COMP47750/COMP47990

Clustering - Overview

Part I
Overview
k-Means

Pádraig Cunningham

School of Computer Science

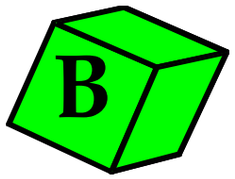
Overview

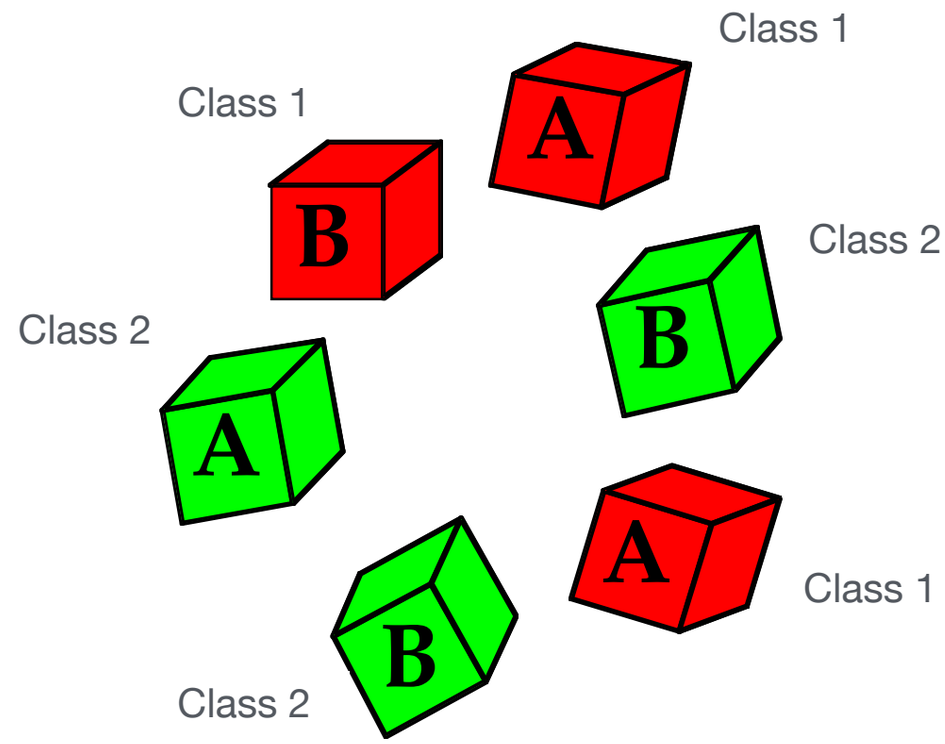
- Supervised v Unsupervised Learning
- Partitional Clustering
  - *k*-Means clustering
  - Cluster initialisation
- Hierarchical Clustering
- Cluster Validation

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.

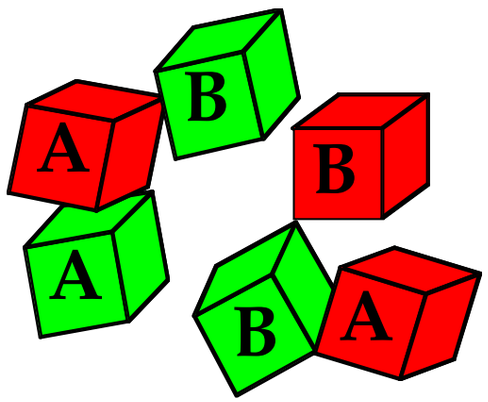
For many tasks, annotated class labels for data are not available - either unknown or too expensive to obtain.

Unsupervised Learning

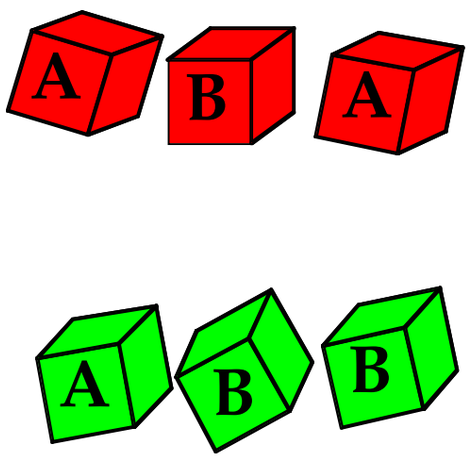
Unsupervised learning algorithms attempt to identify patterns by relying solely on the intrinsic characteristics of the data, without referring to any class information.

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

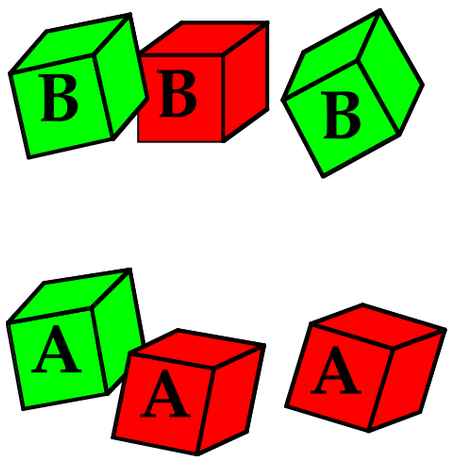
Organise these blocks into groups



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



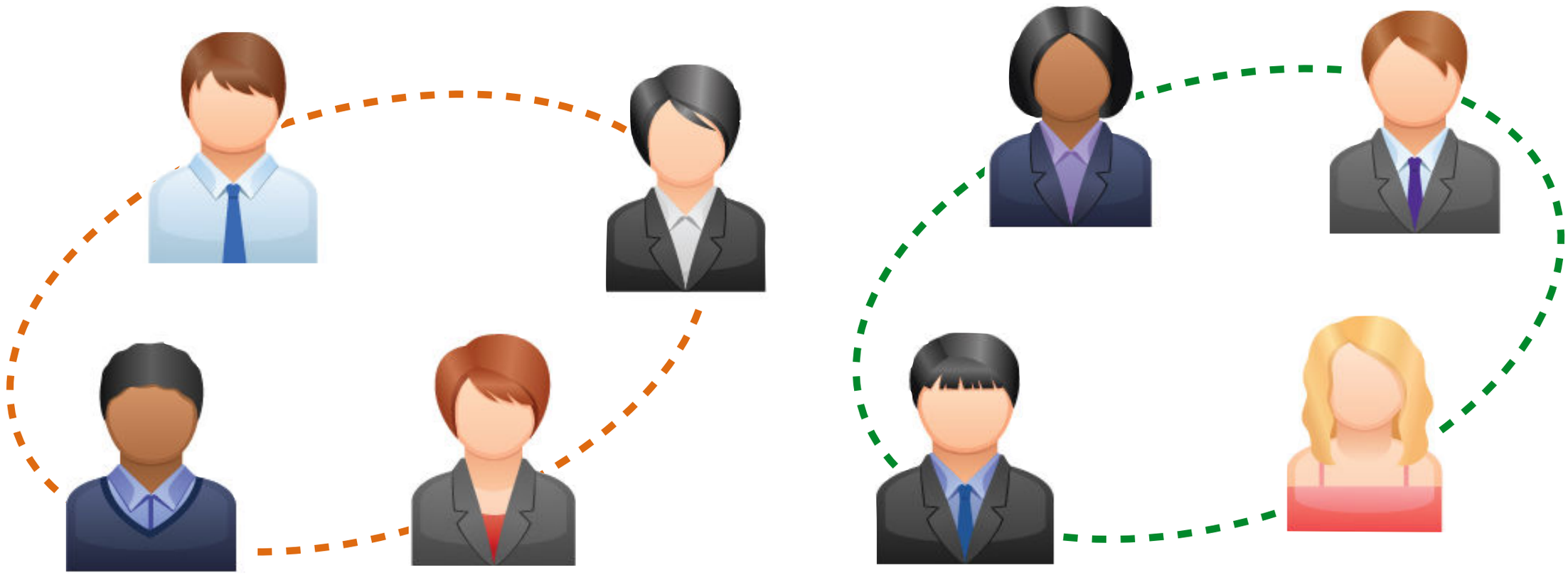
Grouping 1



Grouping 2

Unsupervised Learning: Applications

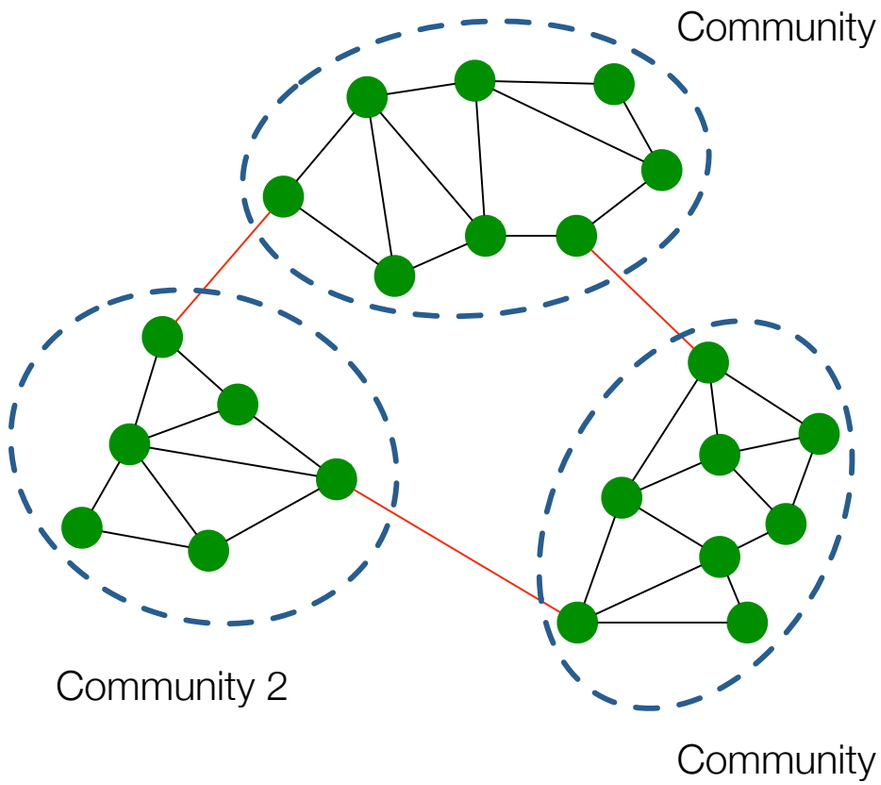
Perform market segmentation by grouping customers into separate clusters, so that customers in the same cluster have similar characteristics.

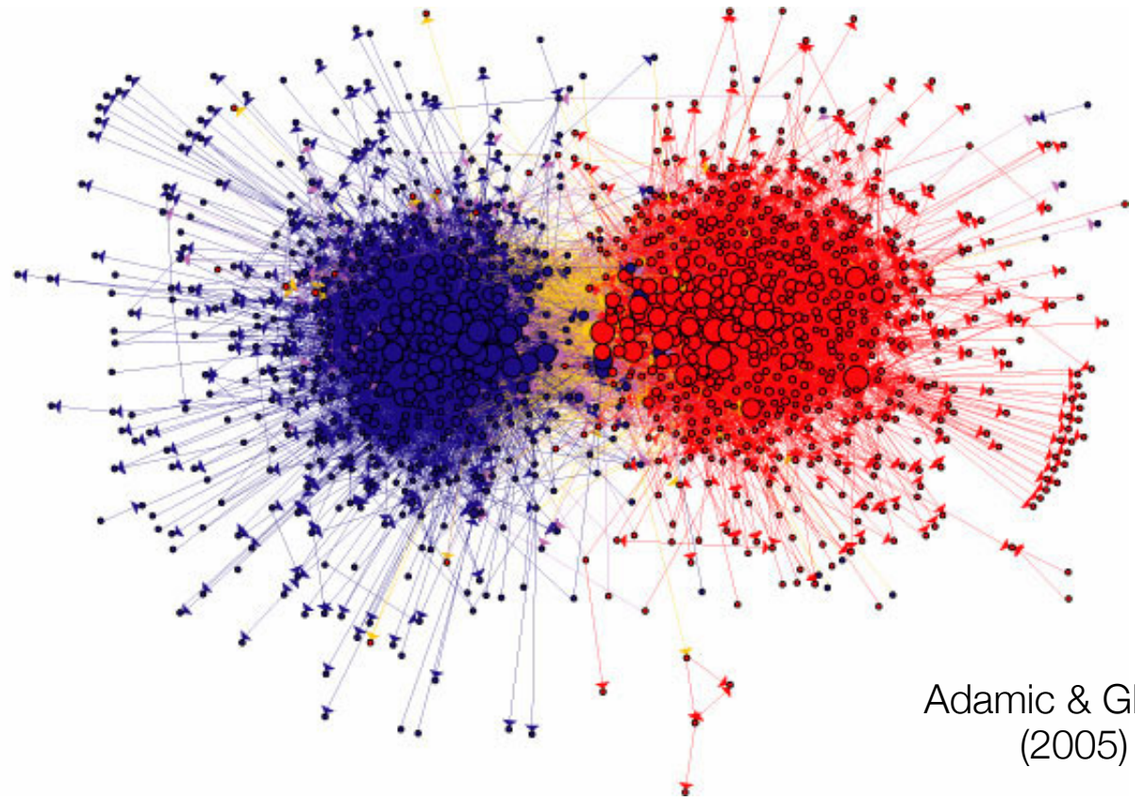


Clustering algorithm output can support targeted marketing and operations for specific sub-groups of customers.

Unsupervised Learning: Applications

Given a social network, apply unsupervised algorithms to identify communities of users who are well-connected to one another, and who are separated from other communities.



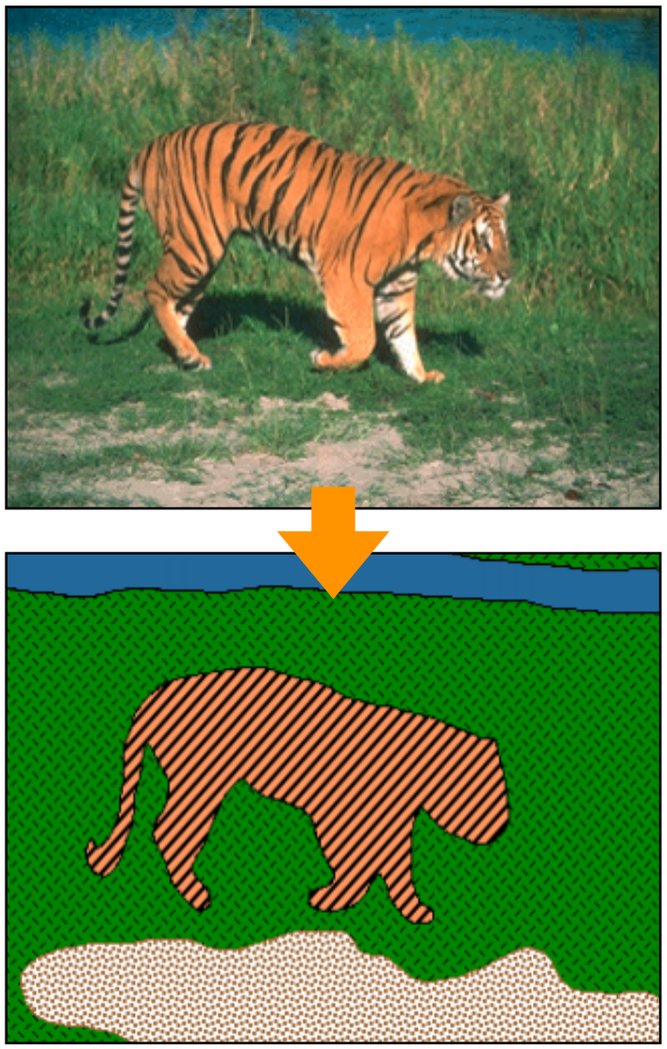


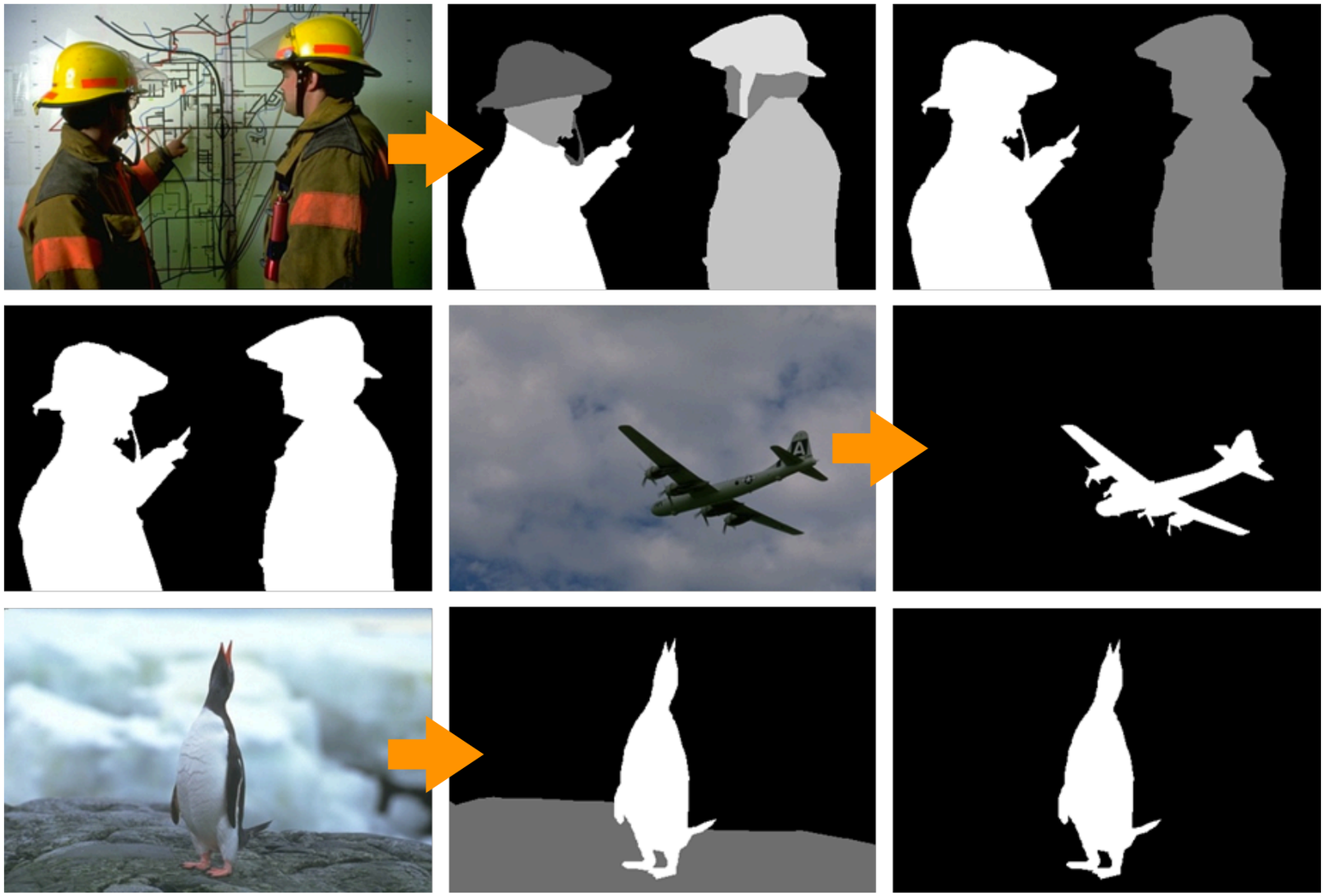
Adamic & Glance (2005)

Users in the network do not need to be manually labeled or annotated in order to apply the community finding algorithm.

Unsupervised Learning: Applications

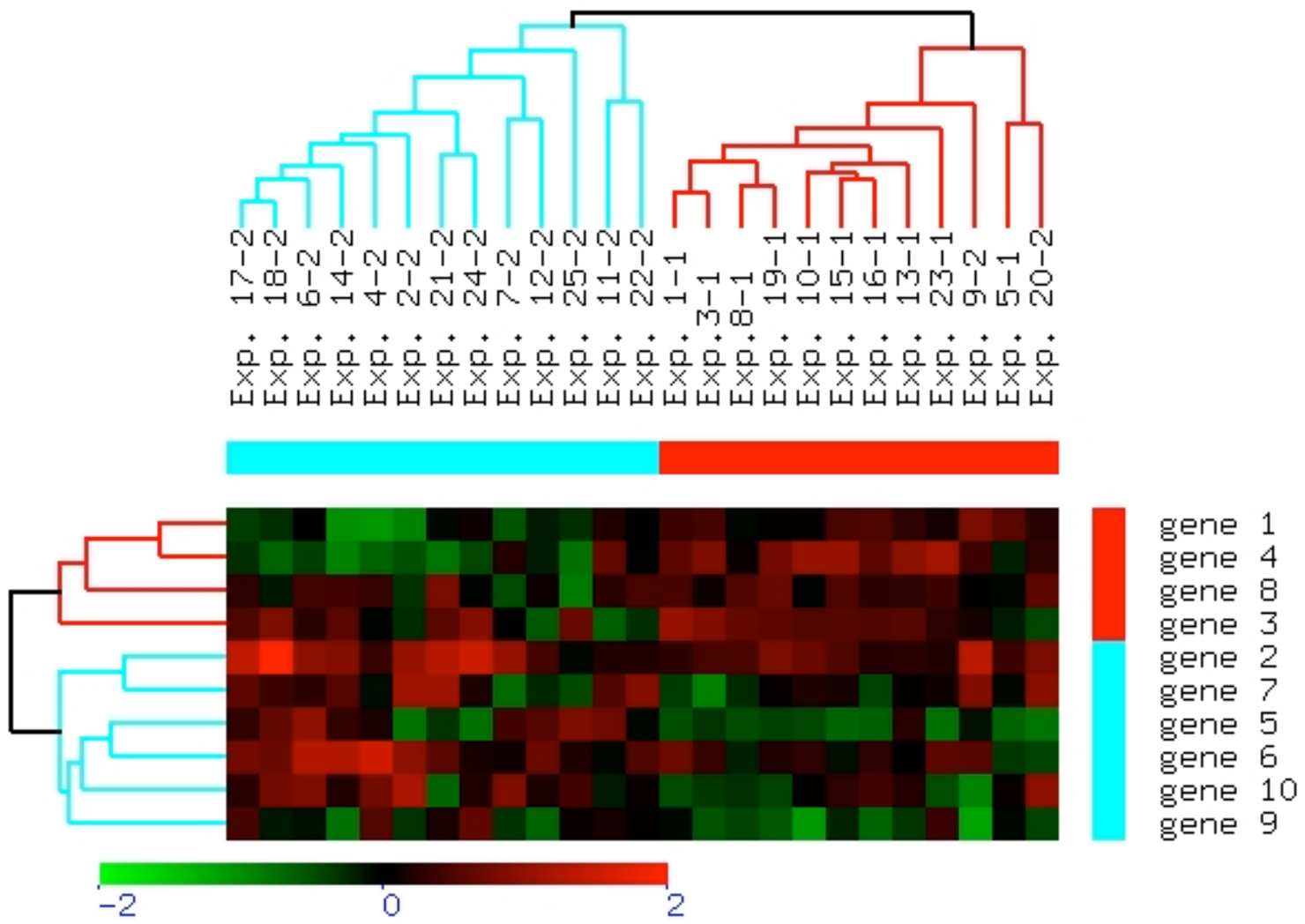
Image segmentation: Unsupervised task in computer vision that attempts to automatically split an image into regions with similar colour or texture, or both. Aim is to partition the image into its constituent “objects”.





Unsupervised Learning: Applications

Hierarchical clustering is frequently applied in biology when studying genetic data to infer biological function of unknown genes. Often we want to cluster both genes and experiments simultaneously.

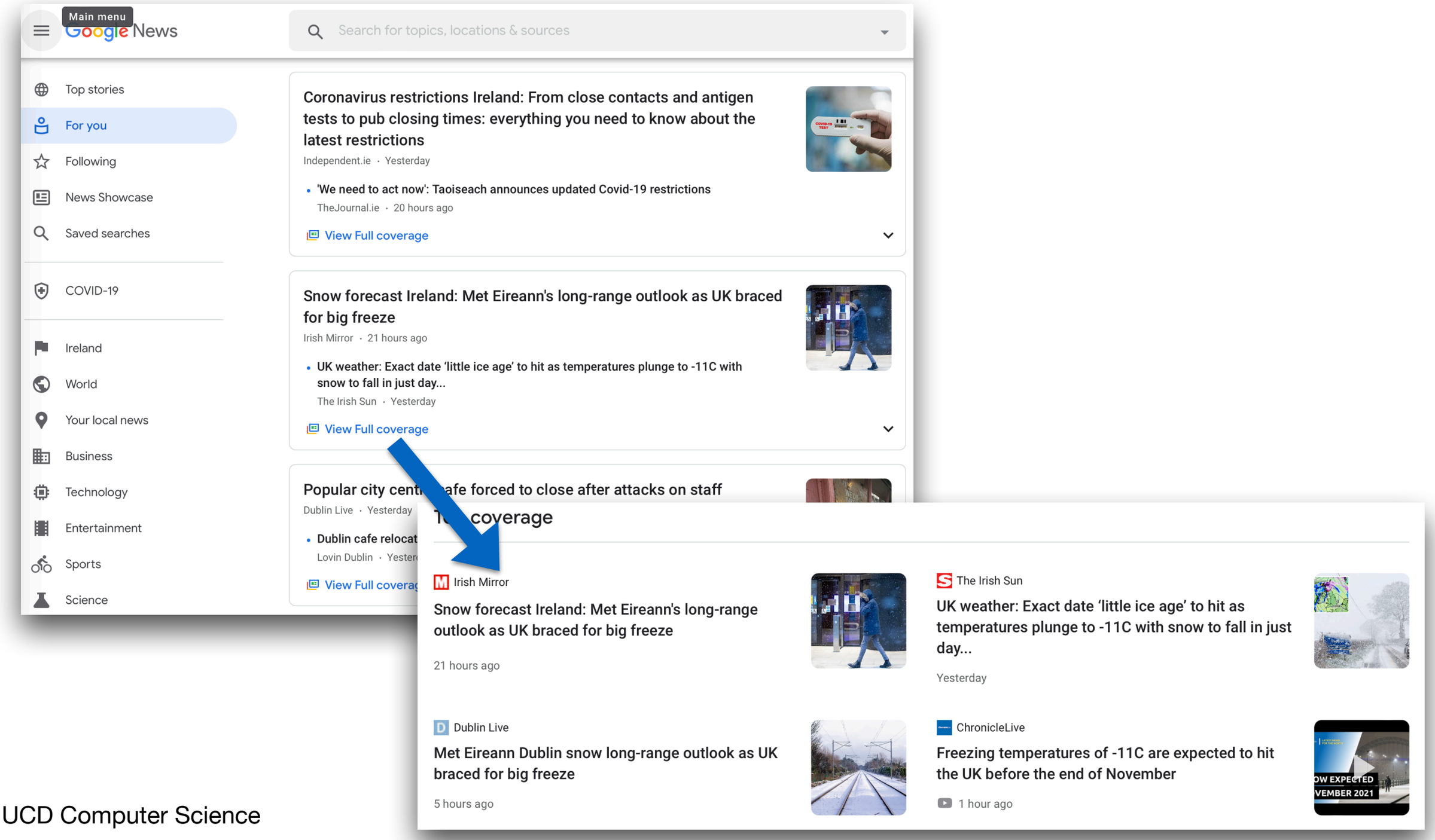


Unsupervised Learning: Applications

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

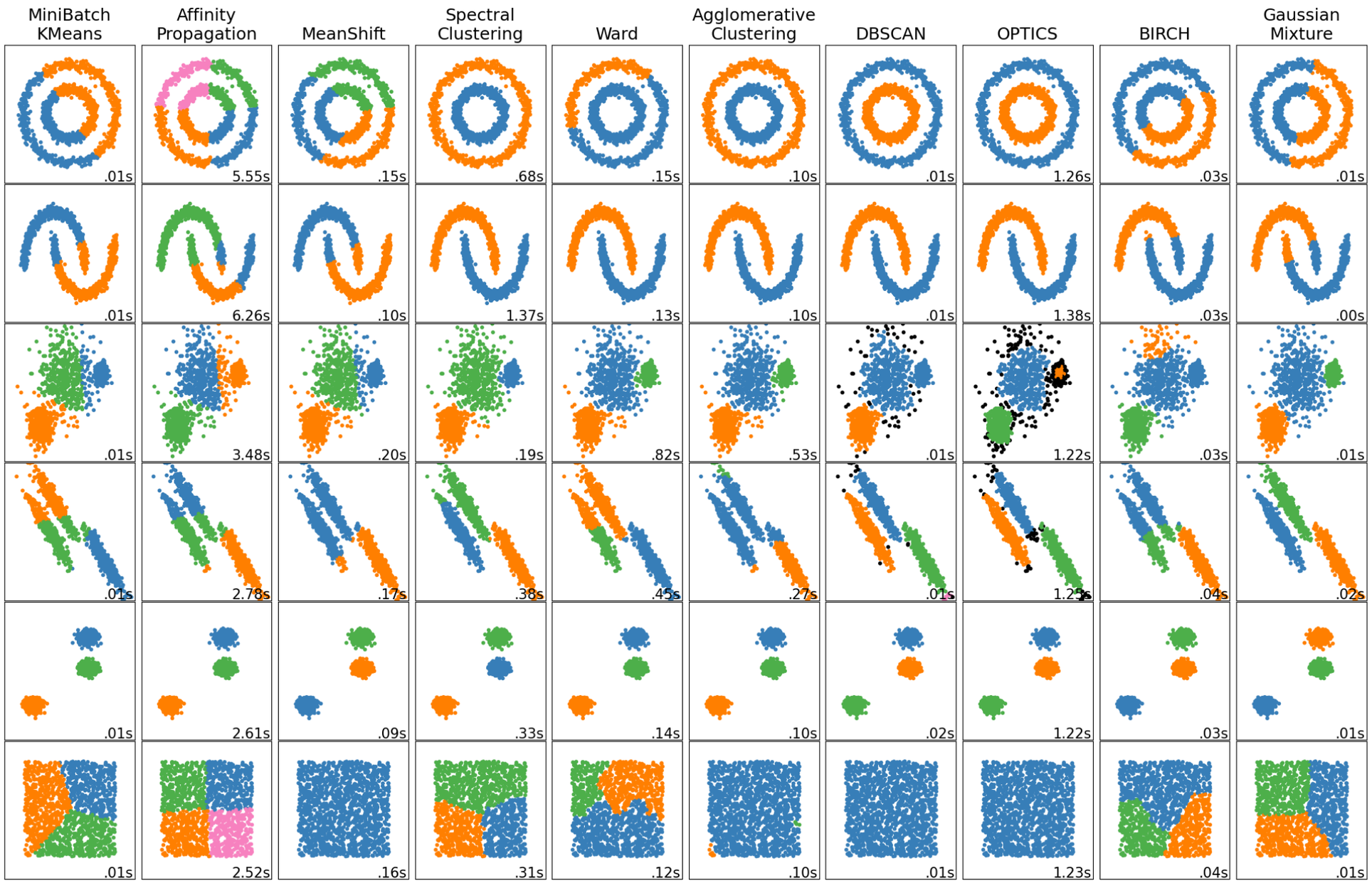
Unsupervised Learning: Applications

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



Clustering

There are usually many different ways to cluster the same data. Clustering algorithms differ significantly in their definition of what constitutes a cluster and the approach used to find them.

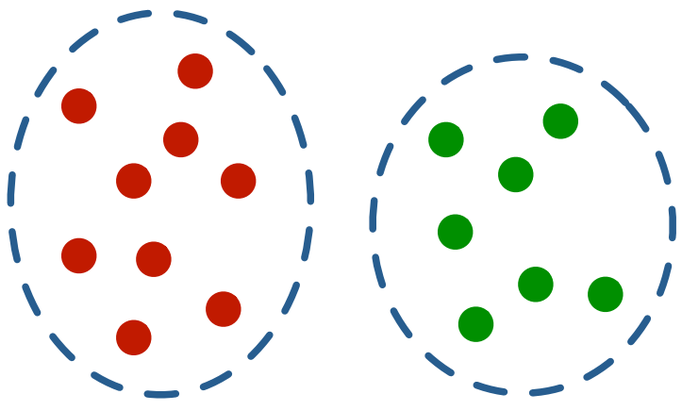


http://scikit-learn.org/stable/modules/clustering.html

Clustering

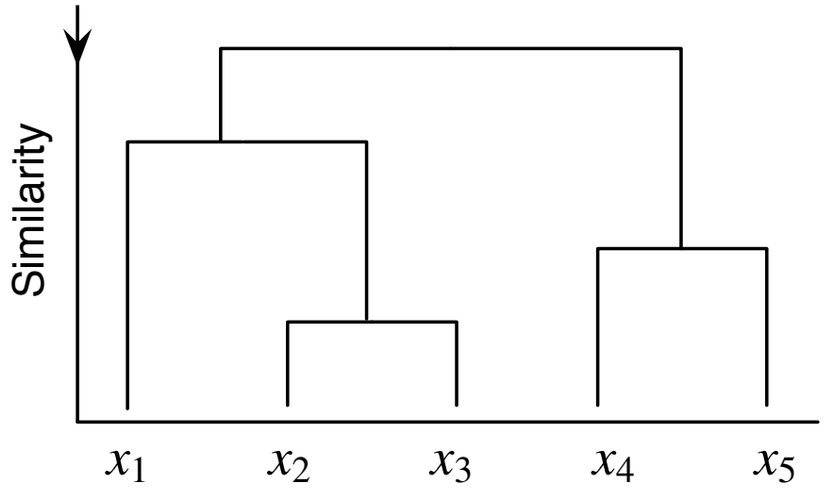
- **General goal**: Assign similar items to the same cluster, keep dissimilar items apart.
- Algorithms employ different definitions of similarity/dissimilarity and objective function for determining a “good” cluster.

Partitional Algorithms



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

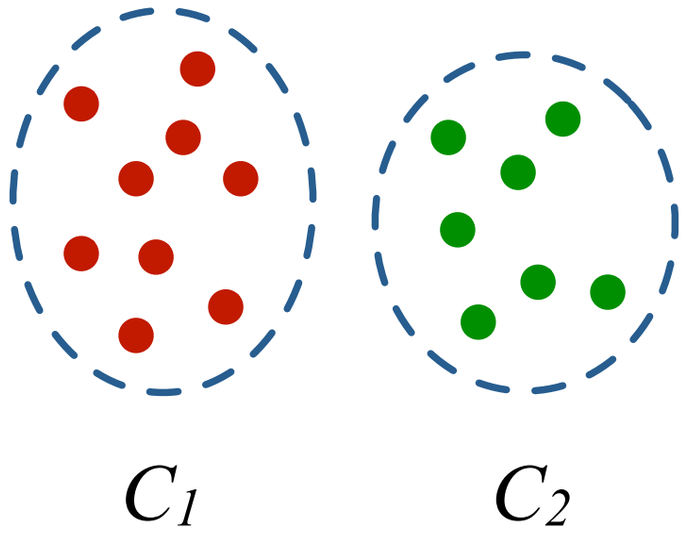
Hierarchical Algorithms



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)



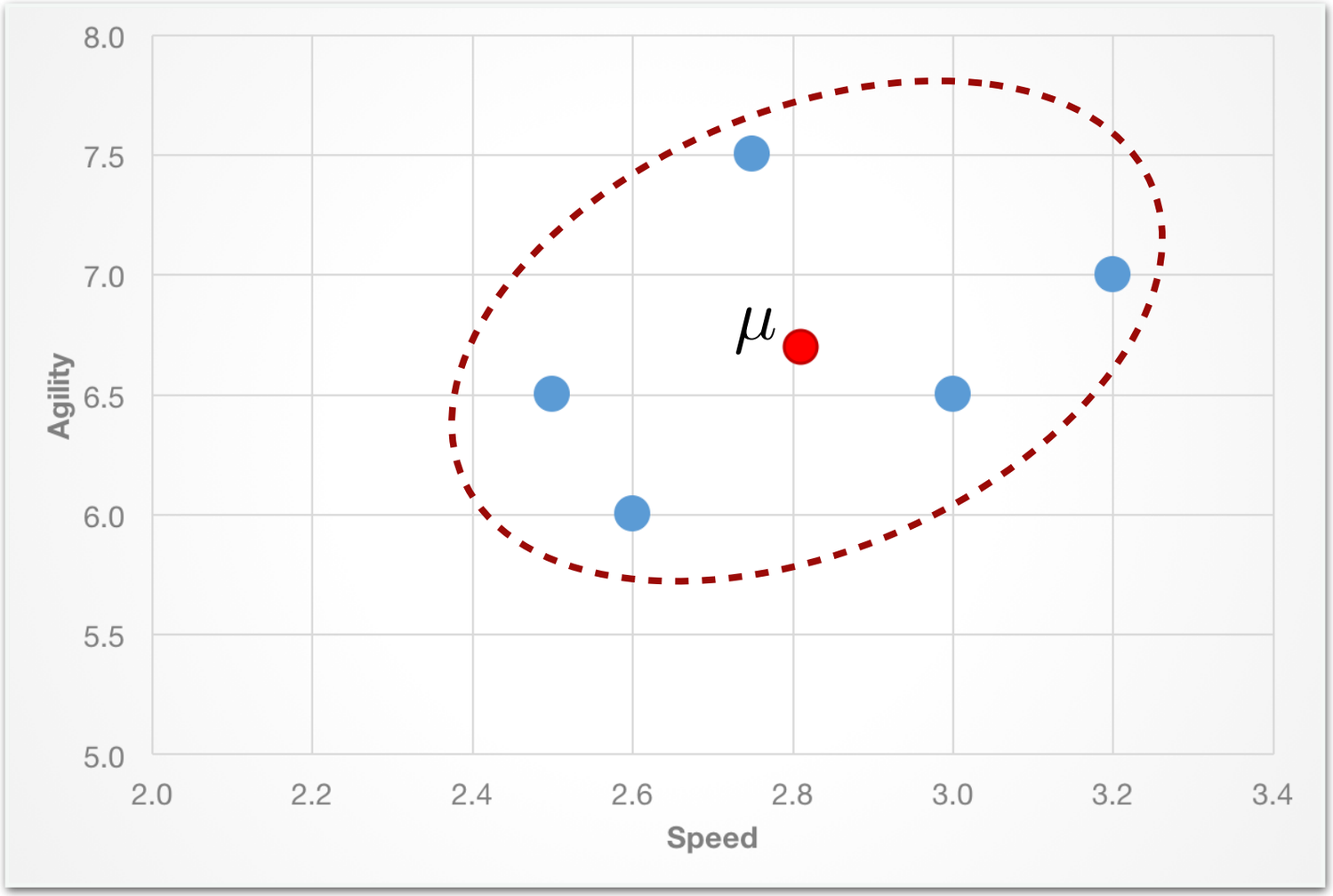
k-Means Clustering

- Centroid: The mean vector 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

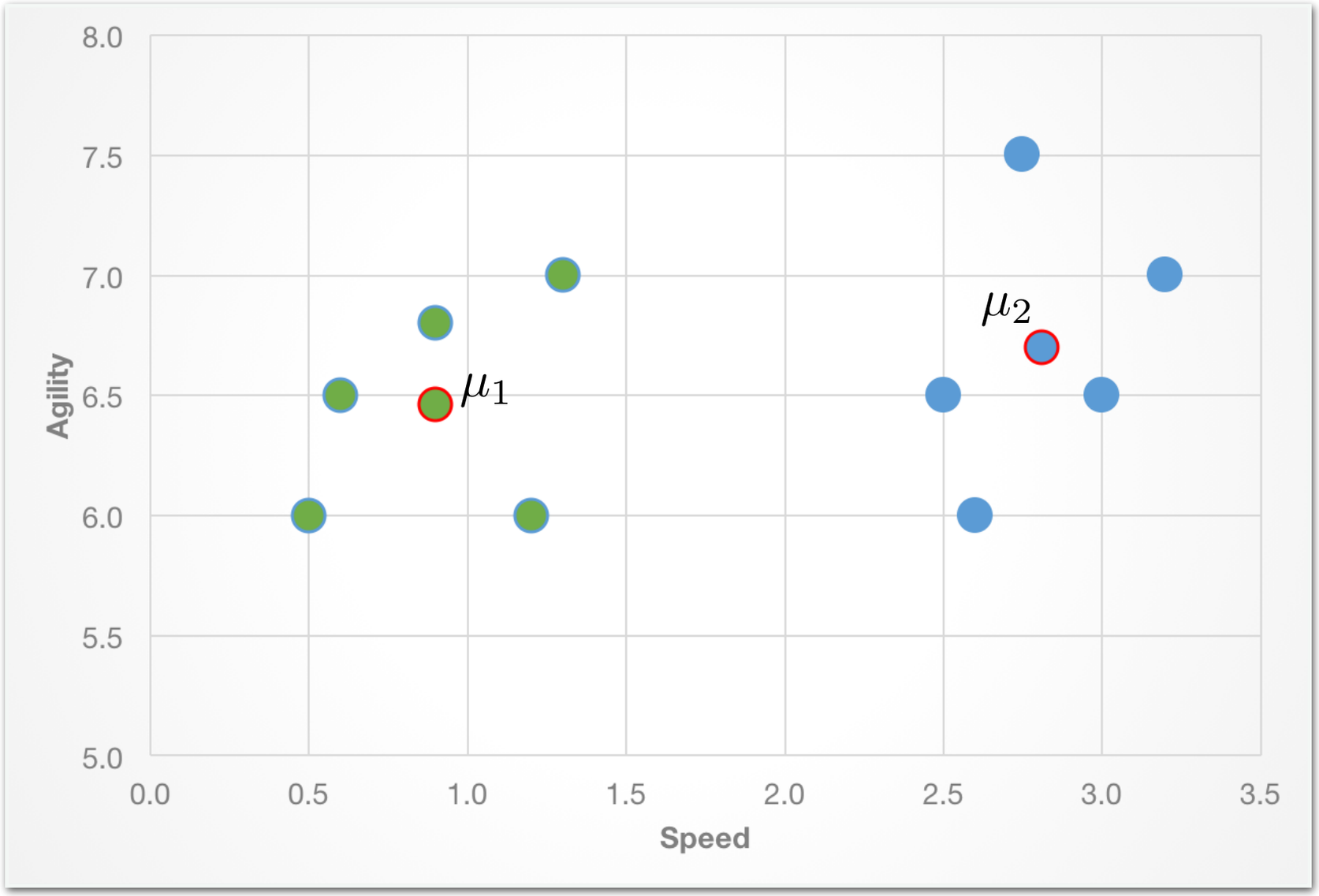
$(2.6 + 3.0 + 2.5 + 3.2 + 2.8)/5 = 2.82$

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



k-Means Clustering

- Each of the *k* clusters in a clustering can be represented by its own centroid μ_i. Example of two clusters, with centroids shown:



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:

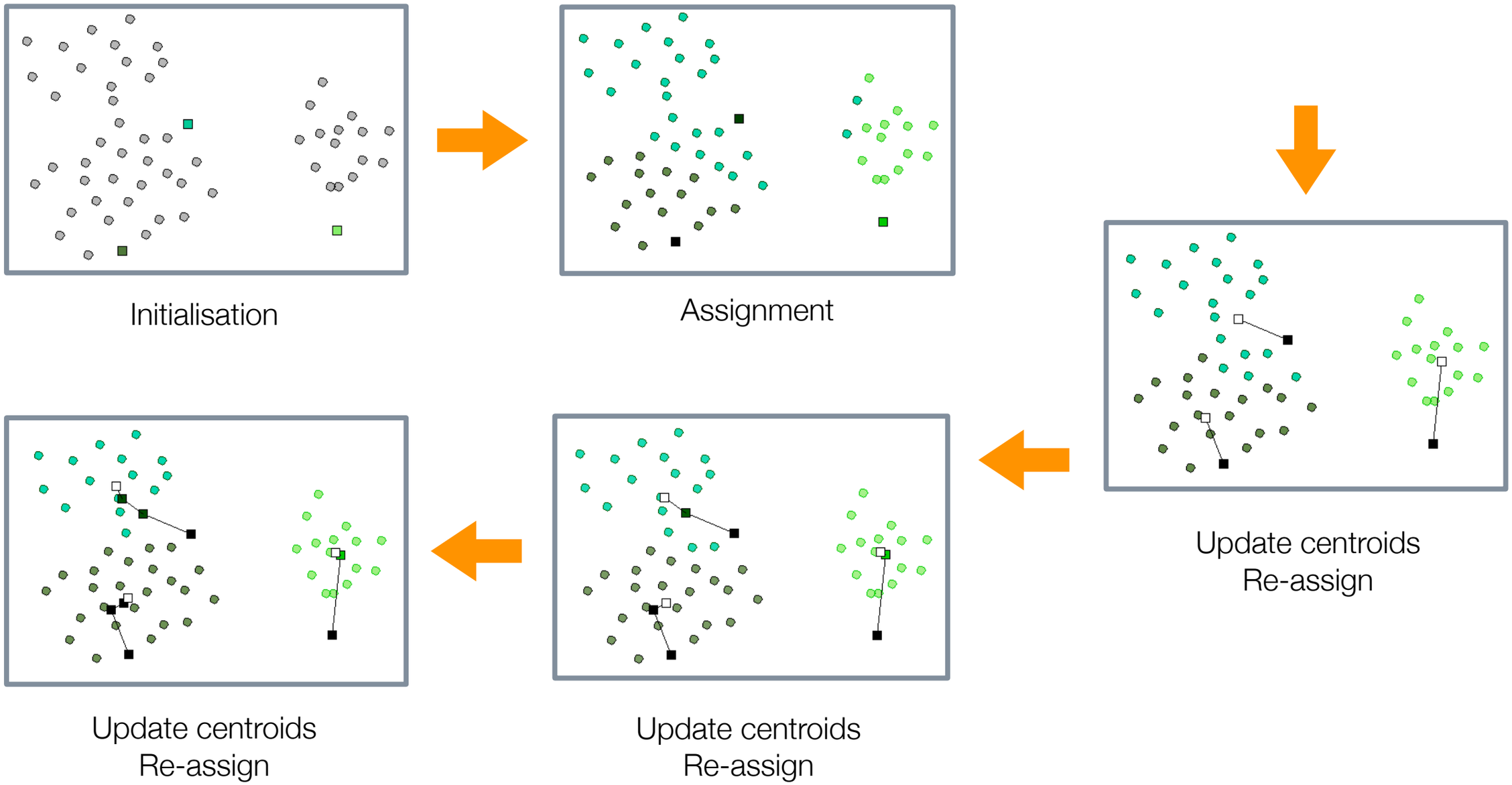
  1) Reassign items to their nearest cluster centroid

  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

Inputs:

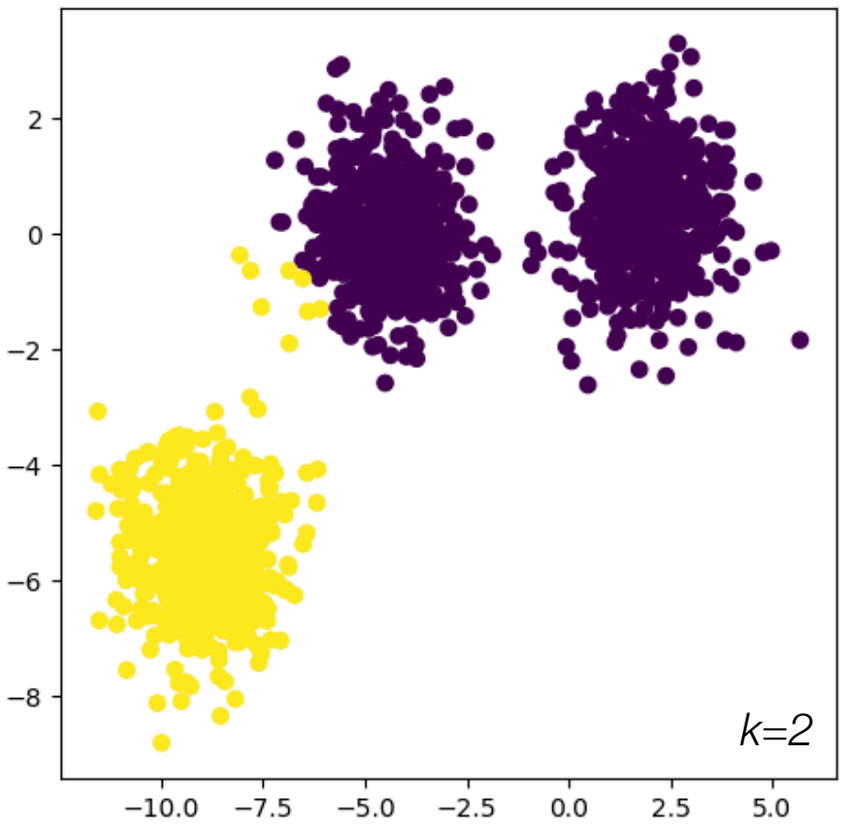
- Data: Set of unlabelled items
- *k*: User-specified target number of clusters
- Maximum number of iterations to run

Algorithm Steps:

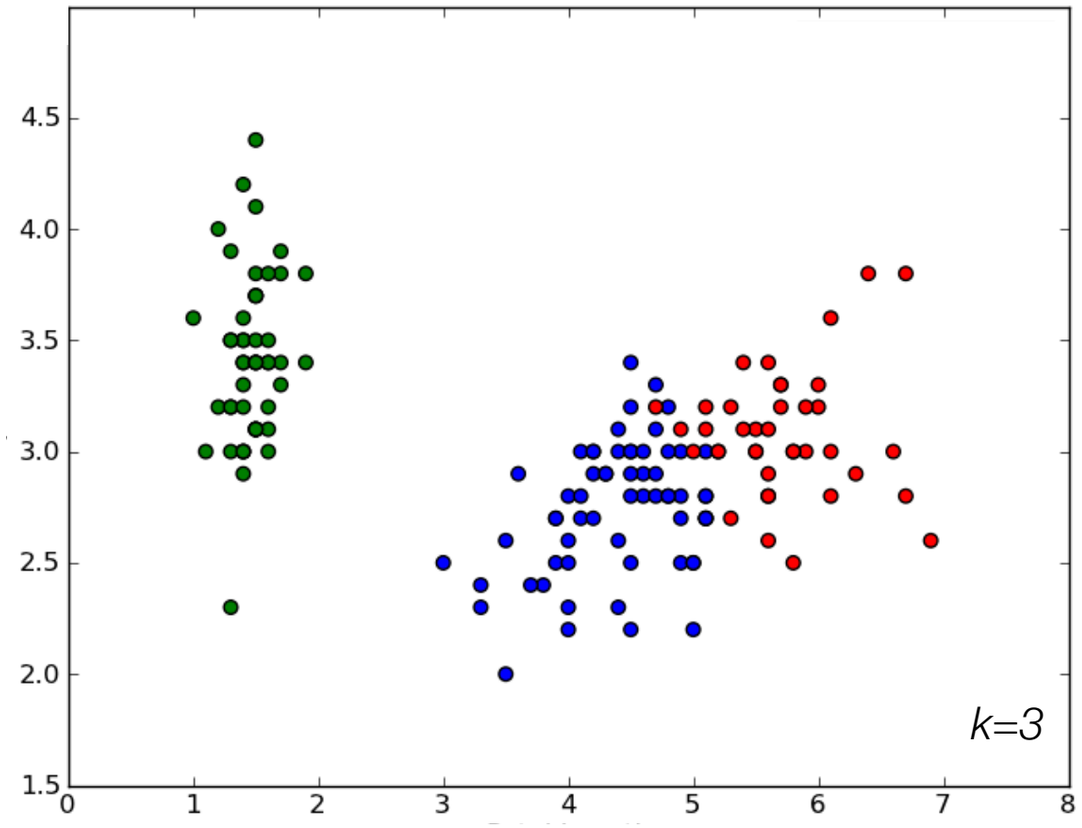
1. *Initialisation*: Select *k* initial cluster centroids (e.g. at random)
2. *Assignment step*: Assign every item to its nearest cluster centroid (e.g. using Euclidean distance).
3. *Update step*: Recompute the centroids of the clusters based on the new cluster assignments, where a centroid is the mean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

k-Means: How Many Clusters?

- 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.



Too few clusters?

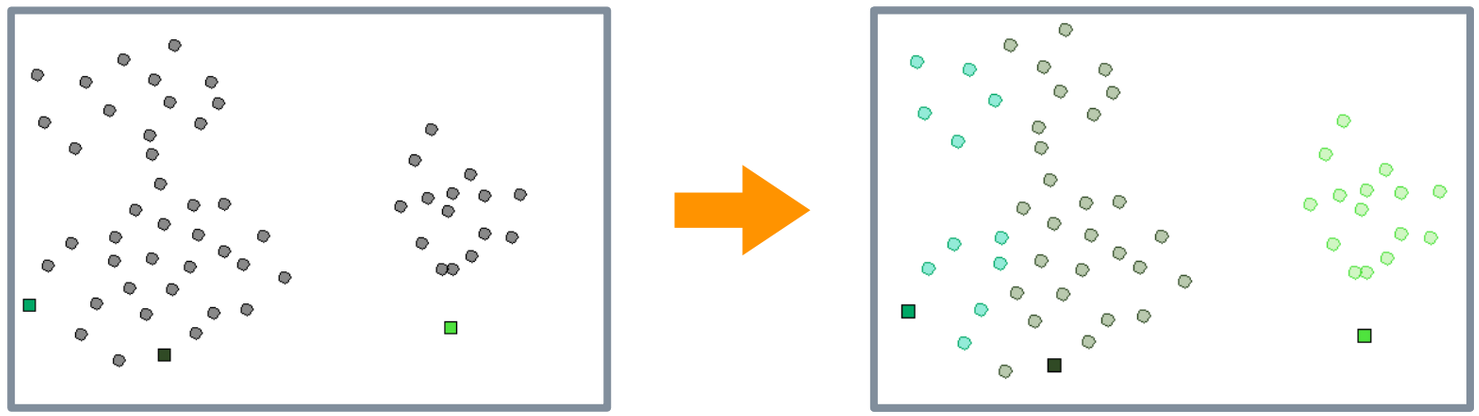


Too many clusters?

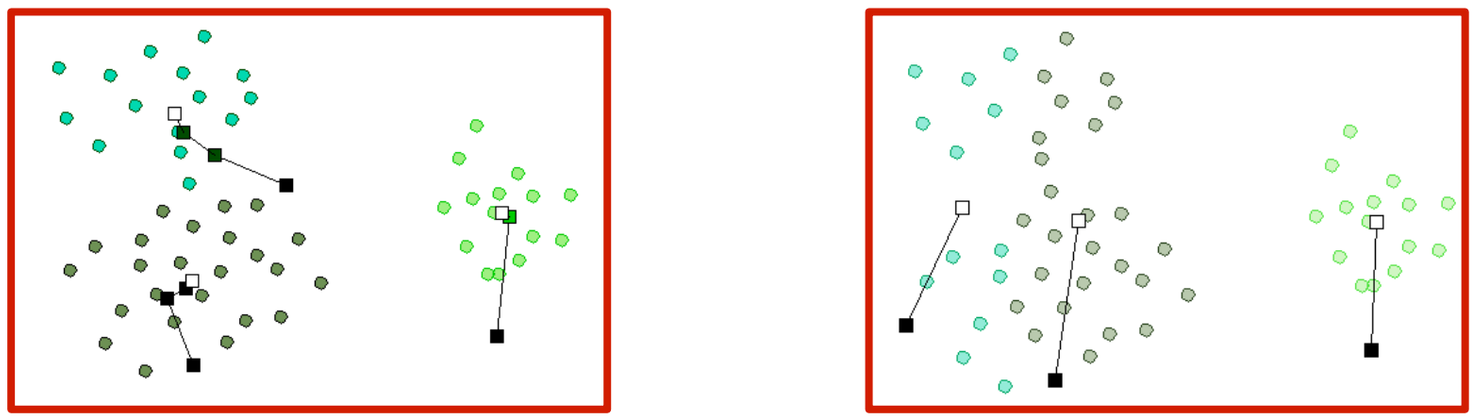
Cluster Initialisation

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*

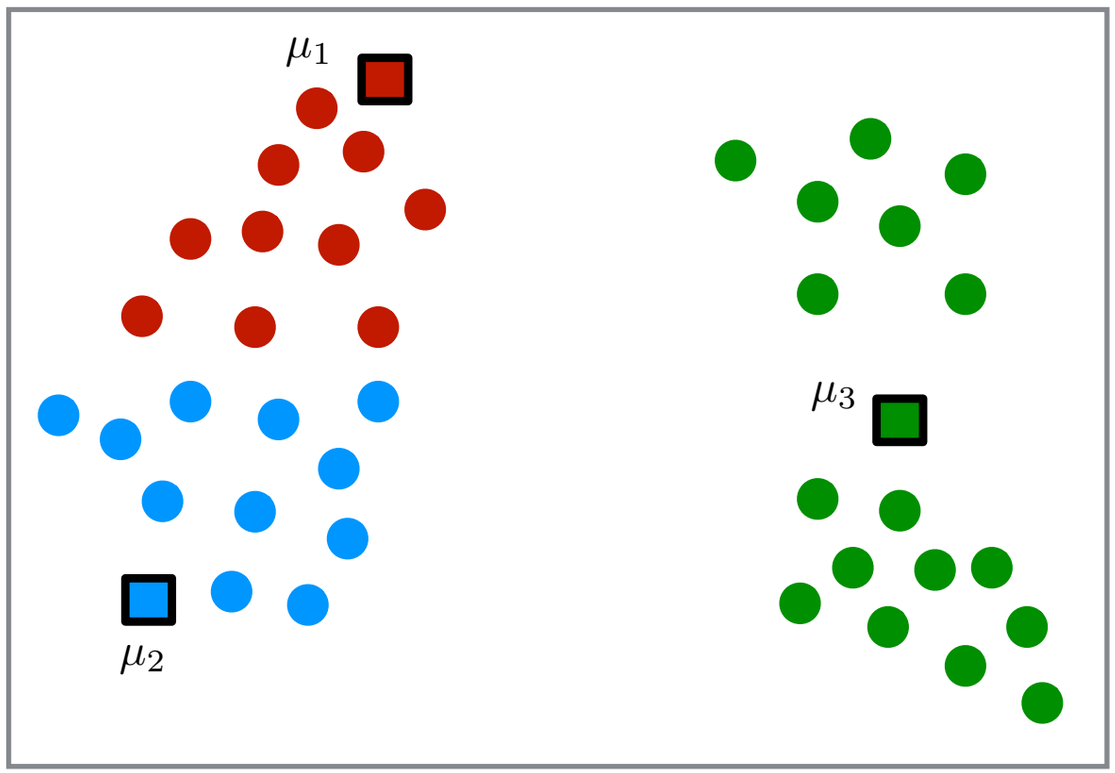


Previous Solution for *k=3*

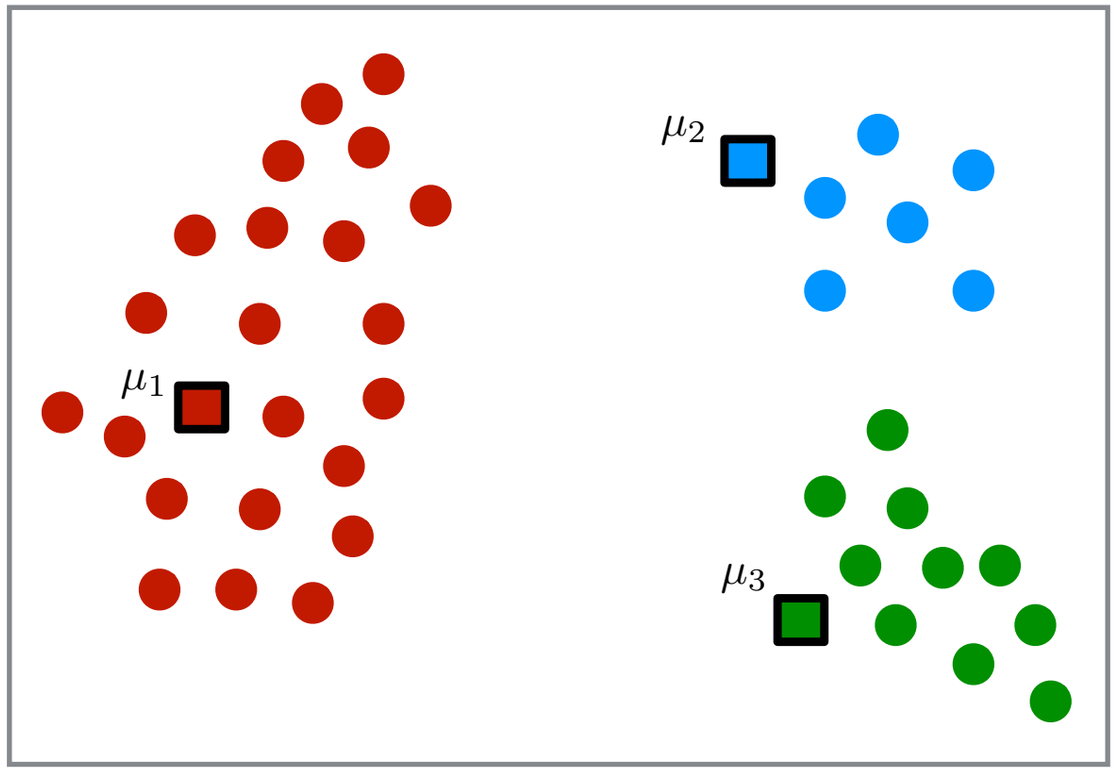


Cluster Initialisation

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

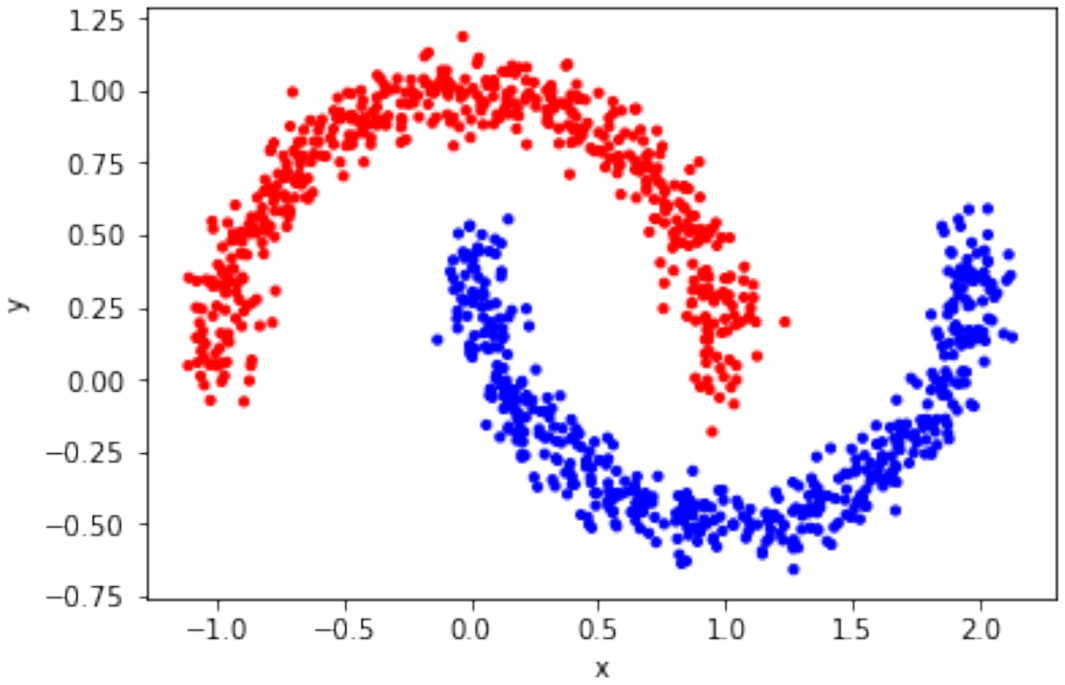
➡ Common strategy: Run the algorithm multiple times, select the solution(s) that scores well according to some validation measure.

Limitations of *k*-Means

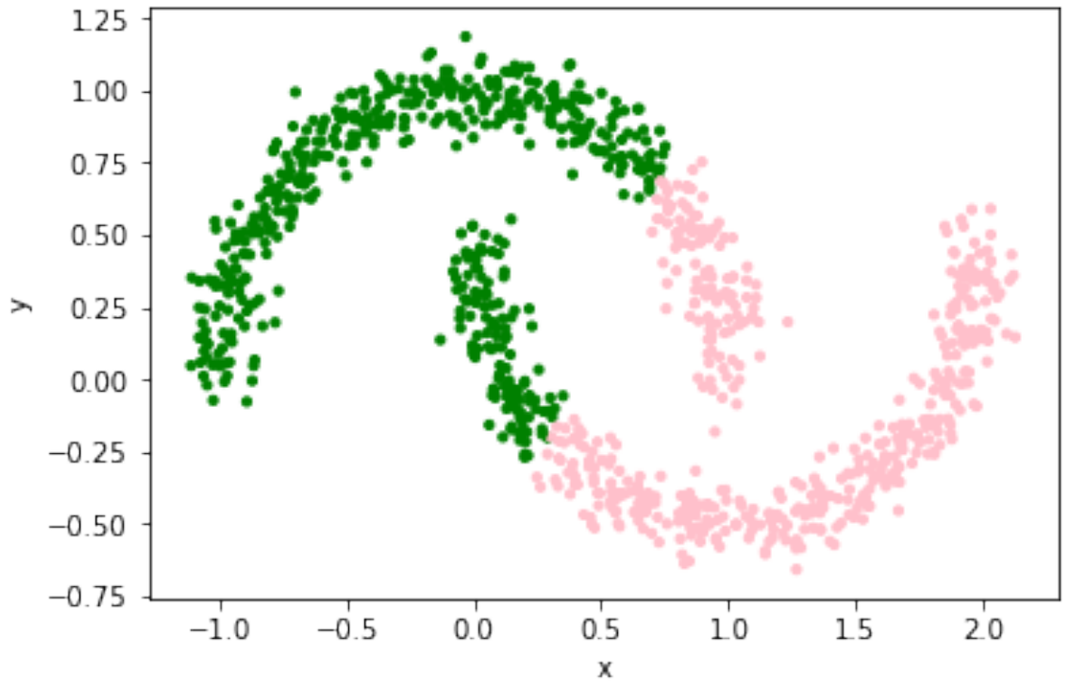
- **Advantages:**
  - Fast, easy to implement.
  - “Good enough” in a wide variety of tasks and domains.
- **Disadvantages:**
  - Must pre-specify number of clusters *k*.
  - Highly sensitive to choice of initial clusters.
  - Assumes that each cluster is spherical in shape and data examples are largely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

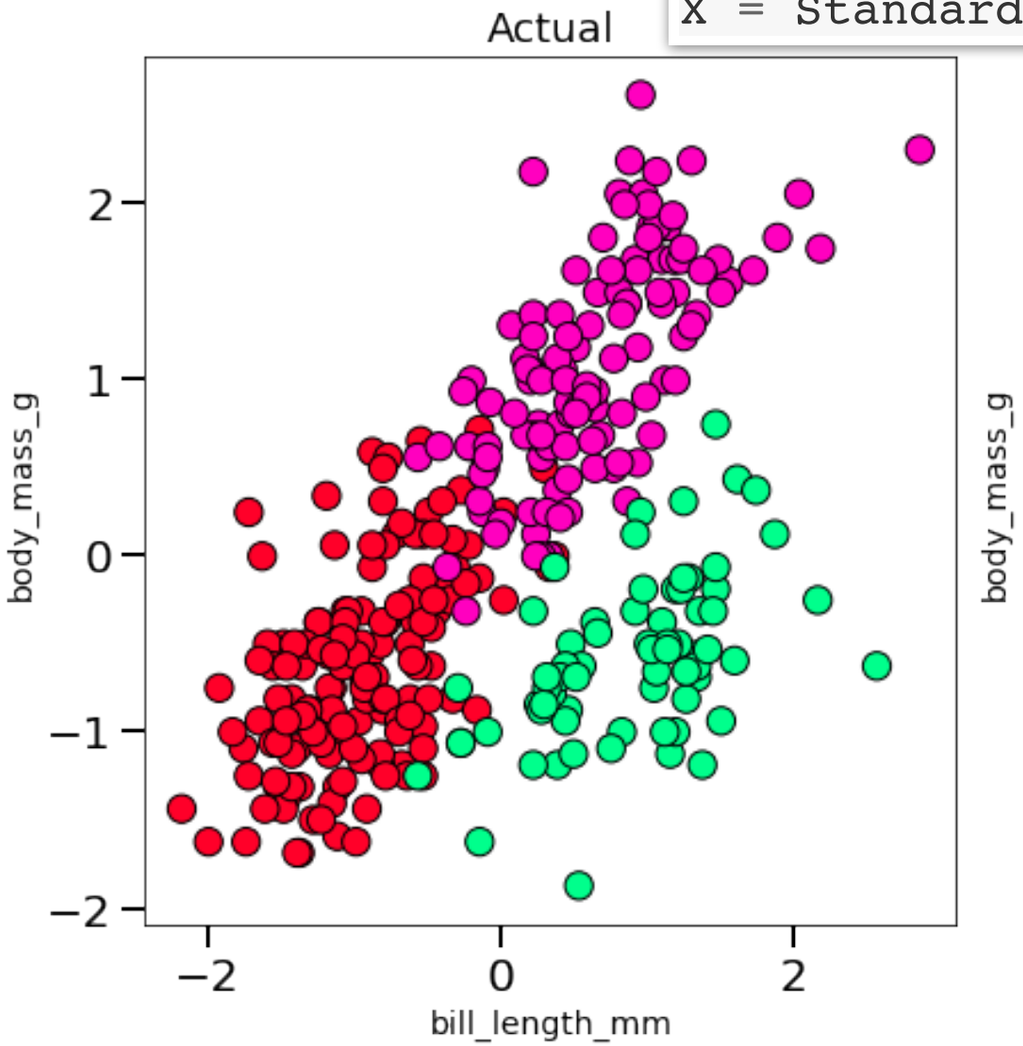
k-Means Clustering sklearn

Encode species as a number

```
penguins_all = pd.read_csv('penguins_af.csv')
penguins = penguins_all[['bill_length_mm', 'bill_depth_mm',
                         'flipper_length_mm', 'body_mass_g']]
LE = LabelEncoder()
penguins_all['code'] = LE.fit_transform(penguins_all['species'])
y = penguins_all['code']

X_raw = penguins.values
X = StandardScaler().fit_transform(X_raw)
```

Scale data

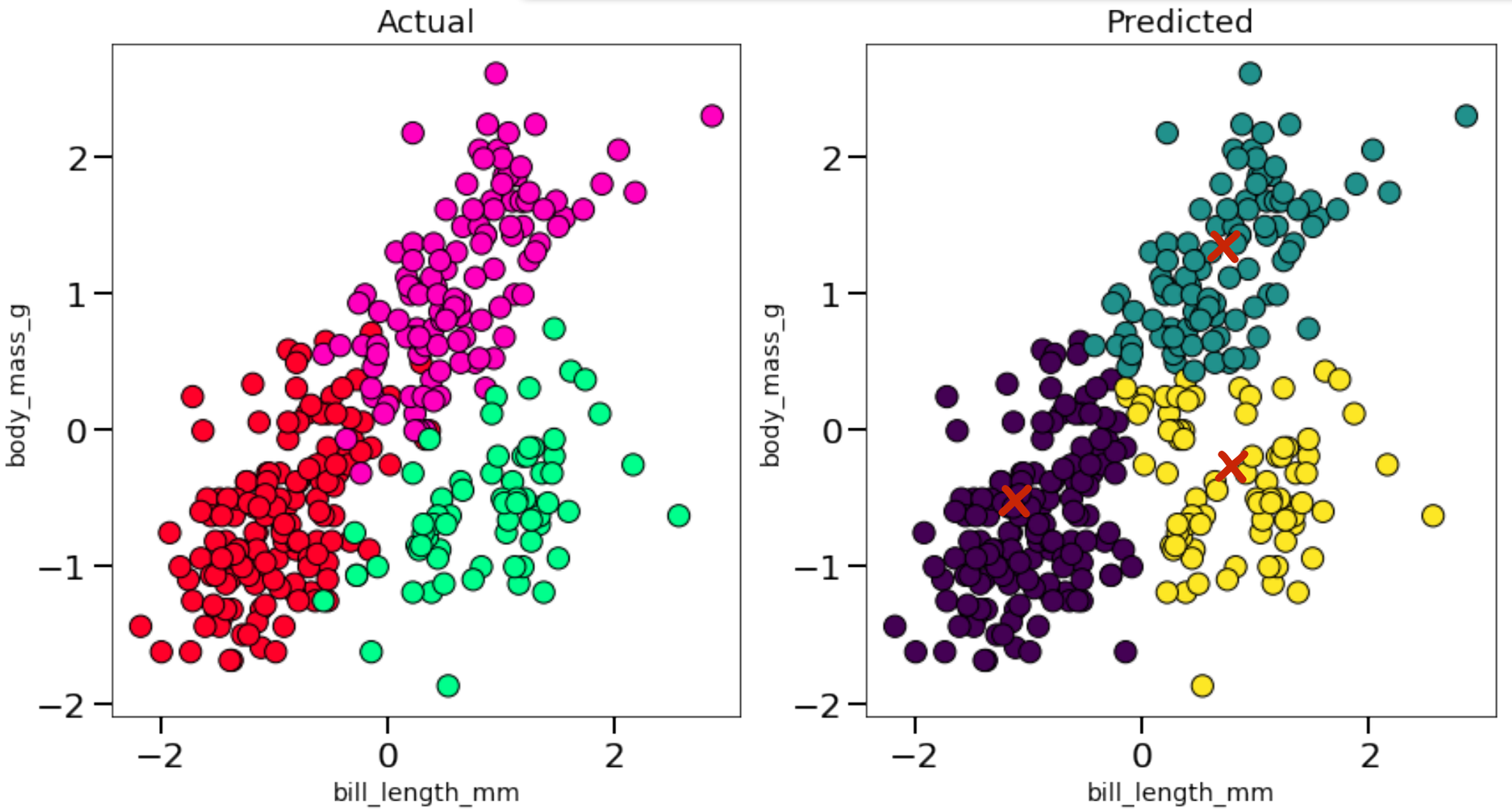



```
features = penguins.columns
f1,f2 = 0,3
```

k-Means Clustering sklearn

```
Xc = X[:, [f1,f2]]  # Select the 0th and 3rd columns
km = KMeans(n_clusters = 3, random_state=1)
km.fit(Xc)
print(km.cluster_centers_)

[[-0.95741688 -0.67079184]
 [ 0.7272949   1.23499466]
 [ 0.81777771 -0.40040812]]
```



Cluster centroids in red

k-Means stability

`inertia_` is SSE

```
for rs in range(5):
    km = KMeans(n_clusters = 3, n_init = 1,
                random_state=rs)
    km.fit(X)
    print("SSE: {:.3f}".format(km.inertia_))
```

One run only per iteration

```
SSE: 372.003
SSE: 370.770
SSE: 371.989
SSE: 371.989
SSE: 372.003
```

Multiple runs

```
km = KMeans(n_clusters = 3, n_init = 10,
            random_state=1)
km.fit(X)
print("SSE: {:.3f}".format(km.inertia_))
```

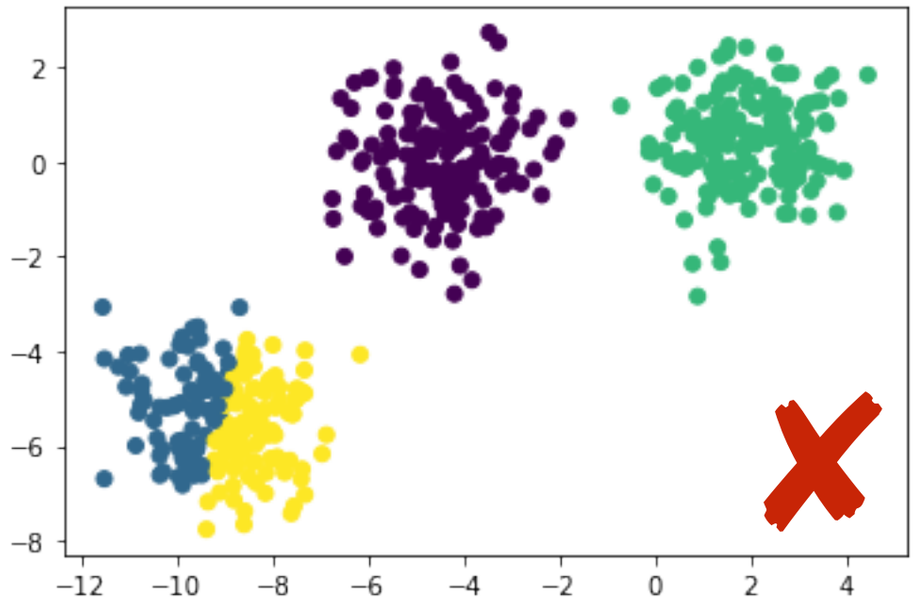
Return best score

```
SSE: 370.766
```

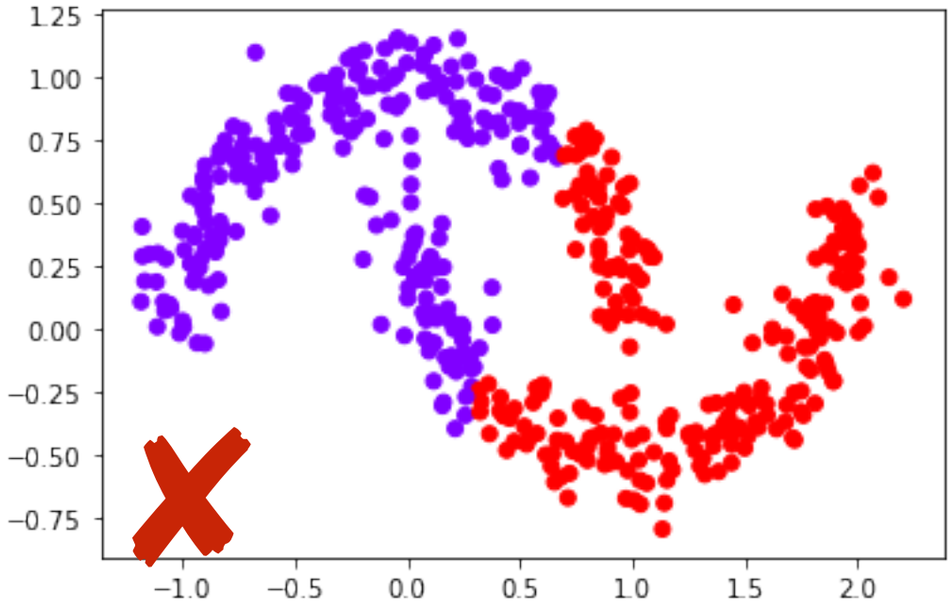
k-Means Shortcomings

Examples in ’17 Clustering k-Means’ notebook

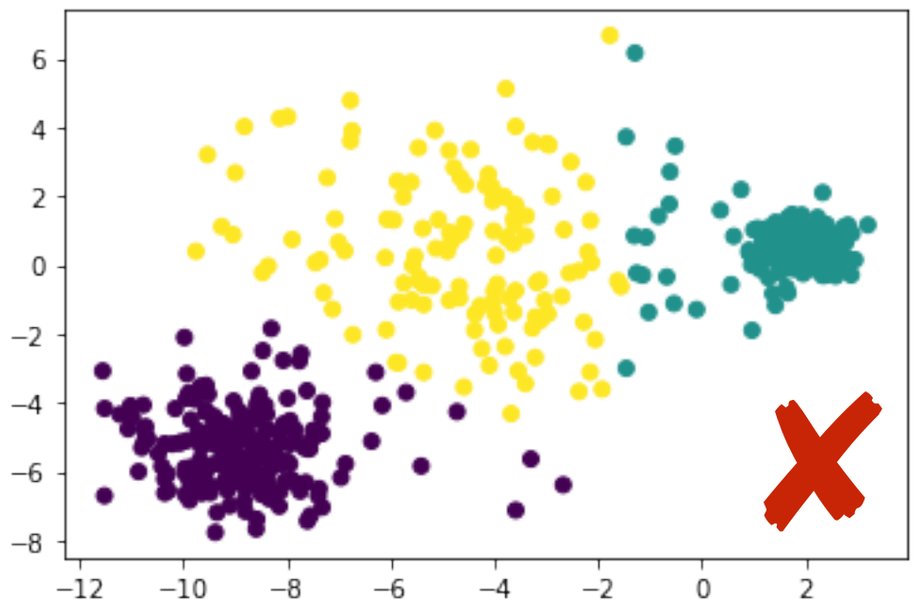
Incorrect *k*



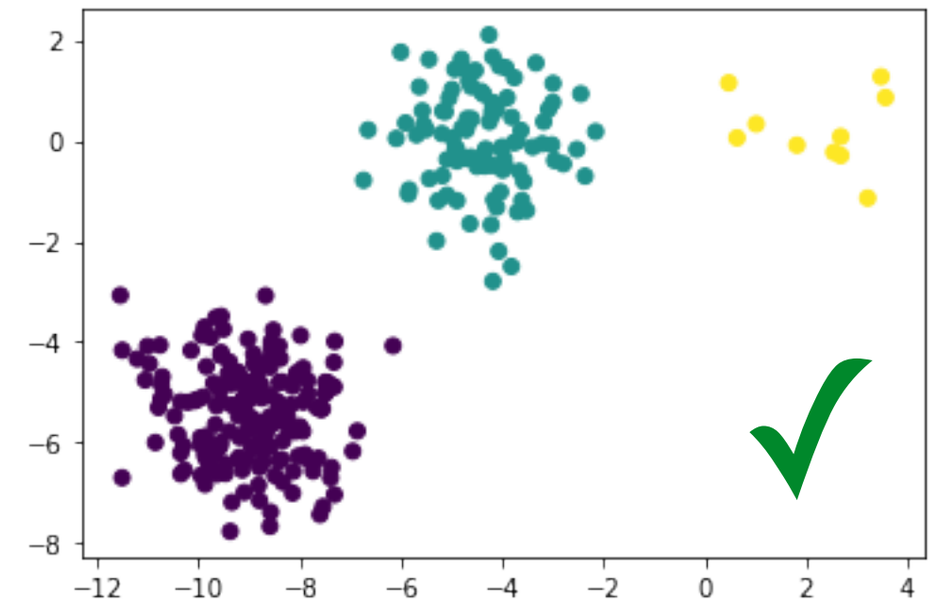
Elongated Blobs



Different size / density



Different density



COMP47750

Clustering - Overview

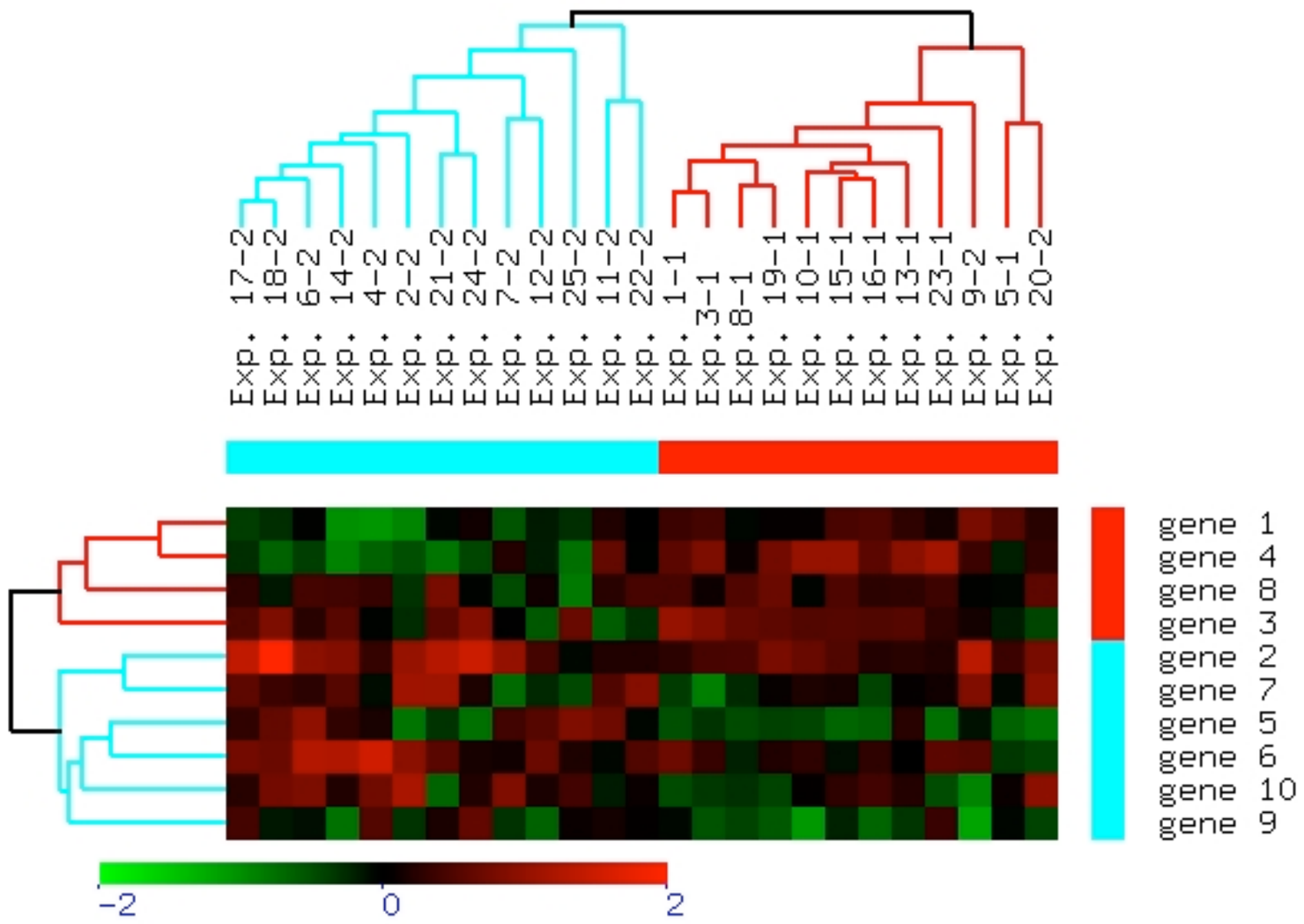
Part II
Hierarchical Clustering

Pádraig Cunningham

School of Computer Science

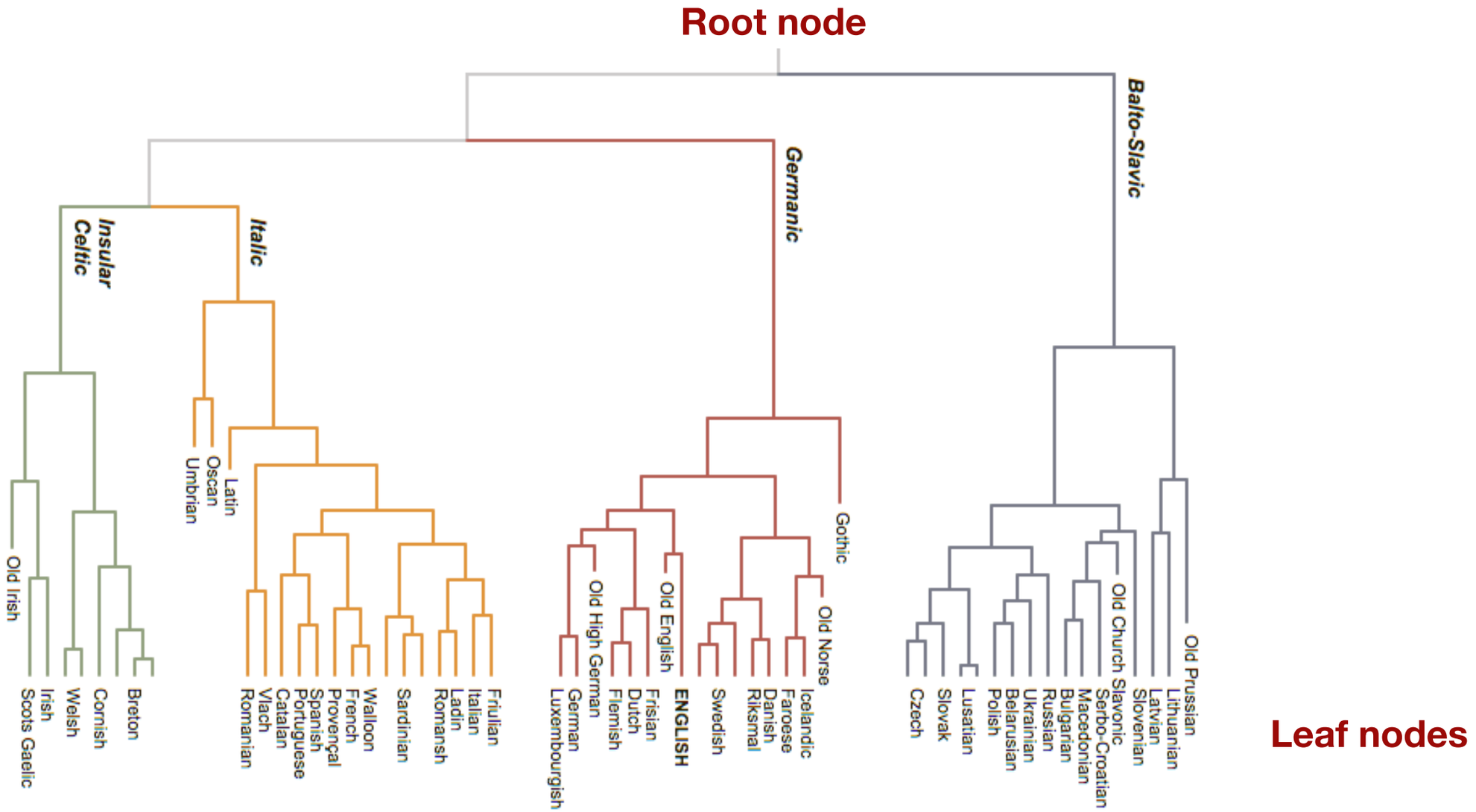
Hierarchical Clustering: Applications

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).



Dendrograms

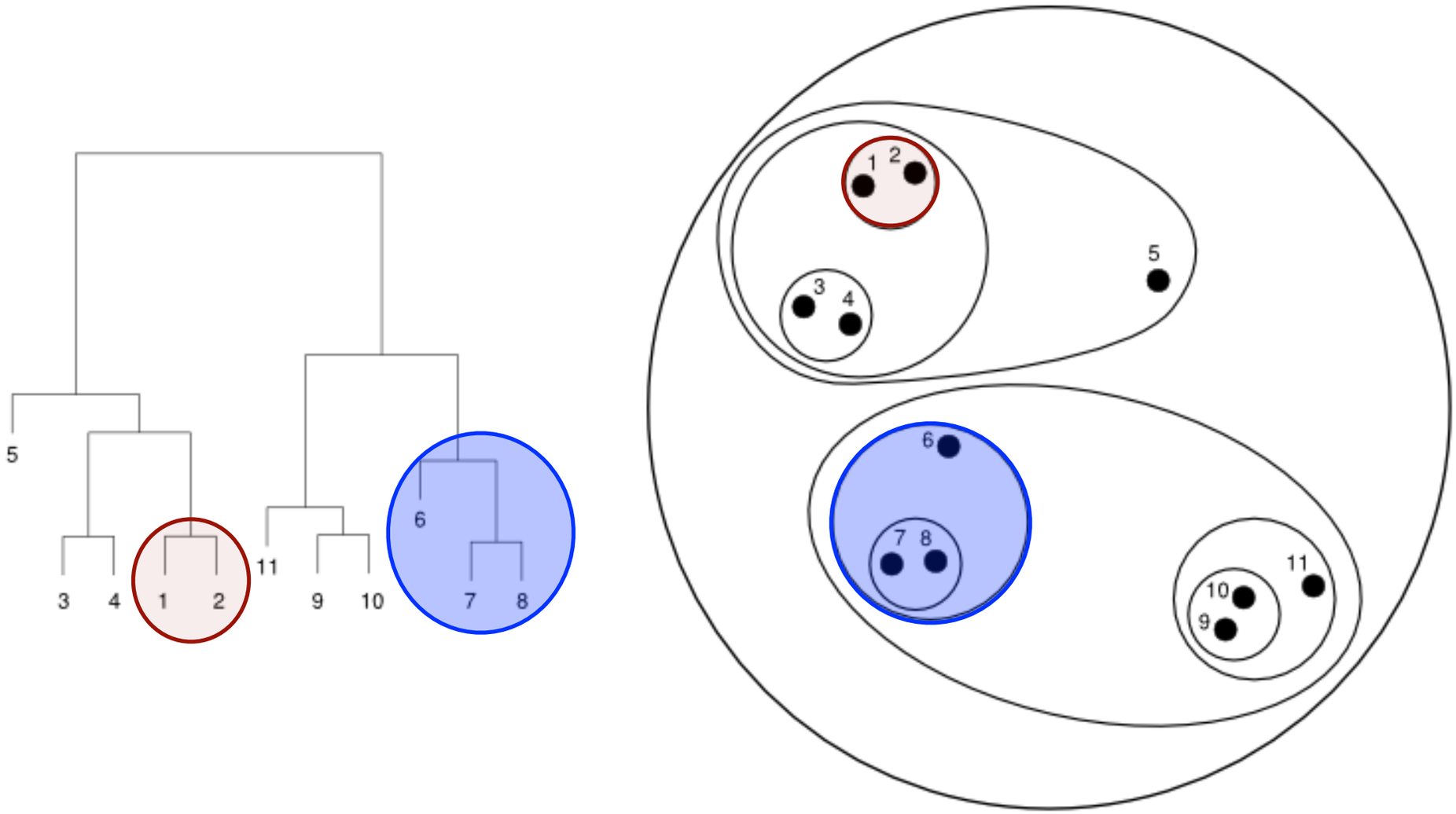
Dendrogram: A tree diagram, frequently used to illustrate arrangement of clusters produced by a hierarchical clustering algorithm. Generic groups are near the top of the tree, more granular groups at the bottom.



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

Dendrograms

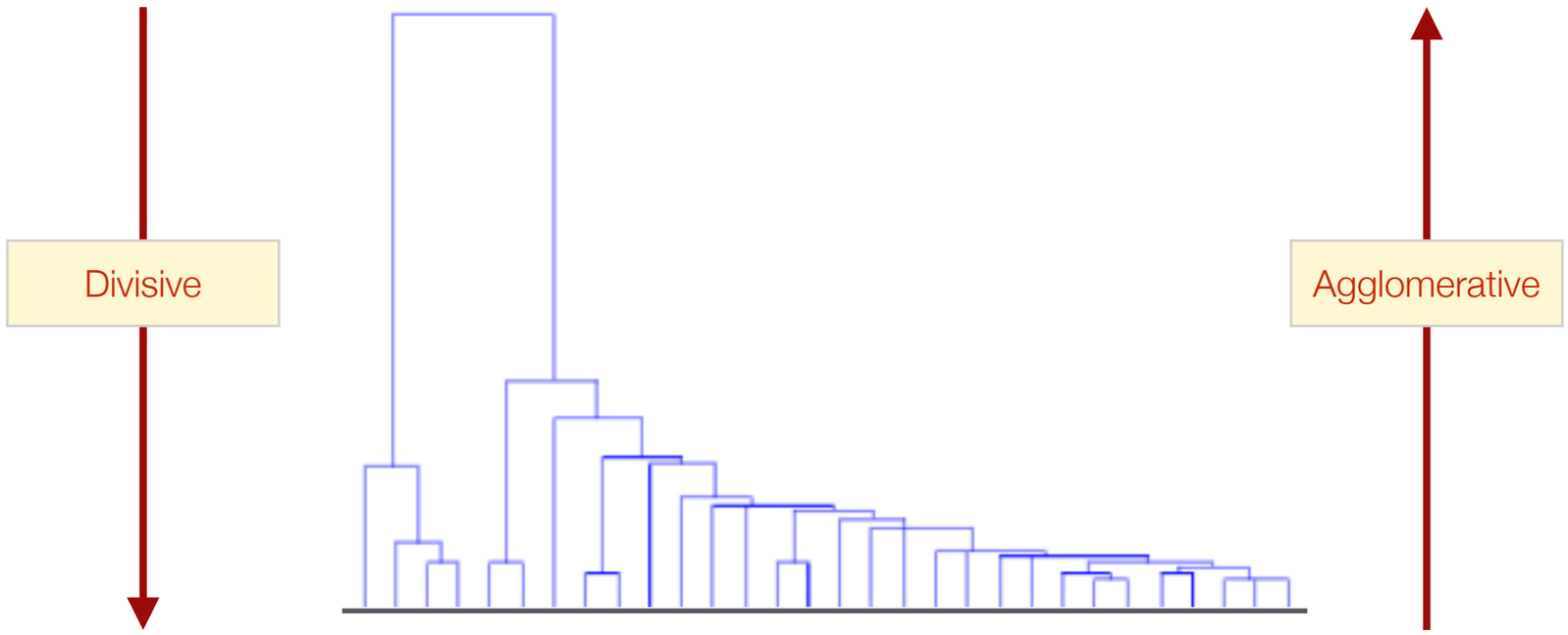
Example: A hierarchical clustering of 11 items in 2 dimensions. Left is a dendrogram representing the clustering, Right is a representation of the same clustering using nested clusters.



http://cs.jhu.edu

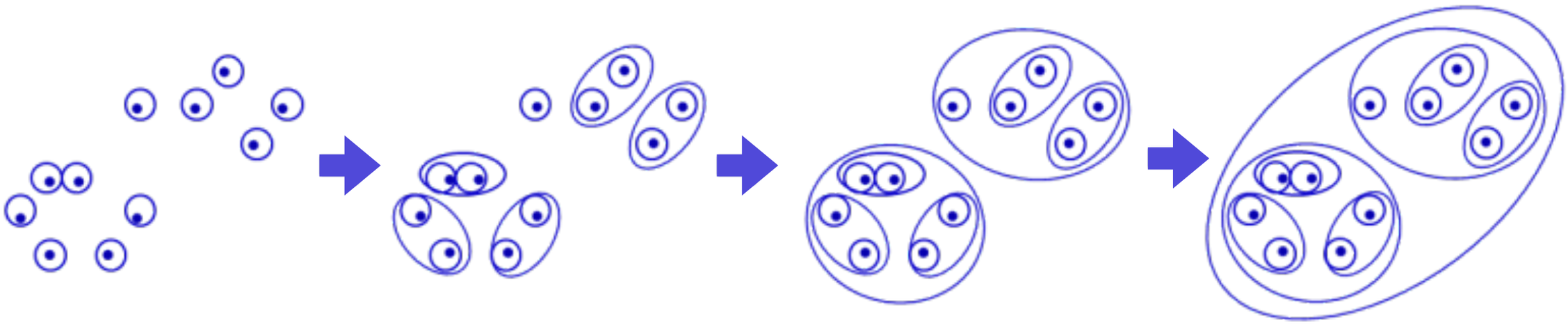
Hierarchical Clustering

- 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.

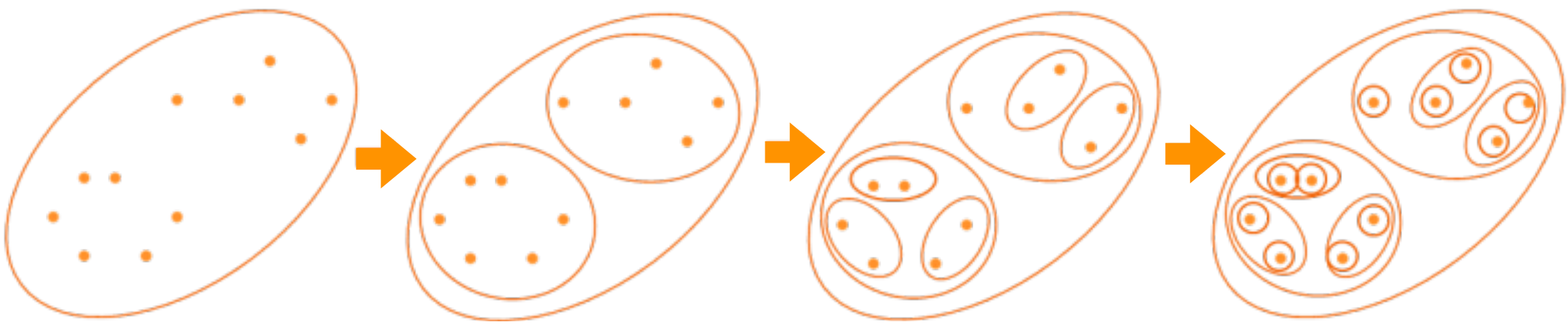


Hierarchical Clustering

Agglomerative Hierarchical Clustering



Divisive Hierarchical Clustering



https://quantdare.com

Agglomerative Clustering

Algorithm Inputs:

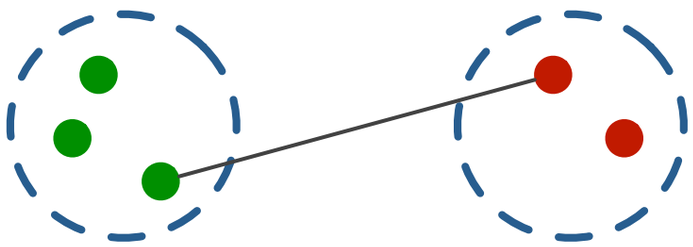
- *Distance matrix* $\mathbf{D}$, specifying the distance between each pair of items in the data, computed using some appropriate measure (e.g. Euclidean).
- *Cluster metric* which helps decide which pair of clusters to merge at each step, using values from $\mathbf{D}$.

Algorithm summary:

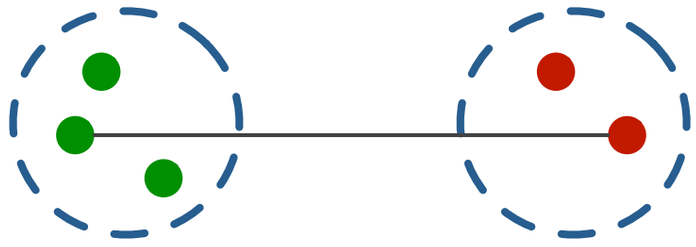
1. Assign every item to its own cluster, each just containing that item. These are the “leaf nodes” of the tree.
2. Find the closest (i.e. most similar) pair of clusters, according to the cluster metric, and merge them into a single cluster, so that now you have one less cluster.
3. Compute distances (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

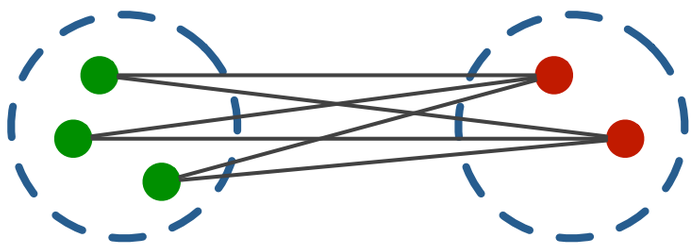
- A variety of metrics exist for determining which pair of clusters should be merged next from among all possible pairs. These specify how we use values from $\mathbf{D}$ to measure the distance between two clusters.
  - Single linkage: Define cluster distance as the smallest pairwise distance between 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 $$d(C_a, C_b) = \frac{\sum_{x_i \in C_a} \sum_{x_j \in C_b} D_{ij}}{|C_a|\,|C_b|}$$

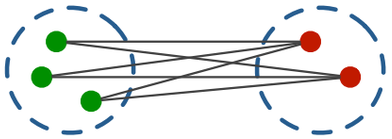
- The choice of cluster distance metric can substantially affect the resulting clustering.

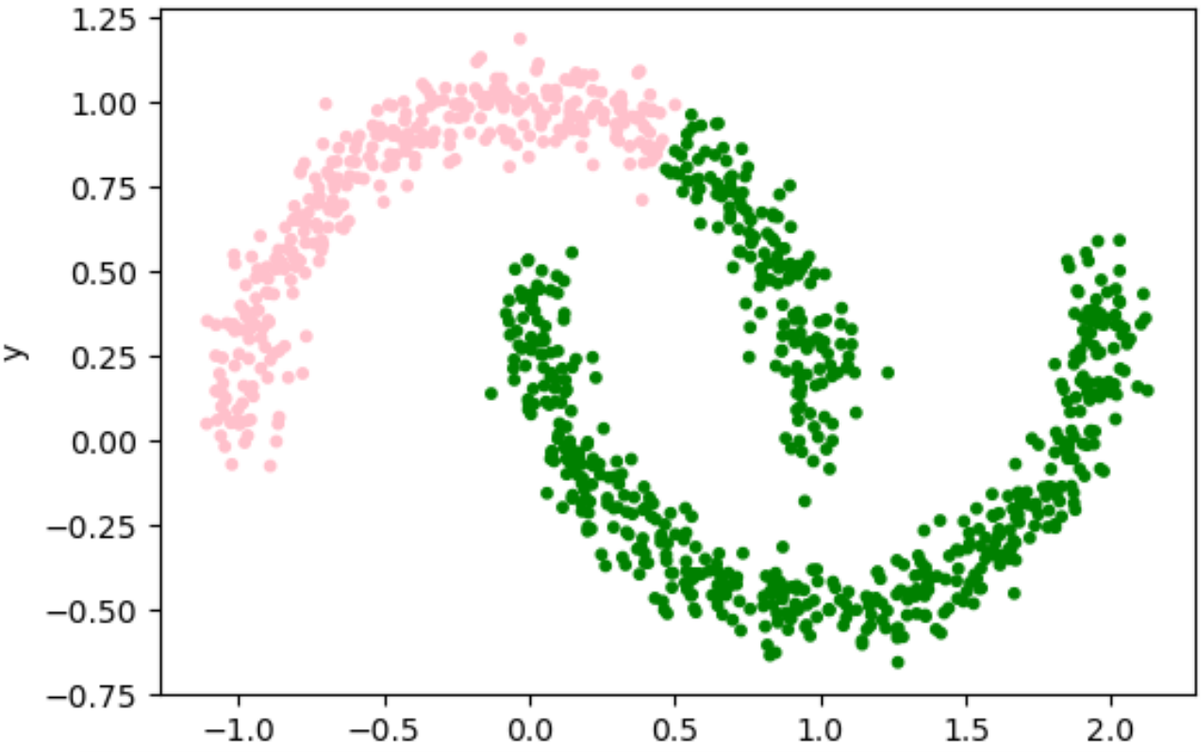
➡ Complete linkage is sensitive to outliers. Single linkage tends to produce long chains, not cohesive clusters.

Hierarchical clustering - metric choice

Single linkage hierarchical clustering can discover clusters that are not compact / spherical.

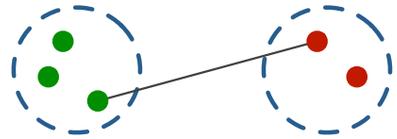
Average

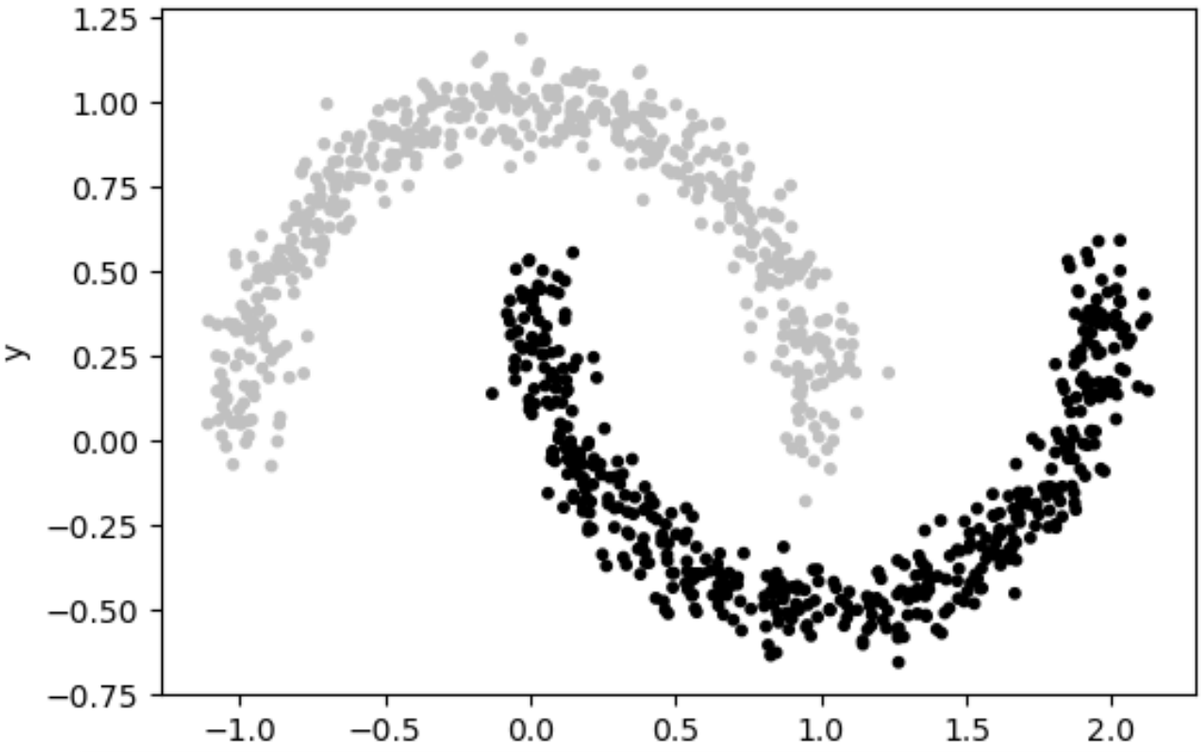





```
hc = AgglomerativeClustering(n_clusters=2,
                             linkage = 'average')
```

Single

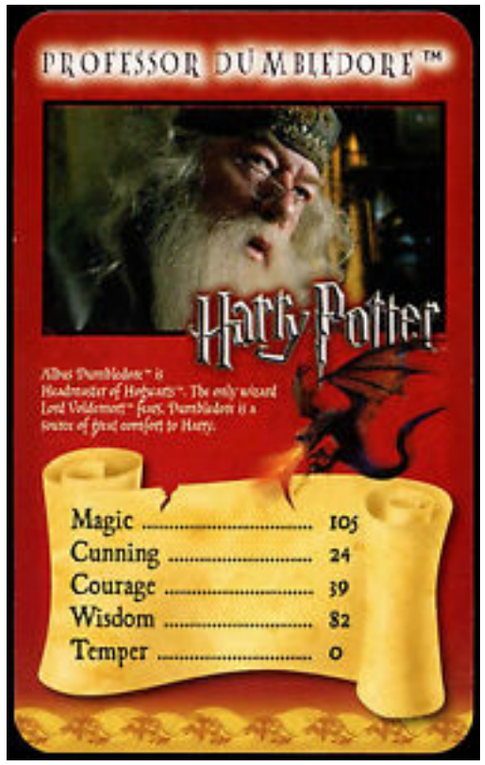


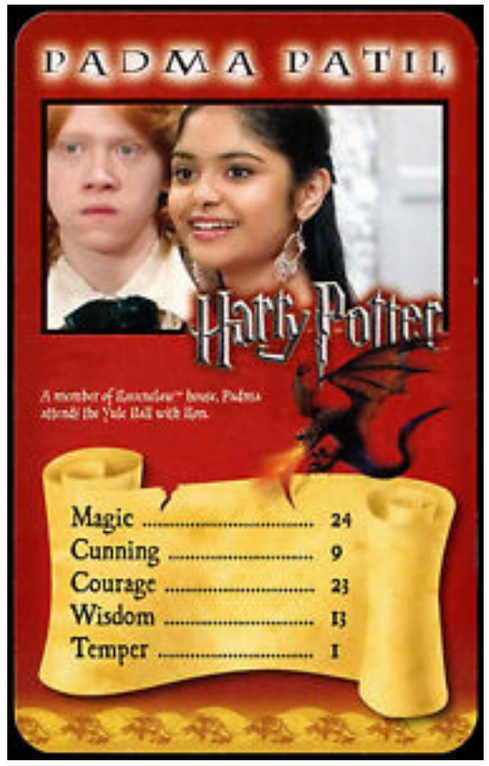



```
hc_s = AgglomerativeClustering(n_clusters=2,
                               linkage = 'single')
```

Notebook:
18 Hierarchical Clustering

Harry Potter Top Trumps data





	Name	Magic	Cunning	Courage	Wisdom	Temper
0	Harry Potter'	62	21	42	26	7
1	Hermione Granger'	60	16	40	73	2
2	Ron Weasley'	45	14	40	22	4
3	Prof. Dumbledore'	105	24	39	82	0
4	Prof. Snape'	85	24	19	71	7
	...	...	...	...	...	...

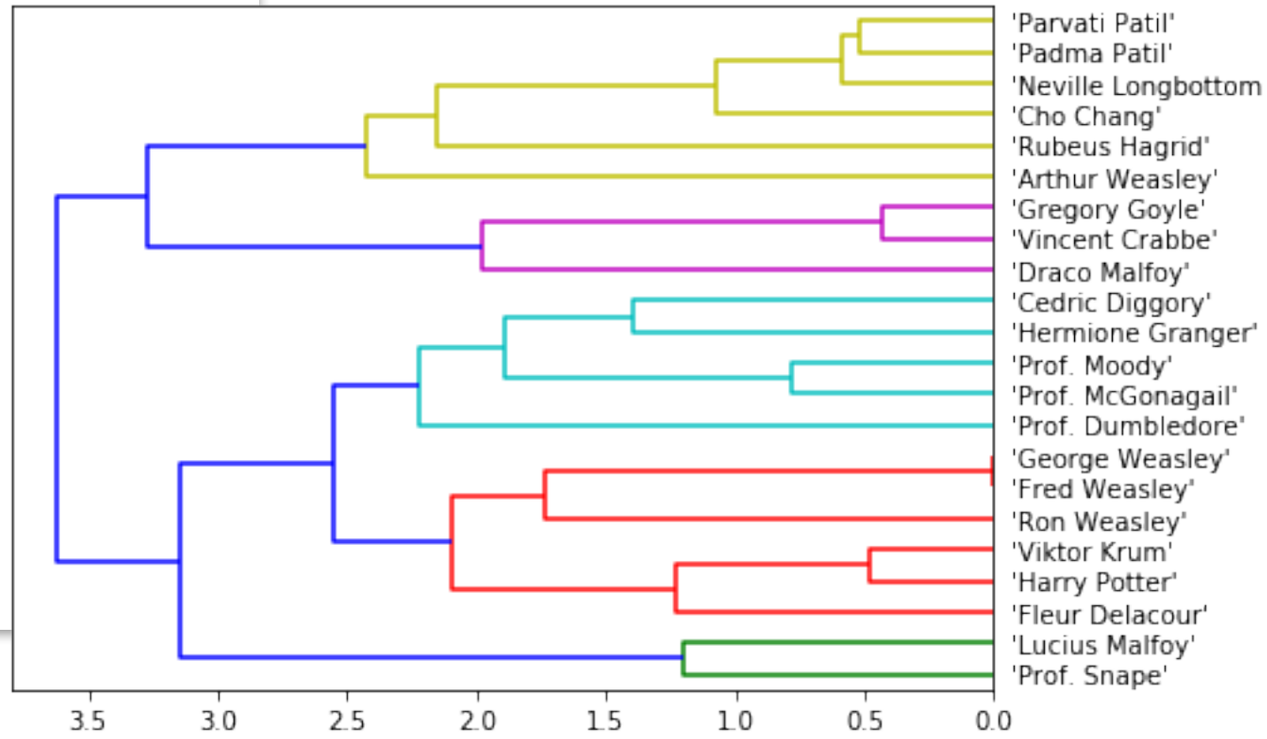
Harry Potter Top Trumps data

```
from scipy.cluster.hierarchy import dendrogram, linkage
from matplotlib import pyplot as plt

linked = linkage(X_scal, 'average')

labelList = list(y)

plt.figure(figsize=(7, 5))
dendrogram(linked,
           orientation='left',
           labels=labelList,
           distance_sort='descending',
           show_leaf_counts=True)
plt.show()
```



	Name	Magic	Cunning	Courage	Wisdom	Temper
0	Harry Potter'	62	21	42	26	7
1	Hermione Granger'	60	16	40	73	2
2	Ron Weasley'	45	14	40	22	4
3	Prof. Dumbledore'	105	24	39	82	0
4	Prof. Snape'	85	24	19	71	7
	...	...	...	...	...	...

Notebook:
18 Hierarchical Clustering

COMP47750

Clustering - Overview

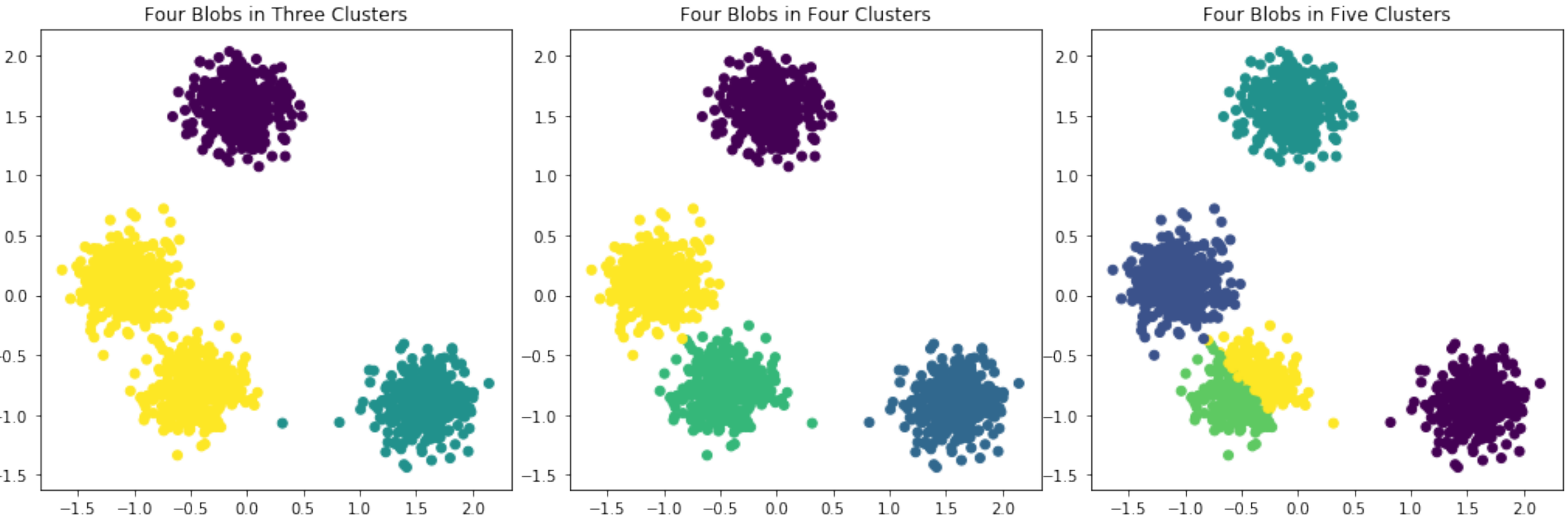
Part III
Cluster Validation

Pádraig Cunningham

School of Computer Science

Cluster Validation

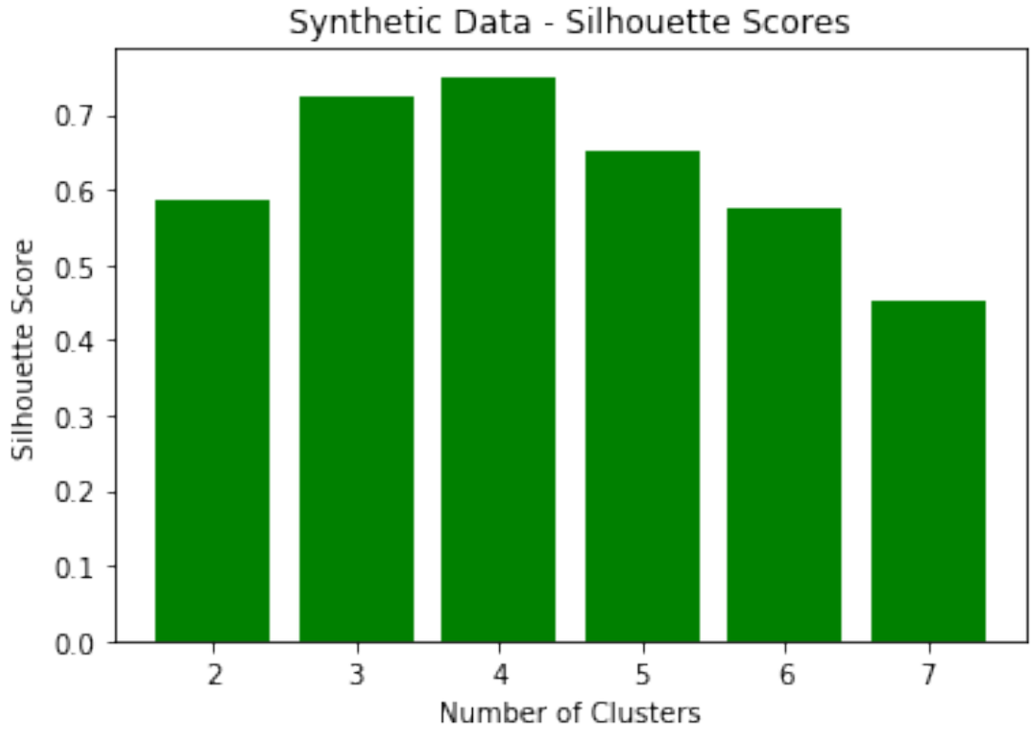
Q. How many clusters in a given data set? Usually will not know in advance…



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

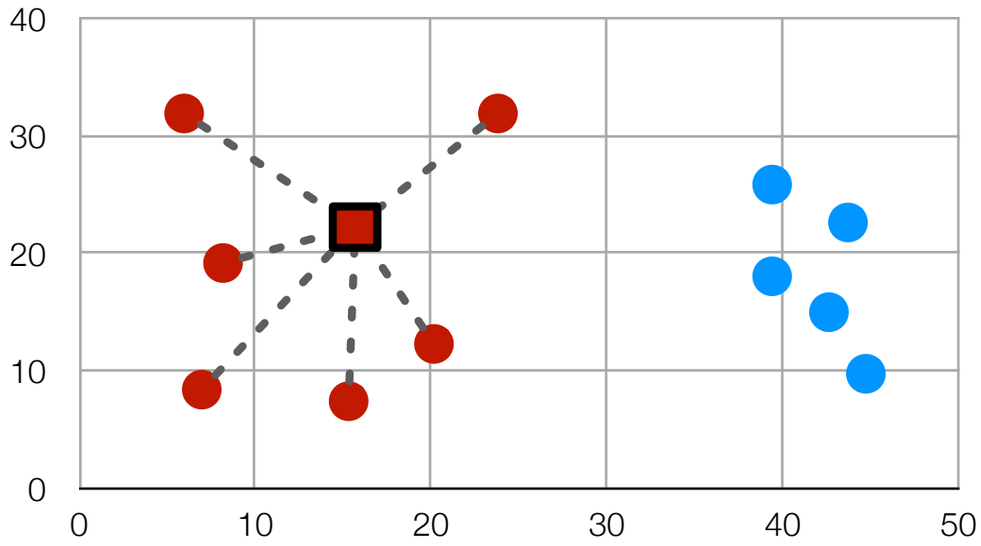
Cluster Validation

- 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*.



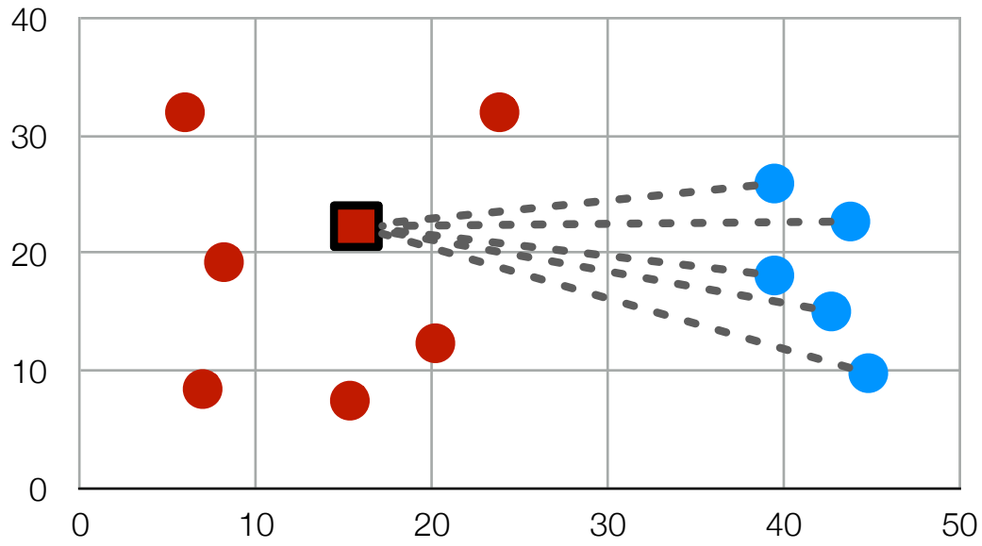
Silhouette Measure

- 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.

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



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

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

- 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

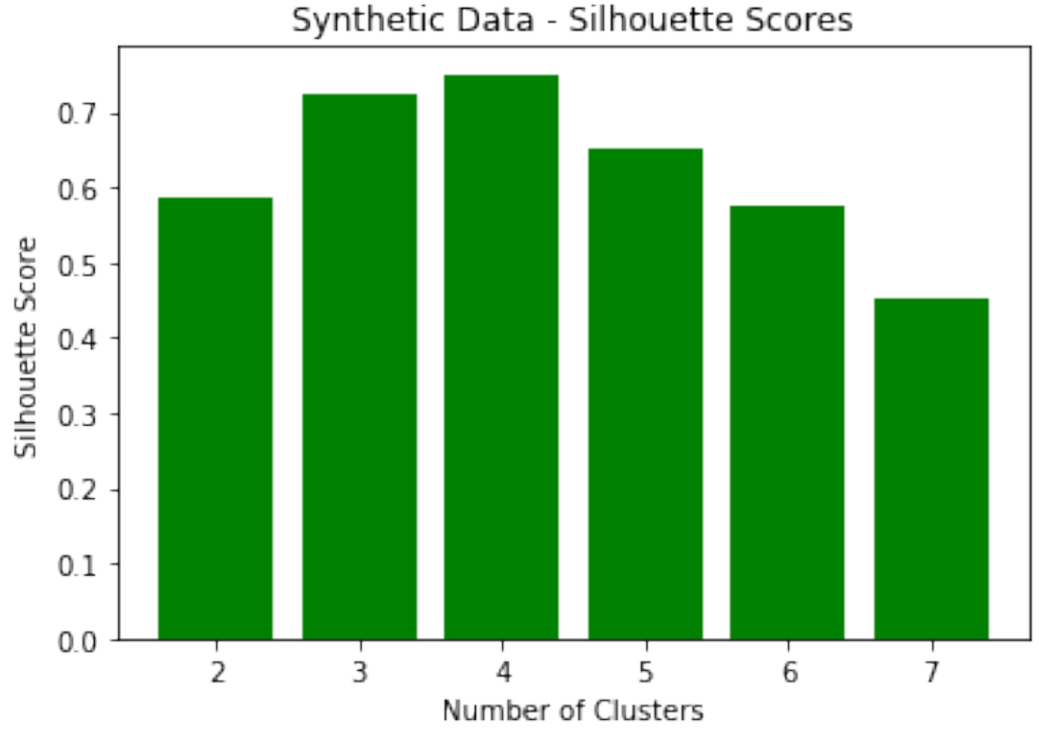
- Silhouette width for a single item x_i is calculated as s_i.
- Average Silhouette Width (ASW): Calculate overall score for a clustering by averaging the silhouette widths for all n items.

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

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

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*.



Overview

- Supervised v Unsupervised Learning
- Partitional Clustering
  - *k*-Means clustering
  - Cluster initialisation
- Hierarchical Clustering
- Cluster Validation