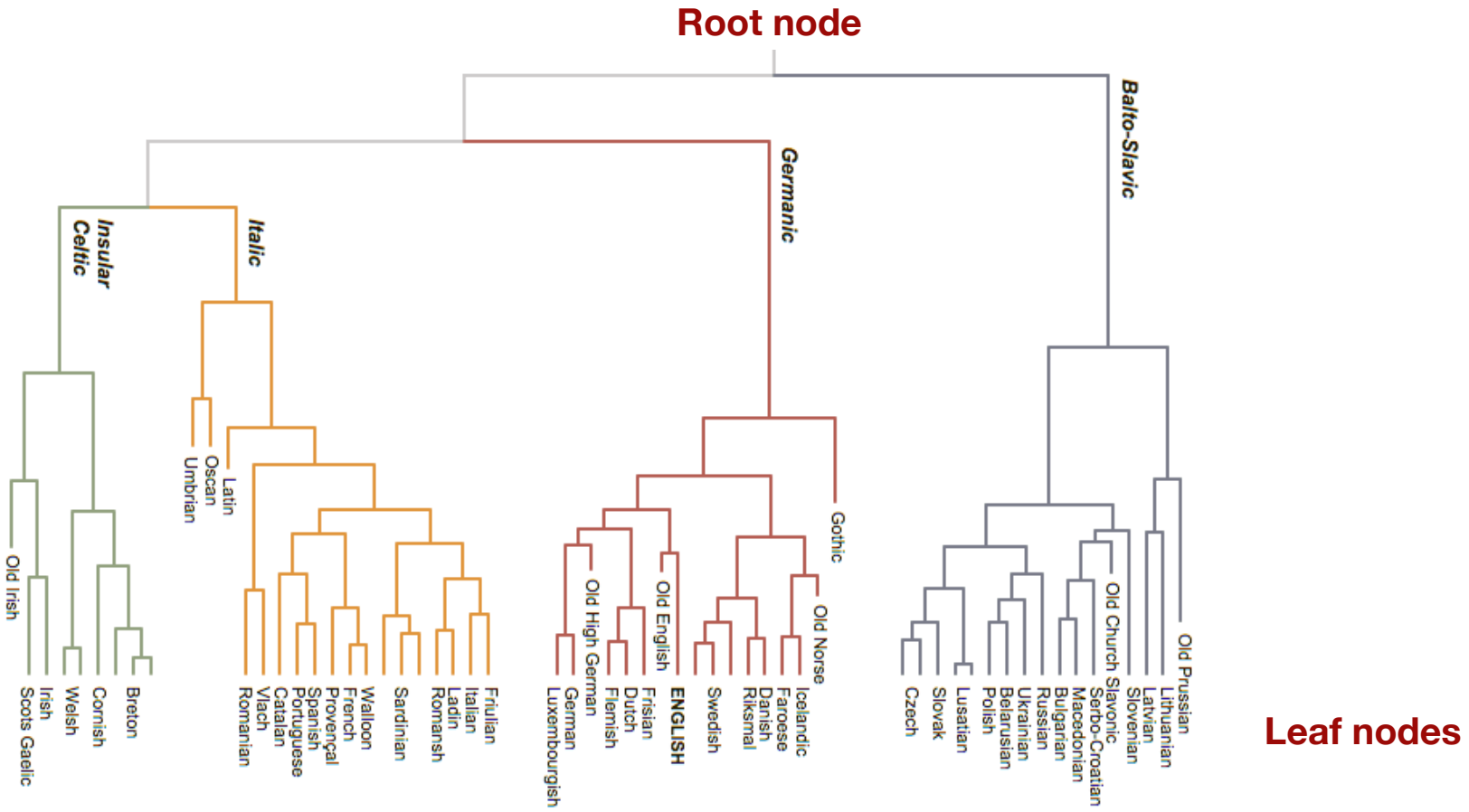
  News → Sport → Olympics → Athletics

  News → Sport → Rugby → World Cup

- Two distinct categories of hierarchical clustering algorithm:
  1. Agglomerative: Begin with each item assigned to its own cluster. Apply a bottom-up strategy where, at each step, the most similar pair of clusters are merged.
  2. Divisive: Begin with a single cluster containing all items. Apply a top-down strategy where, at each step, a chosen cluster is split into two sub-clusters.

Dendrograms

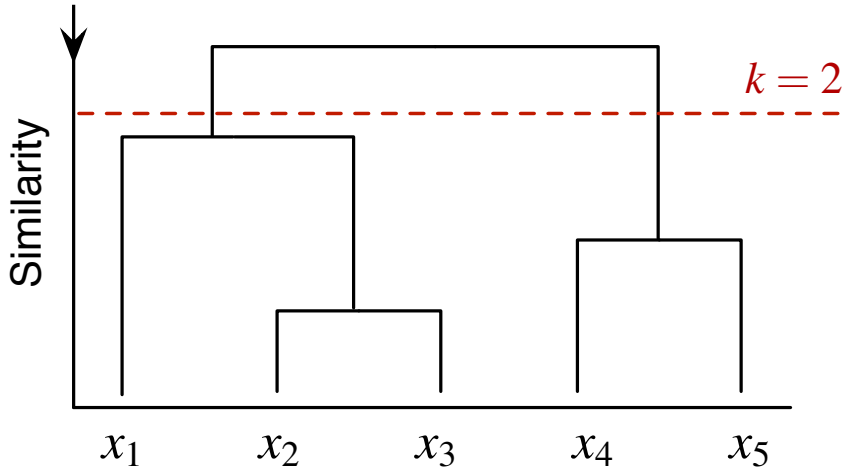
Dendrogram: A tree diagram, frequently used to illustrate arrangement of clusters produced by a hierarchical clustering algorithm. General groups are near the top of the tree, more granular groups at the bottom.



http://www.nytimes.com/interactive/2012/08/24/science/0824-origins.html

Dendrograms

- A dendrogram contains nodes for each cluster, with relations illustrating the merge or split operations that were performed during the clustering process.

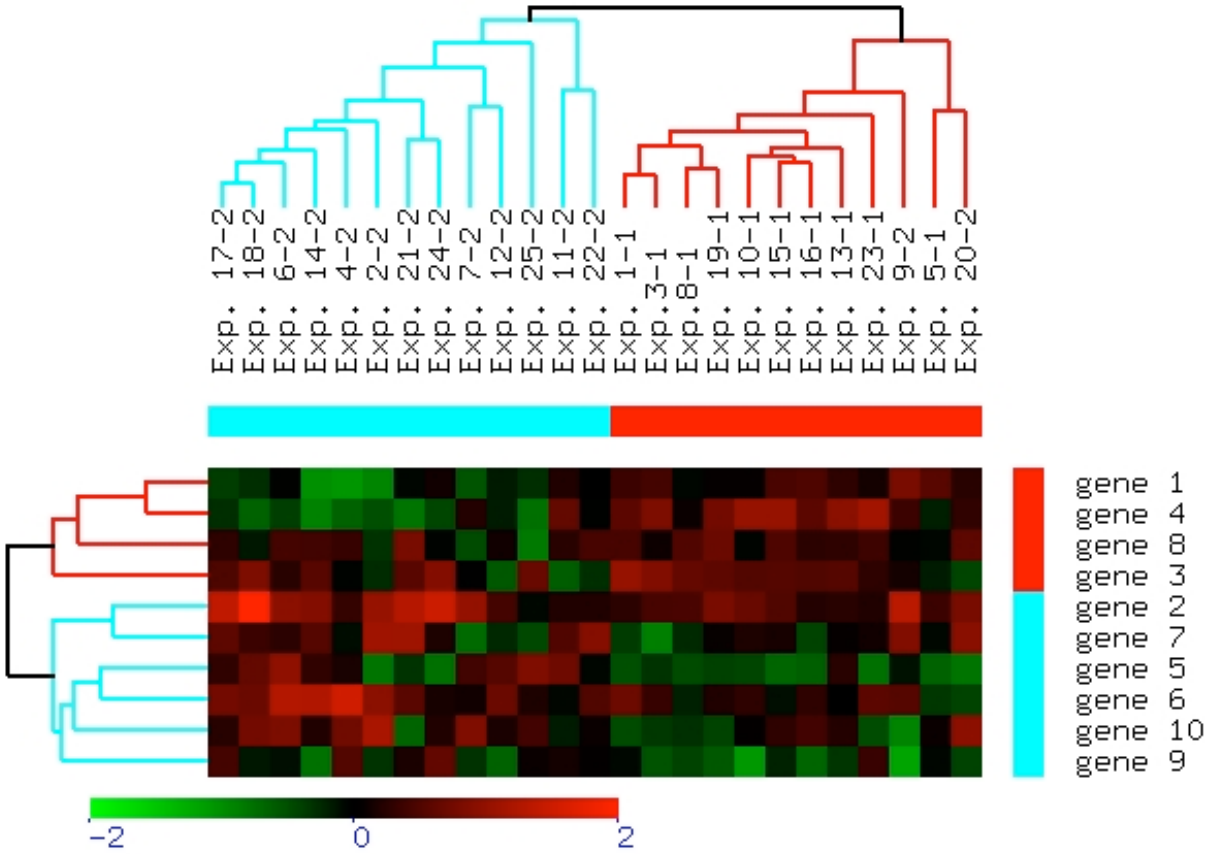


$$\mathcal{C} = \{\{x_1, x_2, x_3, x_4, x_5\}\}$$
$$\mathcal{C} = \{\{x_1, x_2, x_3\}, \{x_4, x_5\}\}$$
$$\mathcal{C} = \{\{x_1\}, \{x_2, x_3\}, \{x_4, x_5\}\}$$
$$\mathcal{C} = \{\{x_1\}, \{x_2, x_3\}, \{x_4\}, \{x_5\}\}$$
$$\mathcal{C} = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}\}$$

- **Advantage:** We generally do not need to specify the number of clusters *k* in advance.
- Construct a tree, then allow the user to manually select *k* by examining the dendrogram to find an appropriate cut-off point.

Hierarchical Clustering: Applications

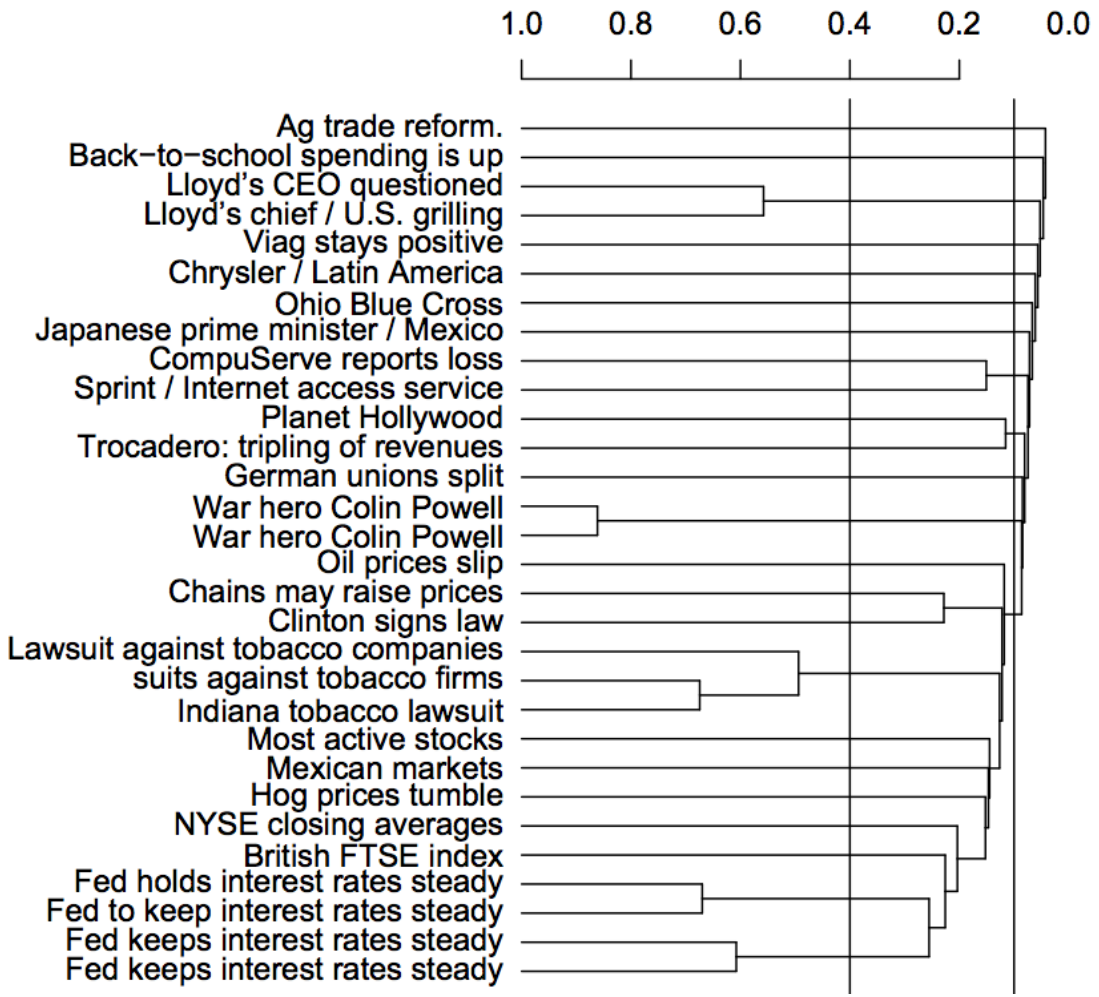
Hierarchical clustering is frequently applied in biology when studying gene expression data to infer biological function of unknown genes. Often want to cluster both genes and experiments (conditions).



Hierarchical Clustering: Applications

Hierarchical clustering also often applied to document collections.

Algorithms can identify both high-level (broad) topics and low-level (more granular) topics.



http://nlp.stanford.edu/IR-book

Agglomerative Clustering

Algorithm Inputs:

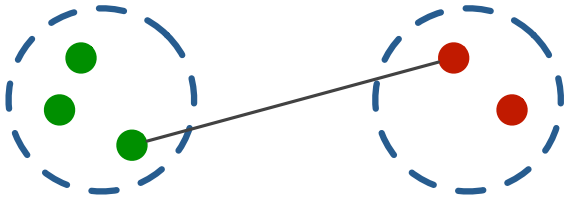
- *Distance matrix* $\mathbf{D}$, specifying the distance between each pair of items in the data, computed using some appropriate measure (e.g. Euclidean).
- *Cluster metric* which helps decide which pair of clusters to merge at each step, using values from $\mathbf{D}$.

Algorithm summary:

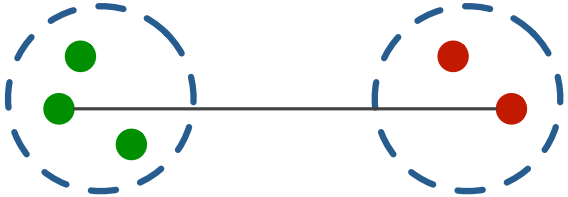
1. Assign every item to its own cluster, each just containing that item. These are the “leaf nodes” of the tree.
2. Find the closest (i.e. most similar) pair of clusters, according to the cluster metric, and merge them into a single cluster, so that now you have one less cluster.
3. Compute distances (similarities) between the new cluster and each of the remaining old clusters.
4. Repeat from Step 2 until all items are clustered into a single cluster. This is the “root node” of the tree.

Cluster Metrics

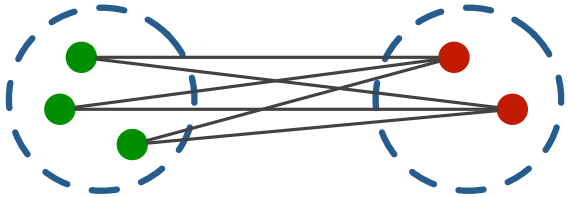
- A variety of metrics exist for determining which pair of clusters should be merged next from among all possible pairs. These specify how we use values from $\mathbf{D}$ to measure the distance between two clusters.
  - Single linkage: Define cluster distance as the smallest pairwise distance between items from each cluster.
  - Complete linkage: Define cluster distance as the largest pairwise distance between items from each cluster.
  - Average linkage: Define cluster distance as the average of all pairwise distances between items from each cluster.



Single Linkage



Complete Linkage



Average Linkage

Cluster Metrics

- Formulae for cluster distance metrics, where D_{ij} is the distance between the *i*-th and *j*-th items in distance matrix $\mathbf{D}$:

Single linkage $$d(C_a, C_b) = \min_{x_i \in C_a, x_j \in C_b} D_{ij}$$

Complete linkage $$d(C_a, C_b) = \max_{x_i \in C_a, x_j \in C_b} D_{ij}$$

Average linkage $$d(C_a, C_b) = \frac{\sum_{x_i \in C_a} \sum_{x_j \in C_b} D_{ij}}{|C_a|\,|C_b|}$$

- The choice of cluster distance metric can substantially affect the resulting clustering.

➡ Complete linkage is sensitive to outliers. Single linkage tends to produce long chains, not cohesive clusters.

Example: Agglomerative Clustering

- Given a data set of four items represented by two features, construct a Euclidean distance matrix **D**.

data matrix

x_1	2	1
x_2	0	0
x_3	1	1
x_4	0	3

$$\mathrm{ED}(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{f \in F} (q_f - p_f)^2}$$

4 x 4 distance matrix

	x_1	x_2	x_3	x_4
x_1	0	2.24	1.00	2.83
x_2	2.24	0	1.41	3.00
x_3	1.00	1.41	0	2.24
x_4	2.83	3.00	2.24	0

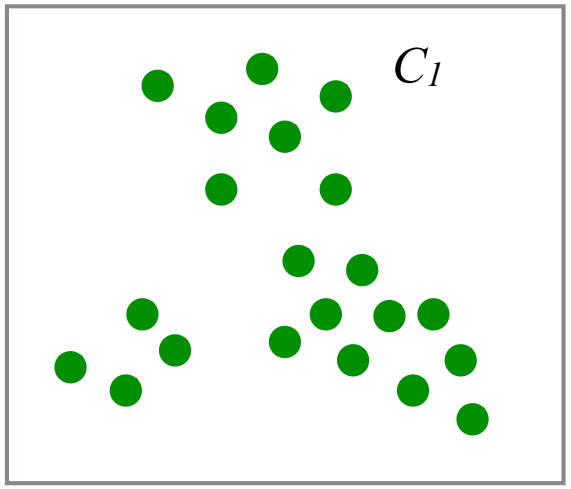
- Two clusters $C_a=\{x_1,x_3\}$ and $C_b=\{x_2,x_4\}$. Then $d(C_a,C_b)$ is…

Single linkage	min(2.24,2.83,1.41,2.24) = 1.41
Complete linkage	max(2.24,2.83,1.41,2.24) = 2.83
Average linkage	(2.24+2.83+1.41+2.24)/4 = 2.18

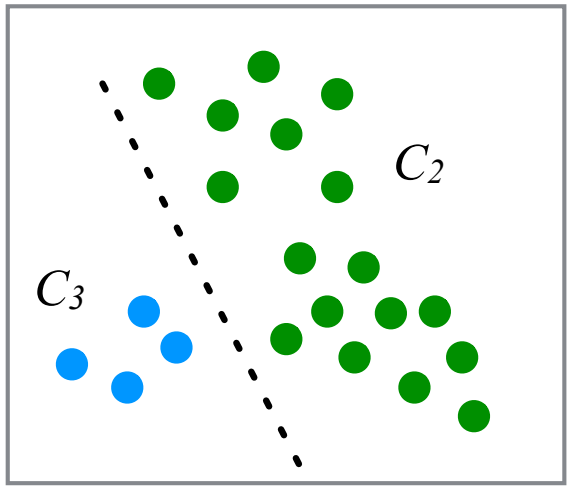
Divisive Algorithms

Divisive Hierarchical Clustering Template:

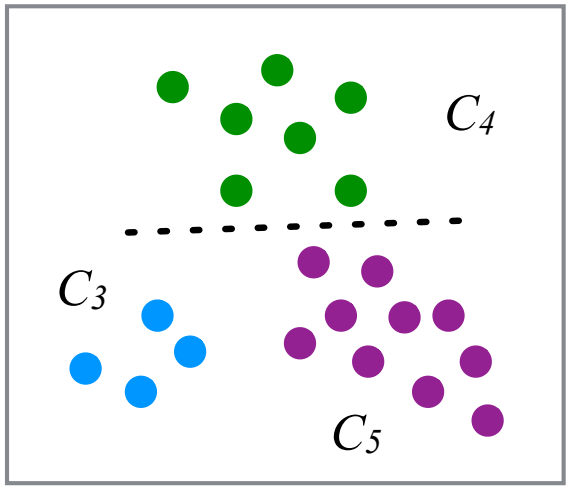
- Start with all items in a single cluster.
- REPEAT until all items are in their own cluster
  - Choose an existing cluster to split using some splitting criterion.
  - Replace the chosen cluster into two sub-clusters.



Assign all items to C_1



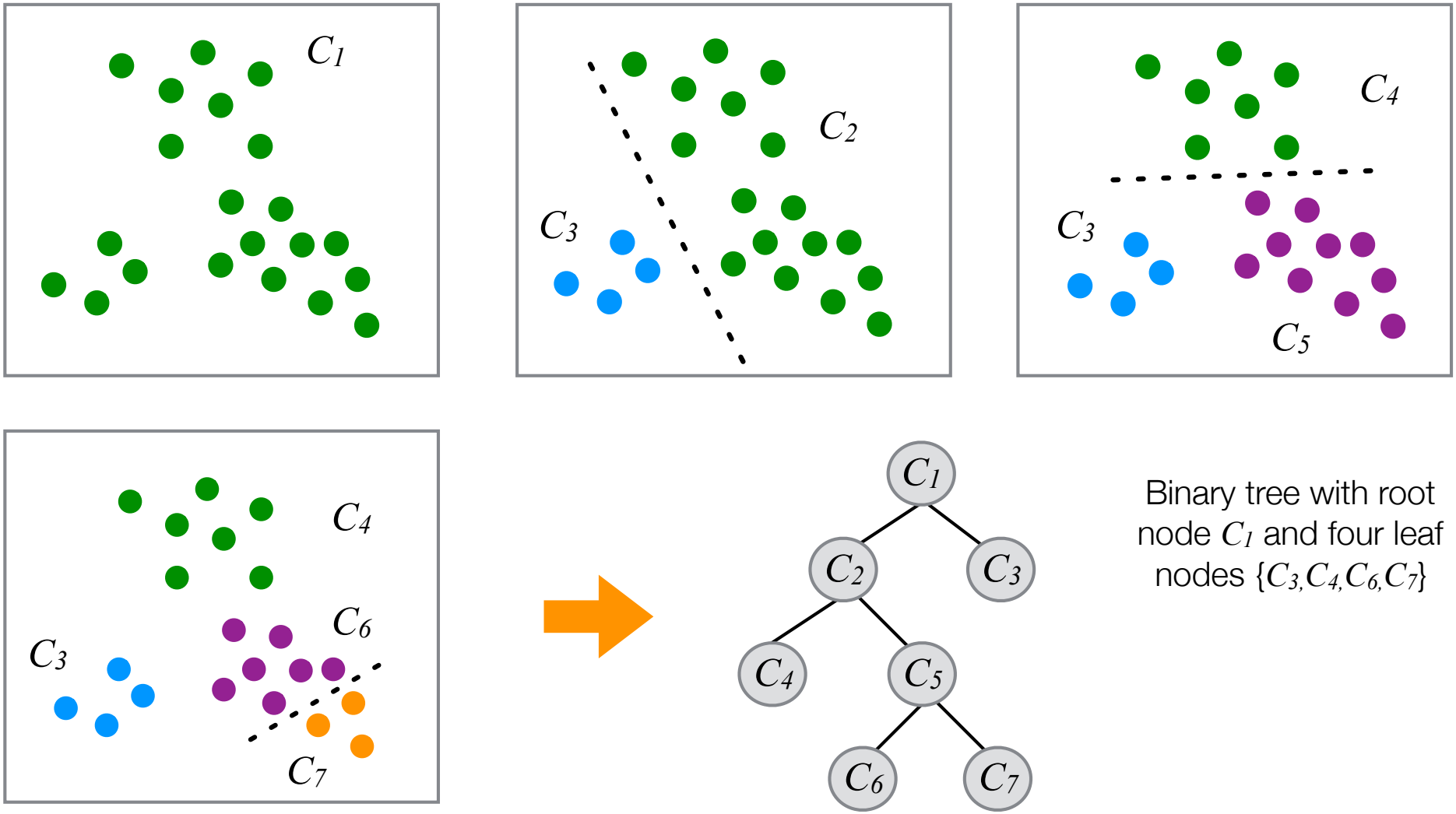
Split C_1 and replace with sub-clusters C_2 and C_3



Split C_2 and replace with sub-clusters C_4 and C_5

Divisive Algorithms

By recursively using a divisive bisecting clustering procedure, the obtained clusters are structured as a hierarchical binary tree.



Binary tree with root node C_1 and four leaf nodes $\{C_3, C_4, C_6, C_7\}$

Bisecting *k*-Means Algorithm

- Key idea: Apply a standard partitional algorithm (i.e. *k*-Means) to split a single cluster into two sub-clusters.

Algorithm Summary:

1. Assign all items to a single cluster.
2. Choose a cluster C_i to split that optimises a given splitting criterion. Common approach: select cluster that has the lowest mean intra-cluster similarity score:

$$MeanIntra(C_c) = \frac{\sum_{x_i, x_j \in C_c} S_{ij}}{|C_c|^2}$$

3. Generate two sub-clusters of C_i by applying *k*-Means with *k=2* to only the items assigned to C_i.
4. Replace the original cluster C_i with the resulting pair of sub-clusters.
5. Repeat from Step 2 until *k* clusters have been generated.

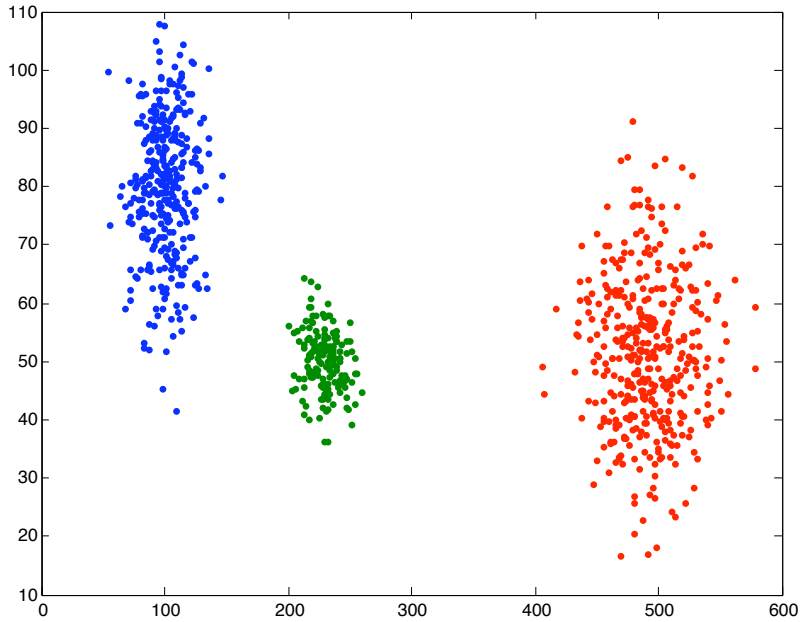
Hierarchical Clustering

- **Advantages:**
  - Allows for multiple levels of granularity, both broad clusters and niche clusters.
  - No requirement to select the “correct” value for number of clusters *k* in advance.
- **Disadvantages:**
  - Poor decisions made early in the clustering process can greatly influence the quality of the final clustering.
  - Once a merging or splitting decision has been made, there exists no facility to rectify a mistake at a later stage.
  - More computationally expensive than partitional methods, particularly for agglomerative clustering.

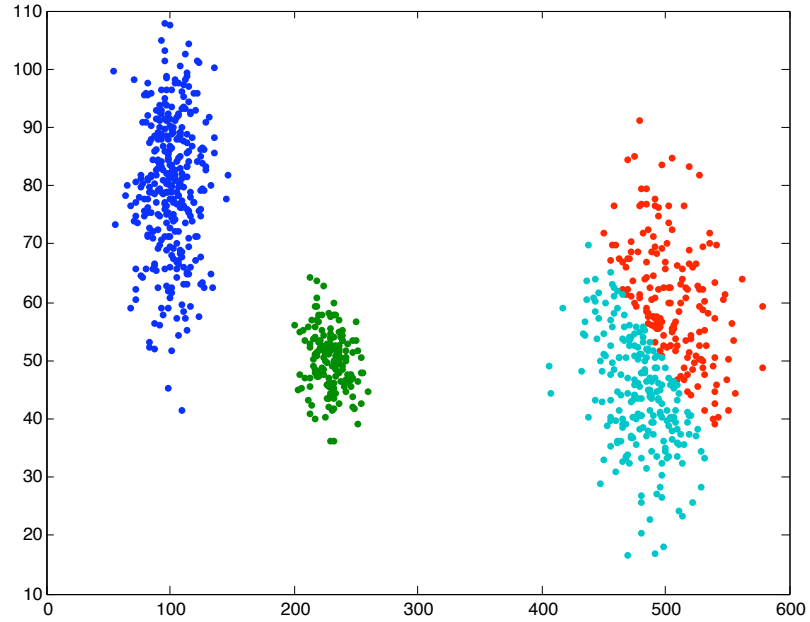
Cluster Validation

Q. How many clusters in a given data set? Usually will not know in advance…

Well-separated clusters *(k=3)*



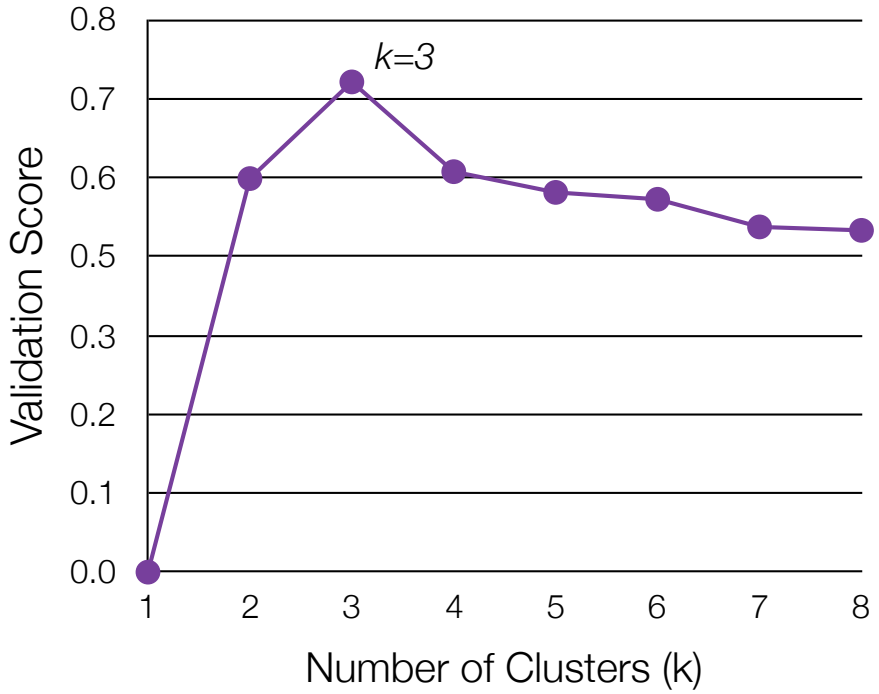
Poorly-separated clusters *(k=4)*



Q. How can we distinguish between a “good” and a “bad” clustering? How can we choose between different clusterings or different clustering algorithms?

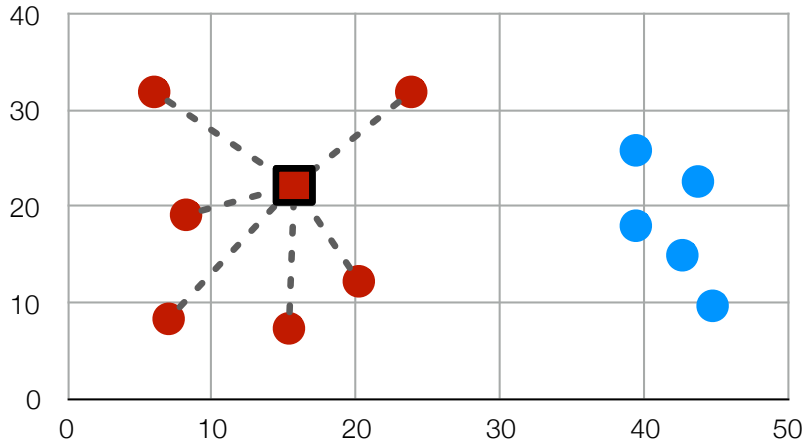
Cluster Validation

- Cluster validation: Measures for automatically producing a quantitative evaluation of the quality of a clustering.
- Common motivation - “good” clusters have the property that cluster members are close to each other and far from members of other clusters.
- Cluster validation is often applied for parameter selection - e.g. select an appropriate value *k* for the *k*-Means algorithm.
- **Typical Strategy:**
  1. Apply *k*-Means for each value from k_{min} to k_{max}.
  2. Calculate score for each clustering using a cluster validation measure.
  3. Examine plot of scores to identify a peak for the best value for *k*.

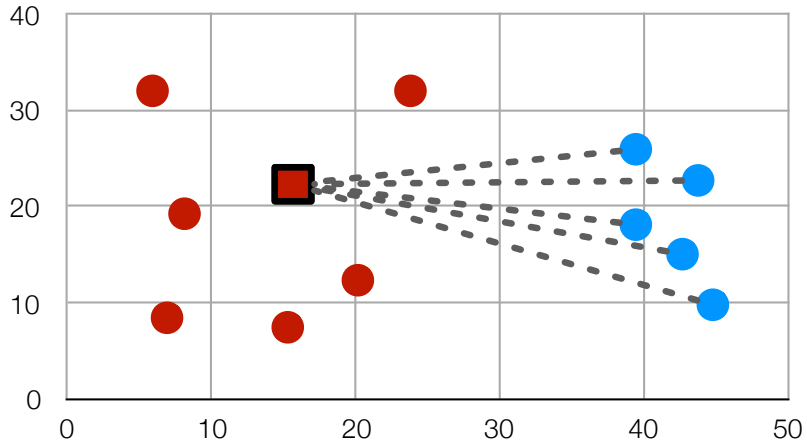


Silhouette Measure

- Validation measure which quantifies degree to which each item belongs in its assigned cluster, relative to the other clusters.



Measure average distance to all other items in same cluster.



Measure average distance to all other items in nearest competing cluster.

$$a_i = \frac{1}{|C_h| - 1} \sum_{j \in C_h, j \neq i} d(i,j)$$

$$b_i = \frac{1}{|C_l|} \sum_{j \in C_l} d(i,j)$$

- Silhouette width for an item x_i is given by s_i. Values are in the range [-1,1], a larger value is better.

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}$$

Silhouette Measure

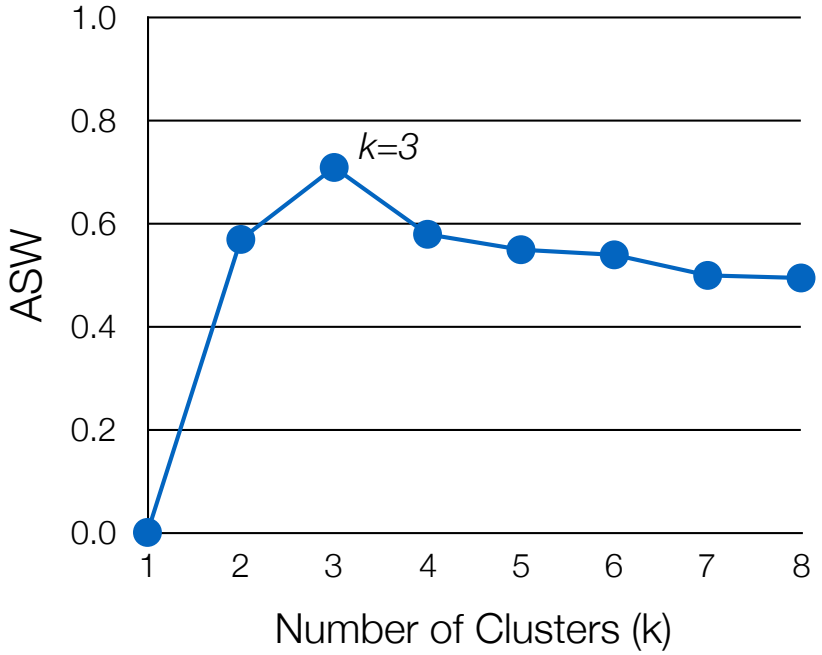
- Silhouette width for a single item x_i is calculated as s_i.
- Average Silhouette Width (ASW): Calculate overall score for a clustering by averaging the silhouette widths for all n items.

$$s_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}$$

$$ASW(\mathcal{C}) = \frac{1}{n}\sum_{i=1}^{n} s_i$$

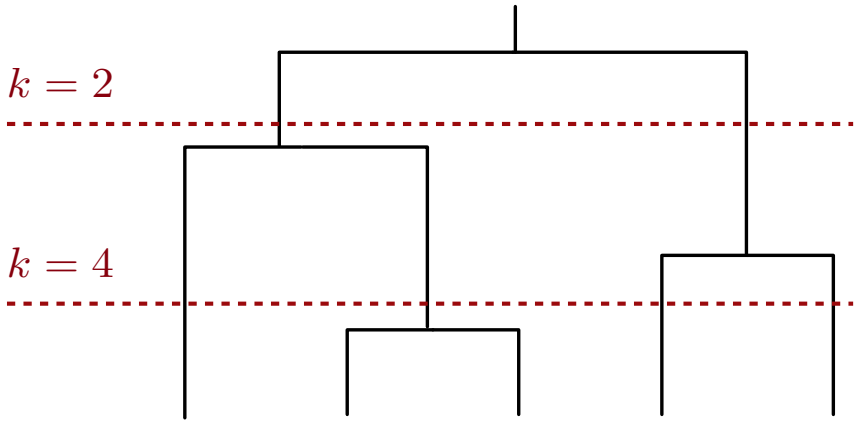
Strategy:

1. Apply *k*-Means for each value from k_{min} to k_{max}.
2. Calculate ASW for each clustering.
3. Examine plot of scores to identify a peak for the best value for *k*.



Hierarchical Cluster Validation

- Hierarchical clusterings are represented as a dendrogram.
- Cutting a dendrogram at a certain level gives a set of flat clusters. Cutting at another level gives another set of flat clusters.
- How can we select where to cut the dendrogram?



Example cuts of a hierarchy for 2 clusters and 4 clusters

➡ Generate hierarchical clustering, then apply cluster validation measures at all possible cut-off levels to identify best level.

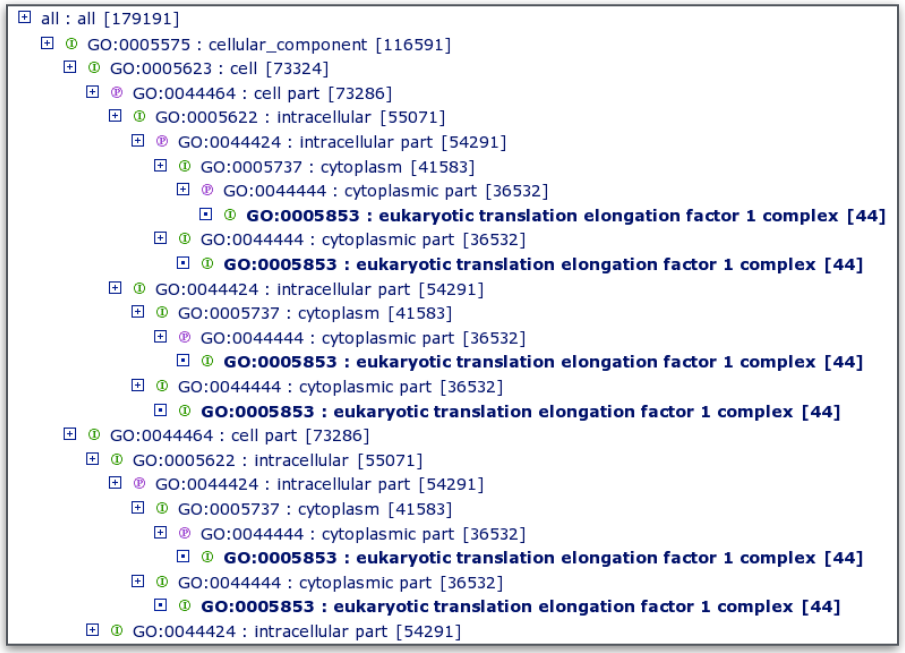
- Many validation measures that can be applied, typically rewarding "compact and well separated clusters" - e.g. Dunn index, DB index

On Clustering Validation Techniques - Halkidi et al (2001)

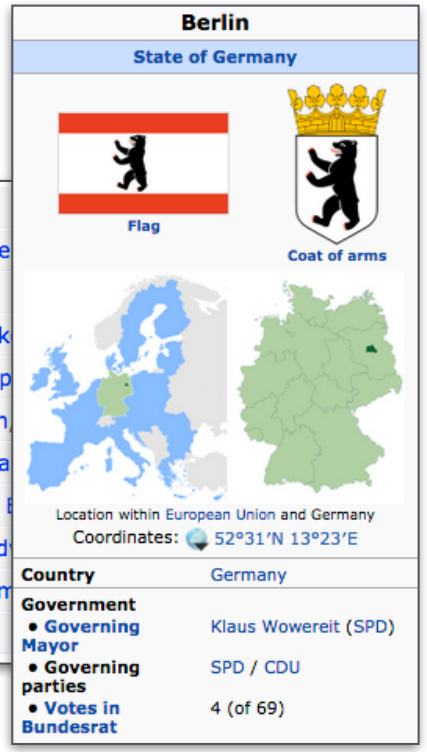
http://dl.acm.org/citation.cfm?id=607609

External Cluster Validation

In practice, we may have some type of external information that can be used to evaluate a clustering, and compare different clusterings.



Class	Examples
City	Cambridge, Berlin, Mancheste
Country	Spain, Iceland, South Korea
Politician	George W. Bush, Nicolas Sark
Musician	AC/DC, Diana Ross, Röyksopp
Music album	Led Zeppelin III, Like a Virgin,
Director	Woody Allen, Oliver Stone, Ta
Film	The Great Beauty, Hysterical E
Book	The Lord of the Rings, The Ad
Computer Game	Tetris, World of Warcraft, Sam
Technical Standard	HTML, RDF, URI



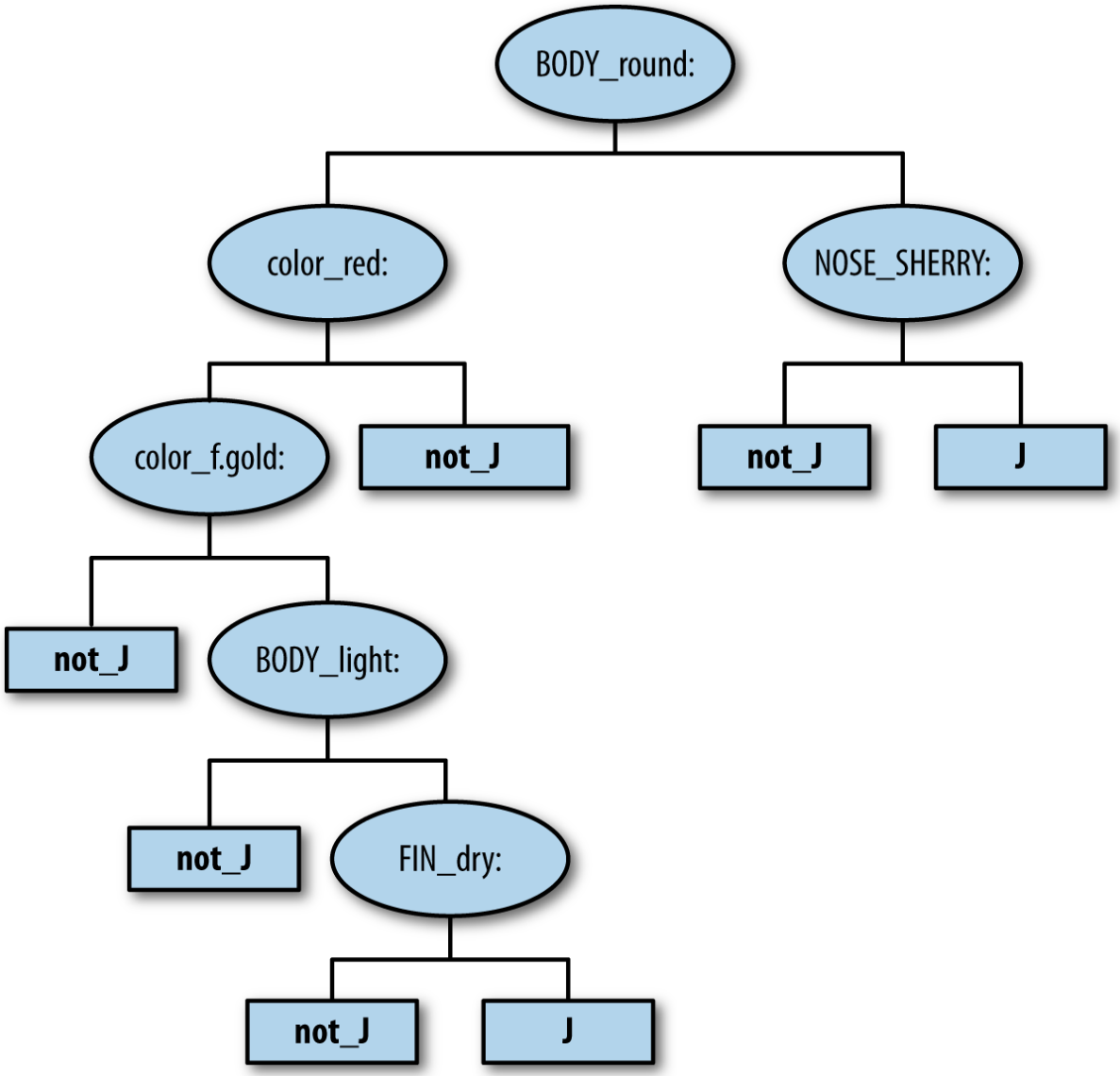
Gene Ontology Database

http://geneontology.org

DBpedia Database

http://dbpedia.org

Generating Cluster Descriptions



Case Study: Credit Line Optimization

- Credit granting businesses face a challenging environment due to the *wide variety* of customer behaviors

Case Study: Credit Line Optimization

Framework for credit customer optimization based on clustering and predictions:

- Customer clusters are formed by using clustering from past credit performance data
- *Within each of the k clusters*, the *expected net present value* to the credit company is estimated
- External data is used to predict for *new accounts* the probabilities of membership for each performance cluster
- The prediction is done using classification and regression trees

Clustering + Expected Value Framework

Summary

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