(MSc) Data Mining

Topic 05 : Clustering

Part 01 : Overview

Dr Bernard Butler and Dr Kieran Murphy

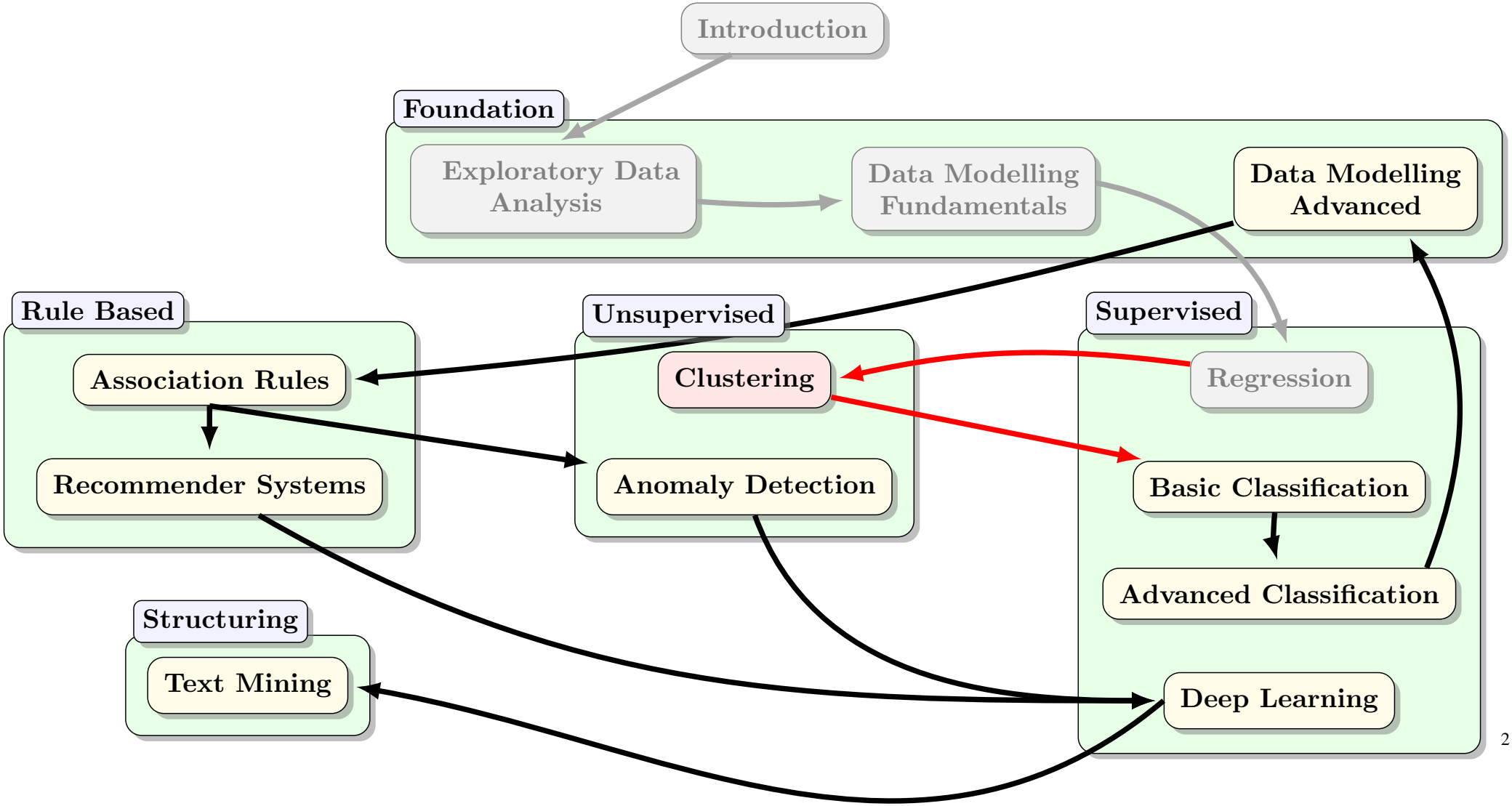
Department of Computing and Mathematics, WIT.
(bernard.butler@setu.ie; kmurphy@wit.ie)

Spring Semester, 2025

Outline

- How to compute distances between instances
- Algorithms that generate multiple clustering schemes
- Algorithms that partition the data

Data Mining (Week 5)



Overview — Summary

This Week's Aim

This week's aim is to introduce the main concepts and representative algorithms used in cluster analysis.

- Introduction to unsupervised learning
- Clustering as a means of understanding data
- Choice of distance function and metaparameters
- Hierarchical Clustering
- Clusters that partition the data
  - Iris data: predicting which of three species

Clustering is a long-established form of analysis, having much in common with exploratory data analysis. We look at the main concepts and algorithms today.

Background: Unsupervised Learning

Definition 1 (Unsupervised Learnng)

With unsupervised learning, the system receives input instances $x_1, x_2, \ldots$ but obtains neither target outputs, nor rewards from its environment. Its goal is to build representations of the input that can be used for decision making, predicting future inputs, efficiently communicating the inputs to another machine, etc. It does this by finding patterns in the data beyond what would be considered pure unstructured noise.

Regression and classification are examples of supervised learning because they require labeled *training* data.
We saw one unsupervised technique (*dimensionality reduction*) previously. Other unsupervised learning techniques include *anomaly detection* and the very "hot" *Generative Adversarial Networks* of deep learning. We look at *clustering* today.

Introduction to Clustering

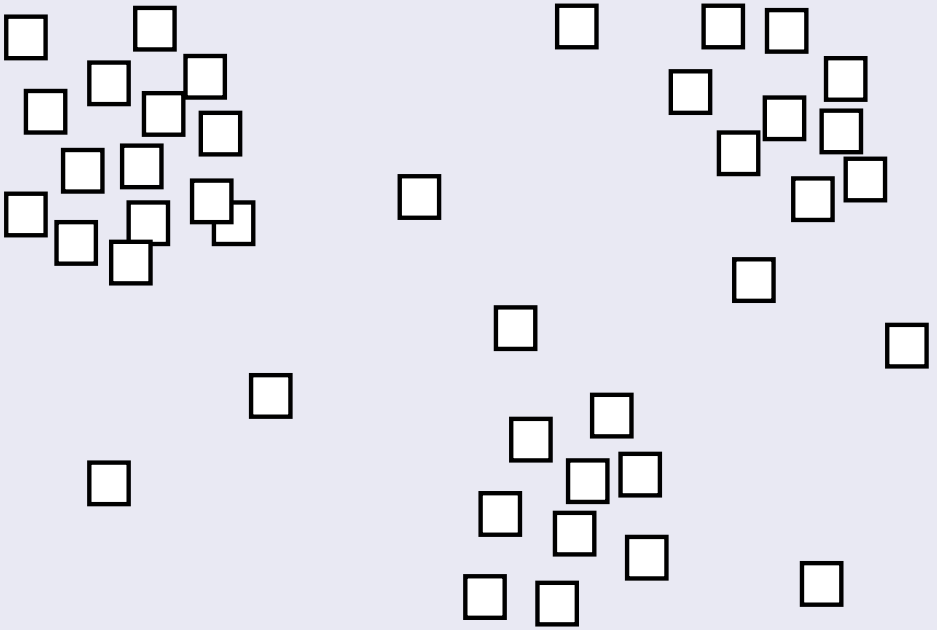
Definition 2 (Clustering)

Clustering is the operation of grouping objects into a smaller number of clusters (or segments), which have two properties. Firstly, they are not defined in advance by an analyst, but are discovered during the operation, unlike the classes used in classification. Secondly, the clusters combine objects having similar characteristics, which are separated from objects having different characteristics (resulting in *internal homogeneity* and *external heterogeneity*.

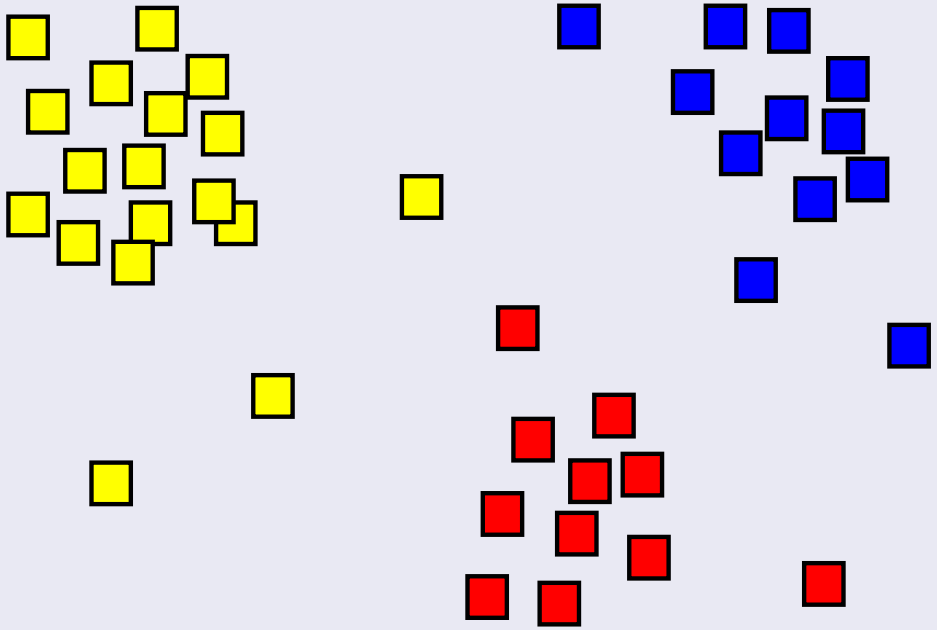
Clustering is usually called *segmentation* in marketing studies.
Usually clustering is not an end in itself: it generates insights that are used to motivate and inform other analyses.
The "quality" of a cluster analysis is difficult to determine objectively - there is no equivalent of *recall*, say.

Example Applications

Before Clustering...



After Clustering...



Picture source: wikipedia

Identify possible applications for clustering

Hierarchical versus Partitional techniques

Hierarchical

- Intermediate steps are interpretable
- More than one clustering generated - see *dendrogram*
- Given choice of linkage, algorithm proceeds *deterministically*
  - no concept of starting values
  - no concept of local versus global optimum
- relatively few parameters to specify
- more complex interpretation

Partitional

- Interpret the final clustering only
- Single clustering returned; repeat with different conditions to improve it
- Optimisation by gradient descent, so
  - result depends on starting values
  - might find local rather than global optimum
- more parameters to specify
- interpretation is relatively easy

Distance Measures and their role in clustering

Definition 3 (Distance Measure)

A *distance measure* (c.f., its complement, a *similarity measure*) is a scalar number $d(x_1, x_2)$ that quantifies the degree of agreement between two (usually vector-valued) observations x_1 and x_2. When $x_1 = x_2$, $d(x_1, x_2) = 0$ and $d(x_1, x_2) > 0$ otherwise. It increases as the difference in the observations increases.

By definition, clustering is based on within-cluster homogeneity (measured by small d) versus large d between clusters. Thus choice of distance measure plays a critical part in generating useful clusters.

Distance Measures for numeric data

Definition 4 (Minkowski $p-$norm)

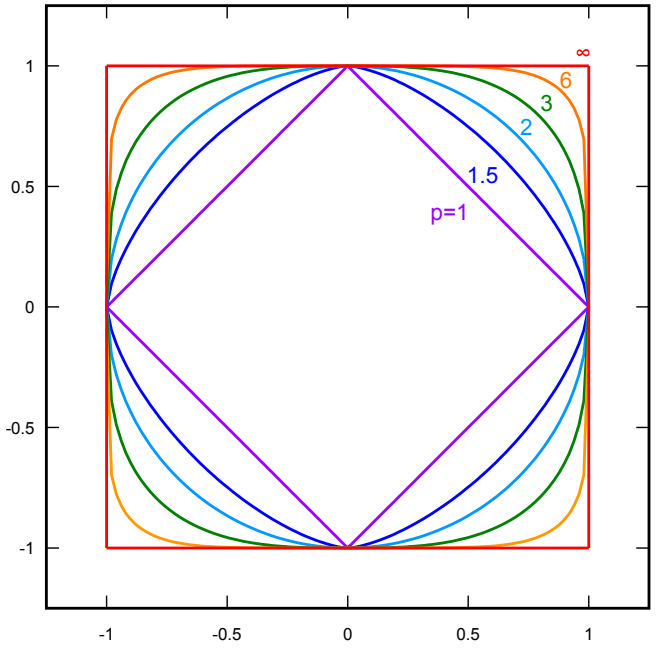
For a real number $1 \leq p < \infty$, the p-norm of $\boldsymbol{x}$ is defined by

$$\|\boldsymbol{x}\|_p \equiv \left(|x_1|^p + |x_2|^p + \ldots + |x_n|^p\right)^{\frac{1}{p}}.$$

The limiting case of $p = \infty$ is defined as

$$\|\boldsymbol{x}\|_\infty \equiv \max\{|x_1|, |x_2|, \ldots, |x_n|\}.$$

See the visualisation of the "unit balls" alongside, for $p = 1, 1.5, 2, 3, 6, \infty$.



Source: wikipedia

The most common norms are when $p = 1, 2, \text{or}, \infty$. Choice of p depends on the application scenario. Can you think of when you would use each?

(Selected) Distance Measures for categorical data

Let $\boldsymbol{x}_1 = [e_{1,1}, e_{1,2}, \ldots e_{1,k}]^T$ and $\boldsymbol{x}_2 = [e_{2,1}, e_{2,2}, \ldots e_{2,k}]^T$. Furthermore let $e_{1,j}e_{2,j} = 1$ if $e_{1,j} = e_{2,j}$ and $e_{1,j}e_{2,j} = 0$ otherwise. To compute s, the number of matching attributes between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, we can just compute the dot product:

$$s = \boldsymbol{x}_1^T \boldsymbol{x}_2$$

and the number of mismatches is $d = k - s$, where k is the number of attributes in $\boldsymbol{x}$.

Definition 5 (Euclidean distance for categorical observations)

$\|\boldsymbol{x}_1 - \boldsymbol{x}_2\| = \sqrt{\boldsymbol{x}_1^T\boldsymbol{x}_1 - 2\boldsymbol{x}_1^T\boldsymbol{x}_2 + \boldsymbol{x}_2^T\boldsymbol{x}_2} = \sqrt{2(k-s)}$. So the maximum distance occurs when $s = 0$ ($\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ share no attribute values in common, as expected).

Definition 6 (Hamming Distance)

This is the number of mismatched values $k - s$.

(Selected) Distance Measures for categorical data - ratios

Definition 7 (Cosine similarity)

$\cos\theta_{1,2} = \frac{\boldsymbol{x}_1^T\boldsymbol{x}_2}{\|\boldsymbol{x}_1\|\|\boldsymbol{x}_2\|} = \frac{s}{\sqrt{k}\sqrt{k}} = \frac{s}{k}$.

because $\|\boldsymbol{x}\| \equiv \sqrt{\boldsymbol{x}^T\boldsymbol{x}} = \sqrt{k}$.

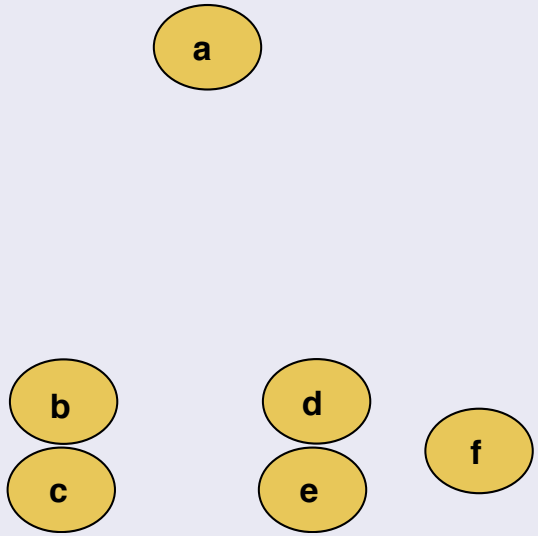
Definition 8 (Jaccard Coefficient)

This is the ratio of the number of matching values s to the number of distinct values that appear in $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, across the d *distinct* attributes of both. It is $J(\boldsymbol{x}_1,\boldsymbol{x}_2) = \frac{s}{2(k-s)+s} = \frac{s}{2k-s}$.

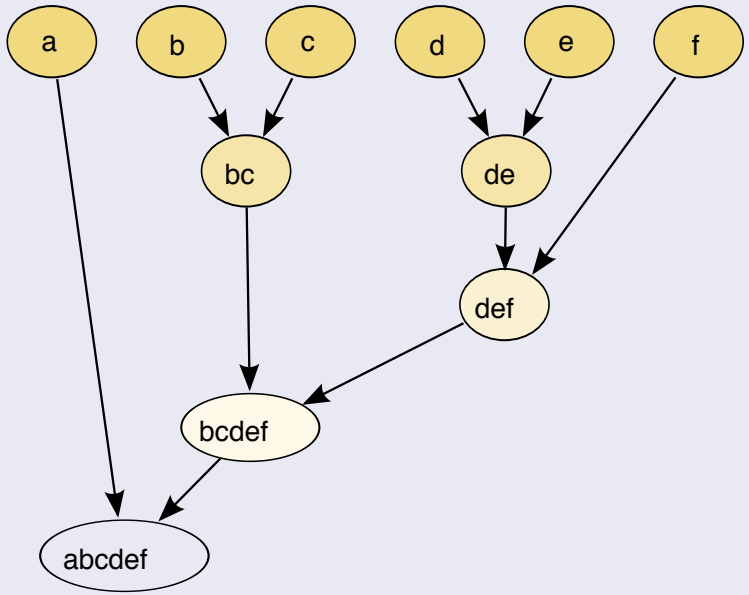
Note that all these distance measures are functions of s and k, where k is a constant and s is a count of the number of matching attribute values across the two observations in question.

Simple example of data and its dendrogram

Data



Dendrogram



This illustrates how *agglomerative* clustering works. *Divisive* clustering works in the opposite direction, but the (flipped) dendrogram is the same in this case.
Sometimes, the vertical separation between Level i and $i + 1$ of the tree indicates the distance between clusters that are merged at Level $i + 1$.
Diagram Source: wikipedia

Dendrograms

Definition 9 (dendrogram)

The dendrogram shows clusters and their subclusters, in the form of a tree. The root cluster contains all elements; each leaf contains a single element. Clusters are merged in order of their similarity.

The dendrogram makes the internal similarity structure of the data more visible.

Sometimes, the vertical separation between Level i and $i + 1$ of the tree is proportional to the distance between clusters that are merged at Level $i + 1$.

An alternative representation is to display the data as a point cloud and to overlay nested clusters over the data.

Overview of Hierarchical Clustering Algorithm

Method (Agglomerative hierarchical clustering (AGNES))

Initialise the Cluster set $C = \{\boldsymbol{x}_i\}, i = 1, \ldots, n$;
$q \leftarrow |\{c_i\}| = n$;
Compute the $n \times n$ proximity matrix D where $D_{ij} = d(c_i, c_j)$;
repeat
  Find i, j associated with $\min_{i,j} D$, where i, j are indices of clusters that are nearest each other;
  Create the merged cluster c_i' containing the elements of cluster c_i and c_j;
  Record the merge operation so the dendrogram data structure can be built;
  Drop the old c_j cluster since it is not needed any more;
  Delete row $D(j, :)$ and column $D(:, j)$ from D
  $q \leftarrow q - 1$;
  Update row $D(i, :)$ and column $D(:, i)$ to compute distance between new cluster c_i' and remaining $q - 2$ clusters;
until $q = 1$ and hence only one cluster remains;

As can be seen, this is a deterministic search algorithm.
However, there is scope for different definitions of the distance function $d(c_i, c_j)$ betwen clusters c_i and c_j.

Distance between clusters: linkage

Earlier, we looked at different ways of computing the *distance between two points*.
For hierarchical clustering, we need to compute the *distance between two clusters*.

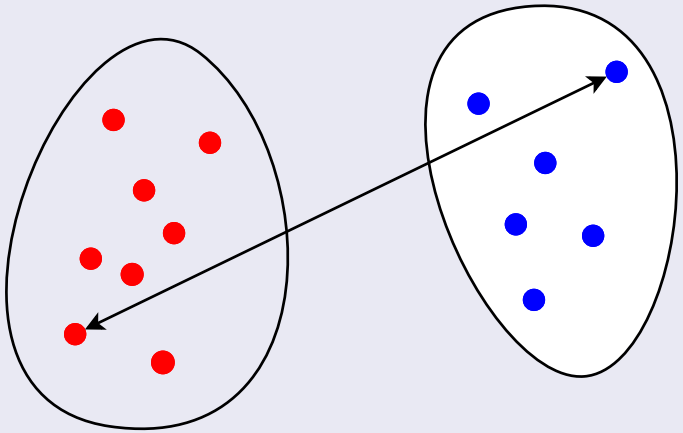
Definition 10 (Linkage function)

For Complete Linkage: $D(X, Y) = \max_{x \in X, y \in Y} d(x, y)$.
For Single Linkage: $D(X, Y) = \min_{x \in X, y \in Y} d(x, y)$.
For Average Linkage: $D(X, Y) = \frac{1}{|X||Y|} \sum_{x \in X, y \in Y} d(x, y)$. This is also known as Unweighted Pair Group Method with Arithmetic Mean (UPGMA) linkage.
For Ward linkage: the initial (single-point) cluster distances are simply the Euclidean distances between the points. The clusters are merged based on a minimum variance criterion. The distance between any point and a merged cluster is calculated using a recursive formula of Lance-Williams type.

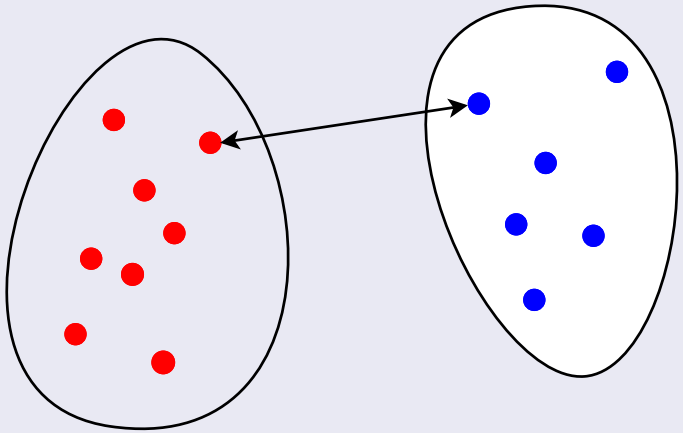
Generally, Complete Linkage and Ward's minimum variance linkage give the most balanced and useful clusters.

Distance between clusters: linkage visualisation

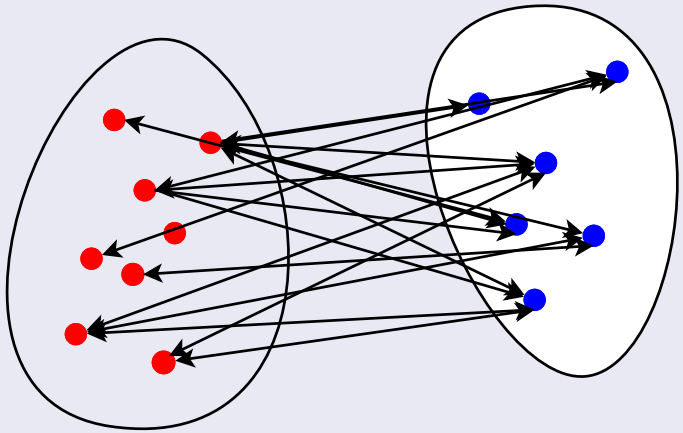
Complete Linkage



Single Linkage



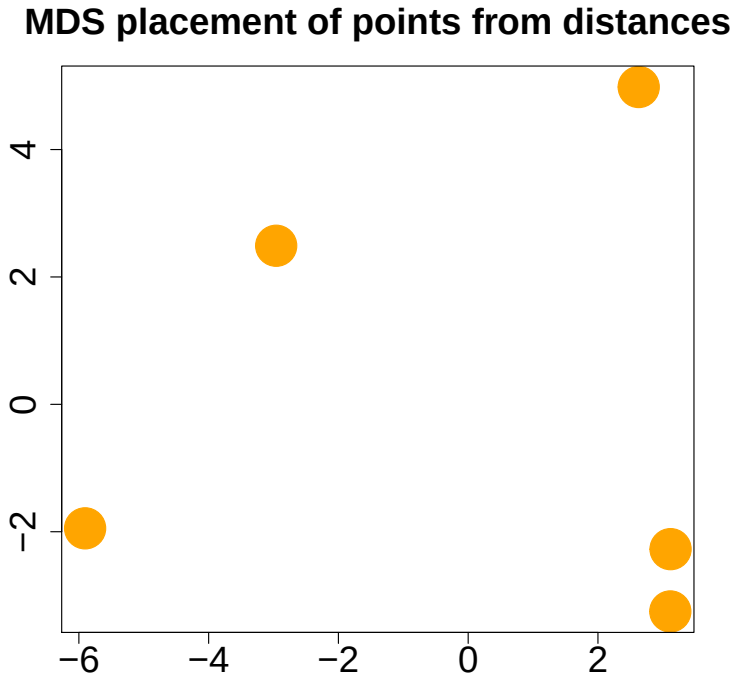
Average Linkage



AGNES worked example: setting the scene

Distance Matrix, Step 0

	A	B	C	D	E
A	0				
B	9	0			
C	3	7	0		
D	6	5	9	0	
E	11	10	2	8	0



Use of distance matrices

Many algorithms in data mining either start from a distance matrix representation, or need to create it themselves. Reversing the process(from distances to locations) is not unique, but MultiDimensional Scaling often gives an attractive placement (as above, centred on origin).

AGNES worked example: Initial iterations

First clustering: CE, A, B, D

The smallest distance is 2 between C-E. We cluster these points and compute the distance of the remaining points from the CE cluster. Because of single linkage, we store the minimum such distance in the revised table beside.

Distance Matrix, Step 1

	CE	A	B	D
CE	0			
A	3	0		
B	7	9	0	
D	8	6	5	0

Second clustering: ACE, B, D

The smallest distance is 3 between A and CE. We cluster these points and compute the distance of the remaining points from the ACE cluster. For example d(B,A) = 9, d(B,C) = 7, d(B,E) = 10, so by single linkage d(B,ACE) = 7 as in the revised table beside.

Distance Matrix, Step 2

	ACE	B	D
ACE	0		
B	7	0	
D	6	5	0

Minimum distances so far: 2,3

AGNES worked example: Final iterations

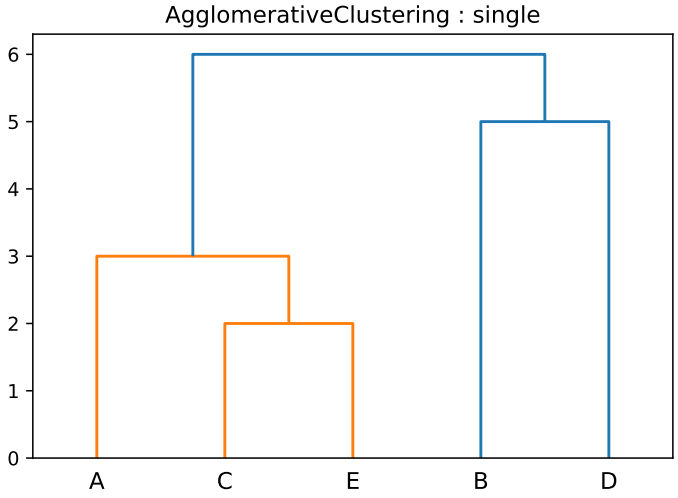
Third clustering: CE, A, B, D

The smallest distance is 5, between B and D, so we create a BD cluster. The new distance matrix is shown alongside. The next step after this would be to merge ACE with BD, creating a single ABCDE cluster. The algorithm ends...

Distance Matrix, Step 3

	ACE	BD
ACE	0	
BD	6	0

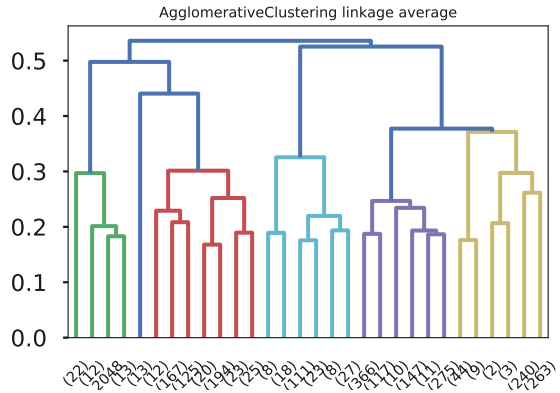
Minimum distances so far: 2,3,5,(6)

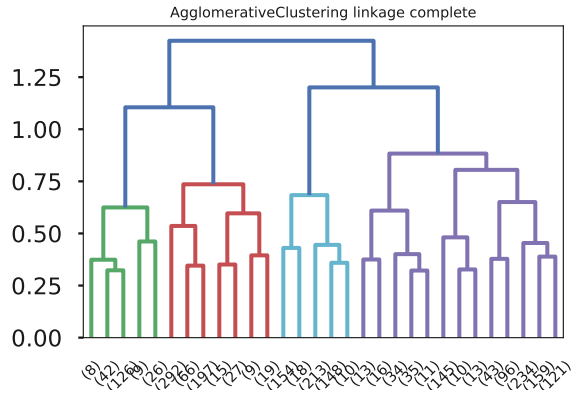


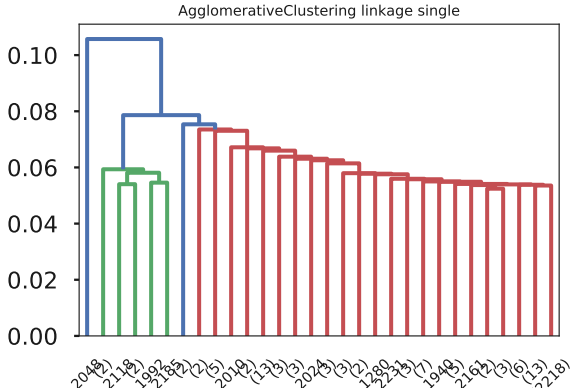
Resulting Dendrogram

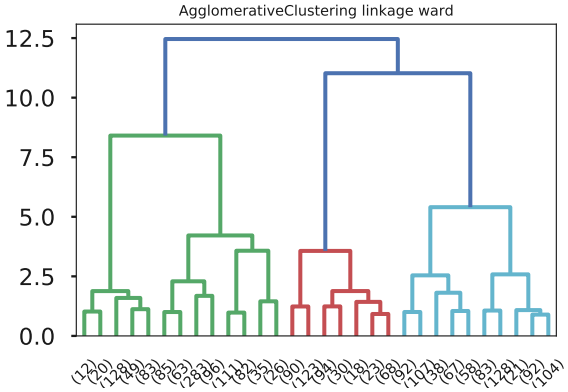
The resulting dendrogram summarises the hierarchical clustering. Note that the joins/splits occur at distances 2, 3, 5 and 6, as noted above.

Comparison of Dendrograms









As can be seen, the choice of linkage function has a dramatic effect on cluster membership.
The underlying data in this instance appeared to have 6 clusters. Can you see this in these dendrograms?

Uses of hierarchical clustering

- Hierarchical clustering can be very helpful for looking at data at a variety of scales, and hence for seeing hierarchical structure in a data set. This can lead to insights that other techniques, which focus on finding a single cluster mapping, cannot offer.
- Hierarchical clustering offers a rich variety of objective functions (primarily relating to linkage), some of which might suit a specific scenario.
- It can be used as a means of estimating parameters for other, perhaps more focused techniques, e.g., to estimate the number of clusters/components in the data.
- Since hierarchical clustering provides more than one candidate clustering, it can be require more system resources (computation and memory) than other techniques. Thus it might not scale very well.
- We have seen agglomerative (bottom-up) clustering. Divisive (top-down) clustering (DIANA) is also available, but has worse scalability.

Clustering as a partitioning problem

- Often the purpose of clustering is to assign one or more labels to each observation, so that "similar" observations are given the same cluster membership label.
- In the standard case, each observation is assigned a single label, and clustering defines a (hard) *partitioning* of the data. Lloyd's *k-means* algorithm does this.
- If each observation is assigned a membership probability for each cluster, this is a *soft partitioning* of the data. A hard partition can be derived by choosing, for each observation, the cluster for which it has the highest probability of membership. *Gaussian Mixture* models can be used for this purpose.
- Some clustering algorithms, notably *density-based clustering*, do not always assign a label to each observation. However, if an observation is assigned a label, it will be just one such label.

Definition 11 (Representation-based clustering)

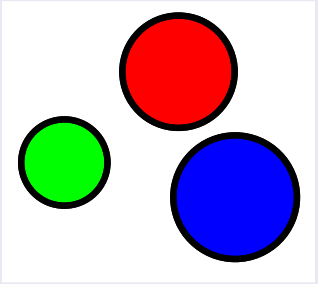
. . . finds a region around a *cluster centre* so that observations can be assigned to the cluster if they are found in that region.

Definition 12 (Density-based clustering)

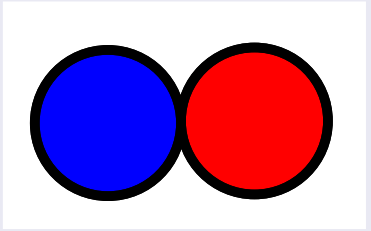
. . . looks for regions, possibly non-convex, where the data density is higher, and assigns observations in those regions to the relevant cluster. Any other observations are assumed to be either "noise" or "border" observations.

Types of partitional clustering

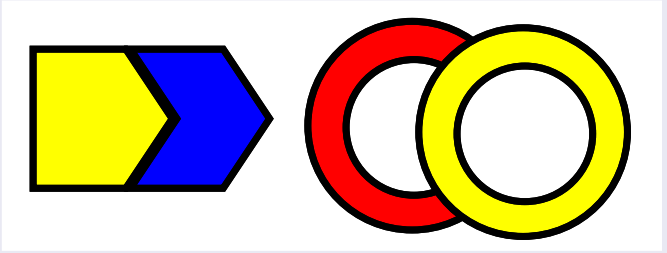
Well-separated clusters



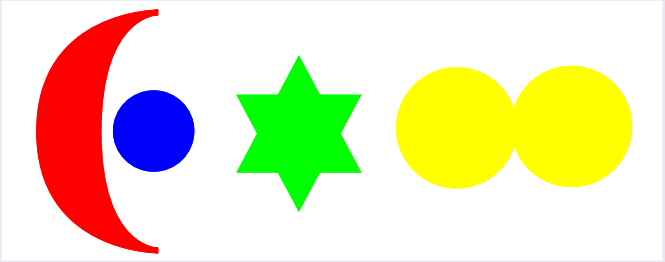
Centre-based clusters



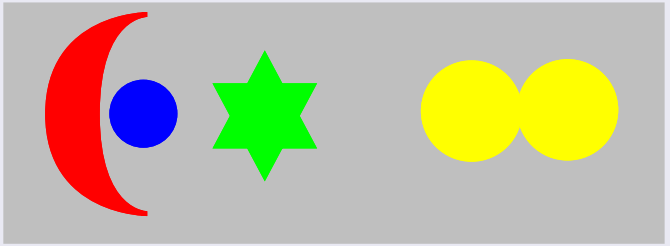
Conceptual clusters



Contiguity-based clusters



Density-based clusters



K-means algorithm: Overview

- The $k-$means algorithm assigns each observation to one of k clusters, by finding the nearest cluster centre for that observation (E-step).
- Each cluster centre is calculated as the centroid of the observations assigned to that cluster (M-step).
- The algorithms proceeds in two steps (E-M). At each iteration, the algorithm finds the nearest centre for each observation, then assigns it to that centre and recomputes the centres.
- Lloyd's algorithm is an example EM-algorithm: Expectation-Maximisation (general algorithm, used in many scenarios, especially clustering).
- Variants include

  Mini-batch k-means : work with a random sample of the data at each iteration: scales better, small loss of accuracy

  kmeans++ : Choose initial centres that are well-separated from each other; "normal" k-means afterwards.

  k-medoids : Manhattan (ℓ_1) distance is used instead of Euclidean (ℓ_2), and centres are constrained to be data points); PAM and CLARA algorithms.
- Generally Lloyd's algorithm is robust, although it is affected by the choice of initial centres, and care must be taken to avoid empty clusters

K-means algorithm: Detail

Method (k-means algorithm)

$t \leftarrow 0$;
Initialise centres $\{\boldsymbol{\mu}_j^t, j = 1, \ldots, k\}$: choose k points randomly, without replacement;
repeat
  $t \leftarrow t + 1$;
  $C_j \leftarrow \emptyset, \forall j = 1, \ldots, k$; ▷ Cluster Assignment Step **E**
  for all $\boldsymbol{x}$ **do** ▷ Assign $\boldsymbol{x}_j$ to the nearest centroid from the previous iteration
    $j^* \leftarrow \arg\min_i \left\{ \|\boldsymbol{x}_j - \boldsymbol{\mu}_i^{t-1}\|^2 \right\}$;
    $C_{j^*} \leftarrow C_{j^*} \cup \{\boldsymbol{x}_j\}$;
  end for ▷ Centroid Update Step **M**
  for all $i = 1$ to k **do**
    $\boldsymbol{\mu}_i^t \leftarrow \frac{1}{|C_i|} \sum_{\boldsymbol{x}_j \in C_i} \boldsymbol{x}_j$;
  end for
until $\sum_{i=1}^{k} \|\boldsymbol{\mu}_i^t - \boldsymbol{\mu}_i^{t-1}\|^2 \leq \epsilon$

The termination condition is that the difference in centre positions should not exceed a small tolerance ϵ.
This happens when points stay in their cluster from iteration p to $p + 1$, so cluster centre stays same.

Clustering categorical data

- k-means uses centroids and Euclidean distance
- So it cannot be applied directly to categorical data...
- Options
  - Either encode categorical columns as integers - can now compute distances
  - Since this can dramatically increase the dimensionality, some dimensionality reduction might be needed
  - Or Use k-modes on the original data if *all* the data is categorical
  - Or Use k-prototypes on the original data if some data is categorical and some is numerical

"K-means" for categorical data: k-modes

k-means uses centroids and Euclidean distance; k-modes uses modes and Hamming distance

Strengths and weaknesses: mostly similar to k-means

- Guaranteed to converge (eventually): helped by good choice of k and initial modes
- Iterates to a local minimum: result quality depends on initial modes
- Distances are integers: need to choose between tied distances when assigning cases to clusters

Implementation

Installation: `conda install conda-forge::kmodes`

```
from kmodes.kmodes import KModes
import pandas as pd
import numpy as np
```

```
model=KModes(n_clusters=3, random_state=42, n_init=4)
```

```
fittedModel=model.fit(df)
print("Cluster centroids - archetypal student grades")
print(fittedModel.cluster_centroids_)
```

```
clusters = fittedModel.predict(df)
df["ClusterID"] = clusters
print("Allocation of students to clusters")
display(df)
```

"K-means" for numerical *and* categorical data: k-prototypes

Combine k-means (on numerical data) and k-modes (on categorical data) in one clustering algorithm.

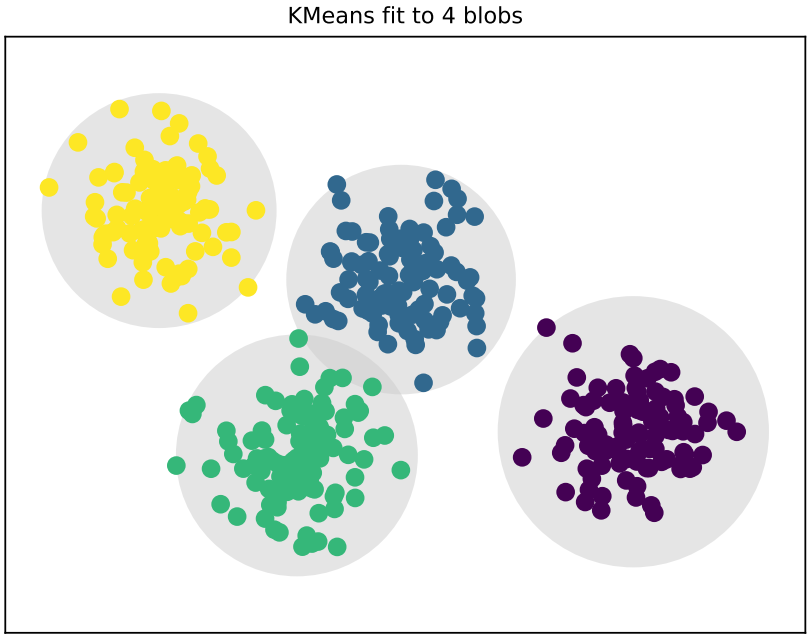
Intuition

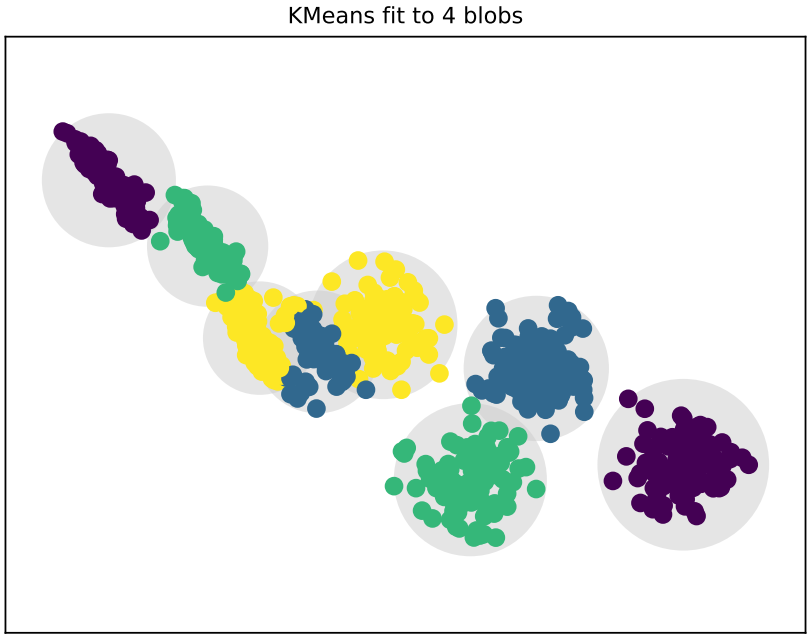
- A prototype instance has representative values of numerical and categorical features.
- Distance is a linear combination of the Euclidean (numerical) and Hamming (categorical) distances
- With this interpretation, the algorithm is much the same as k-means or k-modes, with similar strengths and weaknesses

Implementation

```
kRange = range(1,8)
allCols = numCols + allCatCols
catColIDs = list(range(len(numCols),len(numCols)+len(allCatCols)))
scores = dict()
for k in kRange:
  # Use Huang initialisation, use 5 random starting starting points, turn off logging
  model = KPrototypes(n_clusters=k, init='Huang', verbose=0, random_state=42, n_init=5)
  # Note that we need to tell the model which are the categorical columns
  fittedModel = model.fit(df[allCols], categorical=catColIDs)
  scores[k] = fittedModel.cost_
print(scores)
```

K-means algorithm: In practice





With the original globular clusters, *k*-means was able to find the centres and clusters easily.

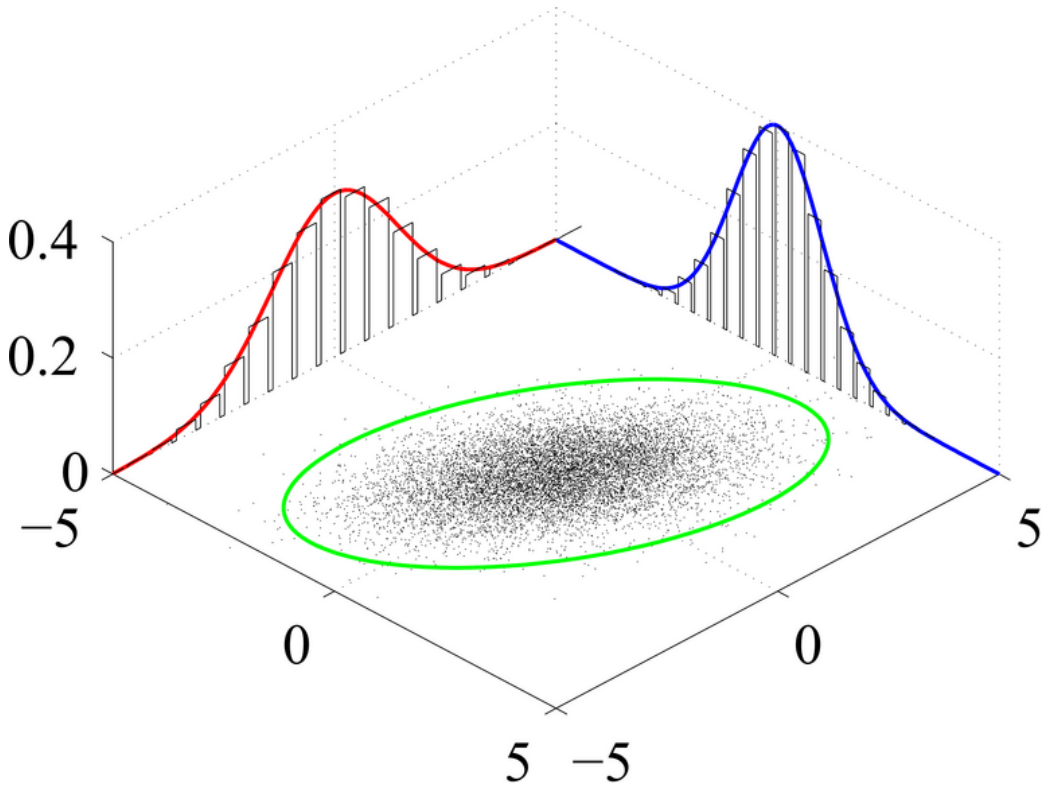
With the stretched clusters, *k*-means had more difficulty, e.g., with the yellow and purple clusters.

k-means minimises the within-cluster sum of squared distances (also known as *inertia*) so the choice of distance function is critical.

Probabilistic models for clustering

- k-means is an example of *hard* clustering, where each data point is mapped to a single cluster
- As such it is best suited to well-separated clusters - but what if they are close or even overlap?
- *fuzzy* clustering: points are assigned to multiple clusters, and are given a membership score in [0,1] for each cluster
- fuzzy c-means algorithm is a straightforward extension of k-means, just using probability $P(x_i, \mu_j)$ to weight each point x_i when calculating the centroid of each cluster μ_j (M-step)
- P is a function of the relative Euclidean distances to the cluster centres $\{\mu_j\}$
- This probability function can be generalised, notably to take account of the *shape* of the clusters and not just their centres, leading to *Gaussian Mixture Model* (GMM) probabilistic clustering

Review: Multivariate (2D) Gaussian/Normal distribution



Source: Wikipedia

- The distribution can have different dimensions that do not need to align with the coordinate axes; captured as a 2x2 covariance matrix C
- The distribution stretches to infinity in the plane, but points far from the centre of the distribution have very low probability.
- A collection of clusters can be modelled by overlaying a *mixture* of such Gaussian distributions on the plane.

Review of Bayes Theorem

Use of Bayes Theorem in Classification

Likelihood is the probability of the data given the label. Prior measures our belief about how likely each label is *before* we have seen any data. The Posterior includes influences of both the Prior and the Likelihood.

$$P(y = c|x) = \frac{P(x|y = c)P(y = c)}{P(x)}$$

The Posterior here is $P(y = c|x)$, the Likelihood is $P(x|y = c)$ and the Prior is $P(y = c)$. $P(x)$ is a normalizing constant that measures how likely the observed data x is.

When used for Gaussian Mixture Models, there is not just a *single* cluster label c, but a linear combination of many.

Overview of EM algorithm for GMM clustering

E-step For each x_i, calculate the probability that x_i belongs to the j^{th} distribution

$$P(\Theta_j|x_i,\Theta) = \frac{P(x_i|\Theta_j)}{\sum_{l=1}^{k} P(x_i|\Theta_l)},$$

where Θ_j is the set of parameters defining Gaussian distribution j, namely its centre $\boldsymbol{\mu}_j$ and covariance matrix C_j.

M-step Maximise the expected likelihood $P(\{x_i\}|\Theta)$ by updating the Gaussian mixture. That is, for each $\boldsymbol{\mu}_j$ and C_j, use all x_i and the $P(\Theta_j|x_i,\Theta)$ computed in the E-step) to derive the new Gaussian distribution parameters.

Note that the E-step computes a membership probability for each point based on all the Gaussian models and their parameters.
By contrast, the M-step computes the new Gaussian models based on all the points and their membership probabilities.
Lloyd's k-means algorithm is equivalent: the membership probability is either 1 (allocated to this cluster) or 0 (not allocated to this cluster) for each point. The M-step re-computes the cluster centres based on all the points and their cluster assignment.

GMM compared with k-means

	k-means	GMM
E-step	Compute membership probability which is either 1 (allocated to this cluster) or 0 (not allocated to this cluster) for each point	Compute membership probability for each point based on all the Gaussian models and their parameters.
M-step	Recompute the new cluster centres based on all the points and their cluster assignment	Recompute the new Gaussian models based on all the points and their membership probabilities
Use for	Well-separated	Centre-based or well-separated
Shape	nondirectional (“spherical”)	directional (“ellipsoidal”) or nondirectional

Relaxing the constraints: density-based clustering

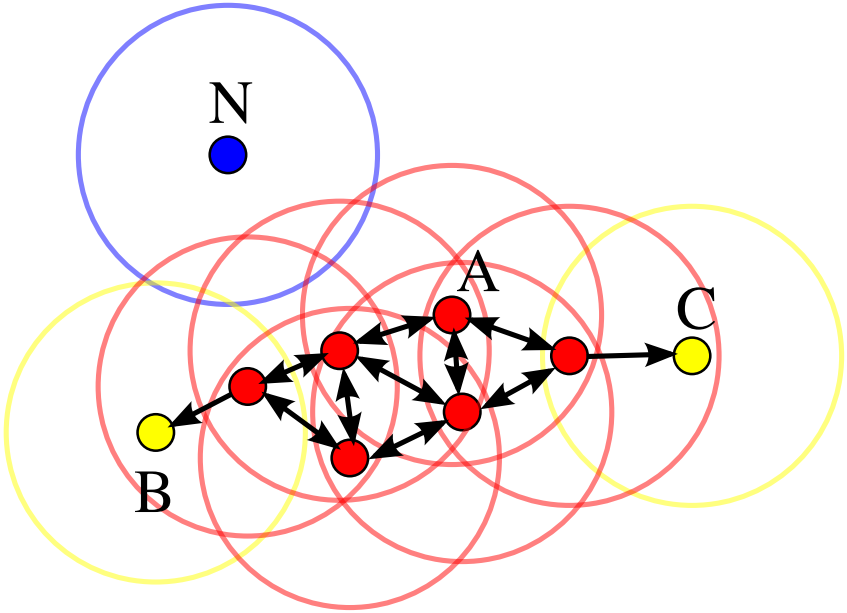
k-means and GMM are both characterised by the following properties:

- the number of clusters k must be specified beforehand
- clusters have a convex shape
- they work best when the clusters are linearly separable
- all points are assigned to clusters, so can be sensitive to outliers

Density-based clustering relaxes these conditions

It uses the heuristic that clusters are (arbitrarily-shaped) contiguous regions with high datapoint density.
Datapoints outside these regions represent *noise* and are ignored.
Rather than specifying k, the user specifies density thresholds.

Relaxing the constraints: density-based clustering



Source: wikipedia

- A, B and C are directly connected points.
- A is a core point
- B and C are border points.
- N is a noise point and so is not assigned to a cluster.
- The connected component of the 8 points (6 red, 2 yellow; including A,B,C) forms a cluster.

DBSCAN algorithm and its concepts

Definition 13 (DBSCAN)

Density-Based Spatial Clustering of Applications with Noise (DBSCAN): an algorithm for deriving clusters in areas of high data density.

Definition 14 (eps-neighbourhood)

Epsilon ϵ parameter defines a region of points $\boldsymbol{t}$ around a point $\boldsymbol{x}$ where $\|\boldsymbol{t} - \boldsymbol{x}\| < \epsilon$.

Definition 15 (core point)

Point with at least `MinPts`-1 other points in its eps-neighbourhood.

Definition 16 (border point)

Point with less than `MinPts`-1 other points in its eps-neighbourhood, but at least one is a core point.

Definition 17 (noise point)

Point with less than `MinPts`-1 other non-core points in its eps-neighbourhood.

Development of the algorithm

Definition 18 (Direct density reachable)

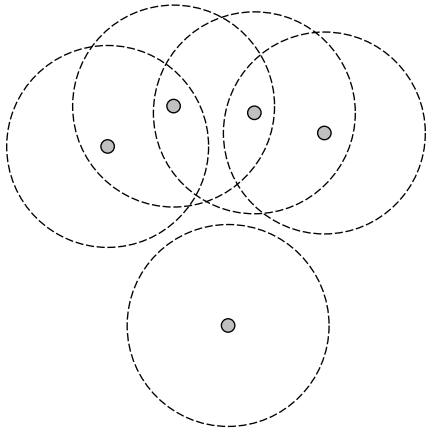
Point $\boldsymbol{x}_A$ is directly density reachable from $\boldsymbol{x}_B$ iff $\boldsymbol{x}_A$ is in the eps-neighborhood of $\boldsymbol{x}_B$ and $\boldsymbol{x}_B$ is a *core point*.

Definition 19 (Density reachable)

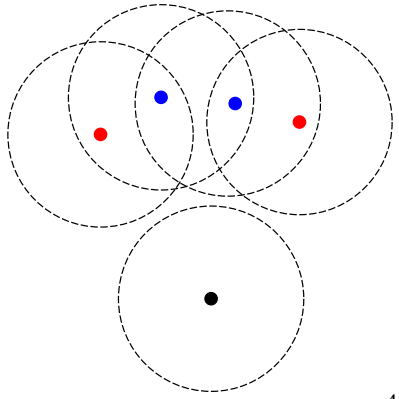
Point $\boldsymbol{x}_A$ is density reachable from $\boldsymbol{x}_B$ if there is a set of core points in each other's eps-neighbourhood between $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$.

Definition 20 (Density connected)

Points $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ are density connected if there exists a core point $\boldsymbol{x}_C$ so that both $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ are density reachable from $\boldsymbol{x}_C$.



Core, border and noise points are coloured blue, red and black below.



Steps of the DBSCAN algorithm

Method (DBSCAN)

1. Find the ϵ neighbors of every point.
2. Identify the core points with more than minPts neighbors.
3. Derive the *connected component* graphs of core points, assigning edges between core points that are less than ϵ apart.
4. Identify the border points and assign them to their nearest cluster.
5. Label any remaining points as *noise*.

- A variant (HDBCSAN) excludes border points from the cluster, treating them as noise points (can be more robust).
- Another vatiant (OPTICS) places the points in a priority queue, ordered by reachability distance (updating is slower, but handles varying density better).

Choosing k, the number of clusters

How can we decide on k for k-means and GMM?

- We can do this *graphically* (plot clusters for each k) or by using *scores*.
- Plot within-cluster sum of squared distances (inertia) against k and look for k at the "elbow".
- Use `kmeans.inertia_` as the score for a given instance of the kmeans classifier.
- Can also compute inertia for other partitional clustering techniques, but this is more work and interpretation is more difficult.

Silhouette scores - derivation

How much is any point in a cluster nearer its peers than it is to points in the nearest of the other clusters?

Method (Silhouette score)

Require: Clustering where the i point is assigned to cluster $C(i)$ and there are k such clusters
for all point i in cluster $C(i)$ **do**
Calculate $a(i)$, the mean distance between i and all the other points in $C(i)$. $\triangleright$ $a(i) \equiv 0$ is there is no other point in $C(i)$.
Calculate $b(i)$, minimum of the mean distances between i and all the other points in each of $C(j)$ where $j \neq i$.
Silhouette $s(i) = 1 - a(i)/b(i)$ if $a(i) < b(i)$, $s(i) = 0$ if $a(i) = b(i)$ and $s(i) = b(i)/s(i) - 1$ if $a(i) > b(i)$.
end for
The mean of $s(i)$ over all points ($\bar{s}_k$) is a measure of the clustering efficiency for that value of k.
The k associated with the *maximum* of these $\bar{s}_k$ silhouette scores is the best choice of k.

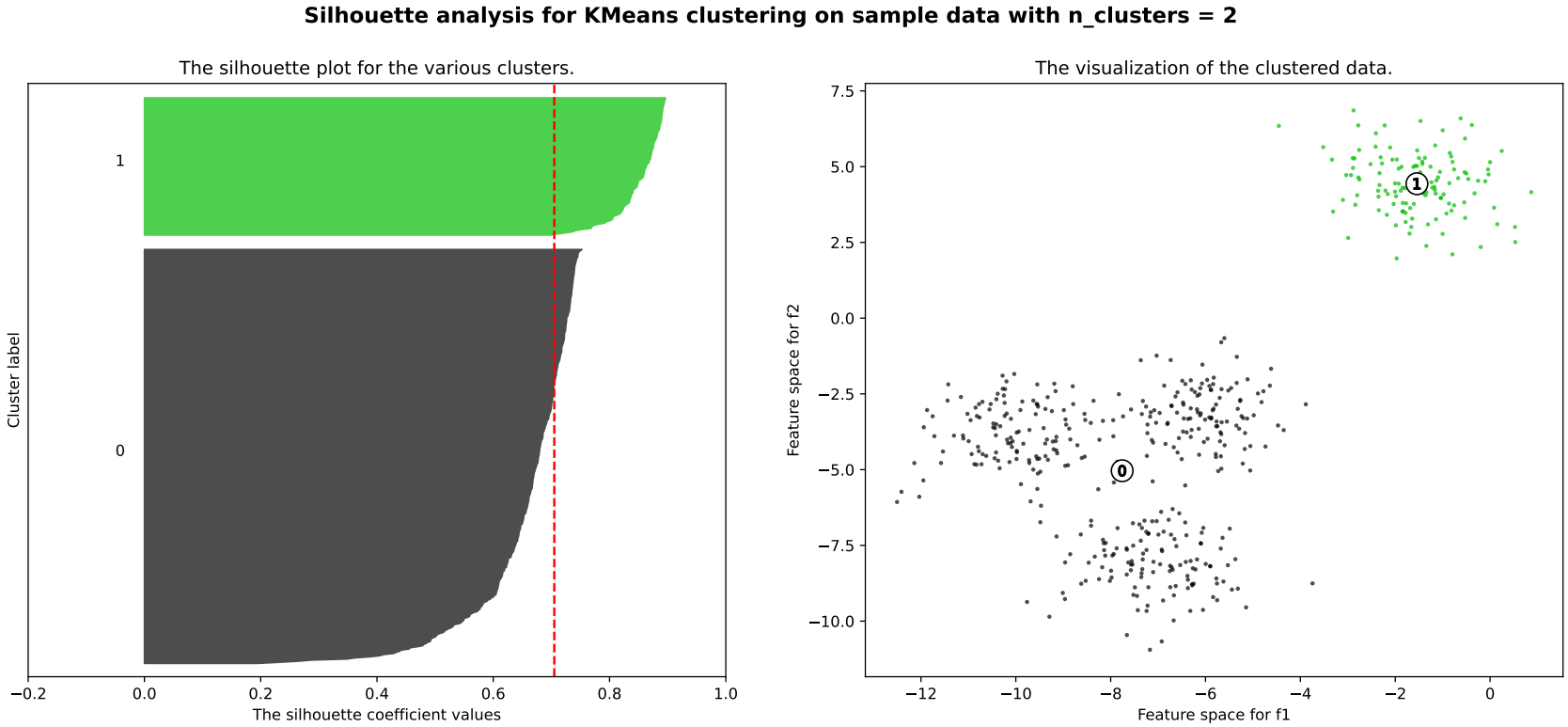
There are many other scores but they require more advanced mathematics and are out of scope for this module.

Silhouette scores - examples

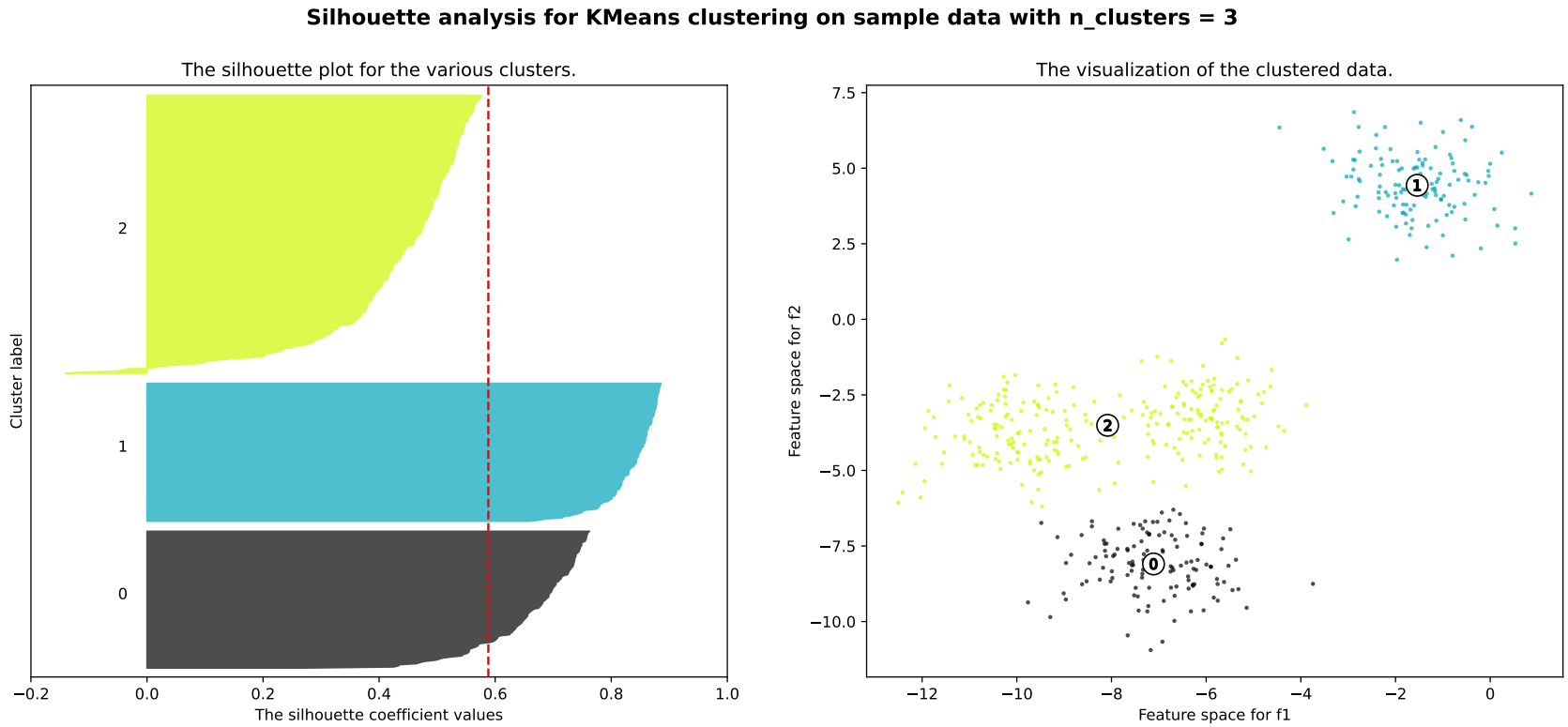
Code to compute the silhouette score

```
from sklearn.metrics import silhouette_samples, silhouette_score

clusterer = KMeans(n_clusters=n_clusters, random_state=10)
cluster_labels = clusterer.fit_predict(X)

# The silhouette_score gives the average value for all the samples.
silhouette_avg = silhouette_score(X, cluster_labels)
```

Silhouette score with $k = 2$ - looking good

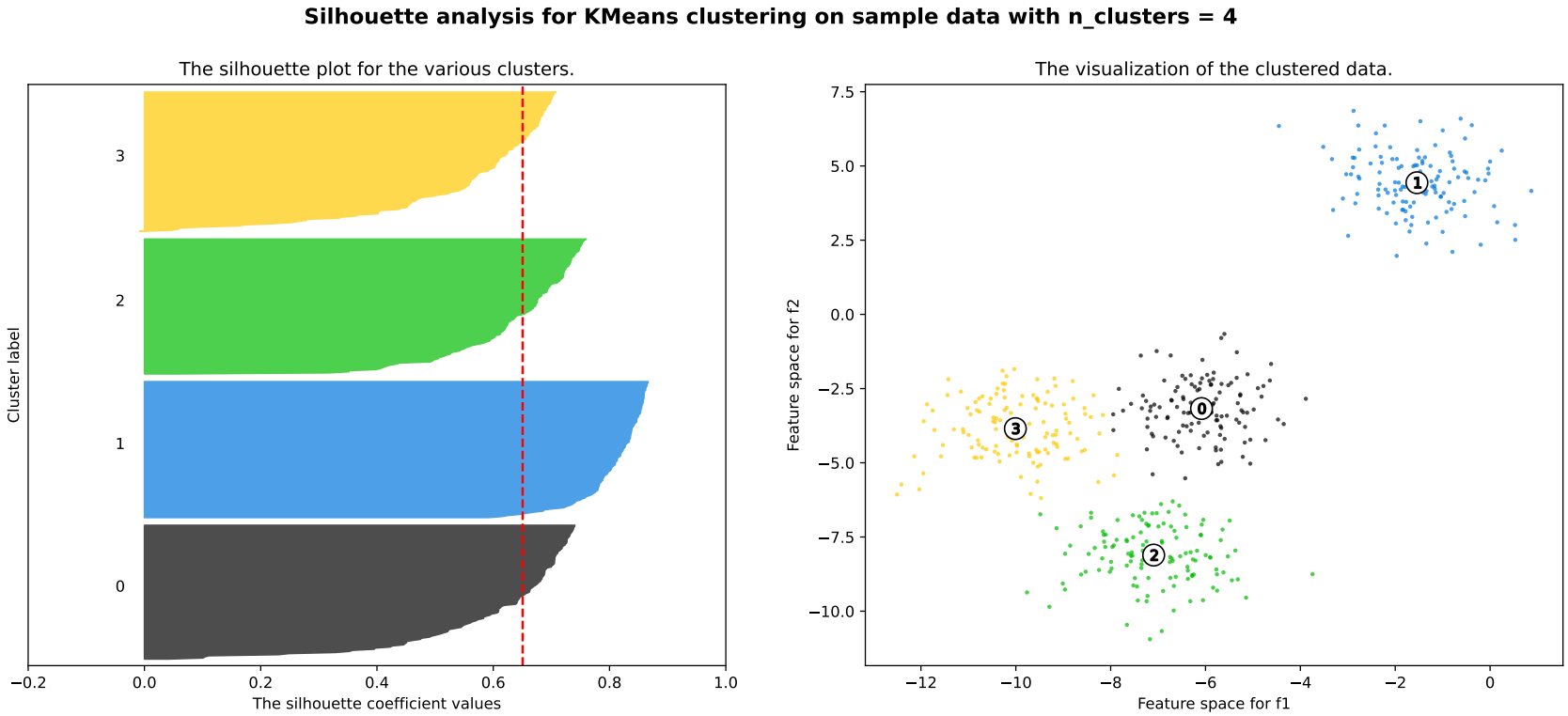


Silhouette score with $k = 3$ - not looking good



Cluster 0's silhouette profile has 2 problems

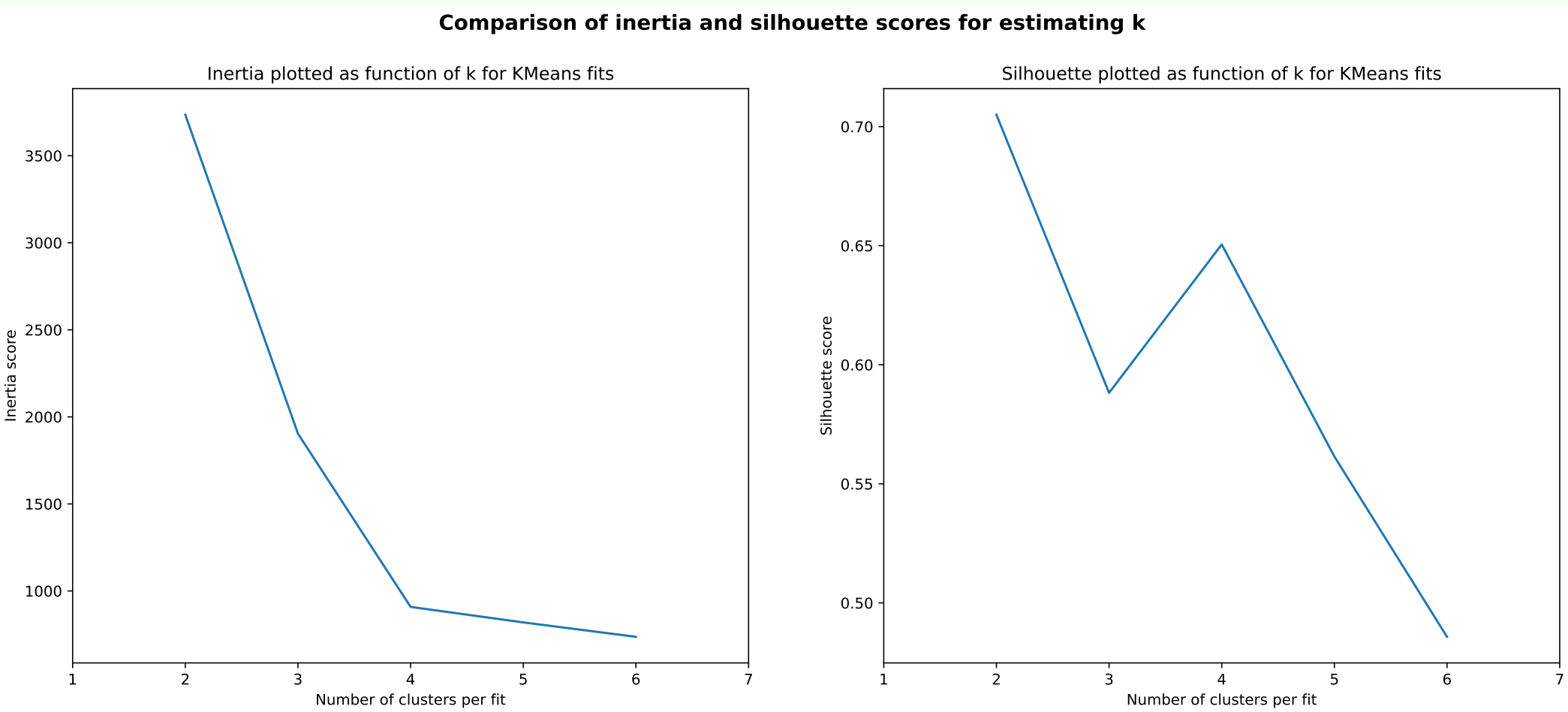
Silhouette score with $k = 4$ - looking good again



Is it better or worse than when $k = 2$?

Comparing both scoring systems

Inertia/elbow plot and silhouette plot on the same data



Clusters for visualising data

After allocating instances to clusters, how can we visualise these clusters?

- If the clusters have 3 dimensions or less, we can use a distinctive marker for each cluster point and plot as normal.
- For example, points in cluster 0 are represented by blue circles, points in cluster 1 appear as red squares, . . .
- But what if there are more than 3 dimensions?
  - Take 2D or 3D slices (pick 2 columns or 3 columns at a time) and plot them as above.
  - But there is no reason that clusters should be well-separated along any of the axis directions!
  - You could use a linear dimensionality reduction technique (say, PCA) to reduce the dimensions to 2 or 3. . .
  - Or you could use a nonlinear reduction technique that squeezes some directions more than others

Nonlinear dimensionality reduction requires careful interpretation, but it can be very useful.

Unlike (linear) PCA, a mapping cannot be learnt during training and applied as is to the test set.

Nonlinear dimensionality reduction techniques

Definition 21 (Multidimensional Scaling (MDS))

Given pairwise distances between observations (conveniently presented as a (symmetric) distance matrix), MDS constucts a representation of that data that preserves these distances as much as possible in the lower dimensionsal space (say 2D (n=2) or 3D ($n = 3$), where n is a hyperparameter). This is done by minimising a loss function; several functional forms are available. Since it focuses only on distances between points, and ignores "distances" between clusters, it can be a poor choice for visualising clusters. It also does not scale very well to really large datasets.

Definition 22 (t-distributed Stochastic Network Embedding (t-SNE))

t-SNE is a nonlinear procedure that takes high-dimensional data and maps it into lower dimensions, while trying to maintain local similarities that are expressed as conditional probabilities. Unlike MDS, the loss function considers points that are further away (but with less "weight"). So it can do a better job than MDS of maintaining the cluster's cohesiveness. However, it has difficulties scaling to larger datasets and is sensitive to many hyperparameters like perplexity.

Nonlinear dimensionality reduction techniques - continued

Definition 23 (Uniform Manifold Approximation and Projection (UMAP))

UMAP is a nonlinear procedure that maps the data into the closest lower dimensional smooth surface (a manifold). Notionally, this surface can be "flattened out" to give the desired representation. UMAP has fewer hyperparameters than t-SNE and tries to balance how it treats local similarity versus global similarity. It also tends to scale better as the dataset increases. However, like any algorithm based on nonlinear optimisation, good results are not guaranteed.

- Finding a "good" visualisation is often a matter of trial and error.
- PCA, MDS and t-SNE are available in `scikit-learn` with standard methods like `.fit()`, etc.
- UMAP can be added using `conda install conda-forge::umap-learn` with a similar API to the others

Choice of hyperparaneters

AGNES : choose distance function and linkage. Usually Ward or single linkage work best.

k-means, etc. : choose distance function, starting condition (cf kmeans++), aggregation (cf k-medoids), k

GMM : choose distance function, k

DBSCAN : choose distance function, `minpts`, `eps`

Tips

- Good idea to scale so that clusters are approximately (hyper)spherical.
- Can be good idea to transform data before clustering.
- Dendrogram is good for visualising structure when data is more than 3-D.

Summary

- Clustering is perhaps the best known form of unsupervised learning
- Hierarchical clustering can provide insights into the structure of a data set - very useful when exploring data for other techniques
- Partitional clustering can be used to label points according to which cluster they belong to
- Partitional classification has many approaches: centre-based and density based are most common
- Clustering can be used to help create training data for classification purposes (c.f., the digits notebook used in the practical)