Queen Mary
University of London

ECS766P Data Mining

Week 6: Classification and Clustering

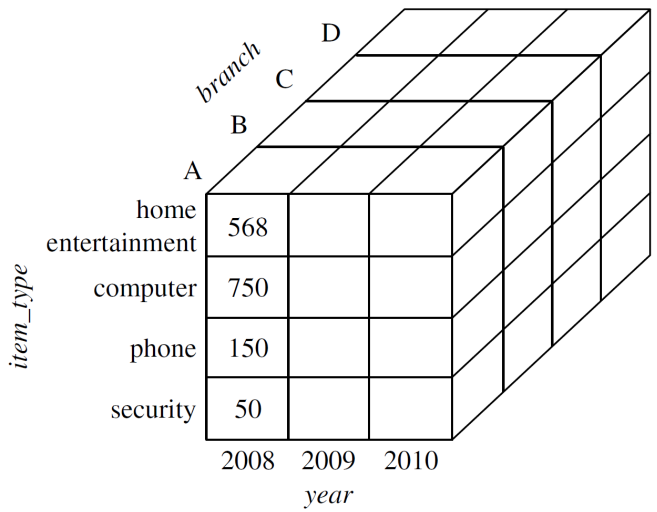
Emmanouil Benetos
emmanouil.benetos@qmul.ac.uk

November 2021

School of EECS, Queen Mary University of London

Last week: Data Warehousing and OLAP

- Data warehouse - basic concepts
- Data warehouse modelling
- Data warehouse design and usage
- Data warehouse implementation

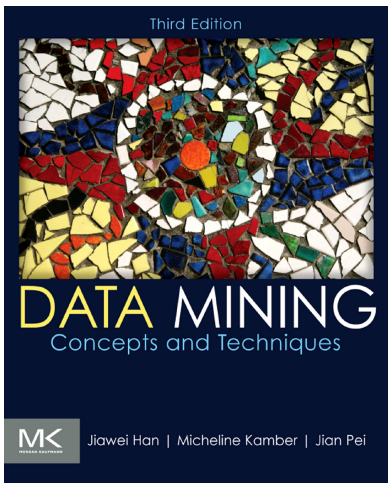


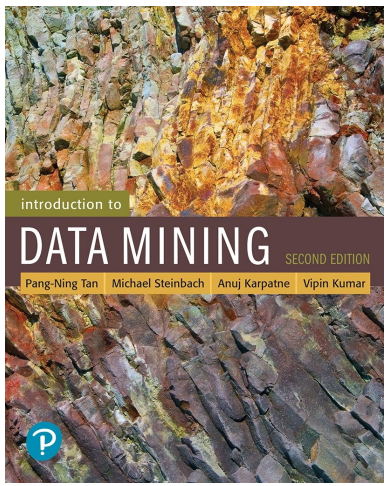
This week's contents

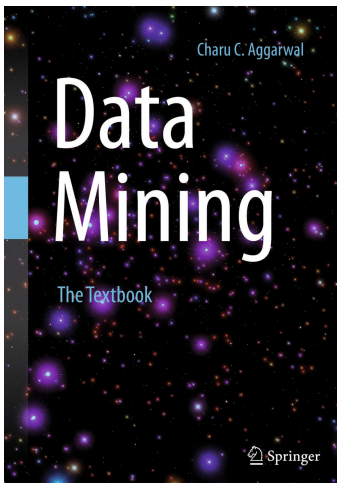
1. Classification
    - Classification task
    - *K*-nearest neighbours classifier
    - Model evaluation
    - Model selection
    - Common classifiers
2. Clustering
    - Clustering task
    - *K*-means clustering
    - Model evaluation and selection
    - Agglomerative hierarchical clustering

Reading

- Chapters 8 and 10 of J. Han, M. Kamber, J. Pei, “Data Mining: Concepts and Techniques”, 3rd edition, Elsevier/Morgan Kaufmann, 2012
- Chapters 3 and 7 of P.-N. Tan, M. Steinbach, A. Karpatne, V. Kumar, “Introduction to Data Mining”, 2nd edition, Pearson, 2019
- Chapters 6 and 10 of C. C. Aggarwal, “Data Mining: The Textbook”, Springer, 2015







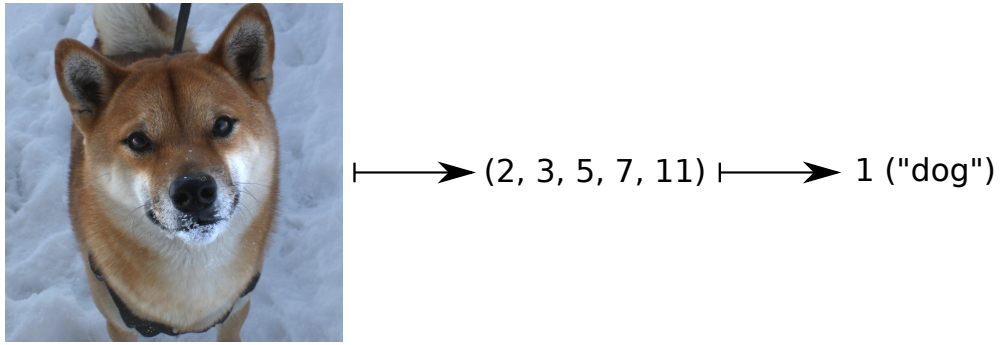
Classification

Classification

Classification task

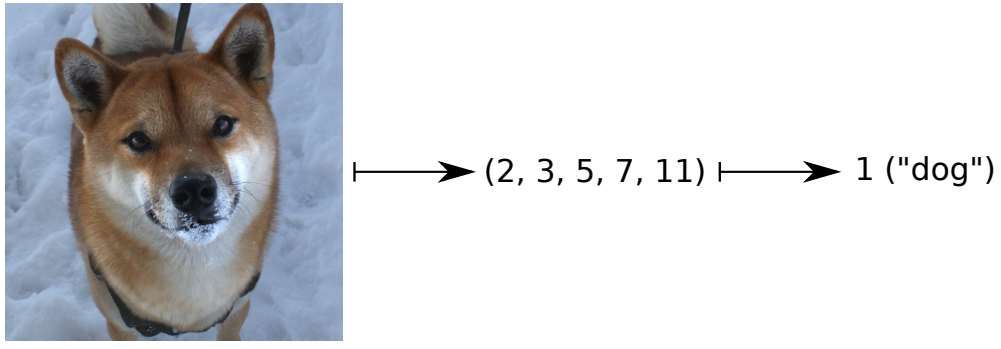
Classification dataset

- Consider a dataset $\mathcal{D} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
- The i-th observation $\mathbf{x}_i$ belongs to class y_i
- An observation $\mathbf{x}_i \in \mathbb{R}^d$ represents some object
- A class $y_i \in \mathbb{N}$ categorises the corresponding object



Classification task

- Consider a dataset $\mathcal{D} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
- Classification task: predicting the class y for a new observation $\mathbf{x}$ based on the examples in $\mathcal{D}$
- Assumptions: each observation belongs to a single class, and there is a finite number of classes

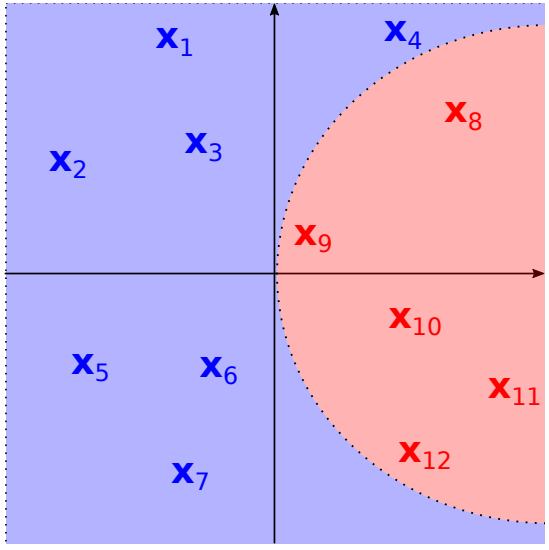


Applications

- Medical image analysis
- Drug discovery
- Natural language processing
- Video surveillance
- Face recognition
- Speech recognition
- Biometric identification
- Credit scoring

Learning algorithm

- A classifier $f : \mathbb{R}^d \to \mathbb{N}$ maps observations to classes
- A learning algorithm receives a dataset $\mathcal{D}$ and produces a classifier



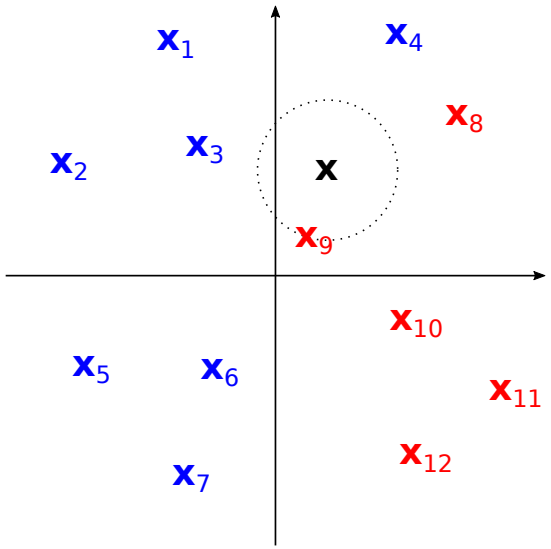
- Regression task: similar to a classification task, but observations are paired with continuous values

Classification

K-nearest neighbours classifier

One-nearest neighbour

- Consider a dataset $\mathcal{D} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
- A one-nearest neighbour classifier classifies a new observation $\mathbf{x}$ as y_i whenever the observation $\mathbf{x}_i$ is the closest observation to $\mathbf{x}$ in the dataset $\mathcal{D}$



Distance function

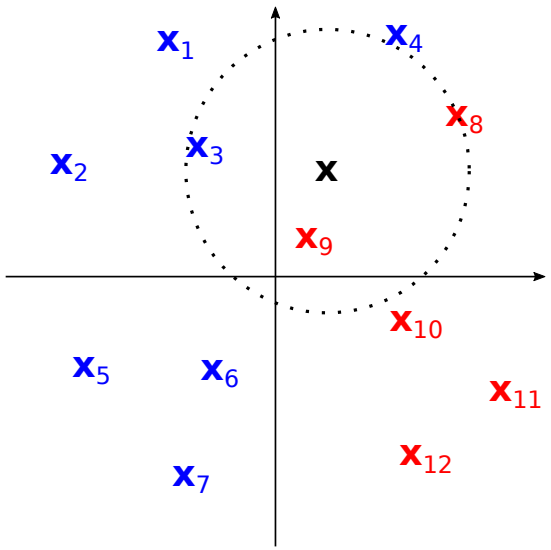
- The distance function between observations is an important choice
- For example, the Euclidean distance e given by

$$e(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'|| = \sqrt{\sum_{j=1}^{d} (x_j - x'_j)^2}$$

requires attributes with comparable units, and one-hot encoded categorical attributes

K-nearest neighbours

- A *K*-nearest neighbours classifier classifies a new observation $\mathbf{x}$ as y if the majority of the K-nearest neighbours of $\mathbf{x}$ in the dataset $\mathcal{D}$ belongs to class y (tie breaking is arbitrary)



- A hyperparameter (such as K) is any setting required by a learning algoritm to produce a classifier

Classification

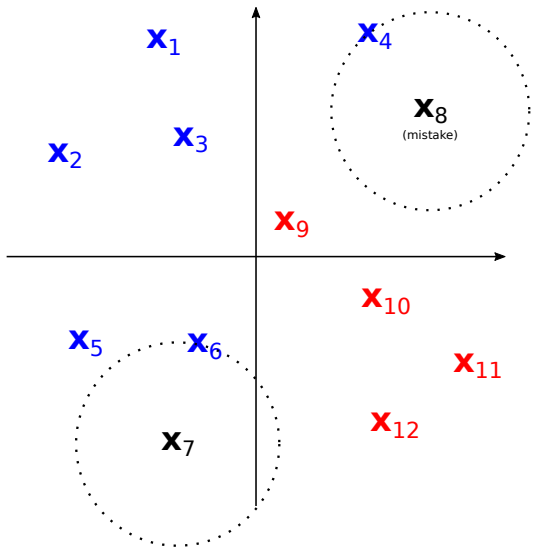
Model evaluation

Generalisation

- A one-nearest neighbour classifier will always predict the correct class for any observation that *already exists* in the dataset $\mathcal{D}$
- However, it is usually necessary to estimate generalisation: the capacity of a given classifier to assign **unseen observations** to correct classes
- How to estimate generalisation given a fixed $\mathcal{D}$?

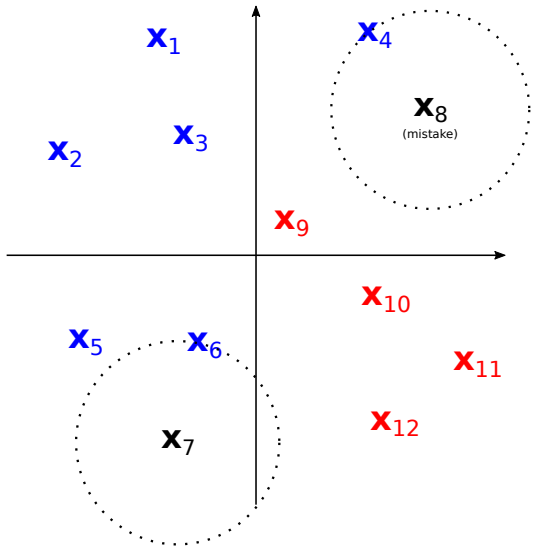
Training and test datasets

- A simple approach is to split an original dataset into a training dataset $\mathcal{D}$ and a test dataset $\mathcal{D}'$
- A learning algorithm is given only the training dataset $\mathcal{D}$ in order to produce a classifier f
- The classifier f may be **evaluated** on the unseen observations in the test dataset $\mathcal{D}'$



Training and test datasets

- The test dataset usually contains at least 20% of the original pairs, which are chosen randomly
- From now on, assume that classifiers are always **trained** on training datasets



Performance metric: accuracy

- The accuracy of a classifier $f : \mathbb{R}^d \to \mathbb{N}$ on a dataset $\mathcal{D} = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ is the fraction of observations in $\mathcal{D}$ that are classified correctly by f:

$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left[f(\mathbf{x}_i) = y_i\right],$$

  where $\mathbb{I}[e] = 1$ if e is true, and $\mathbb{I}[e] = 0$ otherwise
- The accuracy is always between 0% and 100%
- **Note the effect of class imbalance**: if 99% of the observations in $\mathcal{D}$ belong to class y, a classifier that predicts y for every observation has 99% accuracy

Overfitting and underfitting

- Suppose the training and test datasets are **large** and **representative**
- A classifier overfits if its performance is much lower on the test set than on the training set
- A classifier underfits if its performance is low both on the test set and on the training set
- A one-nearest neighbour classifier never underfits, but it may overfit
- Different learning algorithms produce classifiers with potentially different biases, which ultimately leads to different performances

Categorizing predictions: two classes

- Consider a classification problem with **two classes**, labeled **positive** and **negative** for convenience
- A prediction $f(\mathbf{x})$ made for $\mathbf{x}$ is either:
  - True positive (TP): **correct** positive prediction
  - False positive (FP): **incorrect** positive prediction
  - True negative (TN): **correct** negative prediction
  - False negative (FN): **incorrect** negative prediction
- **Note the effect of errors**: incorrectly predicting that a patient is disease-negative (no treatment) may be much worse than incorrectly predicting that a patient is disease-positive (unnecessary treatment)

Precision

- Consider a dataset $\mathcal{D}$ with **two classes**
- The precision of a classifier f on $\mathcal{D}$ is given by

$$\frac{N_{TP}}{N_{TP} + N_{FP}},$$

  where N_S is the number of S-type predictions
- **Interpretation**: percentage of positive predictions that were correct
- **High precision**: high confidence in positive predictions
- **Perfect precision**: no mistakes in positive predictions

Recall

- Consider a dataset $\mathcal{D}$ with **two classes**
- The recall of a classifier f on $\mathcal{D}$ is given by

$$\frac{N_{TP}}{N_{TP} + N_{FN}}$$

  where N_S is the number of S-type predictions
- **Interpretation**: percentage of positive observations that were predicted correctly
- **High recall**: high confidence in *detecting* positive observations
- **Perfect recall**: no *undetected* positive observations

F_1-score

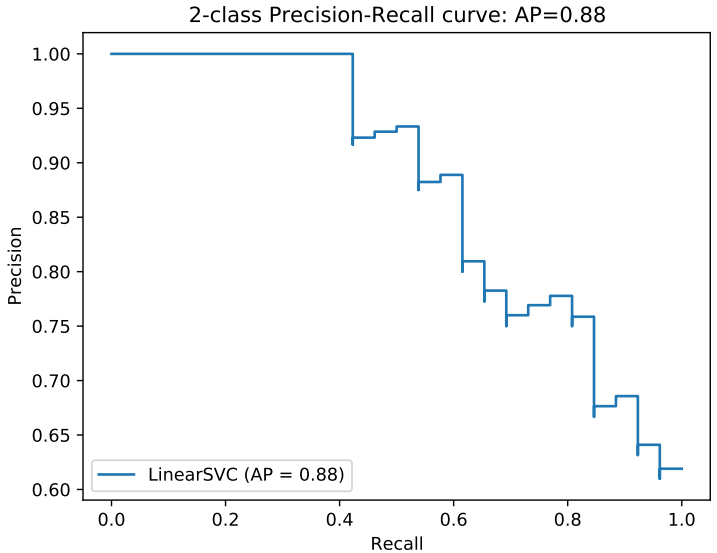
- **Note:** A classifier that predicts *negative* for every observation has **perfect precision**
- **Note:** A classifier that predicts *positive* for every observation has **perfect recall**
- The F_1-score of a classifier f on $\mathcal{D}$ is given by

$$\frac{N_{TP}}{N_{TP} + (N_{FP} + N_{FN})/2}$$

  where N_S is the number of S-type predictions
- **Interpretation**: harmonic mean between precision and recall, which gives both equal importance
- **Perfect F_1-score**: perfect accuracy

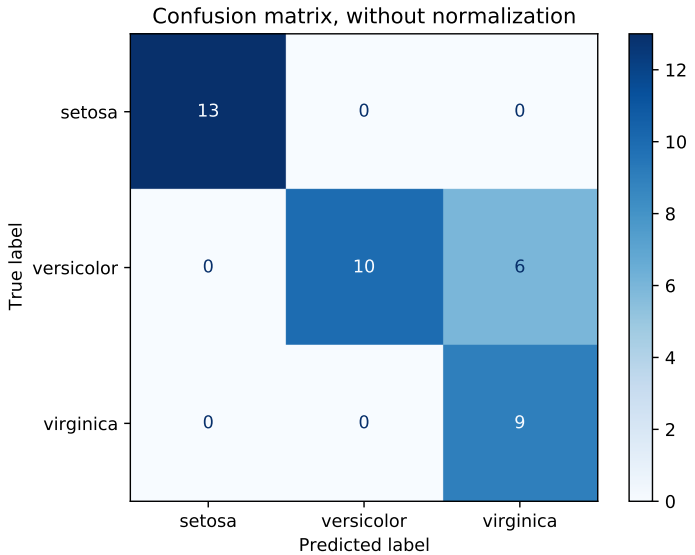
Precision-recall curve

- Consider a dataset $\mathcal{D}$ with **two classes**
- In K-nearest neighbors, instead of majority-based classification, a hyperparameter $0 \leq \alpha \leq K+1$ may be used to control how many positive neighbours are required for a positive prediction
- A precision-recall curve displays precision and recall for every possible value of α



Categorizing predictions: multiple classes

- Consider a dataset $\mathcal{D}$ with **multiple classes**
- Let $N_{i,j}$ denote the number of times that classifier f predicted class j when it should have predicted class i on the dataset $\mathcal{D}$
- A confusion matrix presents $N_{i,j}$ for every i and j



Classification

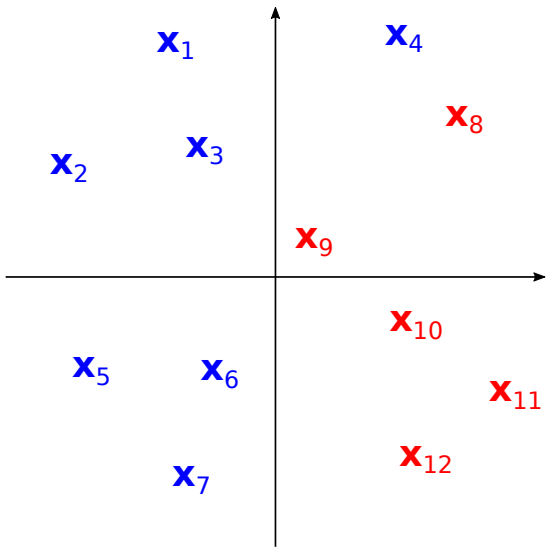
Model selection

Model selection

- Suppose $f_1, \ldots, f_T$ are classifiers trained on $\mathcal{D}$
- Model selection is the process of choosing between these classifiers based on their performance
- This process applies whether the classifiers are the result of different **learning algorithms** or different **hyperparameters** for the same algorithm
- For the sake of concreteness, you may suppose that f_K is a K-nearest neighbours classifier trained on $\mathcal{D}$

Model selection

- A classifier f_k **should not** be selected based on its performance on the **training set** $\mathcal{D}$
- Performance on the training set is not a reliable estimate of generalisation for f_k



Model selection

- A classifier f_k **should not** be selected based on its performance on the **test set** $\mathcal{D}'$
- Performance on the test set could have been obtained by sacrificing performance on other datasets, and there would be **no data left** to enable a reliable estimate of generalisation for f_k
- If model selection were seen as a **learning algorithm** that receives a dataset $\mathcal{D}$ and produces a classifier f_k, this approach would have used the test set $\mathcal{D}'$ during training

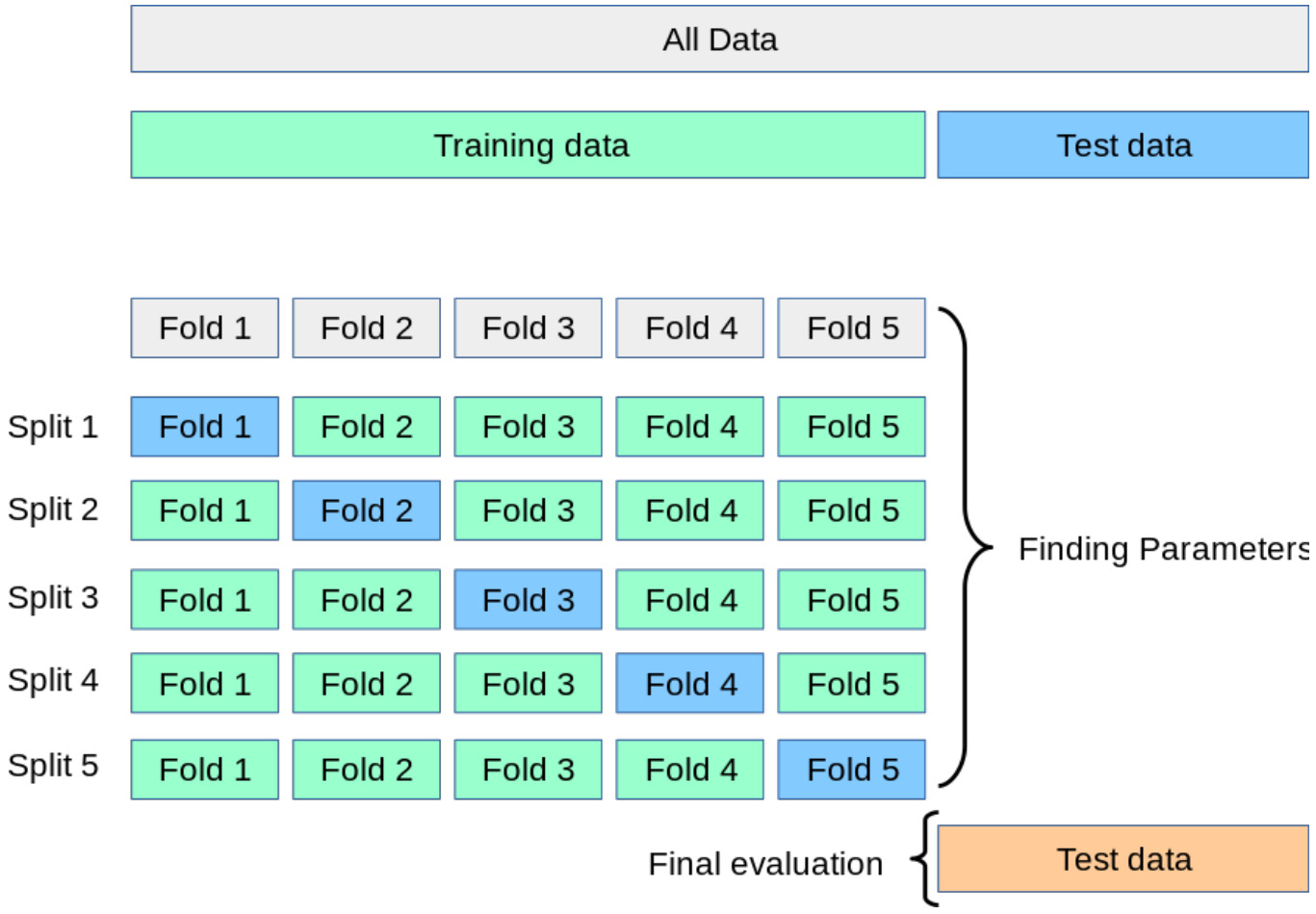
Training, validation, and test datasets

- A fixed dataset may be **randomly** split into:
  - Training dataset: enables training different classifiers
  - Validation dataset: enables choosing the best classifier based on its performance
  - Test dataset: enables estimating reliably the performance of the best classifier
- **Improving model selection after using the test set defeats its purpose: use the test set as a final step**
- In most cases, after the best combination of learning algorithm and hyperparameters is found, all three datasets may be used to train a final classifier

K-fold cross-validation

- A **representative** validation set is crucial to select the best classifier
- However, there is a **trade-off** between the size of the validation set and the size of the training set
- In K-fold cross-validation, a fixed dataset is randomly split into a training dataset $\mathcal{D}$ and a test set $\mathcal{D}'$
- The training dataset is further randomly split into K folds (subsets) of equal size
- Each fold is used as a **validation set** when the remaining folds are used as an **effective training set**
- The classifier that achieves maximum **average performance** across folds (validation sets) is selected

5-fold cross-validation

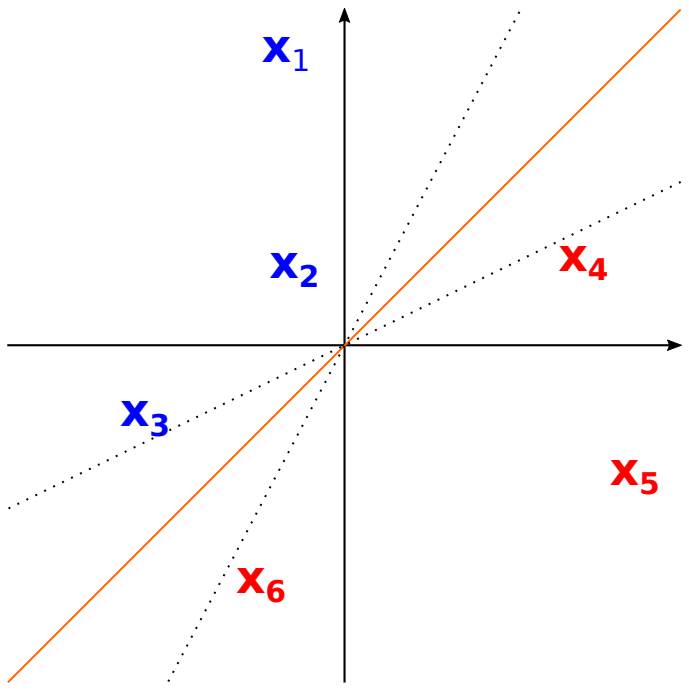


Classification

Common classifiers

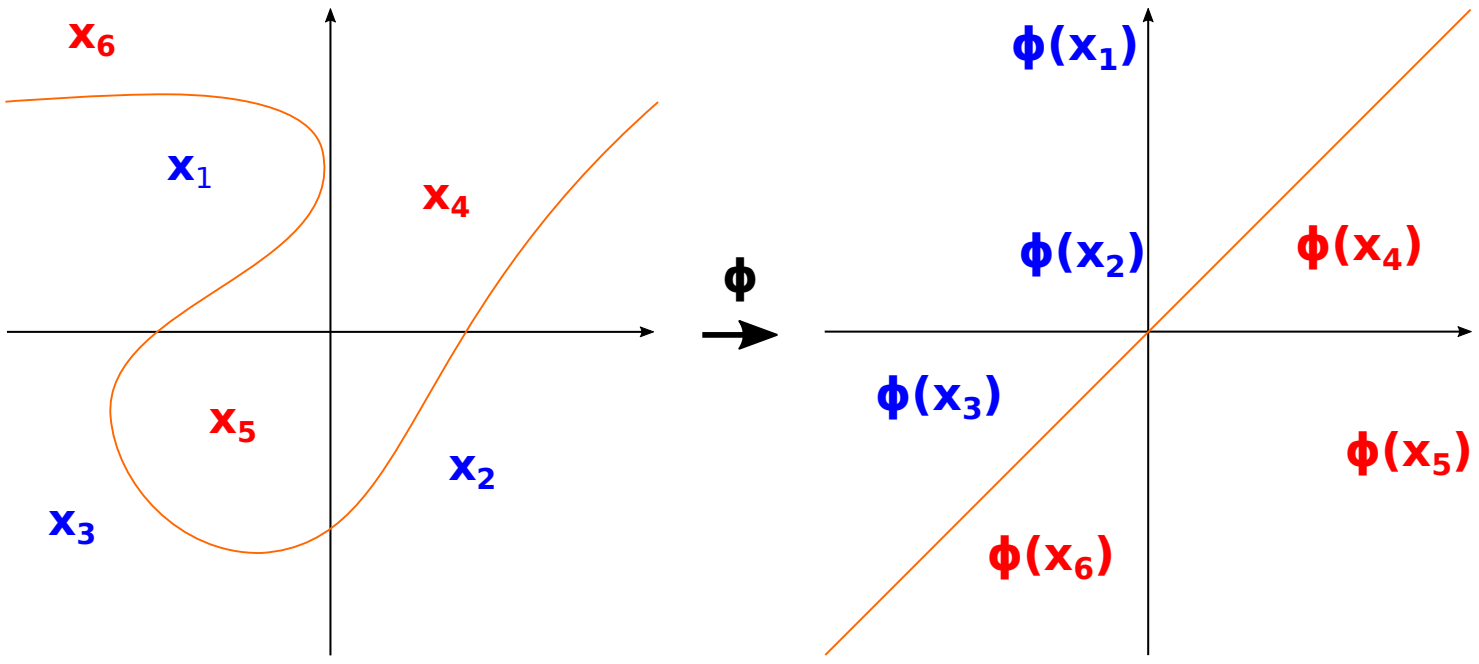
Support vector machines: hard margins

- A hard-margin support vector machine classifies observations based on the **separating hyperplane** with **maximum distance** to the **closest observation**



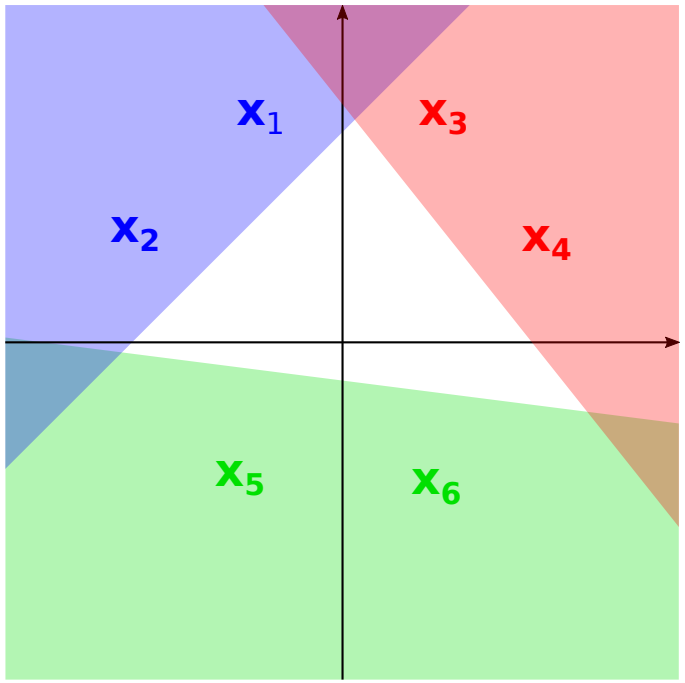
Support vector machines: soft margins and kernels

- A soft-margin support vector machine may **ignore** some observations (subject to a penalty)
- A kernel is a function that can be used to **implicitly** transform observations by a feature map ϕ in order to make a dataset $\mathcal{D}$ linearly separable



Support vector machines: multiple classes

- Support vector machines can be adapted to deal with **multiple classes**



Artificial neural networks

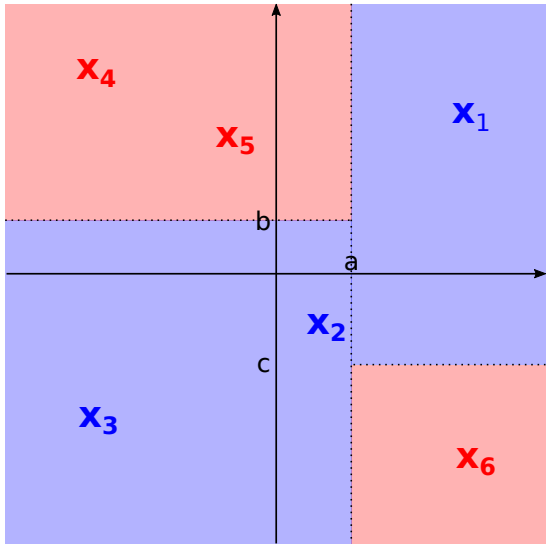
- An artificial neural network classifier f is composed of **parameterised functions** $f_1, f_2, \ldots, f_{l-1}, f_l$:

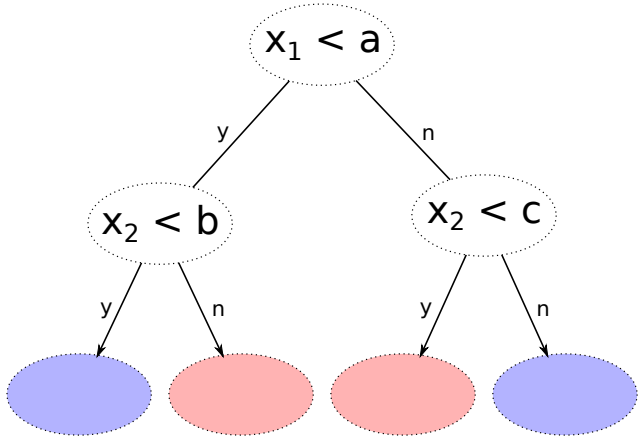
$$f(\mathbf{x}) = (f_l \circ f_{l-1} \circ \ldots \circ f_2 \circ f_1)(\mathbf{x})$$

- An artificial neural network is trained by finding **parameters** for the functions $f_1, f_2, \ldots, f_l$ that jointly maximise performance on the training set
- Some networks currently excel at classifying **unstructured data** (text, video, and audio)
- However, such networks are **not** exceptionally good at classifying **structured data**

Decision trees

- A decision tree classifies an observation according to a sequence of **logical tests** based on its features
- Different learning algorithms lead to different trees





- A random forest classifier combines predictions from different decision trees to reduce **overfitting**

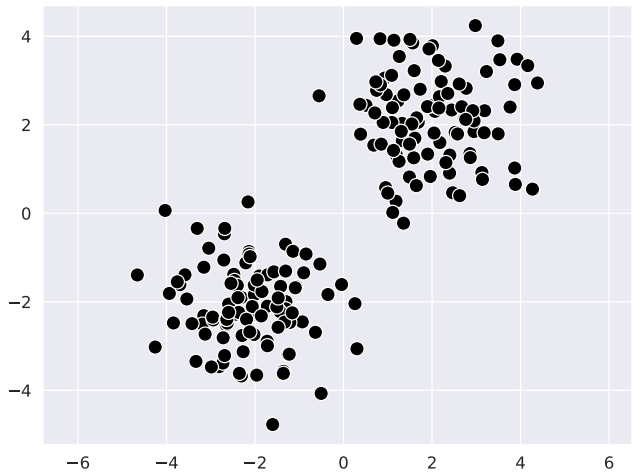
Clustering

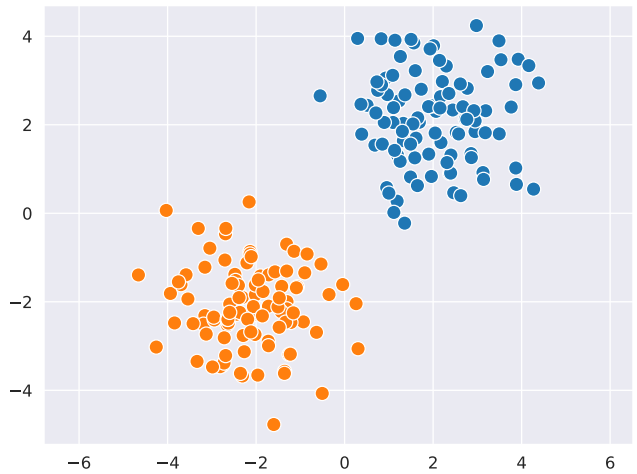
Clustering

Clustering task

Clustering task

- Consider a dataset $\mathcal{D} = \mathbf{x}_1, \ldots, \mathbf{x}_N$
- An observation $\mathbf{x}_i \in \mathbb{R}^d$ represents some object
- Clustering task: partitioning $\mathcal{D}$ into clusters such that **similar** observations belong to the same cluster





Applications

- Market research
- Recommendation systems
- Biology, computational biology and bioinformatics
- World wide web
- Social science
- Finance

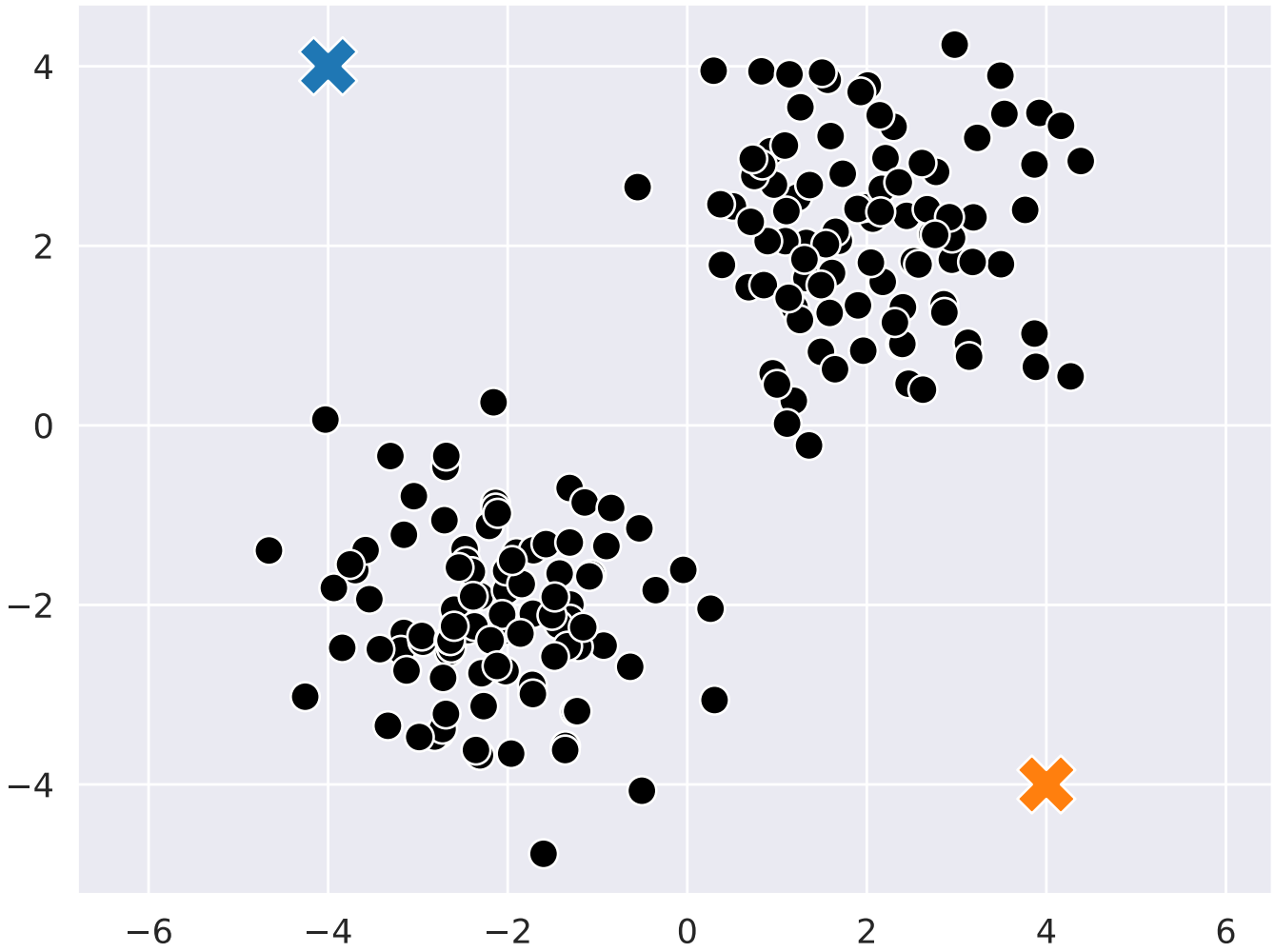
Clustering

K-means clustering

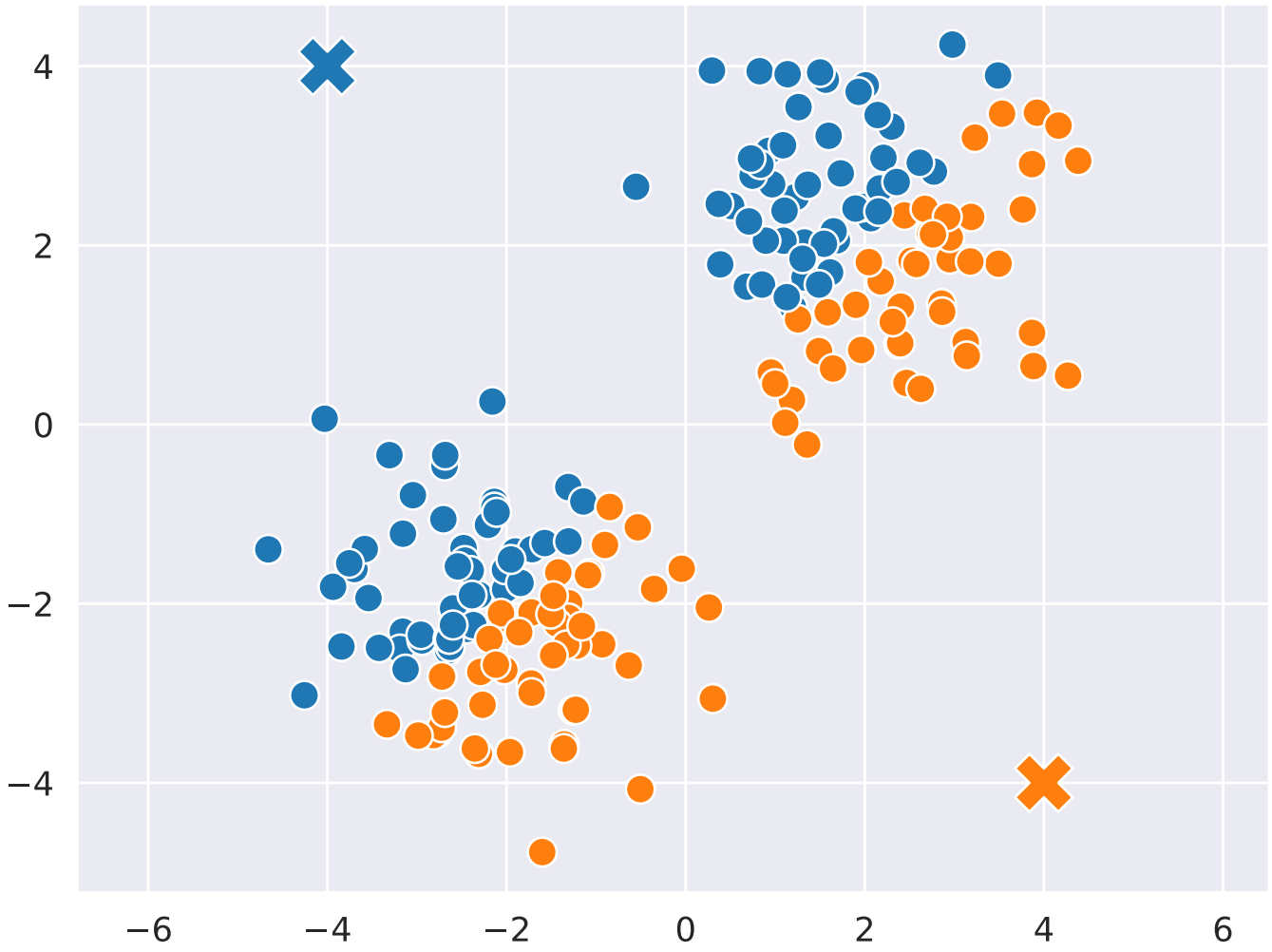
K-means clustering

- K-means is an iterative algorithm that partitions a dataset $\mathcal{D} = \mathbf{x}_1, \ldots, \mathbf{x}_N$ into K clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$
- Initially, arbitrary vectors $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K \in \mathbb{R}^d$ are chosen as cluster centers
- The following steps alternate until convergence:
  - Each observation $\mathbf{x}_i$ is **assigned** to the **closest** cluster center
  - Each cluster center $\boldsymbol{\mu}_k$ is **moved** to the **average** of the observations assigned to it
- Finally, **cluster** $\mathcal{C}_k$ contains the observations assigned to **cluster center** $\boldsymbol{\mu}_k$

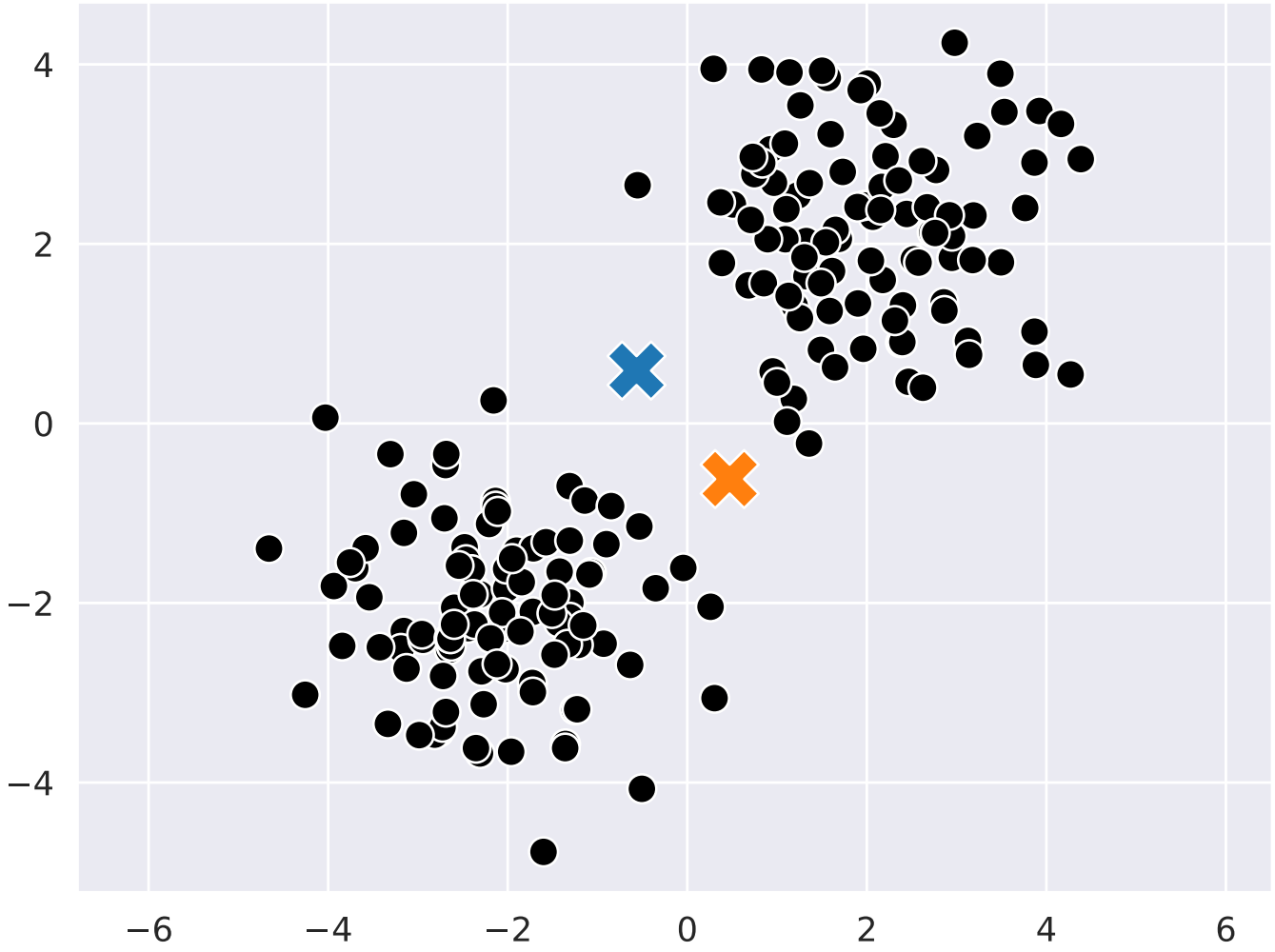
K-means: initialise



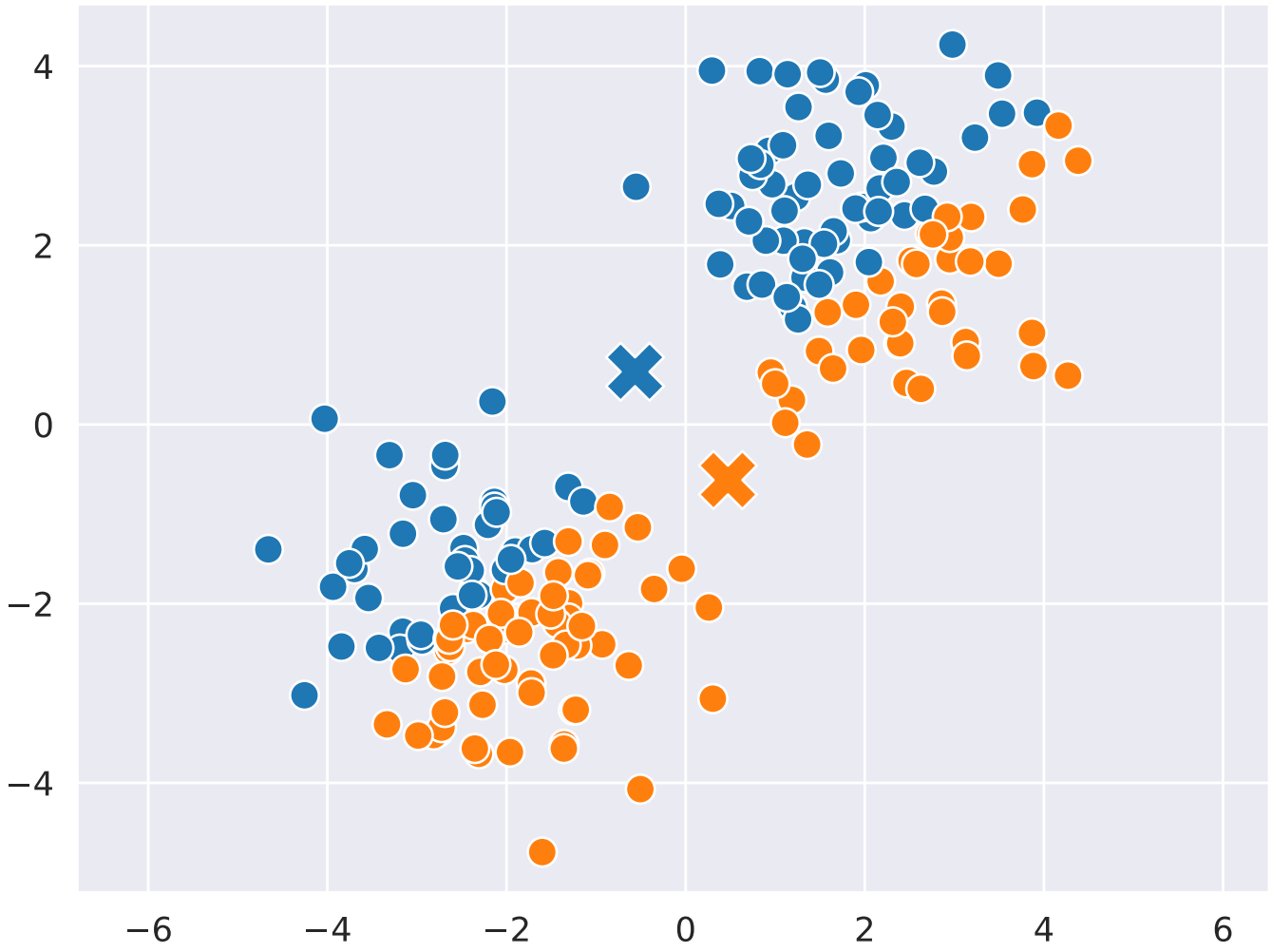
K-means: assign



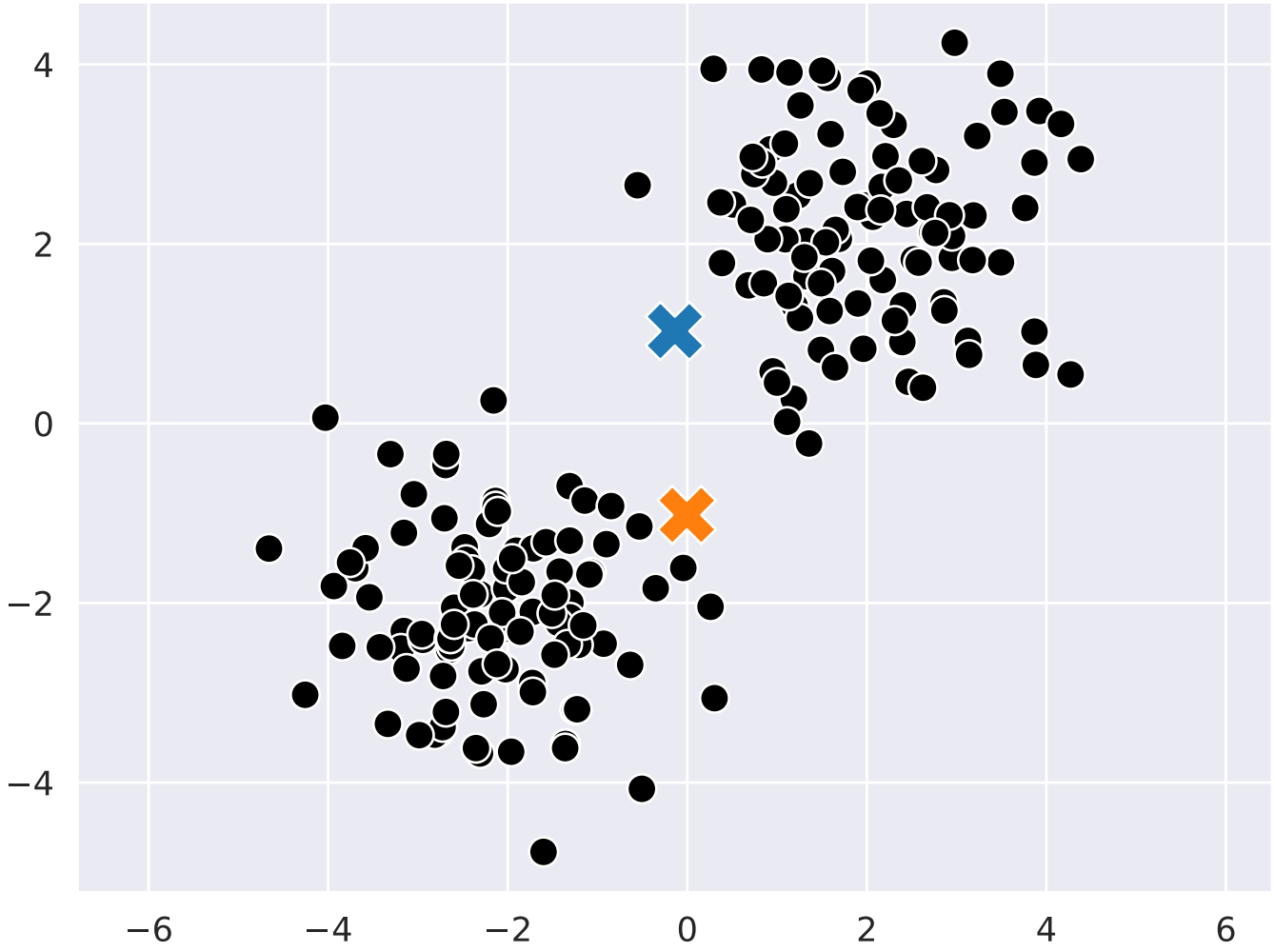
K-means: move



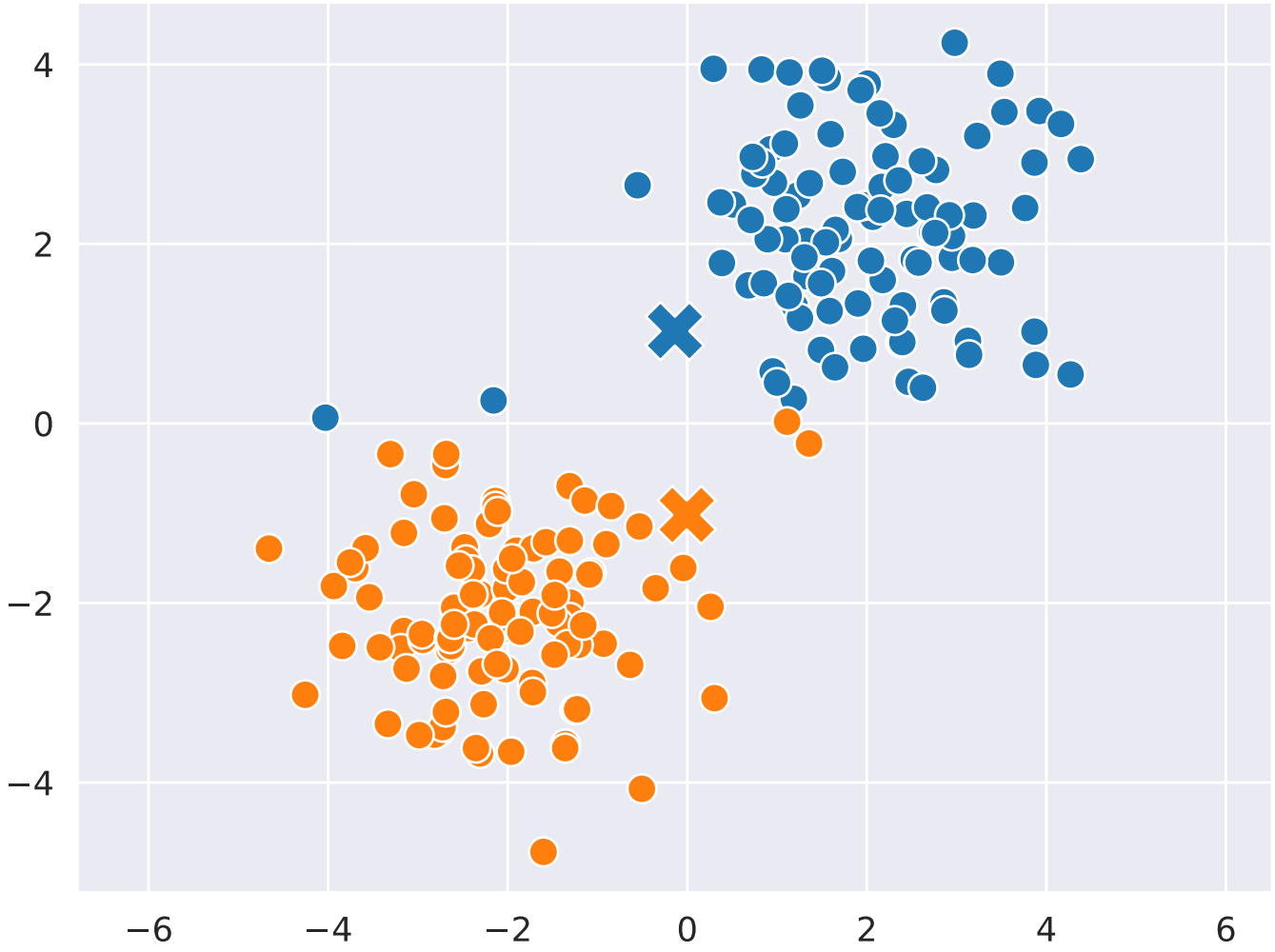
K-means: assign



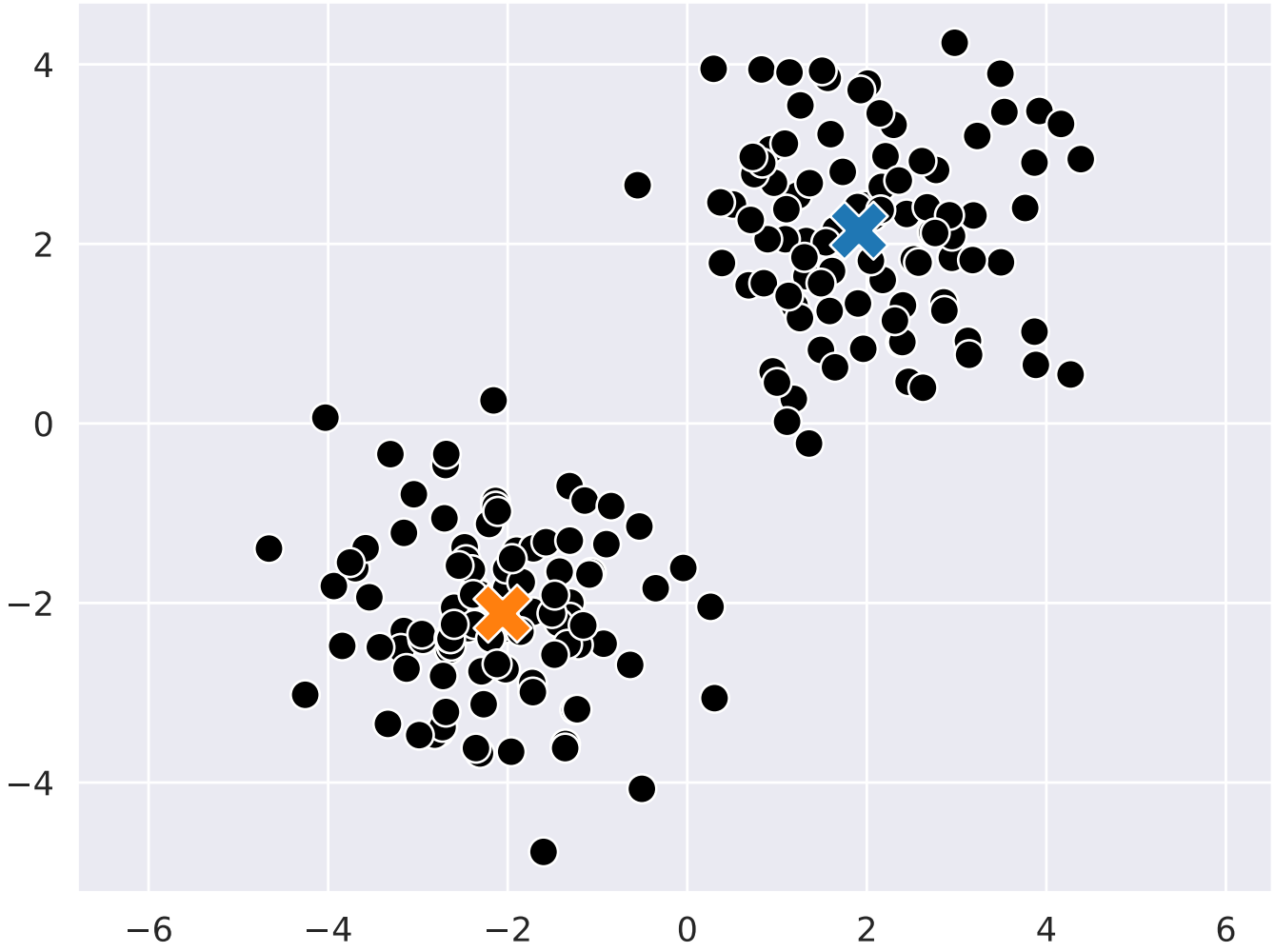
K-means: move



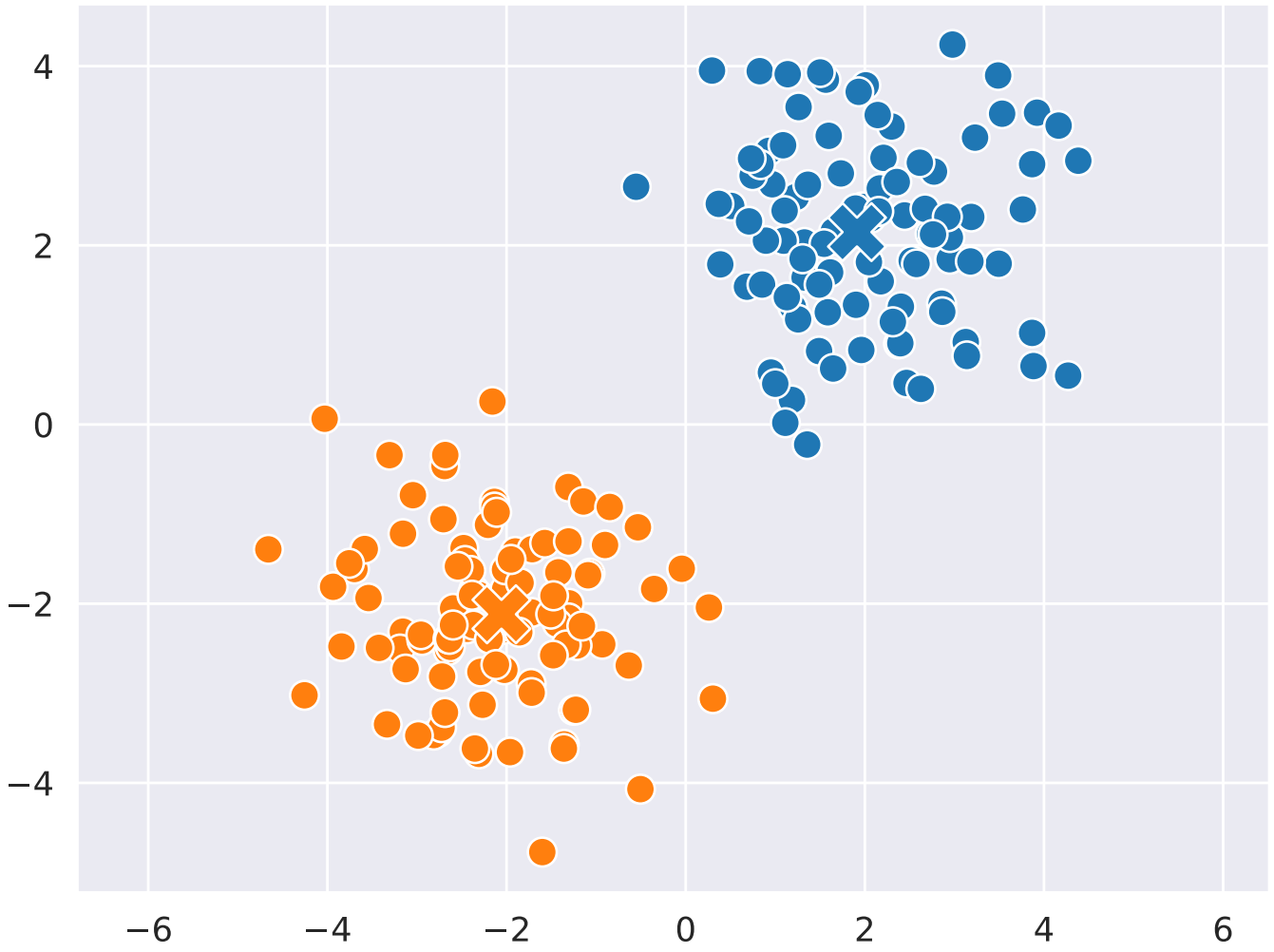
K-means: assign



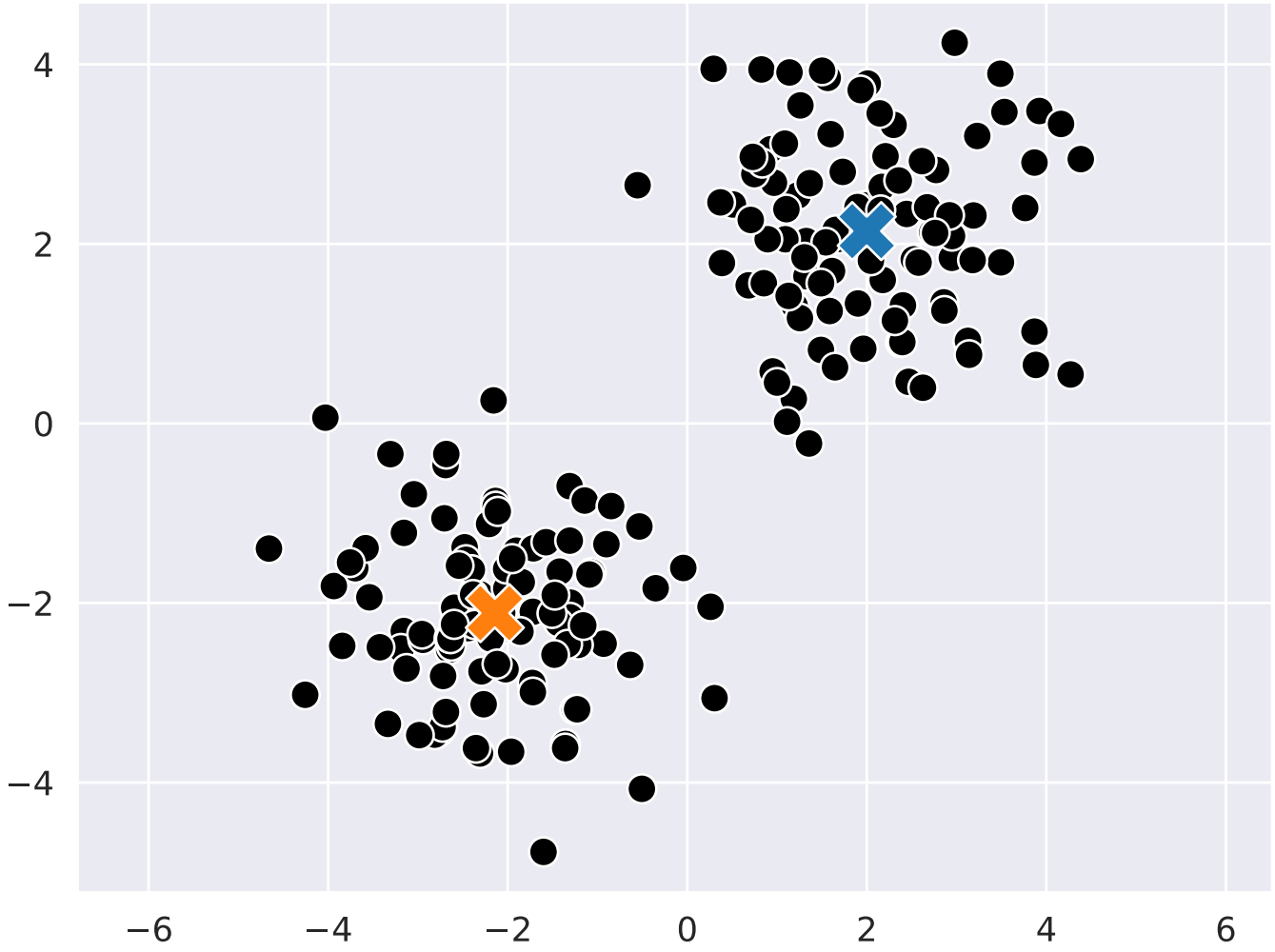
K-means: move



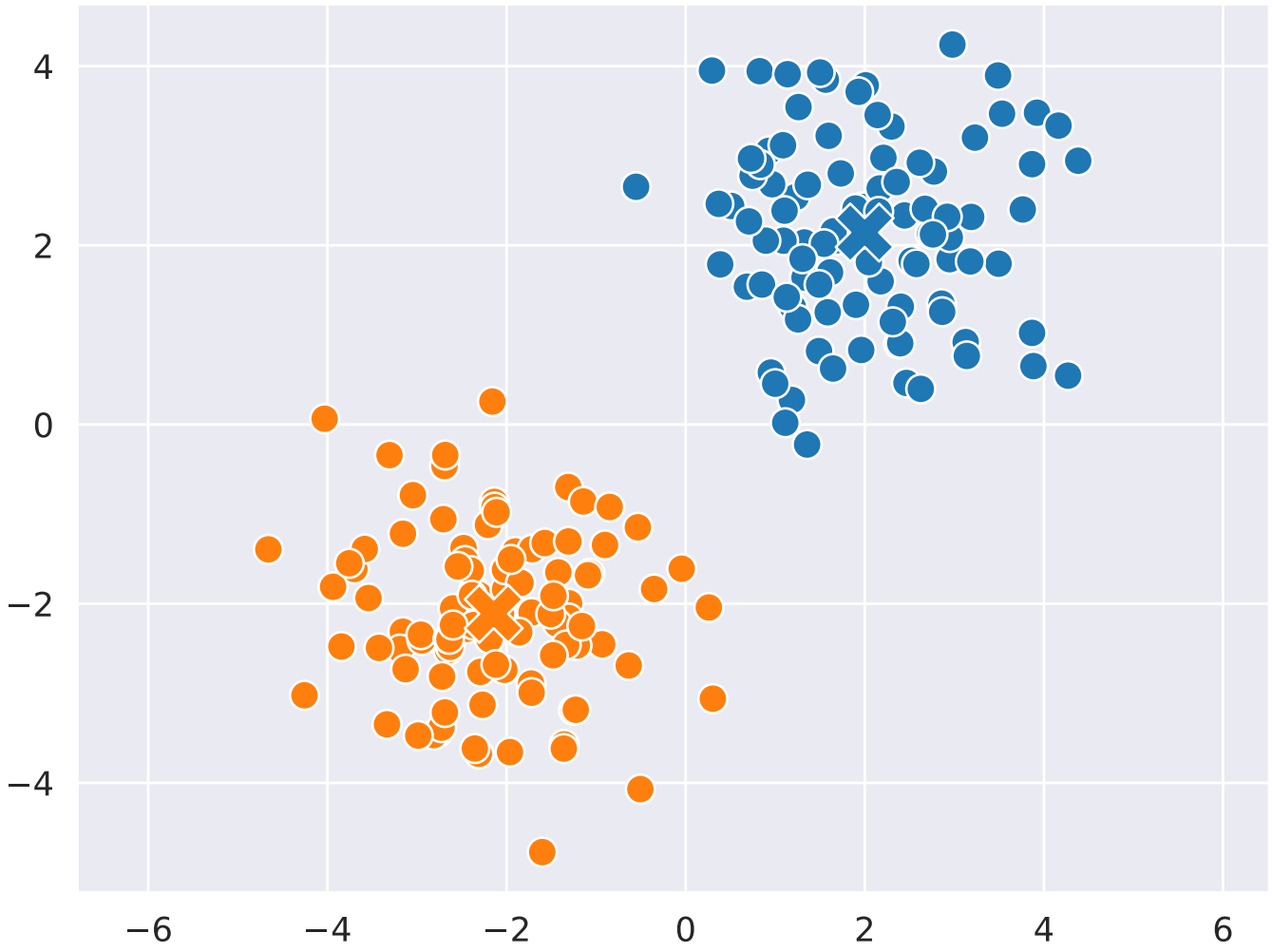
K-means: assign



K-means: move



K-means: assign



Distance function

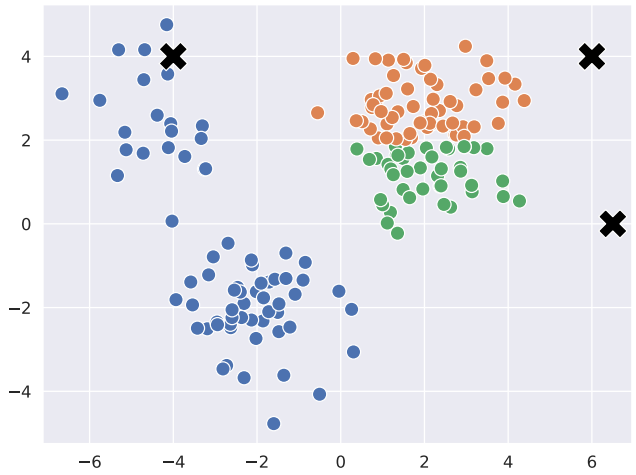
- The distance function between observations is an important choice
- For example, the Euclidean distance e given by

$$e(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'|| = \sqrt{\sum_{j=1}^{d} (x_j - x'_j)^2}$$

  requires attributes with comparable units, and one-hot encoded categorical attributes
- K-means with an Euclidean distance function **always converges**

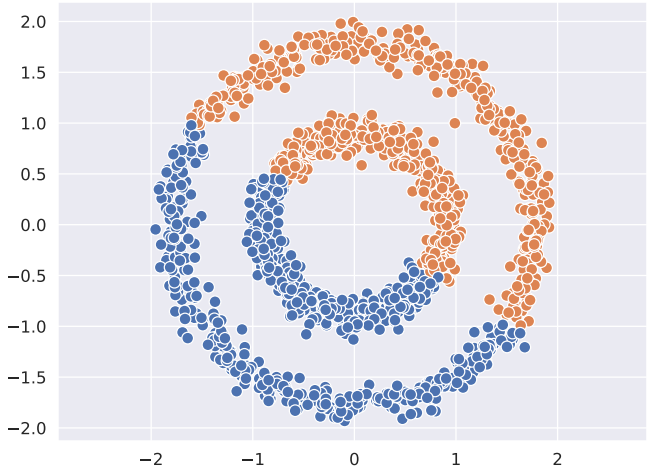
Initialisation

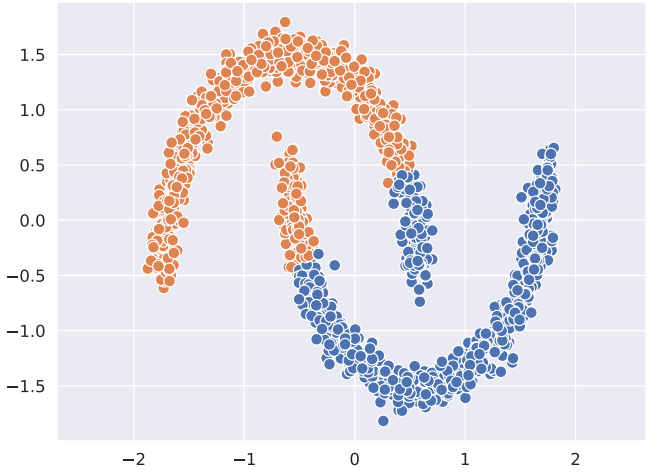
- *K*-means may converge to **different clusters** given **different initialisations**



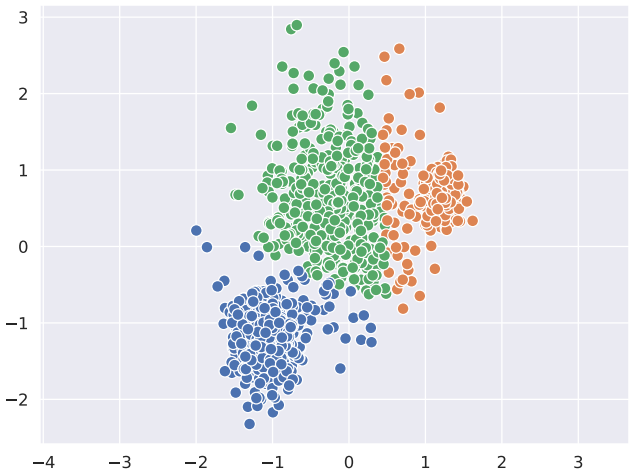
- There are many different initialisation **heuristics**

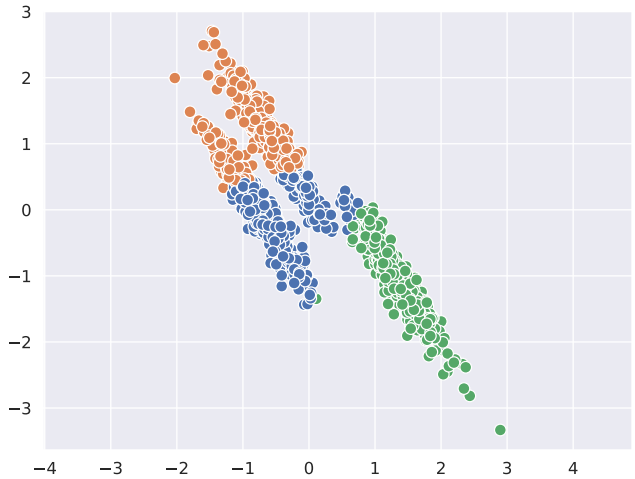
K-means: undesirable behavior (?)





K-means: undesirable behavior (?)





Clustering

Model evaluation and selection

Model evaluation and selection

- There are several metrics to evaluate the **quality** of alternative **clusterings** of the same dataset $\mathcal{D}$
- Model selection is the process of choosing between alternative clusterings based on their quality
- This process applies whether the clusterings are the result of different **clustering algorithms**, different **hyperparameters** for the same algorithm, or even different **initial conditions** for the same algorithm

Quality metric: sum of squared errors

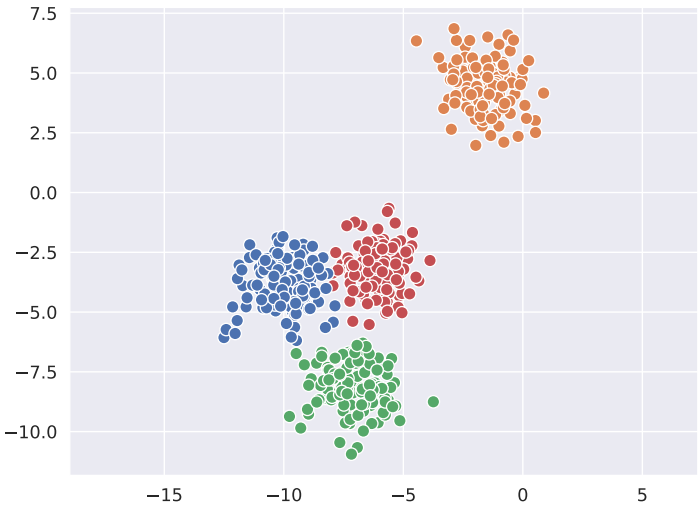
- Consider a dataset $\mathcal{D} = \mathbf{x}_1, \ldots, \mathbf{x}_N$ partitioned into clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$, and let $\boldsymbol{\mu}_k$ be the center of $\mathcal{C}_k$
- The sum of squared errors is given by

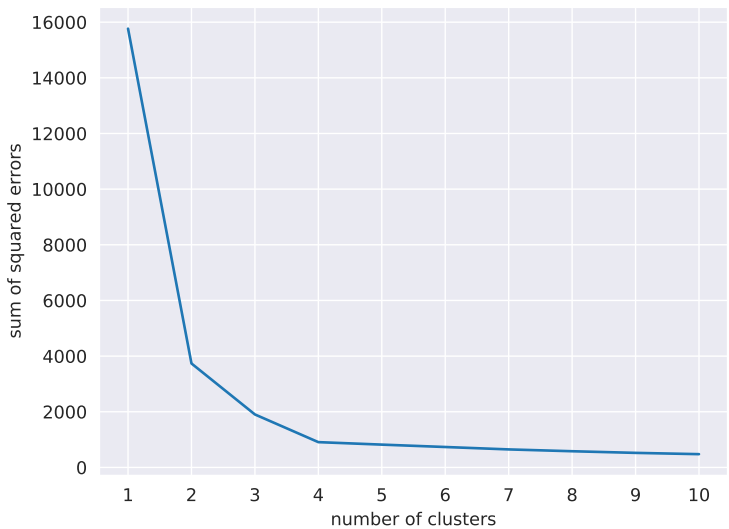
$$\sum_{k=1}^{K} \sum_{\mathbf{x} \in \mathcal{C}_k} ||\mathbf{x} - \boldsymbol{\mu}_k||^2$$

- Model selection may seek to **minimise** this metric while also **minimizing** the number of clusters K

Example: choosing K for K-means clustering

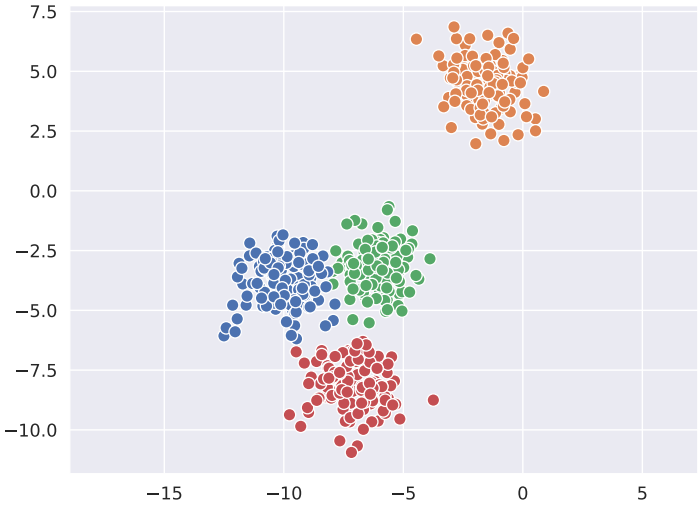
- Elbow heuristic: selecting the **largest** number of clusters before the rate of decrease in sum of squared errors becomes too **small**

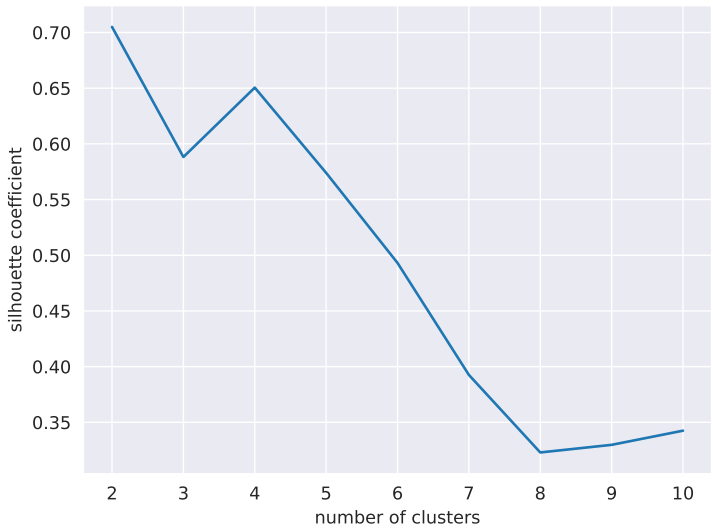




Quality metric: silhouette coefficient

- The silhouette coefficient evaluates intra-cluster **cohesion** and inter-cluster **separation**
- Model selection may seek to **maximise** this metric while also **minimising** the number of clusters *K*





Clustering

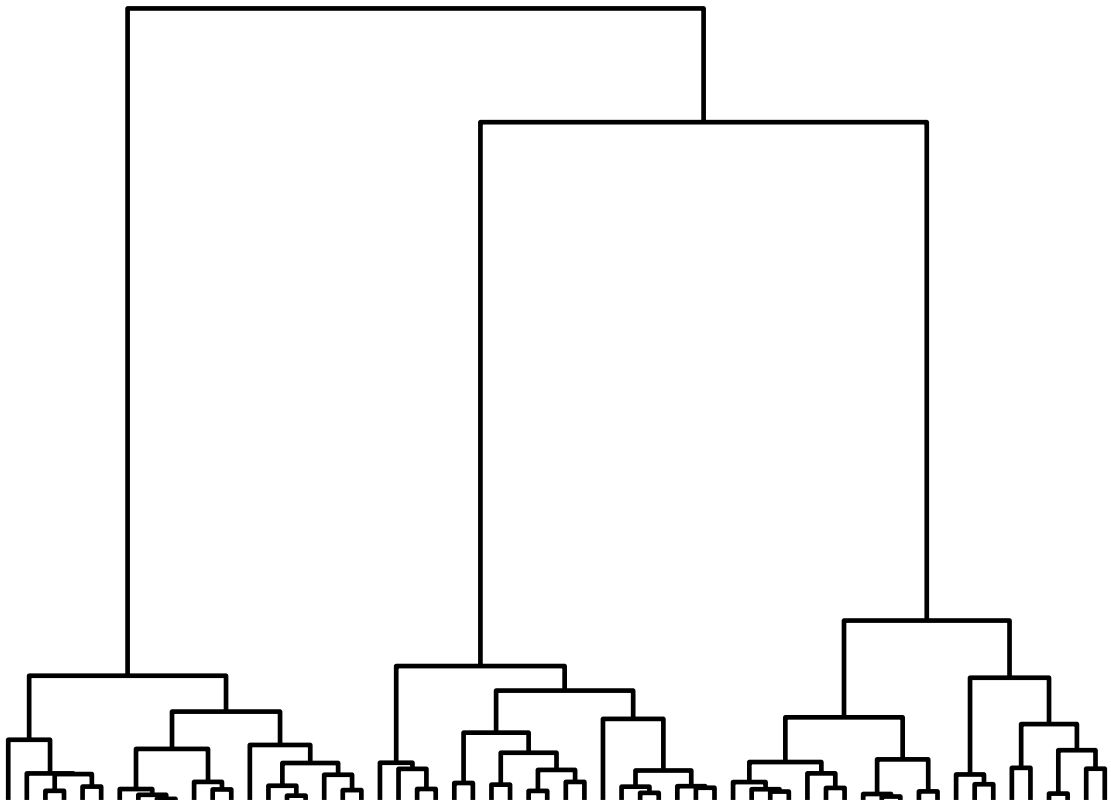
Agglomerative hierarchical clustering

Agglomerative clustering

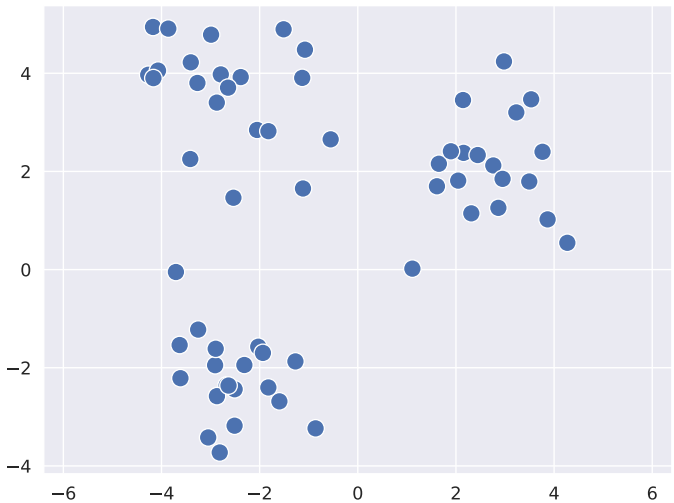
- Agglomerative clustering is an iterative algorithm that produces a **cluster hierarchy** for a dataset $\mathcal{D}$
- Initially, there is one cluster per observation
- The following steps repeat until one cluster remains:
  - The **dissimilarity** between every pair of clusters is computed
  - The **most similar** pair of clusters is **merged** into a single cluster
- The resulting cluster hierarchy may be represented by a **dendrogram**

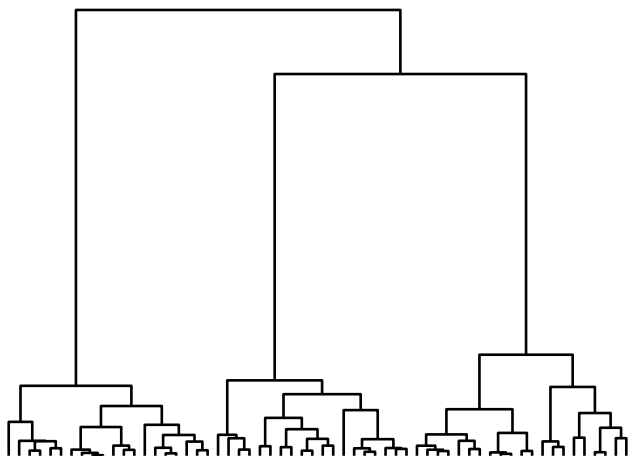
Dendrogram

- In a dendrogram, each **vertical line** represents a cluster and each **horizontal line** splits a cluster.

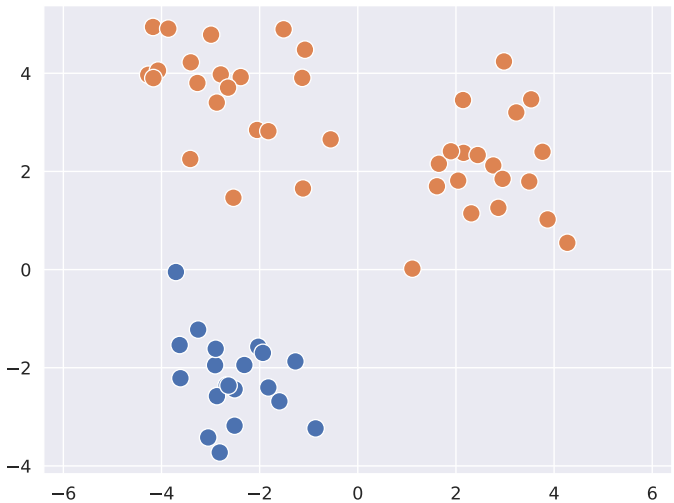


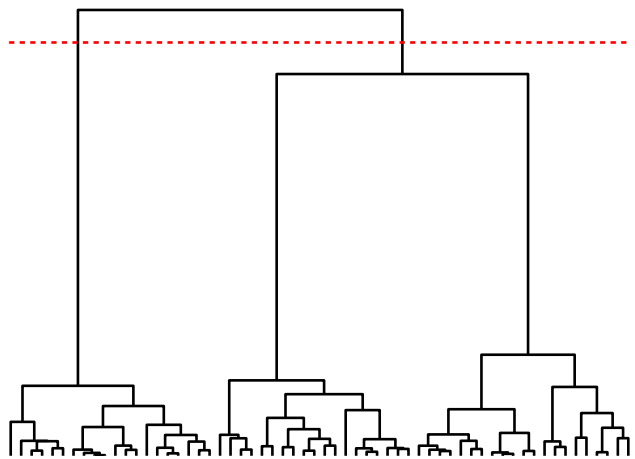
Dendrogram cutting: one cluster



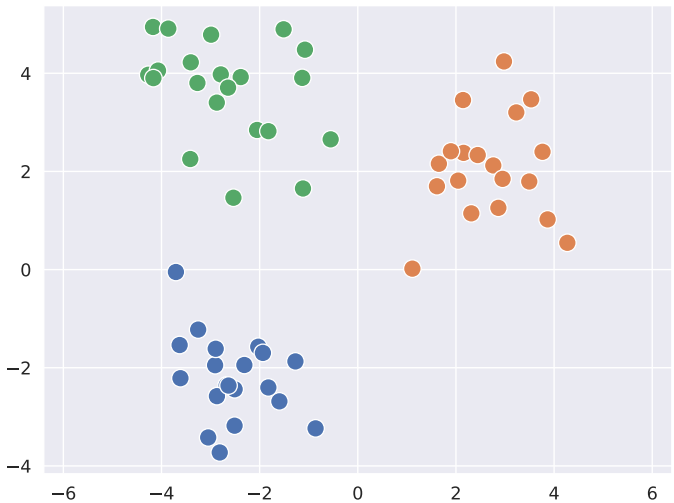


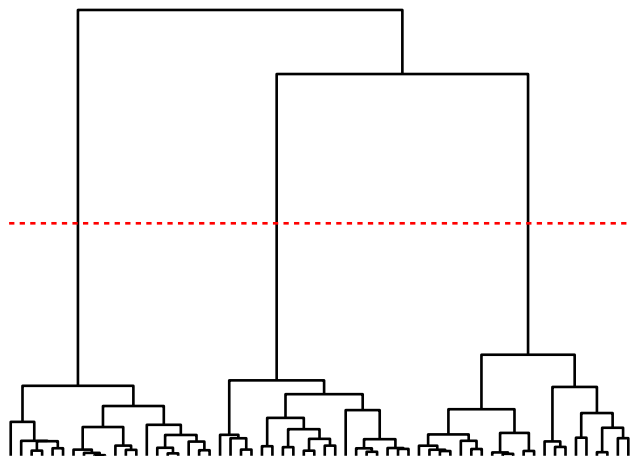
Dendrogram cutting: two clusters



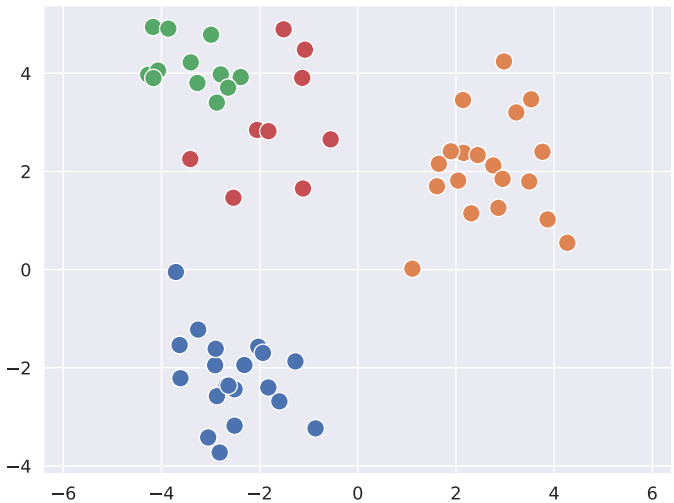


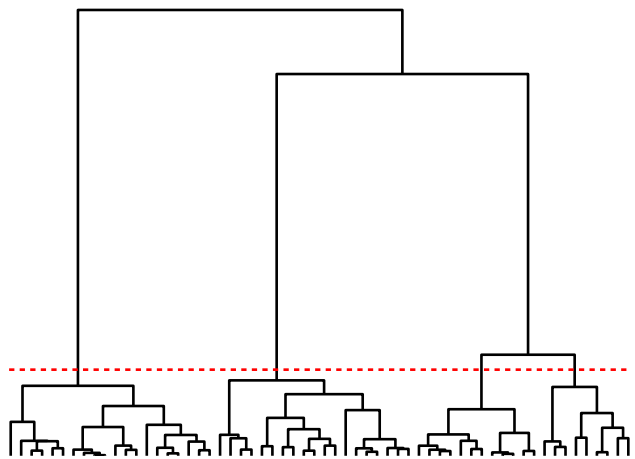
Dendrogram cutting: three clusters





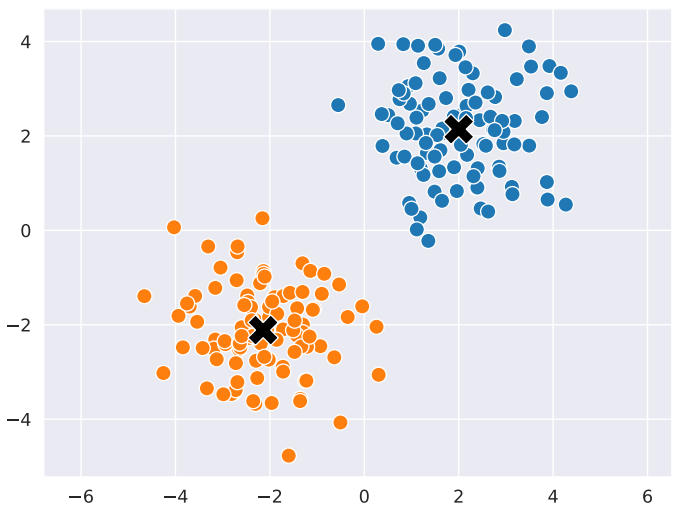
Dendrogram cutting: four clusters

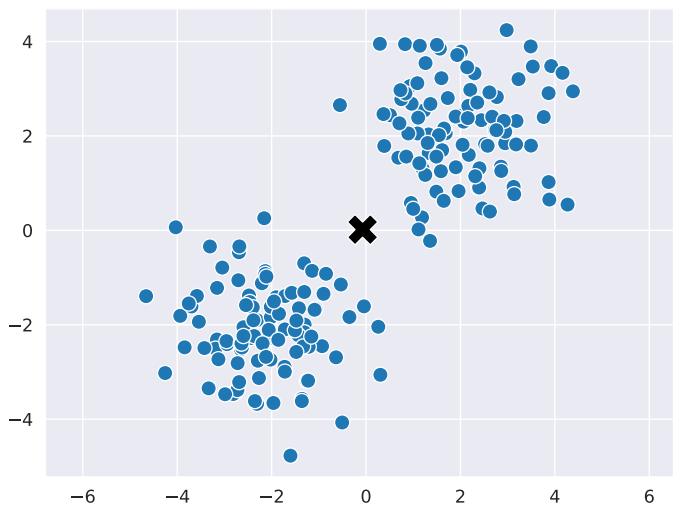




Cluster dissimilarity: Ward's method

- Ward's method is a criterion applied in hierarchical cluster analysis. The method constructs a dendrogram and recursively merges pairs of clusters that minimally increase the within-cluster variance.





Questions?
also please use the forum on QM+