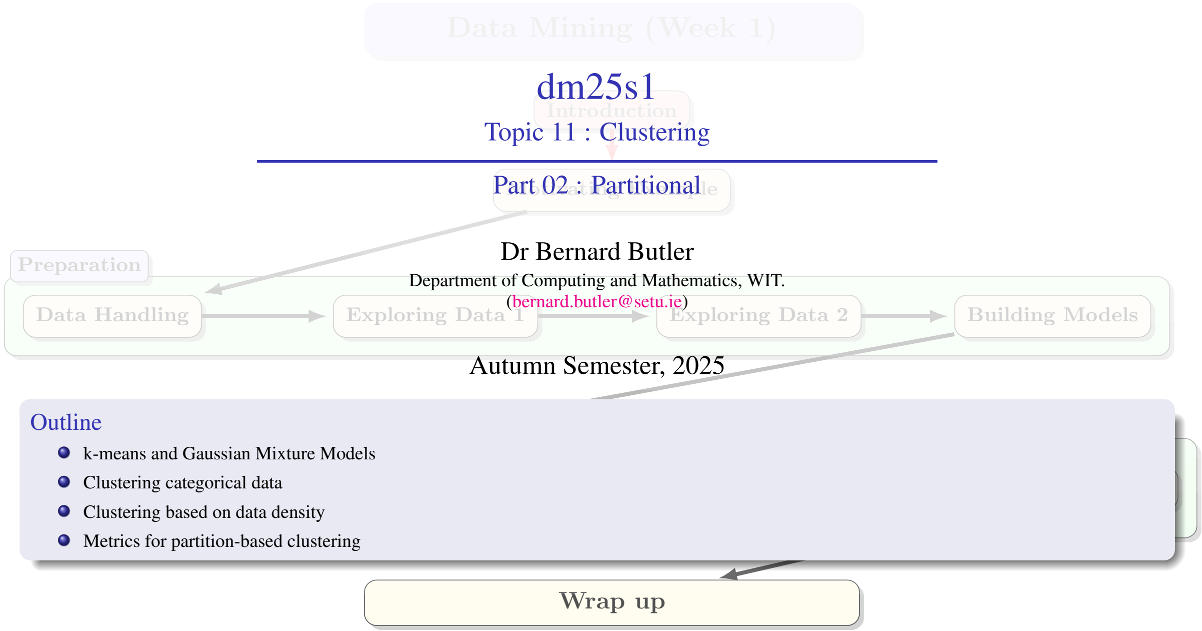


dm25s1

Topic 11 : Clustering

Part 02 : Partitional

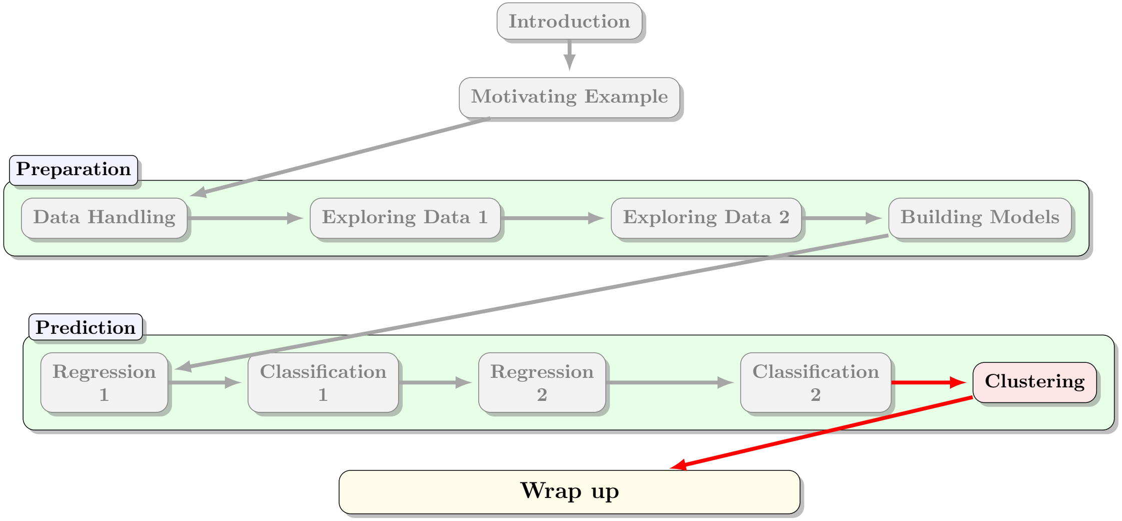
Dr Bernard Butler
Department of Computing and Mathematics, WIT.
(bernard.butler@setu.ie)

Autumn Semester, 2025

Outline

- k-means and Gaussian Mixture Models
- Clustering categorical data
- Clustering based on data density
- Metrics for partition-based clustering

Data Mining (Week 11)



Outline

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.

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.

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.

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.

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 1 (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.

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 1 (Representation-based clustering)

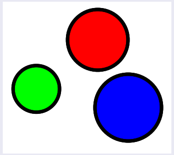
…finds a region around a *cluster centre* so that observations can be assigned to the cluster if they are found in that region.

Definition 2 (Density-based clustering)

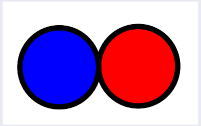
…looks for regions, possibly non-convex, where the data density is higher, and assigns observations in those regions to the relevant cluster. Any other observations are assumed to be either "noise" or "border" observations.

Types of partitional clustering

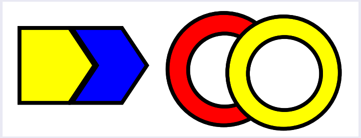
Well-separated clusters



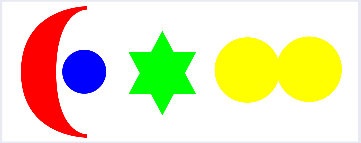
Centre-based clusters



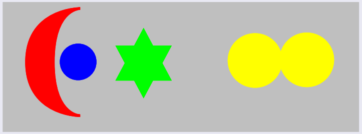
Conceptual clusters



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).

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).

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.

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).

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

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

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-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.

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

Clustering categorical data

- k-means uses centroids and Euclidean distance
- So it cannot be applied directly to categorical data. . .

Clustering categorical data

- k-means uses centroids and Euclidean distance
- So it cannot be applied directly to categorical data. . .
- Options

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

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, dimensionality reduction (e.g., PCA) might be needed

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, dimensionality reduction (e.g., PCA) might be needed
  - Or Use k-modes on the original data if *all* the data is categorical

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, dimensionality reduction (e.g., PCA) 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

"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

“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

“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 - Setup

Installation:

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

Imports:

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

Since k-modes, like most EM algorithms, starts from a random cluster assignment and iterates to improve it, a poor initial choice can result in sub-optimal results. Here, we ask it to start from 4 cluster assignment choices, to iterate to completion for each, and to pick the best overall cluster assignment.

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

k-modes in practice

Deriving cluster centroids

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

```
Cluster centroids - archetypal student grades
[['A' 'A' 'B' 'B' 'A']
 ['A' 'C' 'B' 'A' 'C']
 ['A' 'B' 'A' 'C' 'B']]
```

- Note these centroids indicate a “typical student” in that cluster, but this does not need to match any of the students that were used when learning the cluster assignment.
- That is, the individual grades are the modes of the grades in that cluster, but that does not mean that a student in the cluster has that combination of grades.

k-modes assigns each student to a (0,1,2) cluster

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

```
Allocation of students to clusters
                   English Maths History Geography Science ClusterID
Student
Beryl Smart             A     B       A         B       A         0
Sydney Whitworth        C     C       B         A       C         1
Nora Waite              C     A       B         B       A         0
Carrie Aldridge         B     A       A         B       C         0
Ravinder Townsend       A     B       B         A       C         1
Antonio Hunter          B     A       C         C       C         0
Clive Sheldon           A     A       A         A       A         0
Lynette England         A     C       B         B       B         0
Hilary Farrow           A     B       B         A       A         0
Cristina Rogers         C     C       D         B       A         0
Nana Gilbert            A     C       B         B       C         1
Miriam Moore            A     B       A         C       B         2
Tara Lomas              B     C       C         D       B         1
Sam Humphrey            A     B       B         B       B         0
Olga Pickles            A     A       B         B       A         0
```

"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.

"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

"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
- k-prototypes is similar to k-means or k-modes, with similar strengths and weaknesses

“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.

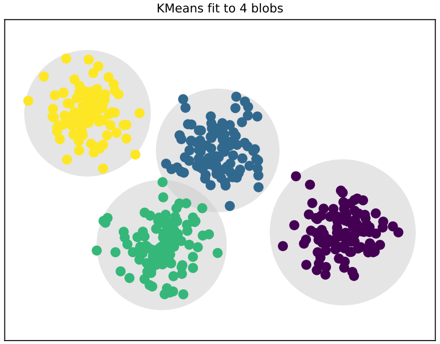
Intuition

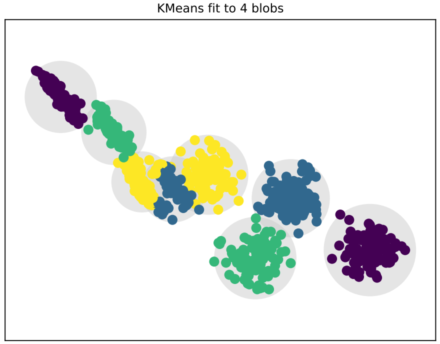
- A prototype instance has representative values of numerical and categorical features.
- Distance is a linear combination of the Euclidean (numerical) and Hamming (categorical) distances
- k-prototypes is similar to k-means or k-modes, with similar strengths and weaknesses

Implementation

```python
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

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?

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

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)

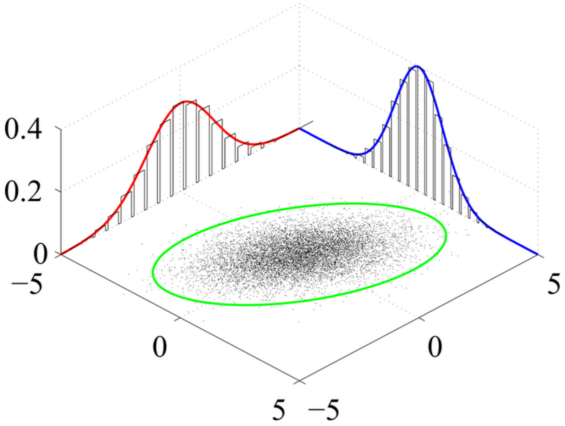
Probabilistic models for clustering

- 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 $\{\mu_j\}$

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



- 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.

Source: Wikipedia

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.

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.

Note that the E-step computes a membership probability for each point based on all the Gaussian models and their parameters.

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.

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.

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

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

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

Relaxing the constraints: density-based clustering

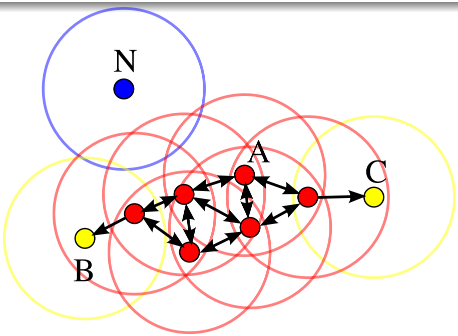
k-means and GMM are both characterised by the following properties:

- 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 3 (DBSCAN)

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

Definition 4 (eps-neighbourhood)

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

Definition 5 (core point)

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

Definition 6 (border point)

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

Definition 7 (noise point)

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

Development of the algorithm

Definition 8 (Direct density reachable)

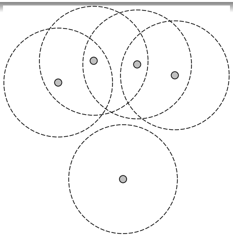
Point $\boldsymbol{x}_A$ is directly density reachable from $\boldsymbol{x}_B$ iff $\boldsymbol{x}_A$ is in the eps-neighborhood of $\boldsymbol{x}_B$ and $\boldsymbol{x}_B$ is a *core point*.

Definition 9 (Density reachable)

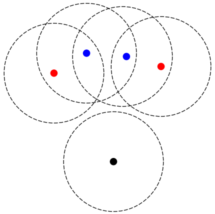
Point $\boldsymbol{x}_A$ is density reachable from $\boldsymbol{x}_B$ if there is a set of core points in each other's eps-neighbourhood between $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$.

Definition 10 (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 variant (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*.

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".

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.

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 $C(i)$.  ▷ $a(i) \equiv 0$ 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.

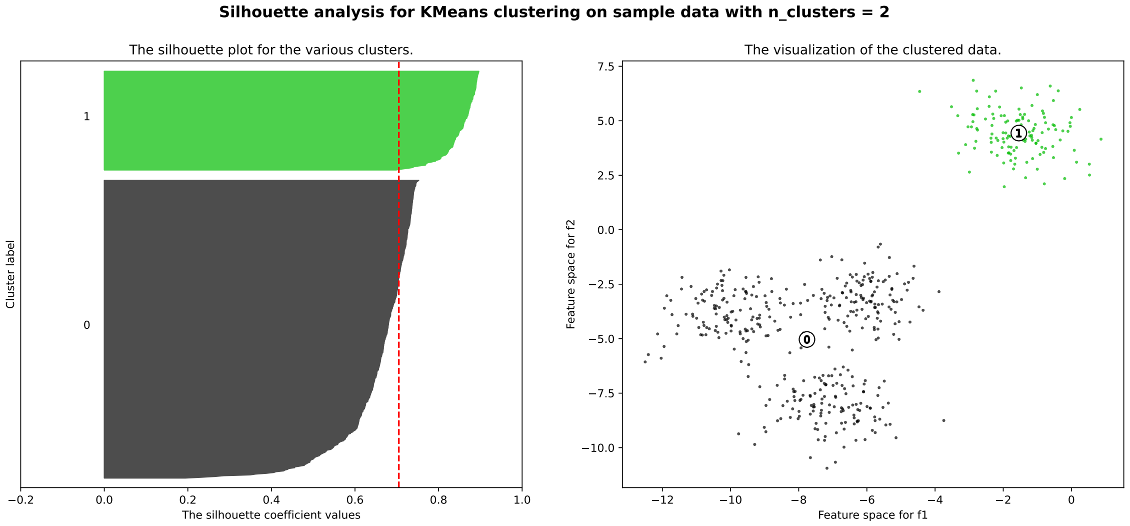
Silhouette scores - examples

Code to compute the silhouette score

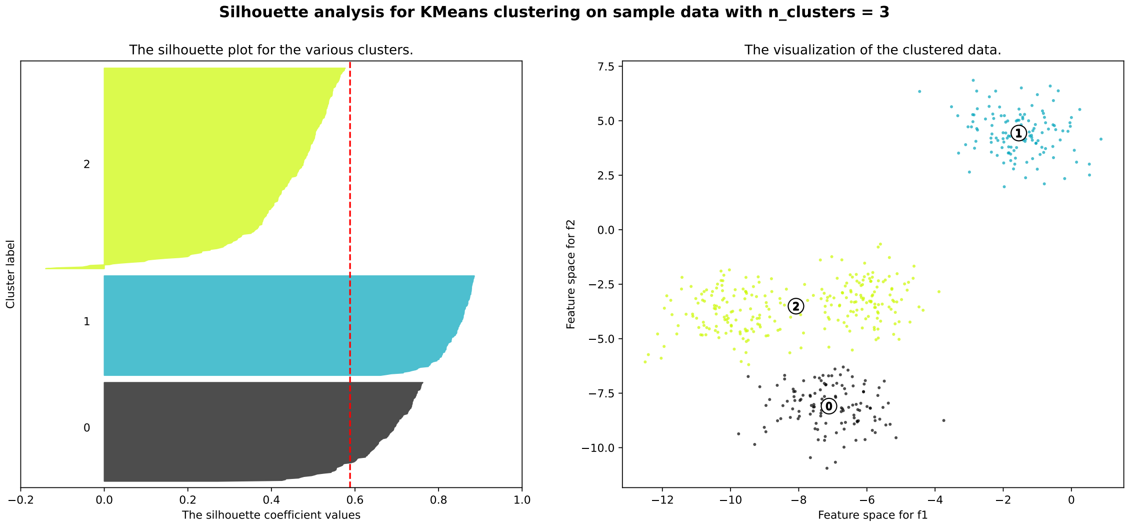
```python
from sklearn.metrics import silhouette_samples, silhouette_score

k = 4
clusterer = KMeans(n_clusters=k, random_state=10)
cluster_labels = clusterer.fit_predict(X)

# silhouette_score is the average silhouette for all the samples
silhouette_avg = silhouette_score(X, cluster_labels)
print(silhouette_avg)
```

```
0.6505186632729437
```

Silhouette score with $k = 2$ - looking good

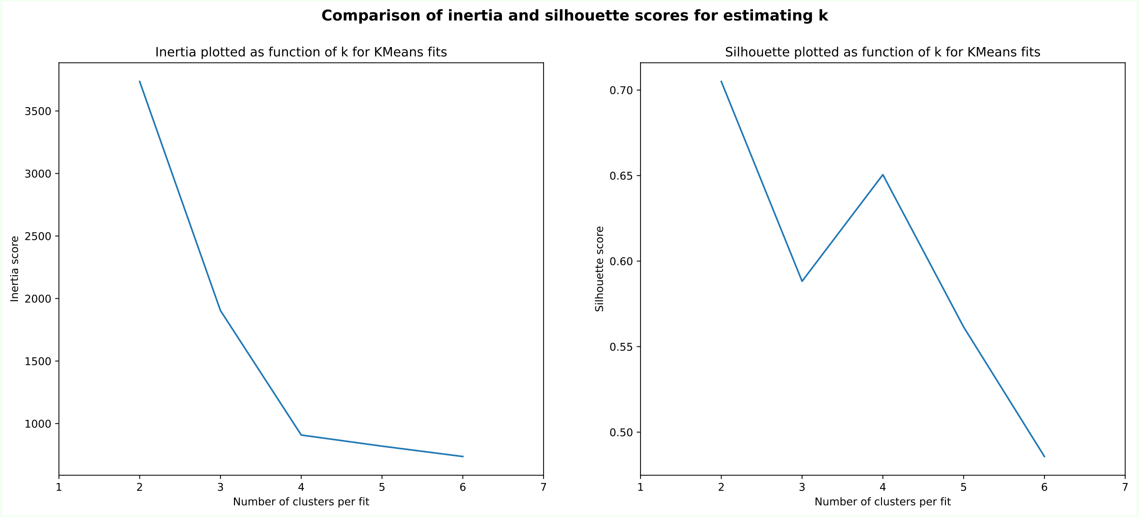


Silhouette score with $k = 3$ - not looking good



Comparing both scoring systems

Inertia/elbow plot and silhouette plot on the same data



Outline

Choice of hyperparameters

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

Choice of hyperparameters

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

Choice of hyperparameters

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

Choice of hyperparameters

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`

Choice of hyperparameters

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.

Choice of hyperparameters

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.

Choice of hyperparameters

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)