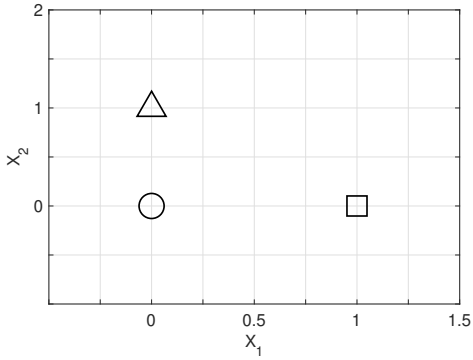
Queen Mary School Hainan
Queen Mary University of London

QHM5703 Principles of Machine Learning Methodology II

Dr Nikesh Bajaj
Dr Jesús Requena Carrión

Week 14: 10/11 Dec 2025

Distances in the attribute space



Which sample is closer to ◯, △ or □?

(a) △ is closer

(b) □ is closer

(c) Both are equally distant

Go to menti.com and use code:

DMT: 2429 5147 — ICS: 1410 4001

Linear separability

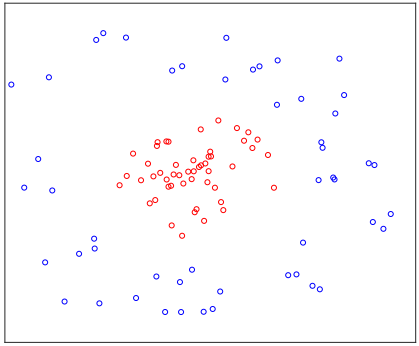
Which dataset is linearly separable, D_1 or D_2?

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

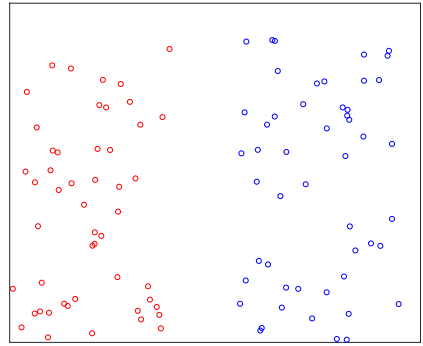
(b) D_2 is linearly separable, D_1 isn't

(c) Both are linearly separable

D_1



D_2



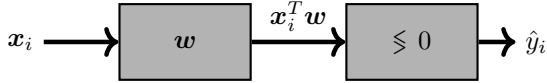
Don't let appearances fool you!

Agenda

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 (
coefficie



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...

Number of parameters and number of samples in dataset

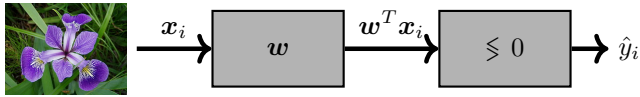
Recall the question in the exercise, about the degree of polynomial that can achieve zero MSE for given dataset.

Q: Given a dataset with 9 samples, which polynomial degree of a regression model can achieve zero MSE?

- Relationship between number of samples and degree of polynomial, or number of co-coefficients
- Overfitting

Rich inputs, high risks

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



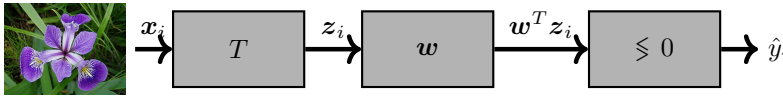
If each RGB picture consists of $A \times B$ pixels:

- 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.

You need more samples than degree of freedom of a model.

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.

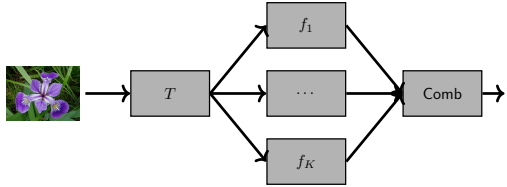


- Our linear model takes 4 values as input.
- The entire machine learning pipeline consists of a feature extraction stage (T) followed by a linear model.
- Both feature extraction stage and model are deployed.
- The final quality depends on the feature extraction stage.

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 need to be tested and deployed.
- The quality of a prediction depends on all the pipeline stages.
- It makes no sense to train a model in a pipeline, then change the other stages in the pipeline.

Agenda

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.

Normalisation (transformation)

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

For a many families of models (such as SVM), data normalisation helps to optimise parameters **efficiently**.

A normalisation stage has **parameters** whose values are determined using **a dataset**. It is not trained though, the values are determined using well-defined operations on the dataset.

We will focus on:

- Min-max normalisation
- Standardisation

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**.

Note: min-max is sensitive to outliers

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 I

- 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 $[0, 0.1]$).

Normalisation: Observations II

- 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

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.

The machine learning task of designing and creating transformation stages is known as **feature engineering**.

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.

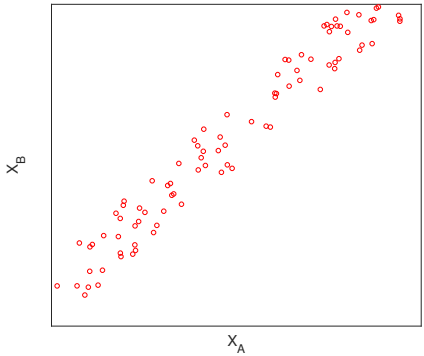
If the destination space has fewer dimensions than the original one, we talk about **dimensionality reduction**.

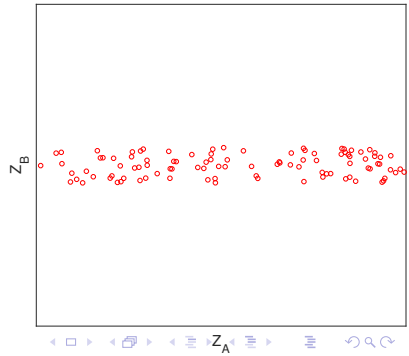
- In particular, after **feature selection** the destination space is defined by a subset of the original attributes.
- In **feature extraction** the new attributes are defined as operations on the original attributes. Common when using complex types, such as photos or audio signals.

Linear transformations: PCA

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

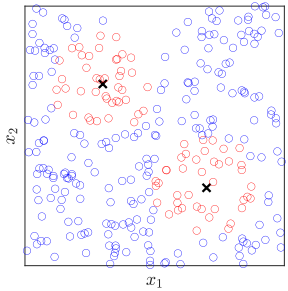
Using a dataset, PCA builds a **linear transformation** and additionally assigns a **score** to each component.



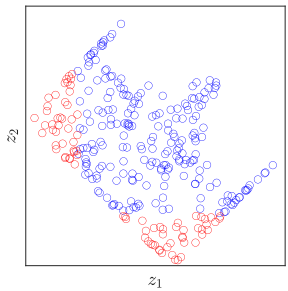


Non-linear transformations: Example

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

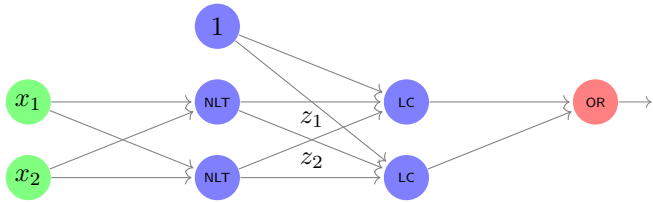
The dataset has two classes that are not linearly separable in the original space. In the destination space, we can use two linear boundaries.

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 machine learning models: Kernel methods

Many complex machine learning models can 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

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

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

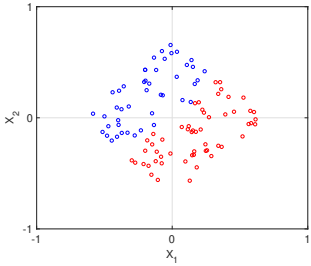
To select the **most relevant features**, we need to be able to assign a score to different subsets of features:

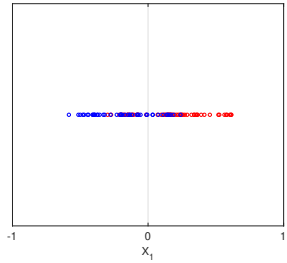
- If our dataset has M attributes, there are a **total of** $2^M - 1$ **subsets** that we could consider (10 attributes define approx 1000 subsets).
- What do we mean by **relevant**? In supervised learning, we can use our target metric to evaluate how relevant a subset of attributes is.
- We need a 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 pipelines** that use different subset of attributes.

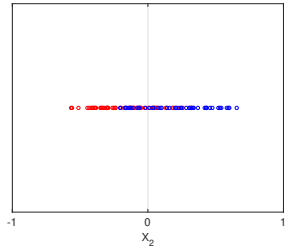
Feature selection: Filtering

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

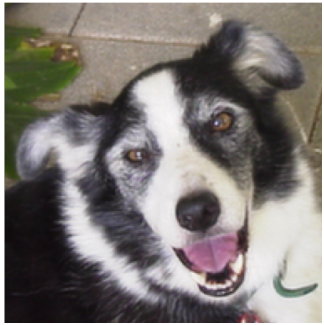
If we suspect that the interaction between features might be crucial, we have no choice but to **evaluate them together**, rather than separately.

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.

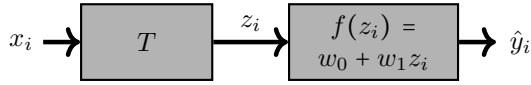
Digital **signal** and **image processing** provide with a wide range of options to extract features from rich input data types.

Pipeline: Question

Training Dataset

x	y
10	1
20	2
15	1
30	3
35	3

Building a pipeline



- Transformation T is a min-max normalisation.
- Linear model $f(z_i) = w_0 + w_1 z_i$ trained using least squares $w_0 = 0.9$, $w_1 = 2.3$

Testing the above pipeline

Testing Dataset: (x,?)

x	$\hat{y}$
1	
25	
35	
100	

Testing: The above pipeline is built using a training dataset and deployed. You observe following incoming values for x (shown in table on left side) after deployment. Compute the output $\hat{y}_i$ value for each input x_i.

Agenda

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?

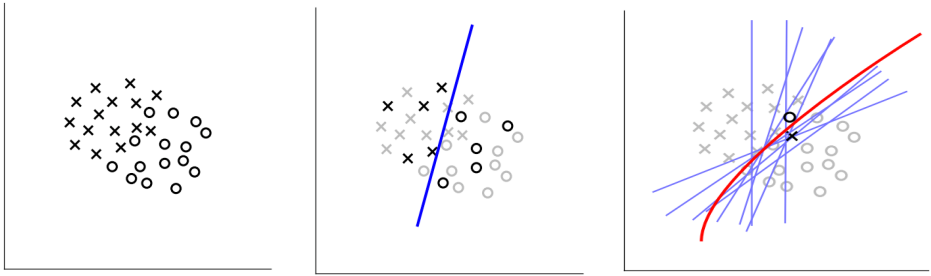
Ensemble methods allow us to create a new model that **combines** the strengths of **base models**. Base models need to be as **diverse** as possible and can be created by training:

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

Ensembles: Bagging

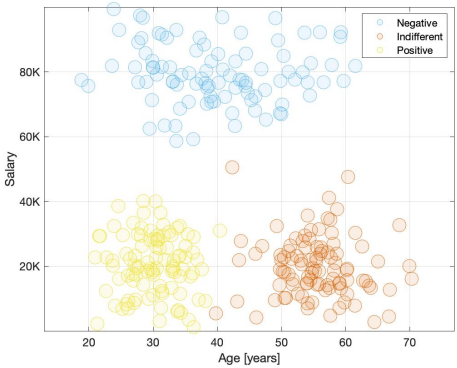
Bootstrap is a statistical method that extracts random samples from a dataset. Given a training dataset, **bagging** generates K sub-datasets by bootstrapping and trains K simple base models with each sub-dataset.

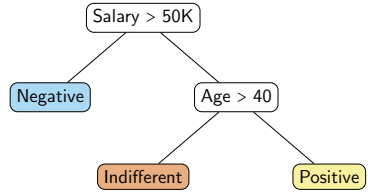
The final model $f(x)$ **combines the predictions of the base models** $f_k(x)$ by averaging or voting, for instance $f(x) = \sum f_k(x)/K$.



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.





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**:

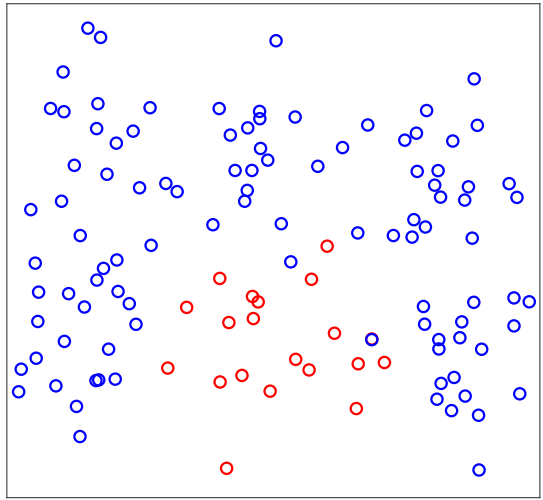
- Starting from the root, we recursively **split each region into two**. Splits are **axis-parallel** (decisions using one predictor).
- The chosen split is such that the purity of the resulting regions is higher than any other split.
- We stop when a given criterion is met, such as the number of samples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

How are decision trees built?

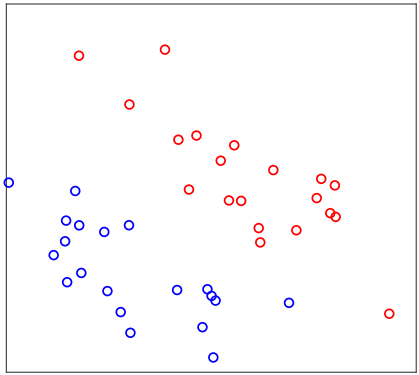
Decision trees are built **recursively**, for instance by **binary splitting**:

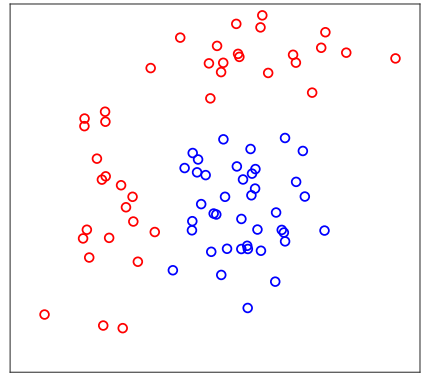
- Starting from the root, we recursively **split each region into two**. Splits are **axis-parallel** and implement simple decisions using one predictor ($\text{predictor}_A > \theta$).
- The chosen split is such that the purity (e.g. entropy, Gini index) of the resulting regions is higher than in any other split.
- We stop when a given criterion is met, such as the number of samples in a region.

Binary splitting



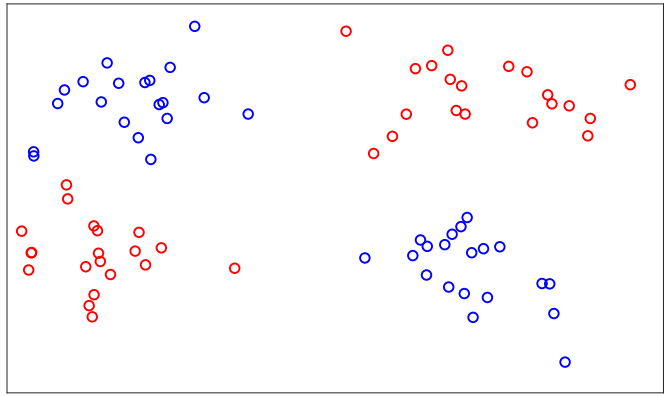
Linear classifier Vs decision tree





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 said to be *harder to interpret than a single tree* (this is not my personal view, trees can be as hard to interpret).

Ensembles: Boosting

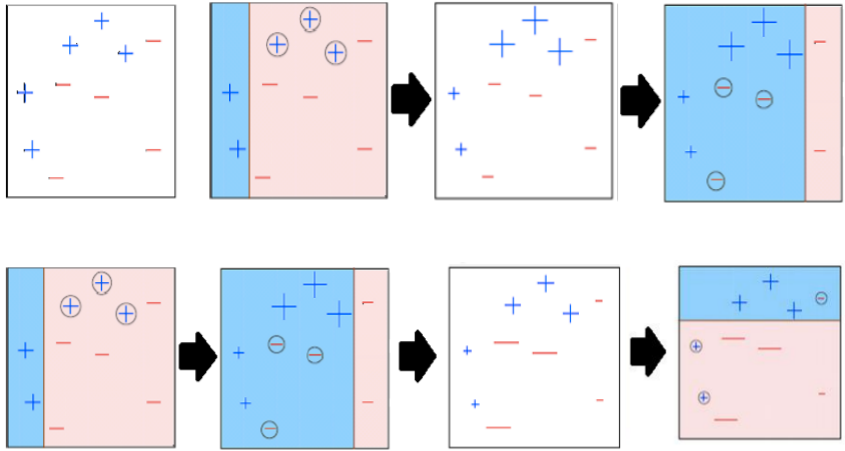
So far we have considered ensembles that create base model diversity by using some form of **randomisation**.

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.

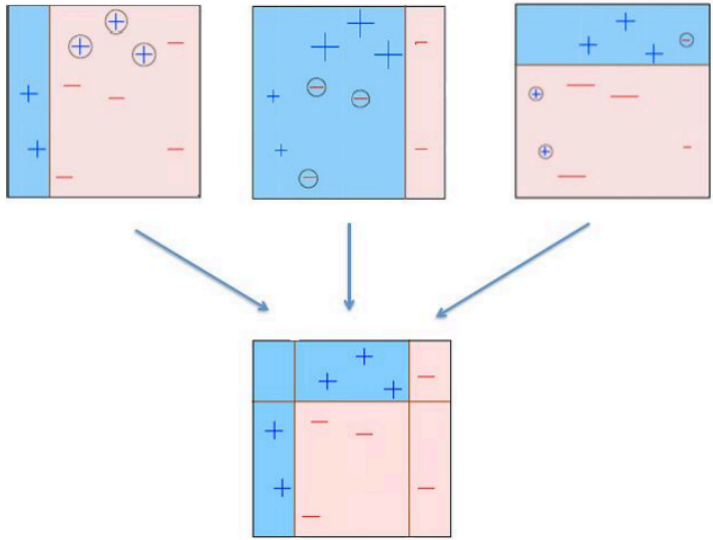
Therefore, each new base model generated by boosting depends on the previous models unlike bagging and random forests.

Bagging reduces variance by training models in parallel, while boosting reduces bias by training models in sequence, correcting errors at each step.

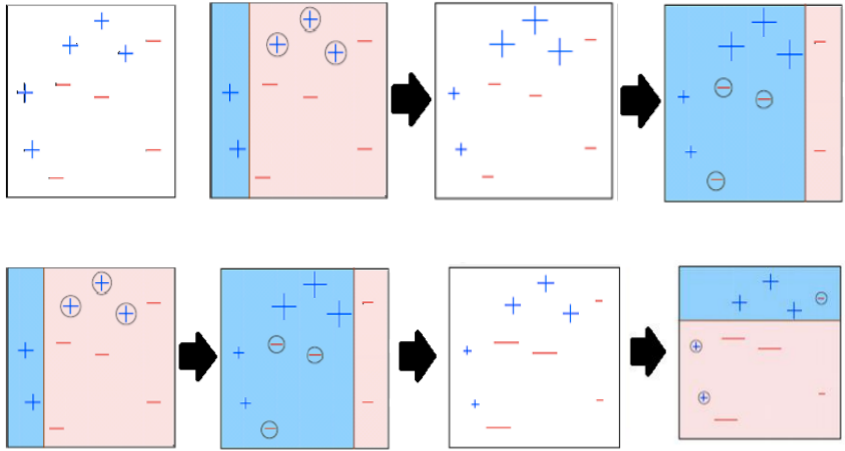
Ensembles: Boosting



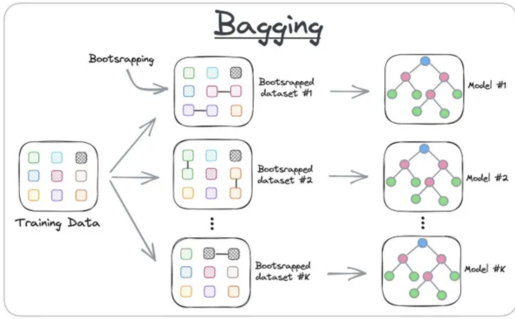
Ensembles: Boosting

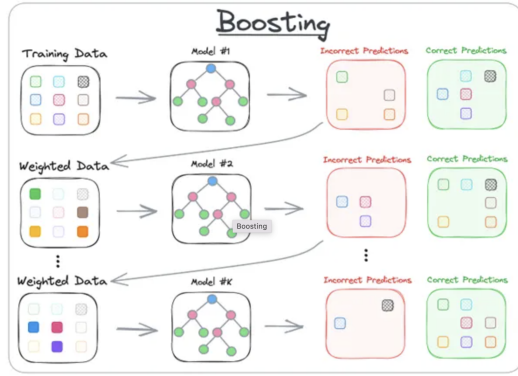


Ensembles: Boosting



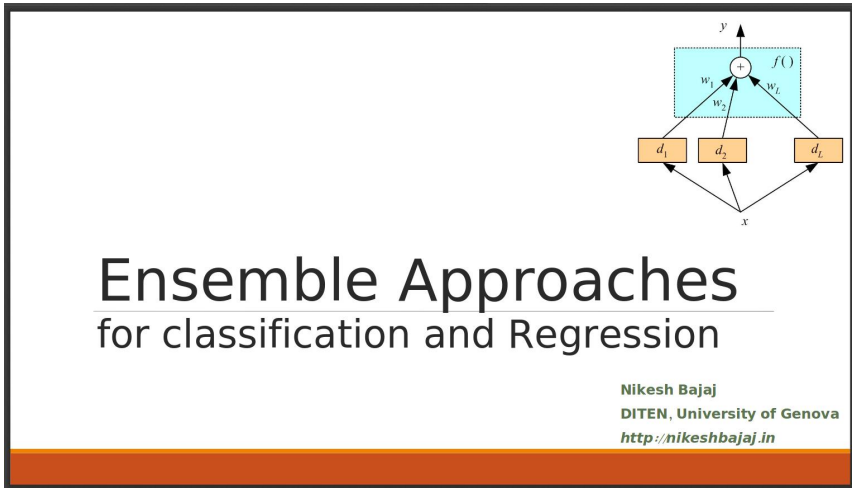
Ensembles: Bagging & Boosting





Source: Science Direct

Ensembles: Resources



Link: https://nikeshbajaj.github.io/teaching/presentations/Ensemble_Approches_NB_2017.pdf

Agenda

Diagnosis: When things don't go as expected

While working on a machine learning problems, usually things do go as well as expected, unless you are working on the trivial or textbook problem.

At times, it gets frustrating when you are working on a problem and things don't go anywhere.

What we can do, when this happens? Assuming we have good background knowledge of the problem, we diagnose the problem with following empirical approaches.

- Feature/Data Analysis
- Model Complexity
- Bias vs Variance
- Error analyses
- Analyse your trained model

These approaches are empirical - suggested by experts in the field as good practices.

Diagnosis: Feature/Data Analysis

Feature Analysis or Data Analysis is usually the first step we do, before we even begun to choose and train models.

Machine learning engineers/ Data scientistic, usually (should) **spend many hours exploring the data and features**, to understand the complexity of the problem, which gives a good idea which model will work and which will not.

Diagnosis: Model Complexity

Model Complexity and description is next important thing to check. You should always check the model description properly.

- Underlying assumptions of the model
- Complexity of the model - **not too small not too large**
- **Number of parameters in the model**

Recall lab activity, minimum order of polynomial for perfect prediction. **For a dataset of 12 samples, minimum degree of polynomial was 11** *for perfect prediction.*

You need more samples than degree of freedom of a model.

Diagnosis: High bias or high variance?

Recall **High bias or high variance issue**. Plotting the curves for training and validation gives you a good idea, if you have high bias or high variance.

This issue is related to the model complexity.

Diagnosis: Error analyses

Once model is trained and it is not performing as well as expected, one approach to find out why, is to perform the error analysis

Error analysis is a way to check those samples in the dataset where model is making error (far from true label). Often it is useful to see, if there is any particular reason for making mistakes for selected samples.

It is not always useful, however, sometime, it might reveals a specific attributes for those samples being mistaken by model.

Diagnosis: Analyse your trained model

A trained model can be analysed by evaluating the model parameters and their interpretations.

It often reveals the decision process of the model, which can be validated for the problem, given the background knowledge.

Analysing a trained model is not used just for the diagnosis purpose but also for **explainability of the model**.
explainability of the model (explainable AI) is large field that focuses on explaining the complex models.

Agenda

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.

Queen Mary
University of London