School of Electronic Engineering and Computer Science
Queen Mary University of London

ECS7020P Principles of Machine Learning
Unsupervised learning: Structure analysis

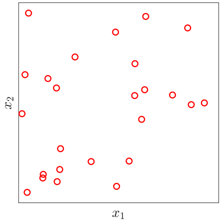
Dr Jesús Requena Carrión

23 Nov 2021

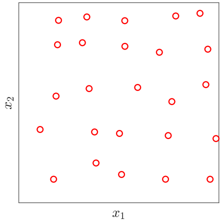
Patterns and structure

Patterns are regularities in our data. **Structure** is a regularity in our target population.

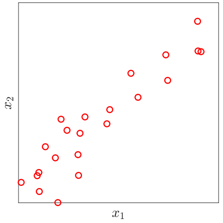
Each one of the three random patterns below comes from a different population. Which one suggests the **least underlying structure**?



(a)



(b)



(c)

Fooled by randomness

Any machine learning project relies on discovering some underlying **structure** by extracting **patterns** in data.

The main challenge is to distinguish **relevant** from **spurious patterns**, or **essence** from **irrelevant details**.

The **hot hand fallacy** illustrates this point: witnessing a player sinking several baskets in a row can lead us to conclude that they are on a roll.

In fact, **overfitting** is nothing but a machine memorising irrelevant details due to chance, rather than extracting the true underlying pattern.

Don’t let randomness fool you!

Dataset dimensionality

Attributes can be interpreted as **dimensions** in a space. Datasets that include many attributes are hence said to be **high dimensional**.

Examples of high dimensional scenarios include:

- **Complex** data types, such as grid data (pictures, audio files).
- Situations where we have **little prior knowledge** and we record everything, just in case.

Collecting as many attributes as possible seems like a good idea. Is it? Not necessarily: it **gives Randomness more opportunities to fool you**.

Note: This is what I call Gullible Big Data.

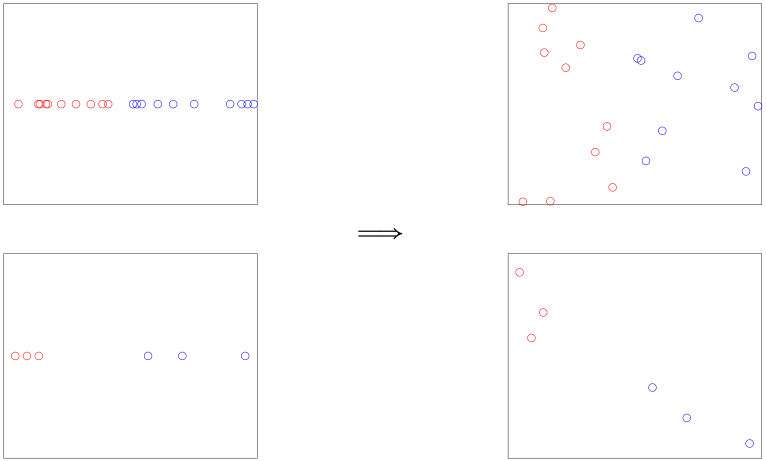
Short and wide datasets

Using the dataset below, train a model that predicts the salary of an individual based on their age, height, weight, gender and postcode.

ID	Age	Height [m]	Weight [kg]	Gender	Postcode	Salary
0	35	1.7	75	M	EC2Y 8DS	100
1	40	1.68	70	F	E14 0QR	200
2	30	1.88	85	M	WC2H 8LH	150

The Curse of Dimensionality

As the dimensionality increases, **data becomes sparser**. What happens if we add irrelevant attributes? What if we have fewer samples?



The Curse of Dimensionality

Overfitting occurs when our models interpret spurious patterns as true structure.

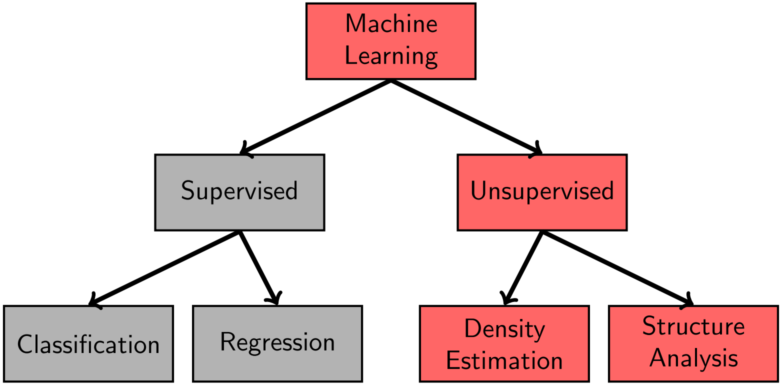
High-dimensional settings are fertile ground for **overfitting**, specially when attributes are weakly relevant or many are irrelevant.

The **curse of dimensionality** is a warning. Irrelevant attributes **do not cancel each other out**, they show up as spurious patterns. Adding more attributes (*just in case*) can actually result in worse-performing models.

We can use **feature selection** techniques to reduce the dimensionality of the problem but first, let's use our **domain knowledge**.

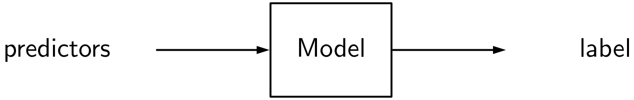
Agenda

Machine Learning taxonomy



Supervised learning

In supervised learning, we **designate** one attribute as a **label** and treat the rest as **predictors**. We set out to build a model that estimates the value of the label based on the value of the predictors.



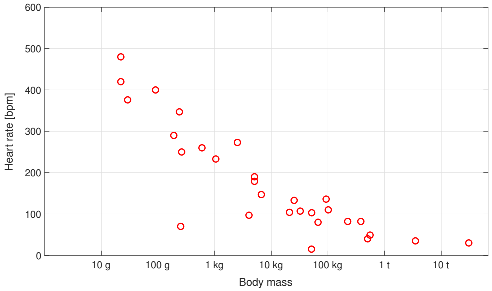
Note that:

- The model mirrors the underlying **structure** of the dataset.
- The notion of quality is defined as a function of the **discrepancy between the true and estimated** values of the label.

The space is empty

Unsupervised learning does not elevate any attribute to the category of label: all the attributes are treated equally. The essence of unsupervised learning is encapsulated in the simple question **where is my data**?

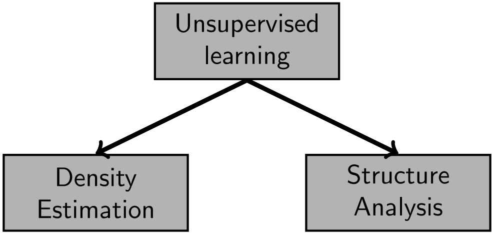
The attribute space is infinite and mostly empty, and the answer to this question will be a **model** that will allow us to identify the regions where we could expect to find samples.



Unsupervised learning

There are two main approaches to answer the question *where is my data*:

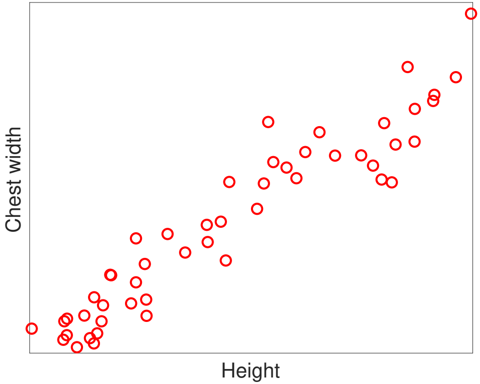
- **Density estimation**: Creates models that allow us to quantify the probability of finding a sample within a region of the attribute space (**probability density**).
- **Structure analysis**: Creates models that identifies regions within the attribute space (**cluster analysis**) or directions (**component analysis**) with a high density of samples.



Unsupervised learning: Summarising data

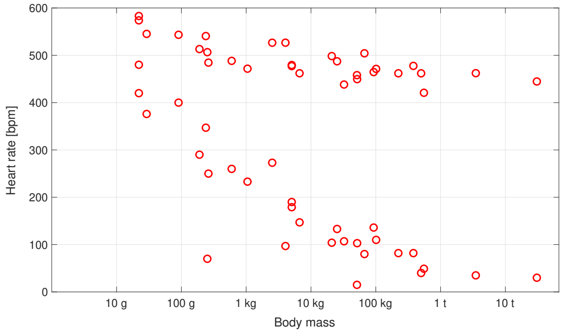
Unsupervised learning can be used to provide summaries of a population in the form of **prototypes** samples.

Look at the dataset below. If you had to produce 3 different t-shirt sizes, which sizes would you choose?



Unsupervised learning: Discovery

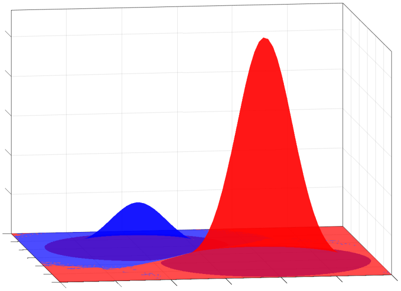
Unsupervised learning can also be used to **discover structure**. This can provide advantages (e.g. market segmentation), generate new knowledge (e.g. genetics-based migration studies) or be used to change the way we represent our data (e.g. compression).



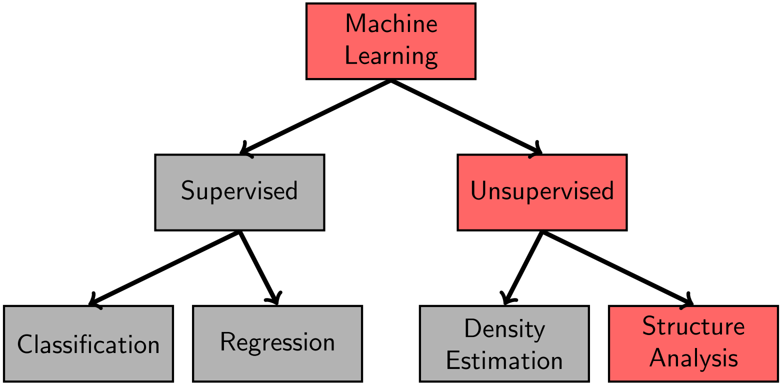
Unsupervised learning: Quantitative applications

Unsupervised learning can be used to build **class densities** that describe the probability of finding a sample from a given class in a region.

A probability density is also useful to identify **anomalies**, i.e. samples that are likely to belong to a different population.



Machine Learning taxonomy

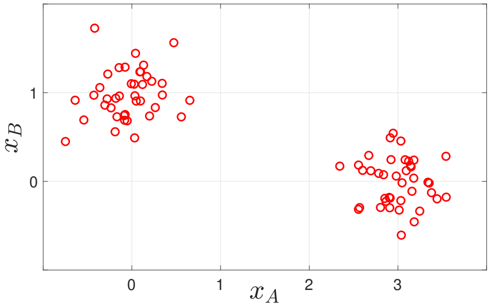


Agenda

Cluster analysis

Clustering is a family of unsupervised learning algorithms that describe the structure of a dataset as groups, or clusters, of **similar samples**.

A notion of **similarity** is therefore needed in order for us to partition a dataset into clusters.

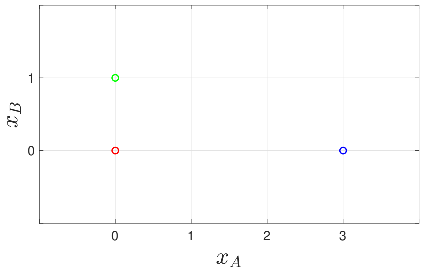


Similarity as proximity

Clusters can be defined as groups of samples that are **close** to one another. In this case, we use proximity as our notion of similarity.

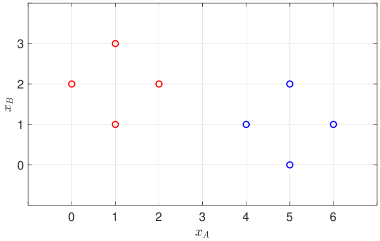
Mathematically, there are different ways of defining a distance. Given two samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ consisting of P attributes, $x_{i,1}$, ..., $x_{1,P}$ and $x_{j,1}$, ..., $x_{j,P}$, the **squared distance** $d_{i,j}$ is defined as

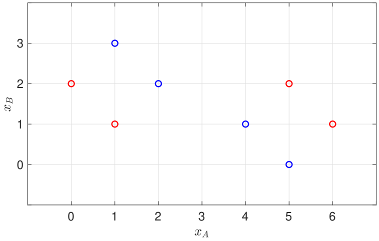
$$d_{i,j} = (x_{i,1} - x_{j,1})^2 + \cdots + (x_{i,P} - x_{j,P})^2$$



A proximity-based quality metric

Using distance as our notion of similarity, samples within the same cluster should be close to one another and samples from different clusters should be far apart.





Our next step will be to create a **quality metric** for a clustering arrangement based on the notion of distance between samples.

A proximity-based quality metric

Assume we have two clusters C_0 and C_1. The **intra-cluster sample scatter** $I(C_0)$ and $I(C_1)$ is the sum of the square distances between samples in the same cluster:

$$I(C_0) = \frac{1}{2} \sum_{x_i, x_j \text{ in } C_0} d_{i,j}, \qquad I(C_1) = \frac{1}{2} \sum_{x_i, x_j \text{ in } C_1} d_{i,j}$$

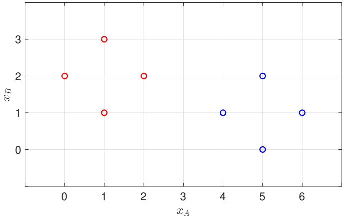
and the **inter-cluster sample scatter** $O(C_0, C_1)$ is defined as the sum of the distances between samples in different clusters:

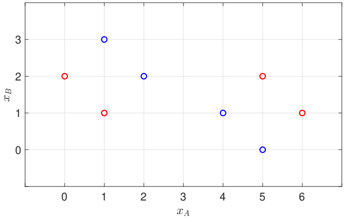
$$O(C_0, C_1) = \sum_{x_i \text{ in } C_0, x_j \text{ in } C_1} d_{i,j}$$

The **best** clustering arrangement has the **lowest intra-cluster** sample scatter and **highest inter-cluster** sample scatter. We can show that reducing the intra-cluster scatter increases the inter-cluster scatter!

A proximity-based quality metric

The intra-cluster and inter-cluster sample scatters allow us to compare and rank different clustering arrangements.

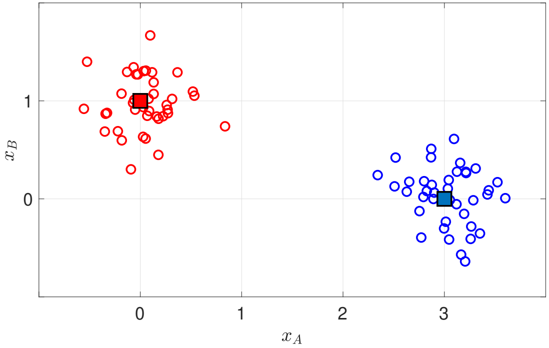




The question is, can we create an algorithm capable of automatically identifying the **best clustering** arrangement? This is an **optimisation** question.

K-means clustering: Prototypes

A simple way to describe a cluster is by using **cluster prototypes**, such as the centre of a cluster.



K-means clustering: Intra-cluster sample scatter

Given a cluster C_0 consisting of N_0 samples, its centre (or mean) $\boldsymbol{\mu}_0$ can be calculated as:

$$\boldsymbol{\mu}_0 = \frac{1}{N_0} \sum_{\boldsymbol{x}_i \text{ in } C_0} x_i$$

Interestingly, the intra-cluster sample scatter can be calculated using the **distance between each sample and the cluster prototype** d_i:

$$I(C_0) = N_0 \sum_{\boldsymbol{x}_i \text{ in } C_0} d_i$$

Therefore, our notion of clustering quality can be expressed as follows: in a good clustering arrangement, samples are **close to their prototype**.

K-means clustering

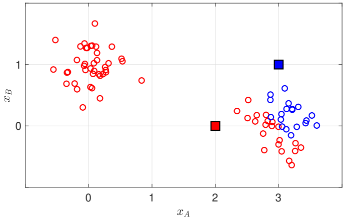
K-means partitions a dataset into K clusters represented by their **mean** and proceeds iteratively as follows:

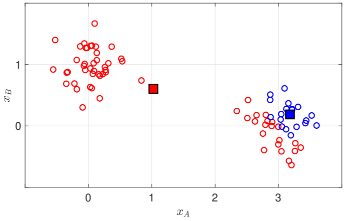
- Prototypes are obtained as the centre (or mean) of each cluster.
- Samples are re-assigned to the cluster with the closest prototype.

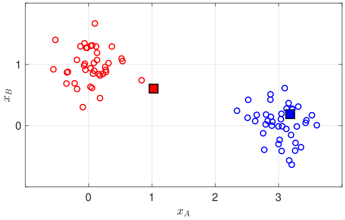
As the K-means algorithm proceeds, we will see samples been reassigned to different clusters until at some point we reach a stable solution, where no sample is reassigned.

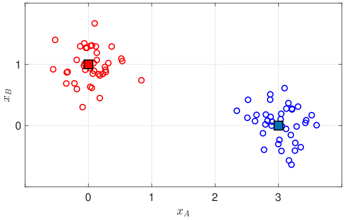
The final solution is a **local optimum**, not necessarily the global one.

K-means clustering









How many clusters?

K-means requires that we specify the number of clusters K that we want to partition our dataset into.

A natural question is, what's the right number of clusters? The answer to this question depends on the application:

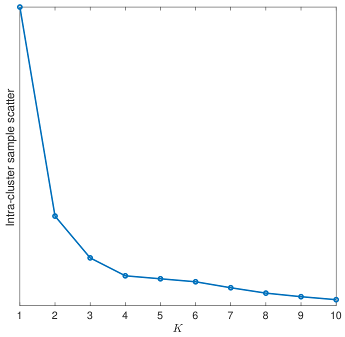
- In some cases we are given the number of clusters (e.g. t-shirt sizes).
- In a discovery scenario we want to find the underlying structure and the number of clusters is unknown.

Validation strategies can suggest a suitable value for the hyperparameter K. Choosing the value of K producing the lowest $I(C_0)$ would not work however, as $I(C_0)$ always decreases as the number of clusters increases.

The elbow method

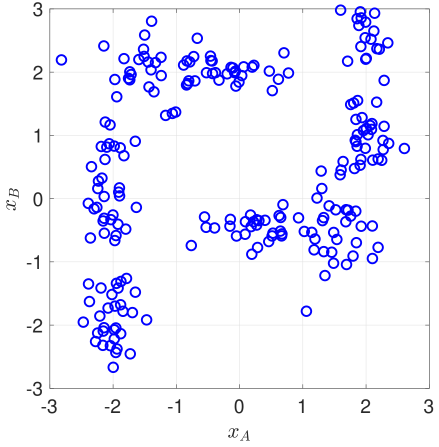
Assume the true number of clusters is K_T. For $K > K_T$, we should expect the increase in quality to be slower than for $K < K_T$, as we will be splitting true clusters.

The true number of clusters can be identified by observing the value of K beyond which the improvement slows down.



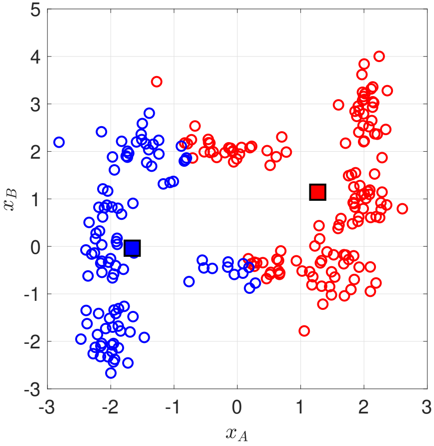
Agenda

Non-convex clusters



Non-convex clusters: K-means

K-means produces spherical clusters, as samples are arranged around a prototype.



Density-based clustering: DBSCAN

In a **non-convex** cluster, we can reach any sample by taking small jumps from sample to sample.

Non-convex scenarios suggest a different notion of cluster as group of samples that are **connected**, rather than simply close: *if I am similar to you, and you are similar to them, I am similar to them too*.

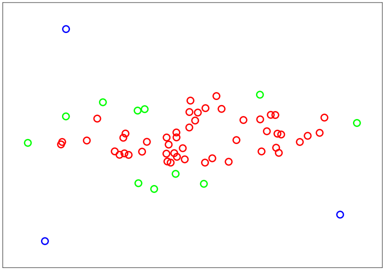
This notion of cluster as a group of connected samples is behind many clustering algorithms, such as DBSCAN (*density-based spatial clustering of applications with noise*).

DBSCAN belongs to the family of **density-based** algorithms, where an estimation of the density of samples around each sample is used to partition the dataset into clusters.

DBSCAN

DBSCAN defines two quantities, a **radius** r and a **threshold** t. A density is first calculated as the number of samples in a neighbourhood of radius r around each sample. Then, three types of samples are identified:

- **Core**: its density is equal or higher than the threshold t.
- **Border**: its density is lower than the threshold t, but contains a core sample within its neighbourhood.
- **Outlier**: Any other sample.



DBSCAN

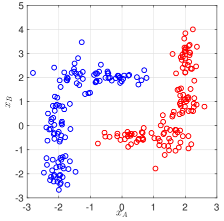
The DBSCAN algorithm proceeds as follows:

- Pair of core samples that are within each other's neighbourhood are connected. Connected core samples form the **backbone** of a cluster.
- Border samples are assigned to the cluster that has more core samples in the neighbourhood of the border sample.
- Outlier samples are not assigned to any cluster.

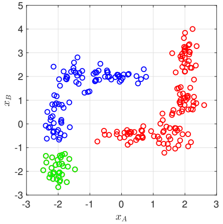
DBSCAN

Solutions for a threshold $t = 3$ and different radii.

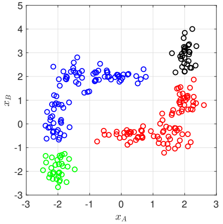
$r = 0.8$



$r = 0.5$



$r = 0.44$



Agenda

Hierarchical clustering

Given a dataset consisting of N samples, there exist two **trivial** clustering solutions: **one single cluster** that includes all the samples, and the solution where **each sample is a cluster** on its own.

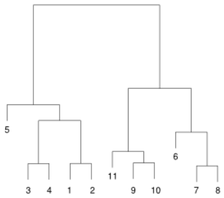
K-means produces K clusters, but we need to choose K within 1 ≤K≤ N. In DBSCAN clusters are discovered automatically, but the final number of clusters depends on the values of the radius r and the threshold value t.

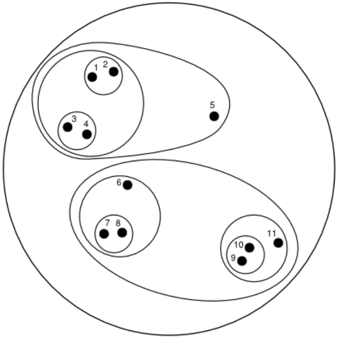
This ambiguity ultimately reveals that **the structure of a dataset can be explored** at different levels that expose different properties.

Hierarchical clustering

Hierarchical clustering is a family of clustering approaches that proceed by progressively building clustering arrangements at **different levels**.

The resulting collection of clustering arrangements is hierarchical in the sense that a cluster in one level contains all the samples from one or more clusters in the level below.





Hierarchical clustering

The representation of the relationship between clusters at different levels is called a **dendrogram**. At the bottom we find the arrangement where each sample is one cluster and at the top, the whole dataset.

There exist two basic strategies to build a **dendrogram**:

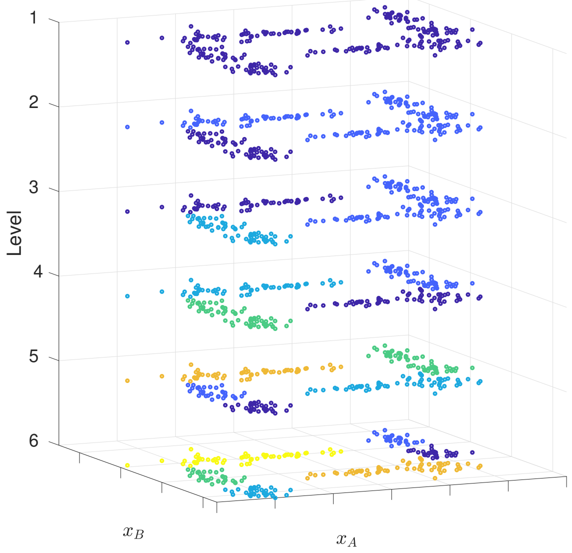
- The **divisive** or top-down approach splits clusters starting from the top of the dendrogram and stops at the bottom level.
- The **agglomerative** or bottom-up merges two clusters, starting from the bottom until we reach the top level.

Hierarchical clustering

There are different options to decide which clusters to merge or split at each level. Common strategies in agglomerative clustering include:

- **Single linkage**: uses the distance between the two closest samples from two clusters. This option results in clusters of arbitrary shapes.
- **Complete linkage**: uses the distance between the two further samples from each pair of clusters. This choice produces clusters that tend to have a spherical shape.
- **Group average**: uses the average distance between samples in two cluster and also produces spherical shapes, although they are more robust to outliers.

Hierarchical clustering



Agenda

Unsupervised learning

- Unsupervised learning provides with answers for the basic question *where is my data? (in the attribute space)*
- Our answer is a mathematical/computer model. This model can tell us where in the space we have samples (clustering) or the probability to find a sample in a region within the space (density estimation).
- Sometimes we say that our data is unlabelled. What we **really mean** is that we don't treat any attribute as a label that we want to predict. Datasets are neither labelled nor unlabelled.
- Lacking such a target as a label means that **our quality metric is not obvious**.

Clustering

- **K-means** is a prototype-based clustering that produces spherical clusters where K is a hyperparameter that has to be set.
- **DBSCAN** is a density-based option suitable for non-convex scenarios and does not require specifying the number of clusters. We need to set r and t and they determine the final number of clusters.
- **Hyerarchical clustering** allows to explore the structure of a dataset at multiple levels.

Comparing clustering solutions

- We could consider **comparing** the solutions from two different algorithms. However, if they use different definitions of clustering quality, this comparisons will make little sense.
- Clustering is ultimately implemented with an **application** in mind so we should create a final notion of clustering quality based on the specific goals of the application.

What about component analysis?

Component analysis allows us to identify the **directions in the space** that our data are aligned with. This can be useful to transform our dataset, clean it and reduce its dimensionality.