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

ECS766 Data Mining
Week 5: Features and dimensionality

Dr Jesús Requena Carrión

23 Oct 2019

Queen Mary
University of London

Agenda

Recap (with some extras)

Data normalisation

Dimensionality reduction

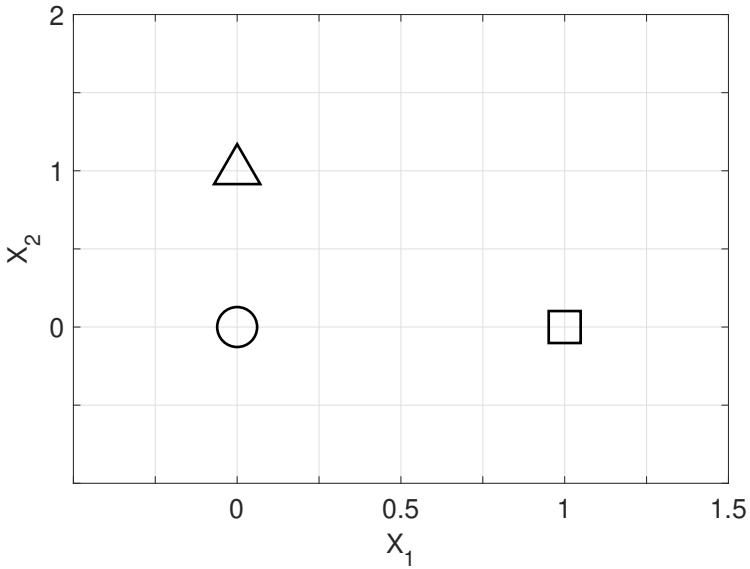
Distances in the predictor space

So far, we have used the notion of **distance** in various occasions:

- In **regression** problems, to define the prediction error $e_i = y_i - \hat{y}_i$ and the MSE cost function, $E_{MSE} = \frac{1}{N}\sum_{i=1}^{N} e_i^2$
- In **classification** problems we used the distance between samples and classifiers'boundaries, and the distance between samples in kNN
- In **clustering** problems, clusters were created based on the square distance between samples and cluster centres, $E_{KM} = \sum |x_i - \mu_i|^2$

The notion of distance is quite intuitive, but is it as straightforward as it seems?

Distances in the predictor space



Which sample is closer to ◯: Is it △ or is it □?

(a) △ is closer

(b) □ is closer

(c) Both are equally distant

Sensitivity to predictors

In a linear regression model, the numerical value of a coefficient indicates how sensitive a prediction is to changes in the value of the corresponding predictor.

In the following linear regression model for a response y:

$$y = \boldsymbol{w}^T \boldsymbol{x} = 3 + 100x_A + 20x_B$$

x_A and x_B are two predictors, $\boldsymbol{x} = [1, x_A, x_B]$ is the extended predictor vector and $\boldsymbol{w} = [3, 100, 20]$ is the coefficient (or parameter) vector.

(a) The response y is more sensitive to x_A than to x_B

(b) The response y is more sensitive to x_B than to x_A

(c) We have insufficient information to tell

Linear separability

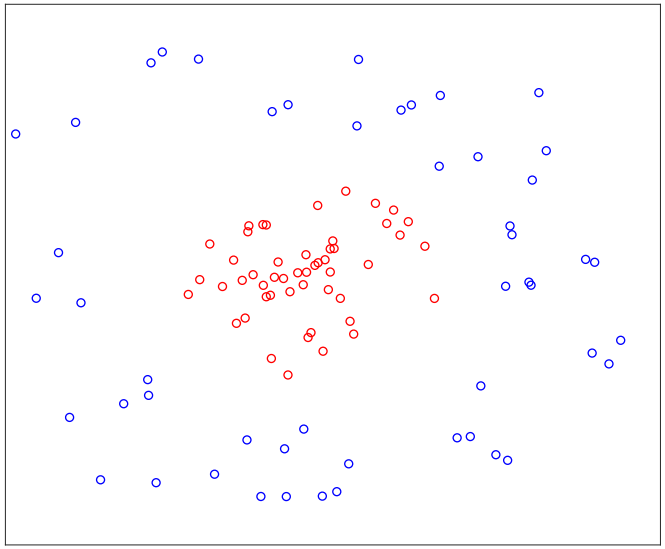
Which dataset is linearly separable, D_1 or D_2?

(a) D_1 is linearly separable, D_2 isn't

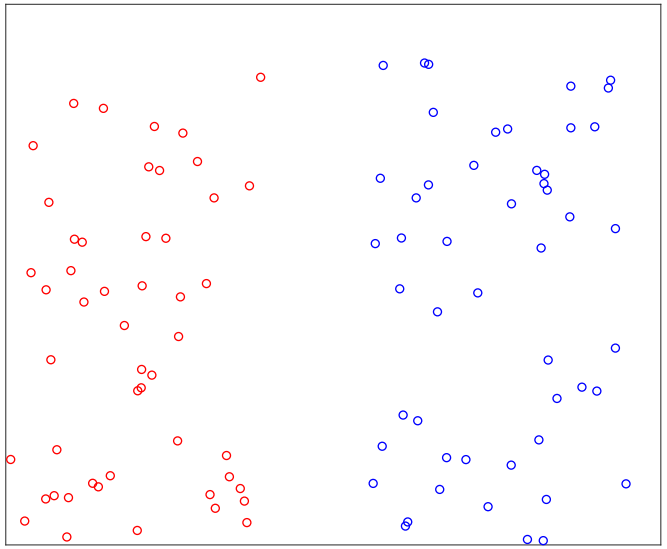
(b) D_2 is linearly separable, D_1 isn't

(c) Both are linearly separable

D_1



D_2



Don't take your representation for granted!

Agenda

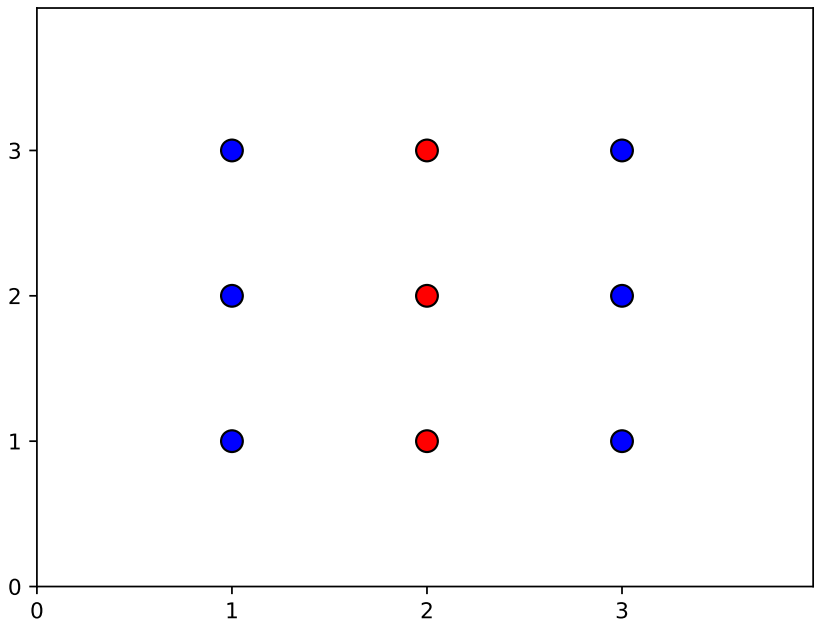
Recap (with some extras)

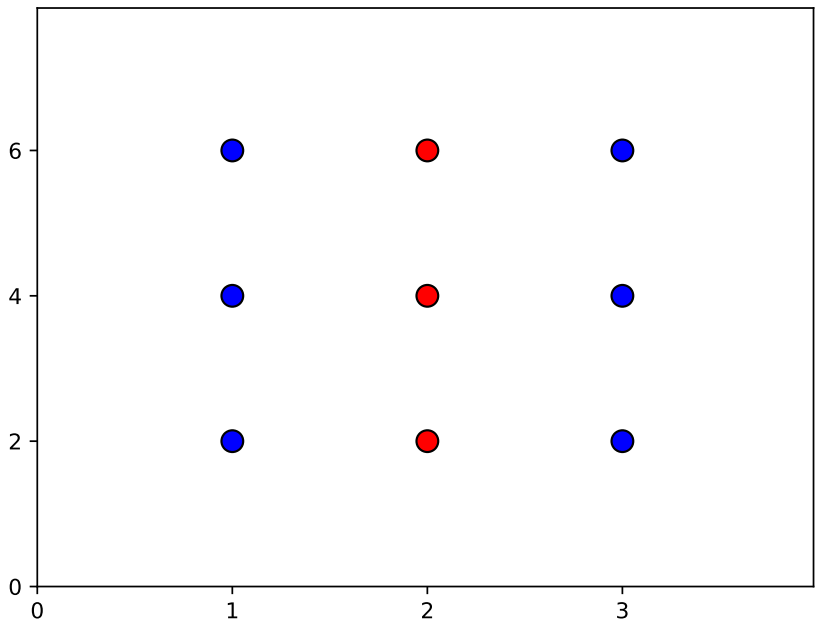
Data normalisation

Dimensionality reduction

kNN and scaling

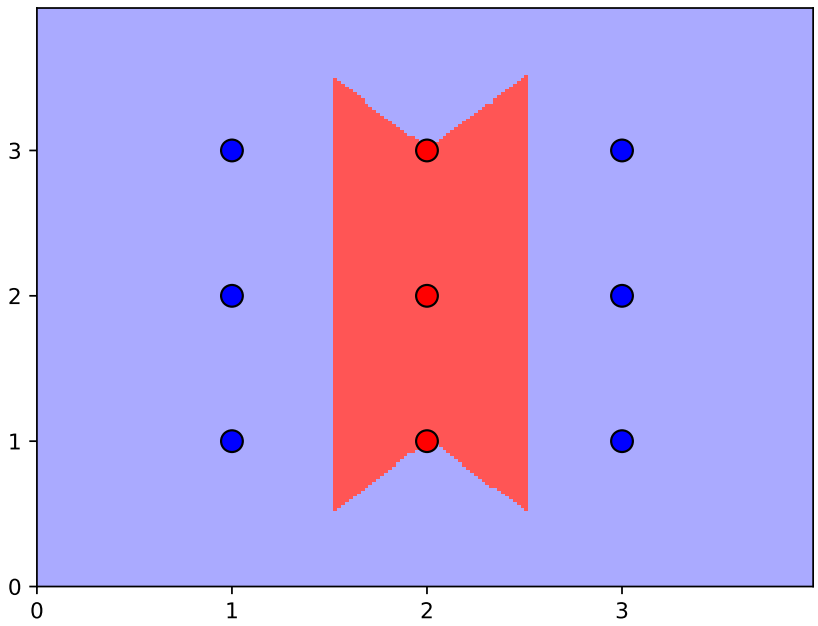
Two identical datasets (except for a scaling factor)

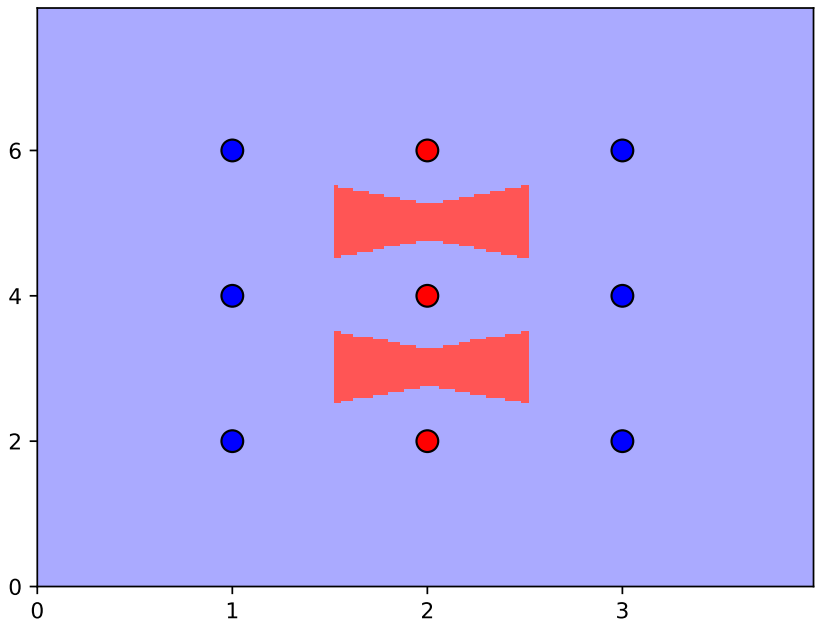




kNN and scaling

kNN solutions ($k = 3$) for each dataset





Numerical representation of attributes

Most of the time, we have ignored the meaning of the attributes under discussion, as our goal has been to discuss general methods that can be applied to any dataset. Hence, we have simply used the **numerical values** of the attributes without much thought. However:

- Attributes can be **incommensurable**, i.e. have different dimensions (for instance, *weight* and *height* cannot be compared)
- Even when attributes have the same dimensions, they might have **different dynamic ranges**
- Having large numerical values **doesn't translate to higher significance**
- Different numerical representations can have an impact on the **final model** and the **performance of our algorithms**

Numerical representation of attributes

The numerical representation of our attributes can be arbitrary and it is possible to find many **equivalent ways of representing numerically each attribute**. So which one is the **most convenient**?

Attributes whose numerical values vary within the same **numerical range** can offer a number of benefits, for instance if the predictors x_A and x_B in the linear model

$$y = \boldsymbol{w}^T \boldsymbol{x} = 3 + 100x_A + 20x_B$$

take on values within the same numerical range, then it makes sense to say that the impact of x_A on the response y is higher than x_B.

Data normalisation (aka *feature scaling*) **techniques** allow us to obtain a **convenient numerical representation** of our attributes.

Min-max normalisation

This technique produces numerical values within the same range $[0, 1]$. In other words, the numerical value of an attribute will always be greater (or equal) than 0 and less (or equal) than 1.

Min-max normalisation produces a normalised attribute x' from the original attribute x by using the following transformation:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

where $\min(x)$ and $\max(x)$ are respectively the minimum and maximum value of x **in the available dataset**.

Notice that x' is a **dimensionless quantity**.

Standardisation

Standardisation is a common procedure in statistics. By applying the following transformation

$$x' = \frac{x - \mu}{\sigma}$$

where μ is the average of the values of x in the available dataset and σ is its standard deviation, the resulting attribute x' is such that the mean of its values in the available dataset is 0 and the standard deviation is 1.

Once again, x' is a **dimensionless quantity**.

Final notes

- Datasets contain samples of a population. During deployment we should expect **out-of-range** values (e.g. $x' = 1.2$ in min-max)
- The effect of **outliers** need to be considered (for instance, an outlier 100 times larger than the second largest value would squeeze the remaining min-max values within the interval $[0, 0.01]$)
- In addition to linear transformations, other non-linear methods exist, for instance **softmax scaling**, which uses the logistic function
- The **distribution of an attribute** can also be normalised
- In general, we need to understand well the **effects and distortions of any data transformation**

Agenda

Recap (with some extras)

Data normalisation

Dimensionality reduction

The Bosch Production Line Dataset

kaggle | Search kaggle | Competitions | Datasets | Kernels | Discussion | Learn

BOSCH

Bosch Production Line Performance

Reduce manufacturing failures

$30,000 · 1,373 teams · 2 years ago

Overview | Data | Kernels | Discussion | Leaderboard | Rules | Late Submission

Overview

- Description
- Evaluation
- Prizes
- Timeline
- Ieee Bigdata 2016

A good chocolate soufflé is decadent, delicious, and delicate. But, it's a challenge to prepare. When you pull a disappointingly deflated dessert out of the oven, you instinctively retrace your steps to identify at what point you went wrong. Bosch, one of the world's leading manufacturing companies, has an imperative to ensure that the recipes for the production of its advanced mechanical components are of the highest quality and safety standards. Part of doing so is closely monitoring its parts as they progress through the manufacturing processes.

The MNIST dataset

Data dimensionality

In Data Science, an attribute can be seen as a **dimension of our datasets**. This interpretation allows us to **represent instances as points in the space**, by using the values of each of its attributes as coordinates.

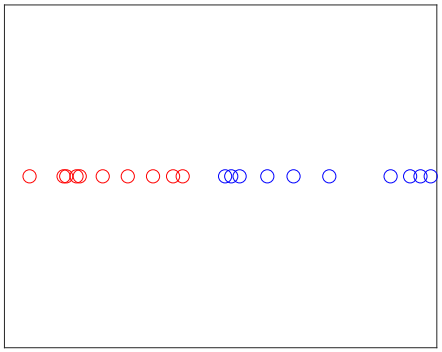
Some datasets may include many attributes (for instance, the Bosch dataset contains 970), and are therefore said to be **high dimensional**. Frequently, this is due to us having **little prior knowledge**, which forces us to record everything (just in case!).

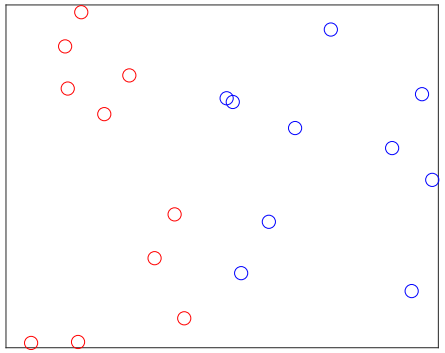
The question arises, what are the **main challenges of high dimensional datasets**? Is it wise to include as many attributes as we can?

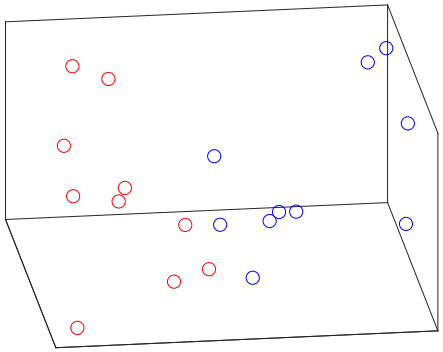
The Curse of Dimensionality

As we increase the dimensionality of our dataset, **data becomes sparser**.

In this example, we add two irrelevant attributes to 20 samples that are initially described by one single attribute. How would a logistic regression boundary change as we include new irrelevant attributes? And a kNN one?

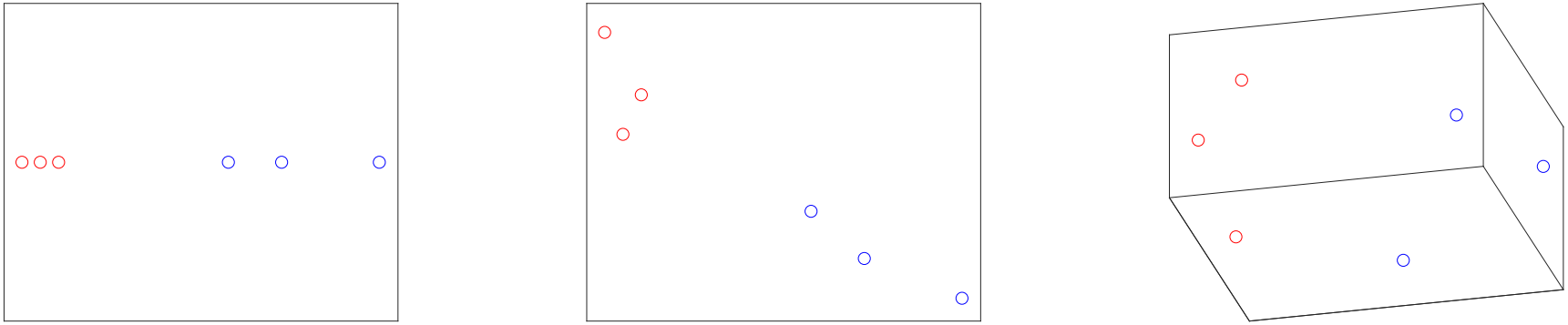






The Curse of Dimensionality

Now we only use 6 samples. Compare their boundaries: It is much clearer that adding new irrelevant attributes can make things worse!



In a high dimensional settings, we need to learn more parameters than in a low dimensional ones, so the **risk of overfitting** increases. This risk is specially dangerous if we have many attributes that are weakly relevant, or some very relevant and many irrelevant.

Dimensionality reduction

High dimensional datasets present many challenges, including:

- Overfitting (curse of dimensionality)
- Irrelevant data
- Computational cost
- Storage cost
- Hard to visualise
- Difficult interpretation

Dimensionality reduction is a family of techniques who goal is to **transform a high dimensional dataset into a more convenient low dimensional dataset**. Two main approaches are:

- Feature selection → *pick* the best predictors
- Feature extraction → *transform* onto a smaller set of predictors

Agenda

Recap (with some extras)

Data normalisation

Dimensionality reduction

- Feature selection
- Feature extraction

Feature selection

In feature selection, we start by wondering whether not all our predictors (a.k.a *features*) in our data set are relevant. Therefore we want to be able to select the **best** ones, in other words, we want to identify the **best subset of predictors**.

This leads to two observations:

- What do we mean by **best**? We will use the **response** to create metrics for subset selection (**supervised problem**!).
- If our dataset has M predictors, there are a **total of** $2^M - 1$ **subsets** that we could consider.

If our dataset has 10 features, we have roughly 1000 options. In the Bosch dataset we have a few more than 10^{270} options. How do we find the best subset?

Filtering

The simplest approach towards feature selection is to consider each one of the features **individually** and assign them a **score**, which we use to **rank** them and **select** the N best ones.

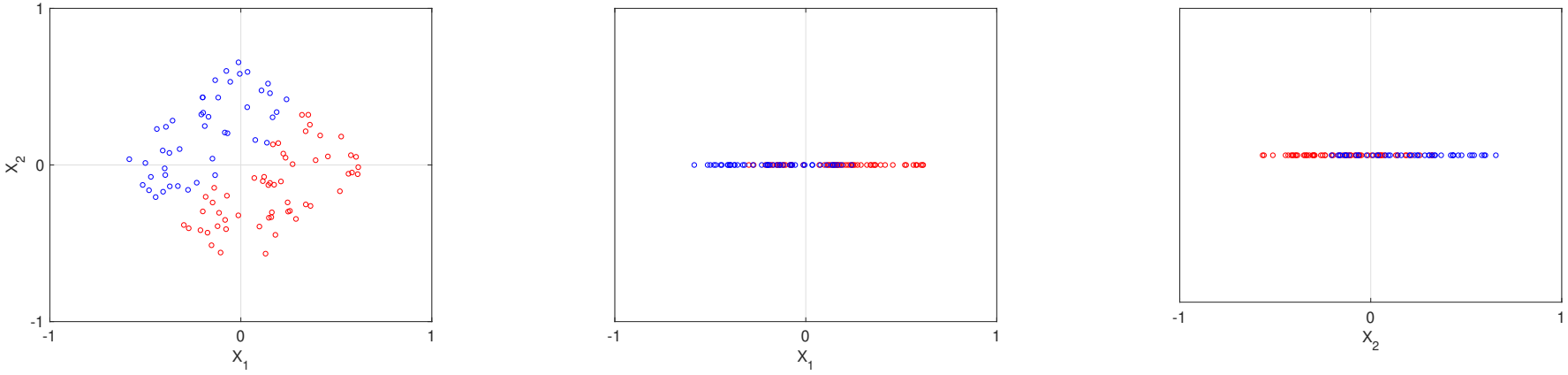
Scores essentially **compare each feature with the desired response**. Common scores include:

- Correlation.
- Mutual information.
- Statistical independence.

Filtering

Filtering is a **simple** and **fast** method. However, its starting point is that the features in the dataset contribute separately to the final response, therefore possible **interactions among predictors are ignored**.

In the following example, predictors X_1 and X_2 do a poor job separately, but together they reveal a clear boundary between the two classes.



Wrapping

If we suspect that the interaction between predictors might be crucial, we have no choice but to **evaluate them together**, rather than separately.

Wrapping approaches consider possible interaction between predictors by:

- Training a model with **different subsets of predictors**
- Evaluating each resulting model by using **validation approaches**.
- Picking the subset with the **highest validation performance**.

Whereas filtering approaches retrain a model M times (where M is the number of predictors), wrapping models can potentially consider up to $2^M - 1$ predictors! The **computational cost** is, therefore, a big concern.

Wrapping: Greedy search

Given the large number of subsets of predictors that we might need to consider, the main challenge when implementing wrapping approaches is how to **search for the best subset**.

In general, considering all the candidate subsets (known as **brute-force** or **exhaustive** search) will be impractical. Greedy search is a strategy for exploring candidate subsets. It comes in two flavours:

- **Forward selection**: We start with a subset containing the best predictor and progressively add the predictor that improves the subset's performance the most.
- **Backward selection**: We start with a subset containing all the predictors and progressively remove the predictor whose elimination improves the performance the most.

In both cases we implement a stop criterion (typically, when the performance stops increasing).

Embedded selection

Some learning algorithms can have some form of feature selection built in the process of learning.

- **Classification trees** can be grow following a strategy that effectively eliminates irrelevant features.
- Properly designed **regularisation strategies** can be seen as a feature selection process, as by attenuating the coefficient w_A corresponding to a predictor x_A, this predictor is eliminated. We have used the so-called L_2 regularisation by using the term $\lambda \boldsymbol{w}^T \boldsymbol{w}$, but other options are available too.

Agenda

Recap (with some extras)

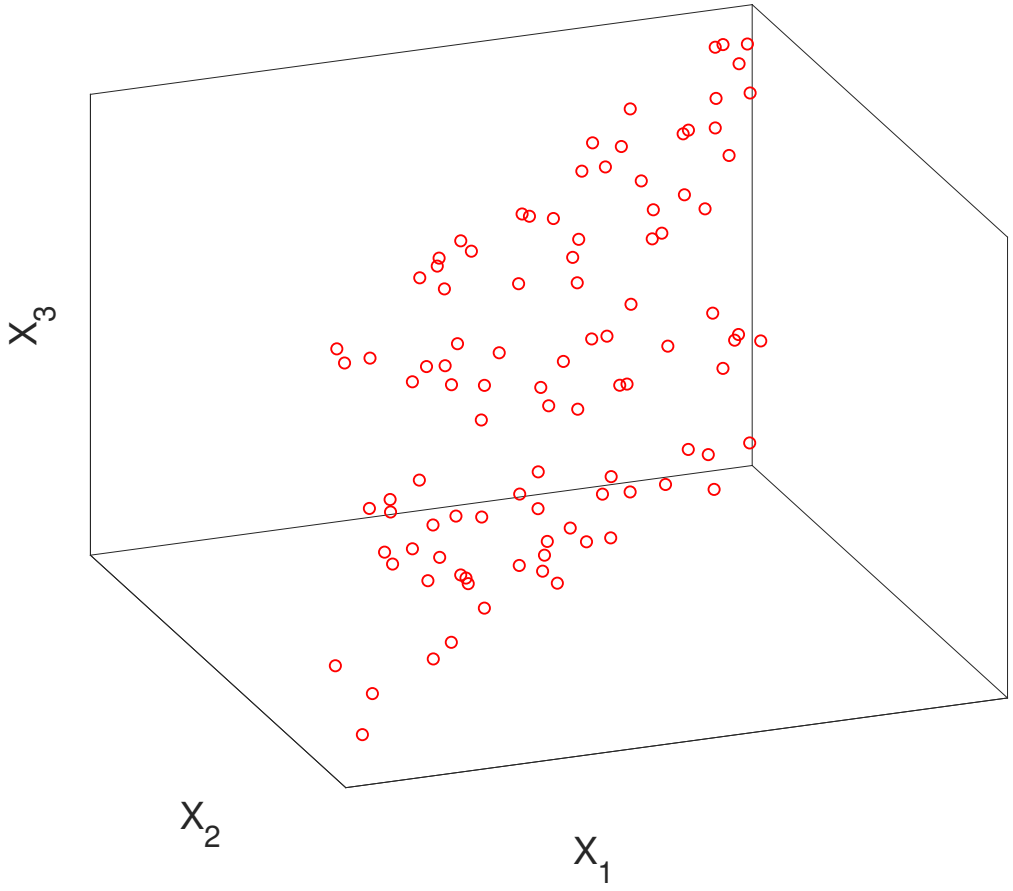
Data normalisation

Dimensionality reduction

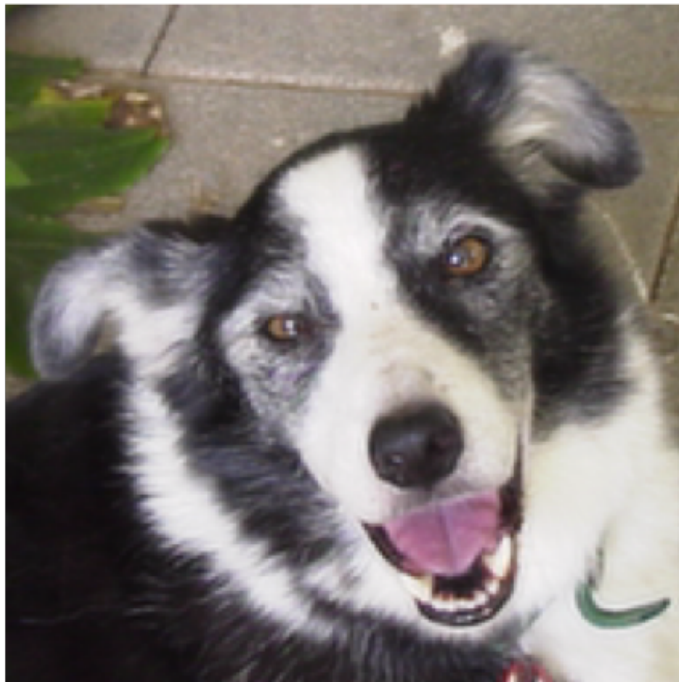
- Feature selection
- Feature extraction

Interesting directions

A dataset with predictors X_1, X_2 and X_3 is represented below. You can move freely in this 3D space. Which way would you go?



What is feature extraction?



This picture consists of $422 \times 424 \approx 180,000$ pixels.

- Do we need 180,000 predictors to tell it's a dog?
- Would a subset of pixels work?
- Can we transform the picture into a new set of predictors?

What is feature extraction?

Feature extraction allows us to reduce the dimensionality of our dataset by creating a **new set of predictors** of lower dimensionality. The new predictors are **non-redundant** and overall should **as much information from the original set** as possible.

There exist many **data, signal and image processing techniques** that allow us to define new features that can be extracted from high-dimensional data, for instance:

- Frequency components of signals (Fourier analysis)
- Texture characterisation of images (wavelet analysis)

Principal Components Analysis is one of the most popular and well understood techniques for dimensionality reduction via feature extraction.

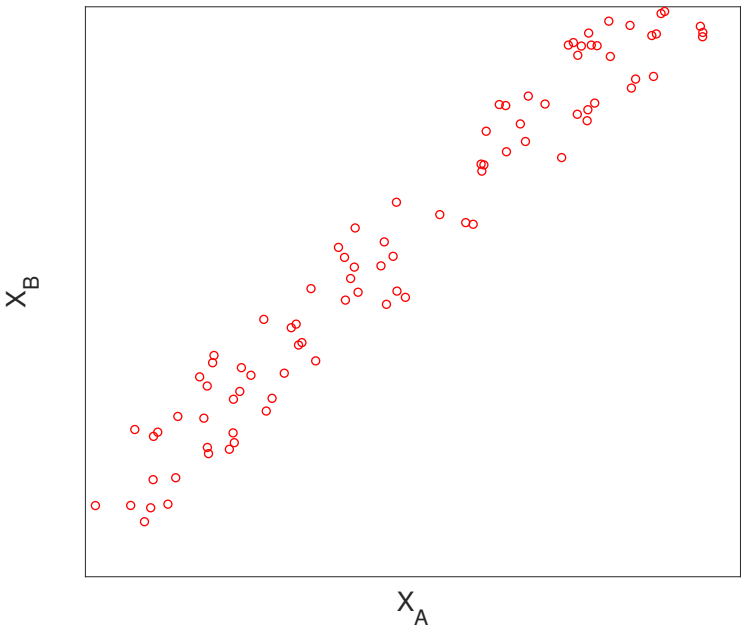
Principal Component Analysis

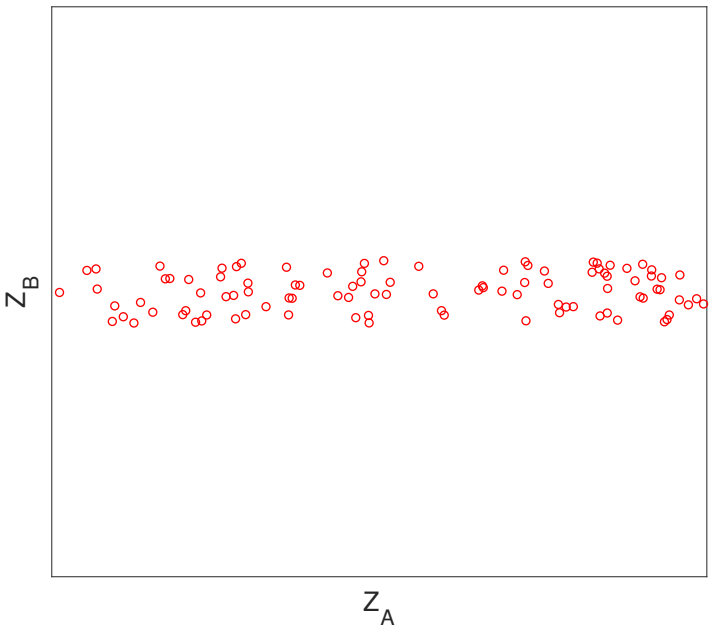
Given a set of N features $x_A, x_B, \ldots$, Principal Component Analysis (PCA) produces:

- Another set of N **orthogonal** new features $z_A, z_B, \ldots$ (in other words, they are non-redundant)
- That can be **combined** to produce the original features (therefore both set of features have the same information)
- Along with a **score** that can be used to rank the new features and select them

PCA doesn't use the desired response to create this transformation, and therefore it is an **unsupervised approach**. How does it work then?

PCA as a projection





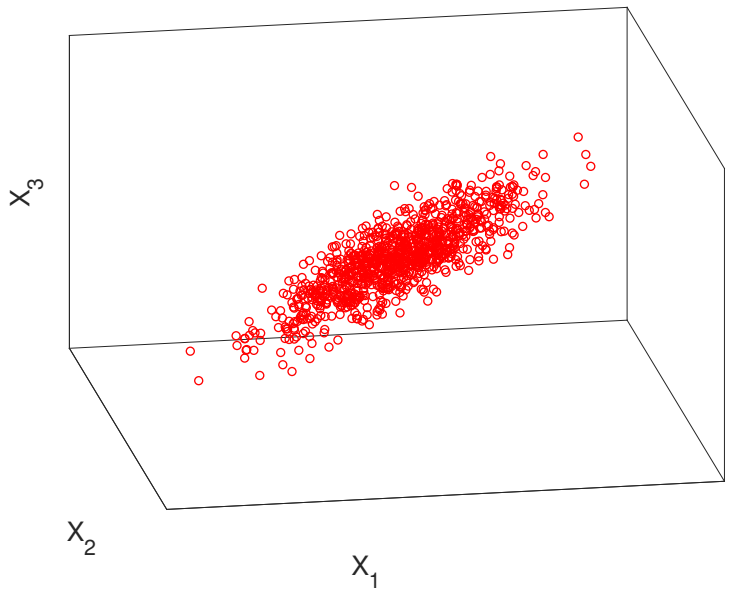
PCA principles

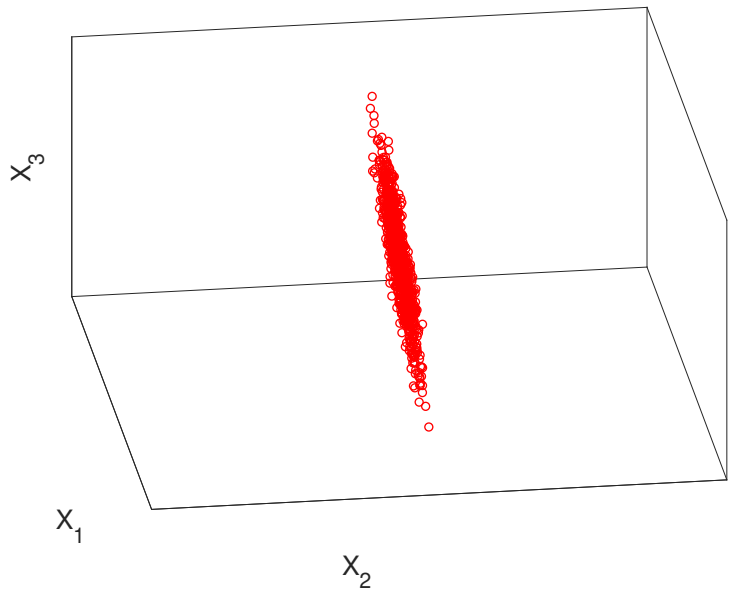
Our starting point is this: **directions along which data varies a lot, contain a lot of information**, and conversely, directions along which data varies little, contain little information.

Our PCA algorithm proceeds as follows:

- We find the direction along which **data varies the most**
- This direction is our first **new feature**
- By subtracting the new feature from our original data, we obtain a residual containing **unexplained** variance
- We extract a new feature and residual from any residual previously obtained in the previous step until we have **exhausted all the dimensions**

PCA principles





Final notes on PCA

PCA is a **linear transformation** of our dataset: the new features $\boldsymbol{z}$ can be obtained by multiplying the original features $\boldsymbol{x}$ by a combination matrix U, $\boldsymbol{z} = U\boldsymbol{x}$. PCA is widely implemented and computationally fast. However:

- It is **not scale-invariant**
- It assumes **variations are gaussian**, and therefore might ignore non-gaussian patterns.
- **Non-linear scenarios** are not accounted for
- Information might be in **low-variance components**