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

Machine Learning Methodology II

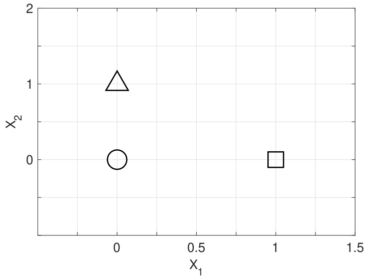
Dr Chao Liu

Nov 2023

Credit to Dr Jesus Requena

Queen Mary University of London

Distances in the attribute space



Which sample is closer to ◯, △ or □?

(a) △ is closer

(b) □ is closer

(c) Both are equally distant

Linear separability

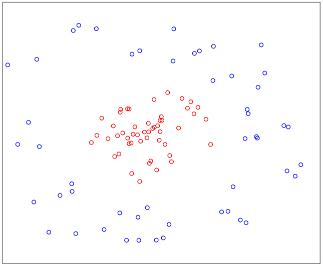
Which dataset is linearly separable, D_1 or D_2?

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

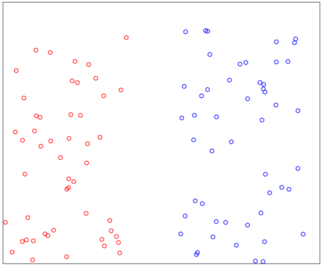
(b) D_2 is linearly separable, D_1 isn't

(c) Both are linearly separable

D_1



D_2



Don't let appearances fool you!

Agenda

Machine learning: More than models

Normalisation

Transformations

Ensembles

Summary

What is a pipeline?

The term pipeline describes a **sequence of operations**. A (supervised) machine learning pipeline is the sequence of operations that produce a prediction (**output**) using a set of predictors (**input**).

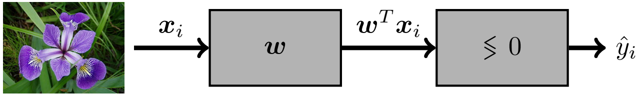
The following pipeline implements a binary linear classifier model defined by a set of coefficients $\boldsymbol{w}$:

$$\begin{tikzcd}
\boldsymbol{x}_i \arrow[r] & \boxed{\boldsymbol{w}} \arrow[r, "\boldsymbol{w}^T\boldsymbol{x}_i"] & \boxed{\lessgtr 0} \arrow[r] & \hat{y}_i
\end{tikzcd}$$

Note that the term **pipeline** is often used to describe **workflows**. We use the term workflow to describe a **sequence of steps that we take**, e.g. we first formulate a problem, then collect data, select a model, train it...

Rich inputs, high risks

Consider a collection of RGB (*red*, *green* and *blue*) photos of setosa and virginica specimens. Can we build a a linear classifier that distinguishes setosa from virginica flowers?

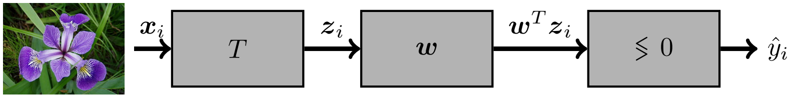


If each RGB picture consists of $A \times B$ pixels:

- The model takes as input $3 \times A \times B$ values (input dimensionality).
- The coefficients vector $\boldsymbol{w}$ consists of $3 \times A \times B + 1$ values.
- To avoid overfitting we need more than $3 \times A \times B + 1$ training pictures.

Few, meaningful features

We can extract features from each picture (e.g. sepal length and width, petal length and width) and build a linear model that takes these features as input.

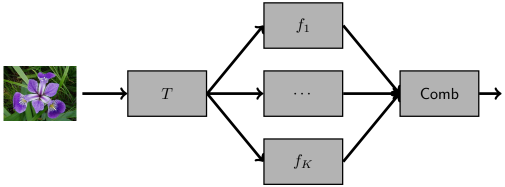


- Our linear model takes 4 values as input.
- The entire machine learning pipeline consists of a feature extraction stage (T) followed by a linear model.
- Both feature extraction stage and model are deployed.
- The final quality depends on the feature extraction stage.

Pilelines, not models

Machine learning solutions are more than just one model. They can include several stages that form a **processing pipeline**:

- Transformation stages (where input data is processed).
- Several machine learning models running in parallel.
- A final aggregation stage (where individual outputs are combined to produce one single output).



We deploy the entire pipeline, not just individual models.

Hand-crafter or trained?

Machine learning models are always built using data (otherwise it's not machine learning!). The other stages in a machine learning pipeline can be either **hand-crafted** or **trained**.

Note that:

- After training, the parameters of a pipeline remain **fixed**.
- Pipelines can be tested and deployed.
- The quality of the prediction depends on all the pipeline stages.

Agenda

Machine learning: More than models

Normalisation

Transformations

Ensembles

Summary

Distances in the attribute space

The notion of **distance** is behind many machine learning techniques:

- In **regression**, the prediction error $e_i = y_i - \hat{y}_i$ can be seen as a distance.
- In **classification**, we used the distance between samples and boundaries (logistic regression), and between samples (kNN).
- In **clustering**, K-means clusters were created based on the distance between samples and prototypes.
- In **density estimation**, the standard deviation quantifies the average distance between samples and the sample mean.

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

Innocent numerical values?

So far we have ignored the meaning of the attributes of our dataset and have simply used their **numerical values**. However, attributes:

- Can be represented using different **units** (e.g. miles or km).
- Can be **incommensurable**, i.e. have different dimensions (e.g., one attribute might represent a *weight* and another a *height*).
- Might also have **different dynamic ranges** (e.g. one attribute might have values in the range of cm, another km).

The numerical representations of our data can have an impact on the **final model** and the **performance of our algorithms**.

Equivalent numerical representation

There are **equivalent** ways of representing numerically one attribute. So which one is the **most convenient**?

Having attributes whose values vary within the same **numerical range** can be beneficial, for instance if the predictors x_A and x_B in the model

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

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

Data normalisation is a **tunable transformation** stage that allows us to scale attributes so that their values belong to similar ranges.

Min-max normalisation

Min-max normalisation produces values within the same continuous range $[0, 1]$, i.e. greater (or equal) than 0 and less (or equal) than 1.

A normalised attribute z from the original attribute x is obtained using the following transformation:

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

$\min(x)$ and $\max(x)$ are the **parameters of this stage** and correspond the minimum and maximum value of x **in the available dataset**.

Standardisation

Standardisation is a common procedure defined by the transformation

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

where μ is the average of x and σ its standard deviation in the dataset. They are the **parameters of this stage**.

The resulting attribute z has **0 mean** and **unit standard deviation**. Standardisation is very common, e.g. in neural networks, as it ensures inputs are treated equally.

Normalisation: Observations

- We use a **dataset** to set the parameters of a normalisation stage. These values are used during test and deployment.
- During test and deployment we should expect **out-of-range** values (e.g. $z = 1.2$ in min-max).
- **Outliers** can have a negative impact (e.g., an outlier 10 times larger than the second largest value will squeeze min-max to $[0, 0.1]$).
- Non-linear scaling options exist, for instance **softmax** scaling, which uses the logistic function, or **logarithmic** scaling.
- The original values in your dataset might be relevant. In general, we need to consider unintended **effects and distortions**.
- Many machine learning algorithms might produce **different solutions** after scaling.

Agenda

Machine learning: More than models

Normalisation

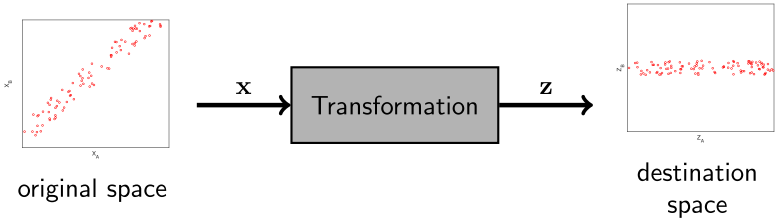
Transformations

Ensembles

Summary

What is a transformation?

Transformations are data manipulations that change the way that we represent our samples. They can be seen as moving samples from one space to another.



Normalisation is one example of a transformation that operates on each attribute separately: it should be included as another stage in our pipeline.

Types of transformations

There are many types of transformations. In some transformations, the original and destination spaces have the **same number of dimensions**.

- A **linear** transformations can be seen as a rotation and scaling.
- There is no unique description for **non-linear** transformations.

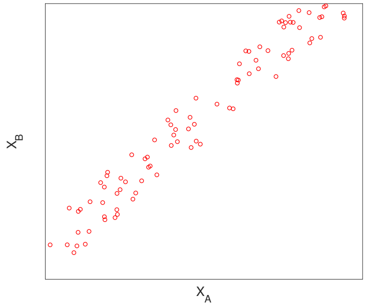
If the destination space has fewer dimensions than the original one, we talk about **dimensionality reduction**.

- In particular, after **feature selection** the destination space is defined by a subset of the original attributes.
- In **feature extraction** the new attributes are defined as operations on the original attributes. Common when using complex types.

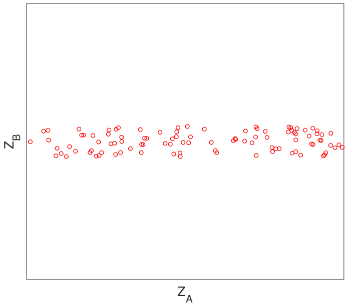
Linear transformations: PCA

Principal components analysis (PCA) identifies the **directions** along which samples are aligned. These directions define a destination space with the same number of dimensions as the original space.

Using a dataset, PCA builds a **linear transformation** and additionally assigns a **score** to each component.



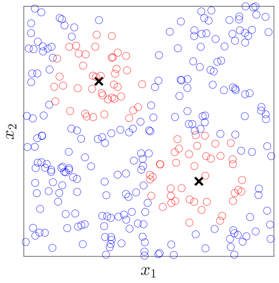
Original space



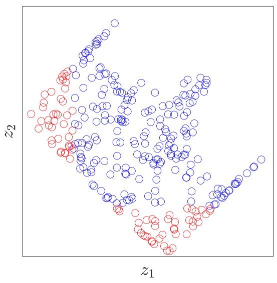
Destination space

Non-linear transformations

In this example, we map a dataset to a space where each new attribute corresponds to the distance between the sample and a centre ($\times$).



Original space



Destination space

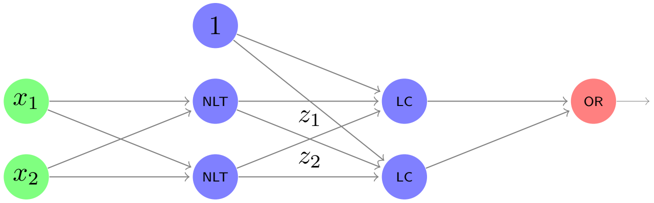
The dataset has two classes that are not linearly separable in the original space. In the destination space, we can use two linear boundaries.

Non-linear transformations

A solution for the previous example could be a pipeline consisting of:

- A suitable **non-linear transformation**.
- Followed by a second layer of **linear classifiers**.
- A final unit implementing a **logical function**.

This pipeline produces circular boundaries in the original space.



(*Note that this pipeline looks like a* **network**)

Complex models and kernel methods

Many complex machine learning models can in fact be interpreted as a transformation followed by a simple model.

There are two scenarios:

- We know how to transform our data and only need to learn the model that operates on the destination space.
- We don't know the transformation, hence we need to learn it too. This can involve **selecting** the right transformation (via validation) or **tuning** the parameters of a given transformation (via training).

Kernel methods, such as support vector machines, implicitly define such transformations using so-called kernel functions.

Dimensionality reduction

PCA defines a **linear transformation** between two spaces that have the the same number of dimensions (i.e. the same number of attributes). In addition, PCA assigns a **score** to each dimension of the destination space.

The score provided by PCA can be used to **rank** the attributes defining the destination space and remove the least important ones, resulting in a reduction of the dimensionality of the dataset.

Machine learning models can then be built that take as an input samples represented in a lower dimensionality space.

Dimensionality reduction

Feature selection is a method to reduce the dimensionality of a dataset that assumes that only a subset of the original attributes are relevant.

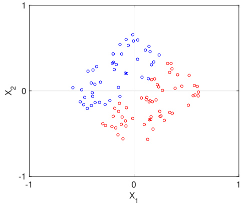
To select the most relevant features, we need to be able to assign a score to different subsets of features:

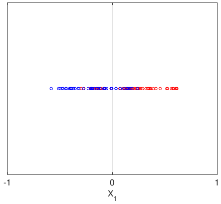
- If our dataset has M attributes, there are a **total of** $2^M - 1$ **subsets** that we could consider (e.g., If our dataset has 10 attributes, we have roughly 1000 options).
- What do we mean by **relevant**? In supervised learning, we can use our target metric to evaluate how relevant a subset of attributes is.
- We still need a **model** trained on each subset of features. The final relevance will also depend on our ability to train a model.
- Feature selection can be seen as a form of **validation**, where we select models that use different subset of attributes.

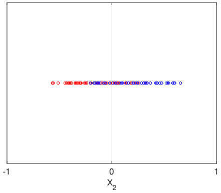
Feature selection: Filtering

The simplest approach to feature selection is to consider each attribute **individually**. A score can be assigned by fitting a model and obtaining its validation performance. Then, the best components are selected.

Filtering is **simple** and **fast**. Possible **interactions** among predictors are however ignored. In the following example, attributes X_1 and X_2 do a poor job separately, but together reveal a clear boundary.







Feature selection: Wrapping

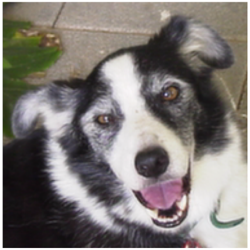
If we suspect that the interaction between features might be crucial, we have no choice but to **evaluate them together**, rather than separately.

Wrapping approaches consider possible interaction between predictors by:

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

Whereas in feature filtering we train M models, where M is the number of features, wrapping can lead to up to $2^M - 1$ options. **Greedy search** can be used to reduce the number of options.

Feature extraction



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

- Do we need 3×180,000 attributes to tell this is a dog?
- Would a subset of attributes work?
- What happens when we change the **format** of the picture?

Feature extraction can reduce dramatically the dimensionality of a dataset summarising it using a few well-designed features.

Digital **signal** and **image processing** provide with a wide range of options to extract features from rich input data types.

Agenda

Machine learning: More than models

Normalisation

Transformations

Ensembles

Summary

Ensembles: Principles

In Machine Learning we have different families of models. So far, our approach has been to use validation to **select the best family**. Can we get them to **work together** instead?

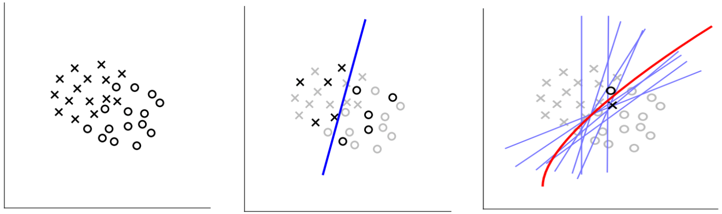
Ensemble methods allow us to create a new model that **combines** the strengths of **base models**. Base models need to be as **diverse** as possible and can be created by training:

- A family of models with **random subsets of the data**.
- Different models with **random subsets of attributes**.
- Different **families of models** altogether.

Ensembles: Bagging

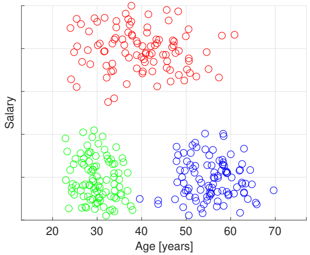
Bootstrap is a statistical method that extracts random samples from a dataset. Given a training dataset, **bagging** generates K sub-datasets by bootstrapping and trains K simple base models with each sub-dataset.

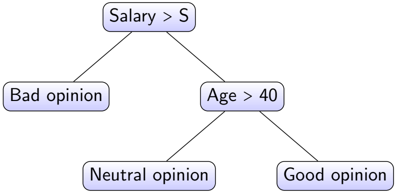
The final model $f(x)$ **combines the predictions of the base models** $f_k(x)$ by averaging or voting, for instance $f(x) = \sum f_k(x)/K$.



Decision trees

Decision tree classifiers partition the predictor space into multiple decision regions by implementing sequences of splitting rules using one **predictor only**. This leads to an algorithm that can be represented as a tree.





(In this dataset o = good, o = neutral and o = bad opinion)

Decision trees

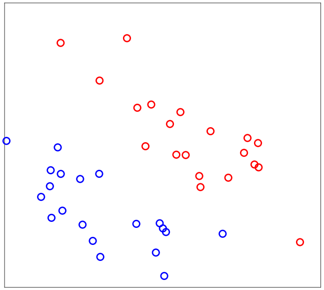
In a decision tree, the **root** corresponds to the whole, unpartitioned dataset and a **leaf** is one of the decision regions.

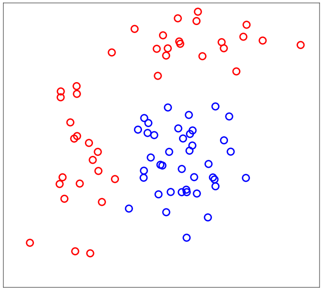
- The goal is to create **pure leaves**, i.e. containing as many samples from the same class as possible.
- During classification, a sample is assigned to the **majority class** in the leaf where the sample is located.

Decision trees are built **recursively**:

- Starting from the root, we recursively **split each region into two**. Splits are **axis-parallel** (decisions using one predictor).
- The chosen split is such that the purity of the resulting regions is higher than any other split.
- We stop when a given criterion is met, such as the number of samples in a region.

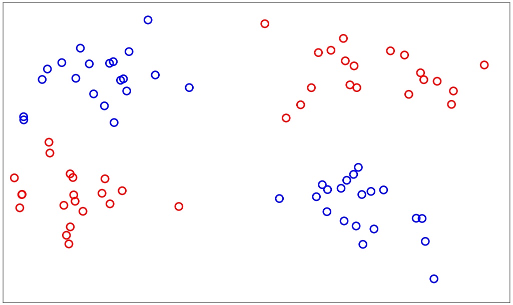
Linear classifier Vs decision tree





The XOR problem

There are problems where a decision tree would be an excellent choice to classify samples, however, existing training algorithms would struggle to find such tree. The XOR problem provides with such an example.



Ensembles: Random forests

Trees are simple and can handle easily both numerical and categorical predictors. However:

- Trees run the risk of memorising training samples, i.e. **overfitting**. Pruning techniques and stop criteria can help to prevent this.
- Different training datasets can lead to a different tree structures.
- Some classification problems can be easily represented as a tree, but the tree might be hard to learn (e.g. XOR).

Random forests are an ensemble of decision trees.

Ensembles: Random forests

Random forests train many individual trees by **randomising** the training **samples** and the **predictors**. Predictions are obtained by averaging the individual predictions.

In general they have great accuracy, but can be expensive to train and are *harder to interpret than a single tree* (this is not my personal view, trees can be as hard to interpret).

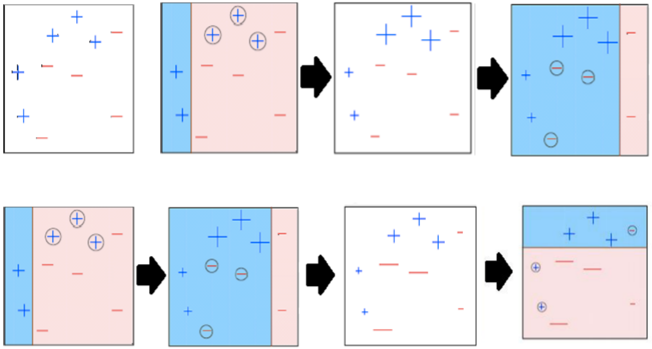
Ensembles: Boosting

So far we have considered ensembles that create base model diversity by using some form of **randomisation**.

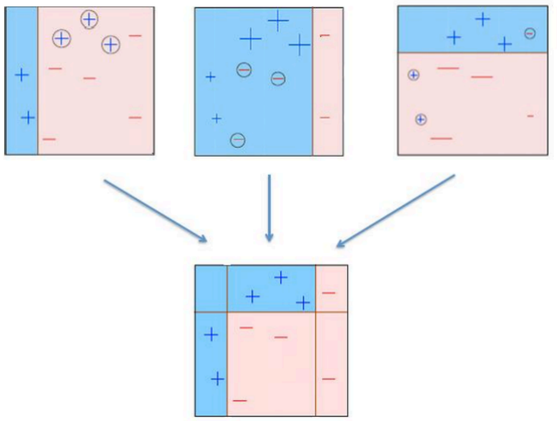
Boosting follows a different approach: it generates a **sequence of simple base models**, where each successive model focuses on the samples that the **previous models could not handle** properly.

Therefore, each new base model generated by boosting depends on the previous models unlike bagging and random forests.

Ensembles: Boosting



Ensembles: Boosting



Agenda

Machine learning: More than models

Normalisation

Transformations

Ensembles

Summary

Deploying machine learning pipelines

- Pipelines define **sequences of operations** on data, including transformations, models and aggregations.
- In machine learning we **deploy pipelines**, not just models.
- In addition to the model, we might need to **learn other stages** in a pipeline using data.
- **Complex machine learning models** can be seen as pipelines, including transformations, simple models and aggregations.

Transformations

- The purpose of a transformation is to use a more **convenient representation** for our data.
- If the destination space has fewer dimensions than the original one, we talk about **dimensionality reduction**.
- **Normalisations** are transformations where each attribute is **scaled individually** so that they all take the same range of values.
- In **feature selection** we select a subset of attributes, whereas in **feature extraction** we produce a new set of attributes by processing the input attributes.
- Once we have designed the transformation stage, it remains **fixed** (same happens with trained models!).

Ensembles

- An **ensemble** is a machine learning strategy that combines the output from individual models.
- The idea is that each model learns **different aspects** of the true underlying model, hence the right aggregation produces better results than each model individually.
- Ensembles can be created by using different subsets of samples, different subsets of attributes or different families of models.