A Not-So-Gentle Introduction To Machine Learning

Marco Del Vecchio [1]

September 8, 2017

[1]Contact info: marco@delvecchiomarco.com, www.delvecchiomarco.com

Contents

Chapter 1

Introduction

1.1 Notation

$\boldsymbol{X}$ denotes an $N \times D$ matrix.

$\boldsymbol{x}_n$ is a row vector and denotes the n^{th} row of $\boldsymbol{X}$.

$\boldsymbol{x}_d$ is a row vector and denotes the d^{th} column of $\boldsymbol{X}$.

$\boldsymbol{x}$ denotes a column vector.

x denotes a scalar.

$f(\boldsymbol{x_n}; \boldsymbol{w})$ is a function acting on $\boldsymbol{x_n}$ with parameters $\boldsymbol{w}$.

t_n denotes the true label of the n^{th} sample.

p denotes a general probability density/mass function.

Note: unless otherwise specified, every other vector not named $\boldsymbol{x}_d$ or $\boldsymbol{x}_n$ is assumed to be a column vector.

1.2 Well-Posed Learning Problems

Definition 1 (Learning)**.** *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

In general, in order to have a well-posed learning problem, one must identify three features:

1. The set of tasks T.
2. The measure of performance P.
3. The source of experience E.

1.3 Common Underlying Assumptions

One of the main assumptions in many standard statistical models is the so called identically and independently distributed (i.i.d.) assumption. According to this assumption:

Independent : The N samples are assumed to be independently drawn. As a consequence, the order of the samples does not matter, which translates to the fact the order of the rows in the dataset is of no importance.

Identically distributed : The samples are assumed to be drawn from the same underlying data generating distribution. This in turn allows the model to be able to generalise to unseen data since this new data is assumed to come form the same distribution. This assumption also implies that the data generating process is stationary. That is, the parameters that define it do not evolve in time.

Definition 2 ((Strictly) Stationary Process)**.** *A process $\{\eta_t\}_{t\geq 0}$ is (strictly) stationary if and only if the joint distribution of any set of k values is invariant under time shits, i.e., if and only if for $k \geq 1$ and $\forall s > 0$,*

$$p(\eta_{t_1}, \eta_{t_2}, \ldots, \eta_{t_k}) = p(\eta_{t_1+s}, \eta_{t_2+s}, \ldots, \eta_{t_k+s})$$

where p is the joint likelihood.

1.4 Loss Functions

In order to pick a good model, we need a way to address the performance of given vector of parameters $\boldsymbol{w}$ in terms of explaining the underlying relationship.

Definition 3 (Loss Function)**.** *A loss function $L(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w})$ is a single, overall measure of loss incurred in taking any of the available decisions or actions. In particular, in this context, we define a loss function to be a mapping that quantifies how unhappy we would be if we used $\boldsymbol{w}$ to make a prediction on $\boldsymbol{X}$ when the correct output is $\boldsymbol{t}$.*

Two simple examples of a loss function are:

Quadratic loss :

$$L(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} (t_n - f(x_n; \boldsymbol{w})))^2$$

Absolute loss :

$$L(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} |t_n - f(x_n; \boldsymbol{w}))|$$

1.5 Generalisation, Overfitting And Course Of Dimensionality

1.5.1 Generalisation and Overfitting

Definition 4 (Overfitting)**.** *A model, is said to overfit to the data if it does not generalise to out-of-sample cases although it fits the training data very well. More specifically, a model which explains the random error or noise in the data instead of underlying relationship is said to be overfitting.*

Definition 5 (Underfitting)**.** *A model, is said to underfit to the data if it cannot capture the underlying trend in the data.*

Ideally, we would like to choose a model which performs best (i.e. minimises the loss) on new unseen data. That is, we would like a model that can generalises well beyond the data used during training. However, by the very nature of the problem, unseen data is not available.

One solution to this is to split the dataset at our disposal into a training set and a validation (testing) set. Then, we would use the training set to train the model and the validation set to validate the performance of the model on new unseen data (data not used for training), and pick the model which achieves the best validation score.

Definition 6 (K-Fold Cross Validation (K-Fold CV))**.** *In k-fold cv, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k-1$ subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds are then averaged to produce a single measure of performance.*

K-Fold Cross Validation is a non-exhaustive cross validation method as it does not compute all ways of splitting the original sample.

Definition 7 (Leave-P-Out Cross Validation (LpOCV))**.** *Leave-p-out cross-validation (LpOCV) involves using p observations as the validation set and the remaining observations as the training set. This is repeated on all ways to cut the original sample on a validation set of p observations and a training set.*

LpOCV is an exhaustive cross-validation methods as it computes all ways of splitting the original sample, and the total number of times the model is trained on the training set and then tested on the testing set is $\binom{N}{p}$, where N is the number of observations in the original dataset.

An extreme case of LpOCV is leave-one-out cross validation, where the validation subsample consists of only one data point. In this case, the average quadratic loss would be is

$$L^{LOOCV} = \frac{1}{N}\sum_{n=1}^{N}(t_n - f(\boldsymbol{x}_n; \hat{\boldsymbol{w}}_{-n}))$$

where $\hat{\boldsymbol{w}}_{-n}$ is the estimate of the parameter obtaining by training the model without the n^{th} data point.

An alternative to K-Fold CV is to use the Bootstrap method according to which the validation set is sampled with replacement from the whole dataset.

1.5.2 Curse Of Dimensionality

As the number of dimensions of the dataset increases, some models become prone to overfitting. As we increase the model complexity, i.e. introducing more parameters, we increase the number of degrees of freedom of the model and therefore, we require even more data for training.

Part I

Supervised Learning

Chapter 2

Non-probabilistic Linear Regression Modelling

Throughout this chapter, let $\boldsymbol{X}$ with dimensions $(N \times (D+1))$ denote the augmented data matrix whose first column is the identity vector.

Definition 8 (Linear Model)**.** *A linear regression model is concerned with learning a linear relationship between the data* $\boldsymbol{X}$ *and the labels* $\boldsymbol{t}$*. That is, relationships of the form*

$$\boldsymbol{t} = f(\boldsymbol{X}; \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{X}$$

where

$$t_n = f(\boldsymbol{x}_n; \boldsymbol{w}) = \boldsymbol{x}_n \boldsymbol{w} = w_0 + x_1 w_1 \ldots, x_N w_D.$$

2.1 Ordinary Least Square Regression

Given, a linear model $f(\boldsymbol{X}; \boldsymbol{w})$, we call the model whose parameters vector minimises the average squared loss, ordinary least square (OLS) regression. That is, in an OLS regression model, the $\boldsymbol{w}$ solves

$$\begin{aligned} \arg\min_w L(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w}) &= \arg\min_w \frac{1}{N} \sum_{n=1}^{N} (t_n - f(x_n; w)))^2 \\ &= \arg\min_w \frac{1}{N} (\boldsymbol{t} - \boldsymbol{X}\boldsymbol{w})^T (\boldsymbol{t} - \boldsymbol{X}\boldsymbol{w}) \end{aligned}$$

Thanks to the fact that this specific loss function is convex and smooth, there exist an analytic solution to the above optimisation problem.

$$
\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{w}} = 0 &\implies \frac{\partial L}{\partial \boldsymbol{w}} \frac{1}{N}(t - Xw)^T(t - Xw) = 0 \\
&\implies \frac{\partial L}{\partial \boldsymbol{w}} \boldsymbol{t}\boldsymbol{w}^T - \boldsymbol{t}^t \boldsymbol{X}\boldsymbol{w} - \boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{t} + \boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{X}\boldsymbol{w} = 0 \\
&\implies \frac{\partial L}{\partial \boldsymbol{w}} \boldsymbol{t}\boldsymbol{w}^T - 2\boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{t} + \boldsymbol{w}^T \boldsymbol{X}^T \boldsymbol{X}\boldsymbol{w} = 0 \\
&\implies -2\boldsymbol{X}^T \boldsymbol{t} + 2\boldsymbol{X}^T \boldsymbol{X}\boldsymbol{w} = 0 \\
&\implies \boldsymbol{w} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{t}
\end{aligned}
$$

Checking the second derivative yields that the solution found above achieves the unique global minimum of L.

Remark 1. *The OLS regression model has the following limitations:*

1. *it is prone to overfitting in high dimensions.*
2. *the parameters are unbounded and therefore in the optimisation problem can become arbitrarily large.*
3. *it is prone to numerical instability (high variance) when the attributes are correlated with each other.*

2.2 Sparsity And Regularisation

Definition 9 (Sparse)**.** *We say that a model is sparse when few attributes have non-zero coefficients associated to them, thus contribute significantly to the resulting model, and the remaining are switched off by having have zero coefficients associated to them.*

Sparsity in a linear model is beneficial for several reasons:

1. for statistical interpretation:
   (a) Claims of the form " attribute x has more predictive power than attribute y" can be made.
   (b) Discover significant correlations.
   (c) Easier for humans to read it.
2. for computational tractability
   (a) less computations and storage
3. for preventing overfitting during training.

We can encode our preference for sparse, and therefore simpler models, by adding a regularisation term to the loss function which takes care of bounding the parameter vector in some way. One commonly used family of regularisers uses the L_p norm.

Definition 10 (L_p Norm)**.** *Let $\boldsymbol{x}$ be an N dimensional vector then,*

$$
L_p(\boldsymbol{x}) = ||\boldsymbol{x}||_p = \Big(\sum_{i=1}^{N} |x_i|^p \Big)^{1/p}
$$

2.2.1 Ridge regression

Ridge regression is a regularised linear regression model where the underlying optimisation problem is:

$$\begin{aligned} &\arg\min_{w} L(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w}) \\ &\text{s.t.} \quad ||\boldsymbol{w}||_2^2 \leq \tau \end{aligned}$$

where $L(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w}) = \frac{1}{N}(\boldsymbol{t} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{t} - \boldsymbol{X}\boldsymbol{w})$. This is equivalent to minimising

$$L(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w}, \lambda) = \frac{1}{N}(\boldsymbol{t} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{t} - \boldsymbol{X}\boldsymbol{w}) + \lambda||\boldsymbol{w}||_2^2.$$

where λ is the strength of the regularisation parameter. This optimisation problem can again be solved analytically and the optimal solution is given by setting

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + N\boldsymbol{I}\lambda)^{-1}\boldsymbol{X}^T\boldsymbol{t}$$

where $\boldsymbol{I}$, the identity matrix, is introduced to make the addition between matrices valid.

Remark 2. *Using the L_2 regularisation (ignoring the fact that we also square it), bounds the parameter vector into a hyper-sphere of size τ. This in turn implies that the parameters cannot get arbitrarily large.*

2.2.2 LASSO regression

LASSO regression is a regularised linear regression model where the underlying optimisation problem is:

$$\begin{aligned} &\arg\min_{w} L(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w}) \\ &\text{s.t.} \quad ||\boldsymbol{w}||_1 \leq \tau \end{aligned}$$

where $L(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w}) = \frac{1}{N}(\boldsymbol{t} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{t} - \boldsymbol{X}\boldsymbol{w})$. This is equivalent to minimising

$$L(\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{w}, \lambda) = \frac{1}{N}(\boldsymbol{t} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{t} - \boldsymbol{X}\boldsymbol{w}) + \lambda||\boldsymbol{w}||_1.$$

where λ is the strength of the regularisation parameter. This optimisation problem cannot be solved analytically due to the fact that the absolute value function is not everywhere differentiable. As a consequence, a numerical solution must be computed.

Remark 3. *Using the L_1 regularisation, bounds the parameter vector into a hyper-diamond of size τ. This in turn implies that sparse solution can be found since this regularisation can force some of the parameters to be identically zero.*

2.3 Non-Linear Responses From A Linear Model

The definition of a linear model $\boldsymbol{t} = f(\boldsymbol{X}; \boldsymbol{w})$ demands the relationship between $\boldsymbol{t}$ and $\boldsymbol{w}$ to be linear. That is, f is a linear function with respect to $\boldsymbol{w}$. However, this definition does not impose any restriction on the form of the relationship between $\boldsymbol{t}$ and $\boldsymbol{X}$. As a consequence, we can have a linear model which is linear with respect to the parameters but possibly non-linear with respect to the input data. Thus being able to generate a non-linear response from a linear model.

2.4 Feature Engineering

One way to establish a non-linear relationship between $\boldsymbol{t}$ and $\boldsymbol{X}$ is to define a mapping

$$\boldsymbol{X} \mapsto \phi(\boldsymbol{X})$$

form $\mathbb{R}^{N\times(D+1)}$ to $\mathbb{R}^{N\times D^*}$, where $D^* \leq (D+1)$ or $D^* \geq (D+1)$, which transforms the input space in to a feature space.

One example of such a mapping is the polynomial expansion: we augment each sample point by mapping

$$\boldsymbol{x}_n \mapsto \Big(\sum_{d=1}^{D+1} x_{n,d} \Big)^k$$

for an order k polynomial expansion of the input data.

Chapter 3

Instance Based Learning

Instance-based learners are all the machine learning algorithms that do not construct an explicit description of the target function but store the training examples and use them, and a notion of distance between them, to generalise to unseen data.

3.1 K-Nearest Neighbours Algorithm (kNN)

Definition 11 (Decision Boundary)**.** *The boundary between two classes where it is equiprobable to belong to either class is called the decision boundary.*

3.1.1 Distance Metrics

Given, a norm L, we can define the distance between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ in the space defined by L as $d(\boldsymbol{x}, \boldsymbol{y}) = L(\boldsymbol{x} - \boldsymbol{y})$.

One distance metric which is not based on the L_p but will be useful in this section, is the so called Hamming distance which establishes a notion of distance for categorical data.

Definition 12 (Hamming Distance)**.** *Let the Hamming stance between two D-dimensional vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ be defined as*

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{D} \delta(x_i, y_i)$$

where,

$$\delta(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{otherwise} \end{cases}$$

Algorithm 1 KNN Regression$(\boldsymbol{x}^*, \{(\boldsymbol{x}_n, t_n)\}_{n=1}^N, k)$

1: Let the input-output pairs $\{(\boldsymbol{x}_n, t_n)\}_{n=1}^N$ be the training examples.
2: Find the k nearest neighbours of $\boldsymbol{x}^*$ in $\{\boldsymbol{x}_n\}_{n=1}^N$ and label them $\{\bar{x}_1, \ldots, \bar{x}_k\}$
3: Return t^* where

$$t^* = \frac{1}{k} \sum_{n=1}^{k} \bar{t}_n$$

and where $\{\bar{t}_1, \ldots, \bar{t}_k\}$ are the target labels of $\{\bar{x}_1, \ldots, \bar{x}_k\}$.

Algorithm 2 KNN Binary Classification$(\boldsymbol{x}^*, \{(\boldsymbol{x}_n, t_n)\}_{n=1}^N, k)$

1: Let the input-output pairs $\{(\boldsymbol{x}_n, t_n)\}_{n=1}^N$ be the training examples.
2: Find the k nearest neighbours of $\boldsymbol{x}^*$ in $\{\boldsymbol{x}_n\}_{n=1}^N$ and label them $\{\bar{x}_1, \ldots, \bar{x}_k\}$
3: Return t^* where

$$t^* = \arg\max_{c \in \{-1,1\}} \sum_{n=1}^{k} (1 - d(c, \bar{t}_n))$$

and where $\{\bar{t}_1, \ldots, \bar{t}_k\}$ are the labels of $\{\bar{x}_1, \ldots, \bar{x}_k\}$ and $d(x, y)$ is the Hamming distance between x and y.

3.1.2 Distance-Weighted KNN

It is possible to specify a variant of the vanilla KNN algorithms introduced above by weighting different neighbours differently. One such way is to weight neighbours in terms of distance:

$$t^* = \begin{cases} \frac{1}{k} \frac{\sum_{n=1}^k w_n \bar{t}_n}{\sum_{n=1}^k w_n} & \text{for regression} \\ \arg\max_{c \in \{-1,1\}} \sum_{n=1}^k w_n (1 - d(c, \bar{t}_n)) & \text{for classification} \end{cases}$$

where

$$w_n = \frac{1}{\tilde{d}(\boldsymbol{x}^*, \bar{\boldsymbol{x}}_n)^2}$$

where $\tilde{d}$ is a distance metric. Popular choices for $\tilde{d}$ are the Euclidean distance for real-valued vectors and the Hamming distance for binary-valued vectors.

Note: if $\boldsymbol{x}^* = \bar{\boldsymbol{x}}_n$ for some $n = 1, 2, \ldots k$ then w_n is undefined. We overcome this by letting $t^* = \bar{t}_n$ when this happens.

Remark 4. *Remarks on KNN model complexity and curse of dimensionality:*

1. *k controls the complexity of the model. As k increases, the decision boundaries become smoother and therefore, the complexity of the model decreases.*

2. *Vanillia KNN will perform poorly in high dimensional spaces because in high dimensions data tends to concentrate and distances, as a result, tend to go to extremes. One way to avoid this is to use the weighted version of the KNN algorithm, by learning the weights via cross validation.*

3.2 Decision Trees

Definition 13 (Entropy)**.** *Let the entropy of a set S be defined as*

$$E(S) = -\sum_i p_i \log_2 p_i$$

where $p_i = \frac{|\{x \in S | x = i\}|}{|S|}$ is the proportion of elements of S equal to i.

Definition 14 (Impurity)**.** *We say that a node is impure if the samples reaching the node are not of the same category.*

Definition 15 (Information Gain)**.** *We define information gain as the amount of information that we have gained after having split the data according to a specific rule.*

3.2.1 Complexity And Overfitting

In order to combat overfitting in decision tree, two general approaches are

Pre-Pruning : Stop growing the tree earlier, before perfect classification on training data. This can be achieved by

1. Stopping growing the tree when there is no statistically significant change in information gain.

Post-Pruning Allow the tree to grow fully and then prune it. This can be achieved by

1. Performing reduced error pruning: propose nodes for removal and test the difference in performance via cross validation. Then, nodes whose removal is found to increase the performance are removed iteratively.
2. Performing rule post pruning: convert the fully grown tree into rules and then prune the rules.

3.2.2 ID3

The ID3 algorithm can be applied to problems which involve nominal data only. If the problem involves real-valued variables, they are first binned into intervals, each interval being treated as an unordered sequence of nominal attributes.

Definition 16 (Entropy Impurity)**.** *Entropy impurity is a measure of impurity calculated using the notion of entropy. We define entropy impurity as*

$$i(S) = E(S) = -\sum_i p_i \log_2 p_i$$

where p_i is the frequency of items of type i in S.

At this point, in order to being able to choose the best attributes which enable us to discriminate between classes, we need to measure the change in impurity once a split occurs. To this end, we define the change in impurity as

$$\Delta i(S, A) = i(S) - \sum_i \frac{|S_{u_i}|}{|S|} i(S_{u_i})$$

where A is the attribute under consideration, u_i are the possible values of A and S_{u_i} is the subset of S where $A = u_i$.

Algorithm 3 ID3(`Observations`,`Targets`, `Attributes`)

```
1: Create a root node for the tree
2: if all the observations belong to the same class then
3:    return a single node tree with label indicating that specific class.
4: end if
5: if Attributes is empty then
6:    return a single-node tree with label indicating the most common value in
      Targets
7: else
8:    A* ← arg max_{A∈Attributes} Δi(Observations, A)
9:    Let A* be the decision attribute for the root node
10:   for each possible value u_i that A* takes do
11:      Add a new tree branch below the root tree for A = u_i
12:      S_{u_i} ← subsets of Observations where A* = u_i
13:      if S_{u_i} is empty then
14:         Add a leaf node with label indicating the most common value in Targets
15:      else
16:         Add below branch ID3(S_{u_i},Targets, Attributes\A*)
17:      end if
18:   end for
19: end if
```

Remark 5. *Remarks on the ID3 algorithm:*

1. *The hypothesis space of the ID3 algorithm contains every finite discrete value function.*
2. *The ID3 algorithm is an hill-climbing (greedy) search algorithm with the change in entropy impurity as the optimisation function. As a consequence, ID3 does not have any backtracking to avoid local solutions.*
3. *The ID3 algorithm posses inductive bias: the preference for shorter trees is encoded in the algorithm and attributes with the highest change in entropy impurity will be closer to the root of the tree.*

3.2.3 C4.5

The ID3 algorithm can been shown to be biased towards favouring attributes with a high branching factor. This is due to how we measure the change in entropy impurity so a way to fix this is to normalise the change in entropy impurity by the entropy impurity of the data split.

Definition 17 (Gain Ratio Impurity)**.**

$$\Delta i_B(S, A) = \frac{\Delta i(S, A)}{\sum_{k=1}^{B} -p_k \log_2 p_k}$$

where p_k is the fraction of the data on branch k, and B is the number of branches coming out of the node.

Another difference between ID3 and C4.5 is that while ID3 does not support real-valued data, C4.5 can handle such a scenario by creating split tests on thresholds for the continuous variables.

3.2.4 CART

The CART (Classification And Regression Trees) algorithm differs from ID3 and C4.5 in various ways:

1. Does not use entropy as a measure of impurity but the so called Gini Impurity.
2. It only deals with binary trees - as every tree can be represented using only binary decisions.
3. Can handle missing values nicely via surrogate splits - similar splits that can be performed when a specific measurement is missing.

Definition 18 (Gini Impurity)**.** *We define Gini impurity as*

$$i_G(S) = \sum_{i \neq j} p_i p_j$$

where $p_k, k \in \{i, j\}$ is the fraction of the observations in S which are of type k.

Remark 6. *In the CART algorithm it is possible to combat overfitting using a variate of post-pruning which consists of merging pairs of neighbouring nodes back to the ancestor. This can be done because CART uses binary trees.*

Chapter 4

Maximum Likelihood Framework

4.1 General Theory

Definition 19 (Likelihood)**.** *The likelihood of a vector of parameters $\boldsymbol{\theta}$, given the outcome of an N dimensional random variable $\boldsymbol{X}$, $\boldsymbol{x}$, is equal to the probability assumed for those observed outcomes given those parameter values, that is*

$$L(\boldsymbol{\theta}|\boldsymbol{x}) = p_{\boldsymbol{\theta}}(\boldsymbol{x})$$

where p is the probability density/mass function which specifies the distribution assumed for $\boldsymbol{X}$. Further, when we assume i.i.d. random variables,

$$L(\boldsymbol{\theta}|\boldsymbol{x}) = \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(x_i)$$

Definition 20 (Maximum Likelihood Estimate (MLE))**.** *The maximum likelihood estimate is the value of $\boldsymbol{\theta}$ which maximises the likelihood, that is,*

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}|\boldsymbol{x})$$

4.1.1 Bias Variance Decomposition

Definition 21 (Bias)**.** *The Bias of an estimator $\hat{\theta}$ with respect to an unknown parameter θ is defined as*

$$Bias(\hat{\theta}) = \mathbb{E}(\hat{\theta}) - \theta$$

Further, we say that an estimator is unbiased when the bias is equal to zero.

Definition 22 (Mean Squared Error (MSE))**.** *The MSE of an estimator $\hat{\theta}$ with respect to an unknown parameter θ is defined as*

$$MSE(\hat{\theta}) = \mathbb{E}((\hat{\theta} - \theta)^2)$$

The MSE can be written as the sum of the variance of the estimator and the squared bias of the estimator, providing a useful way to calculate the MSE and implying that in the case of unbiased estimators, the MSE and variance are equivalent.

Proposition 1. *The MSE of an estimator $\hat{\theta}$ with respect to an unknown parameter θ is can be decomposed in terms of Variance and Bias:*

$$MSE(\hat{\theta}) = Var(\hat{\theta}) + Bias(\hat{\theta})^2$$

Proof.

$$\begin{aligned} MSE(\hat{\theta}) &= \mathbb{E}((\hat{\theta} - \mathbb{E}(\hat{\theta}) + \mathbb{E}(\hat{\theta}) - \theta)^2) \\ &= \mathbb{E}((\hat{\theta} - \mathbb{E}(\hat{\theta}))^2 - 2(\hat{\theta} - \mathbb{E}(\hat{\theta}))(\mathbb{E}(\hat{\theta}) - \theta) + (\mathbb{E}(\hat{\theta}) - \theta)^2) \\ &= \mathbb{E}((\hat{\theta} - \mathbb{E}(\hat{\theta}))^2) + \mathbb{E}((\mathbb{E}(\hat{\theta}) - \theta)^2) \\ &= Var(\hat{\theta}) + Bias(\hat{\theta})^2 \end{aligned}$$

since

$$\begin{aligned} \mathbb{E}((\mathbb{E}(\hat{\theta}) - \theta)^2) &= \mathbb{E}(\mathbb{E}(\hat{\theta})^2 - 2\mathbb{E}(\hat{\theta})\theta + \theta^2) \\ &= \mathbb{E}(\mathbb{E}(\hat{\theta})^2) - 2\mathbb{E}(\mathbb{E}(\hat{\theta})\theta) + \mathbb{E}(\theta^2) \\ &= \mathbb{E}(\hat{\theta})^2 - 2\mathbb{E}(\hat{\theta})\theta + \theta^2 \\ &= (\mathbb{E}(\hat{\theta}) - \theta)^2 \\ &= Bias(\hat{\theta})^2, \end{aligned}$$

and

$$\begin{aligned} -2\mathbb{E}((\hat{\theta} - \mathbb{E}(\hat{\theta})(\mathbb{E}(\hat{\theta}) - \theta)) &= -2\mathbb{E}(\hat{\theta}\mathbb{E}(\hat{\theta}) - \hat{\theta}\theta - \mathbb{E}(\hat{\theta})^2 + \mathbb{E}(\hat{\theta})\theta) \\ &= -2(\mathbb{E}(\hat{\theta})^2 - \mathbb{E}(\hat{\theta})\theta - \mathbb{E}(\hat{\theta})^2 + \mathbb{E}(\hat{\theta})\theta) \\ &= 0. \end{aligned}$$

□

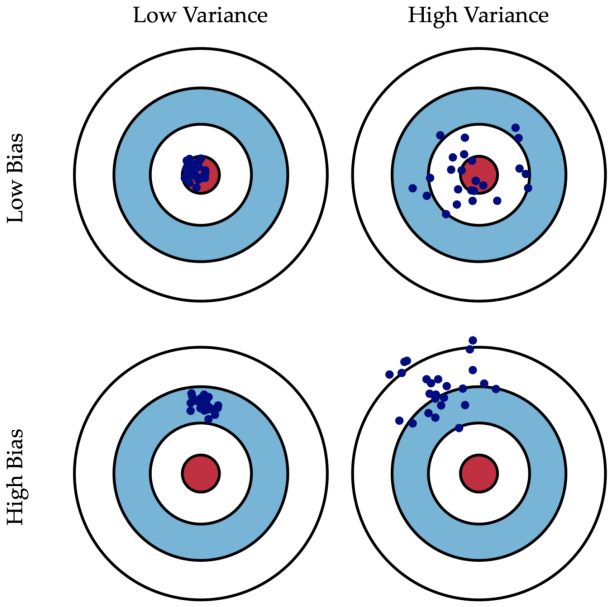


Figure 4.1: Bias Variance Trade-off

4.1.2 Linear Regression In The MLE Framework

Under this probabilistic framework, we can view a linear model as

$$t_n = f(\boldsymbol{x}_n; \boldsymbol{w}) + \epsilon_n = \boldsymbol{x}_n \boldsymbol{w} + \epsilon_n, \quad \epsilon_n \sim \mathcal{N}(0, \sigma^2)$$

That is, we assume that the observed data is coming form an hyperplane corrupted by some white Gaussian noise.

Then, by treating $\boldsymbol{t} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon}$ as a random variable, and assuming that the ϵ_n are i.i.d random variables,

$$\boldsymbol{t}_n \sim \mathcal{N}(\boldsymbol{x}_n \boldsymbol{w}, \sigma^2 | \boldsymbol{x_n}) \implies p_{(\boldsymbol{w}, \sigma^2)}(\boldsymbol{t}|\boldsymbol{X}) = \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{x}_n \boldsymbol{w}, \sigma^2 | \boldsymbol{x_n}),$$

we can write down the likelihood of the linear regression model:

$$L((\boldsymbol{w}, \sigma^2)|\boldsymbol{X}) = \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{x}_n \boldsymbol{w}, \sigma^2 | \boldsymbol{x_n})$$

By solving

$$(\hat{\boldsymbol{w}}, \hat{\sigma}^2) = \underset{(\boldsymbol{w}, \sigma^2) \in \mathbb{R}^D \times \mathbb{R}_+}{\arg\max} L((\boldsymbol{w}, \sigma^2)|\boldsymbol{X})$$

we find that

$$\begin{aligned} \hat{\boldsymbol{w}} &= (\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}^T\boldsymbol{t}, \\ \hat{\sigma}^2 &= \frac{1}{N}\sum_{n=1}^{N}(t_n - \boldsymbol{x}_n\hat{\boldsymbol{w}})^2. \end{aligned}$$

We can see that under this framework, the MLE for $\boldsymbol{w}$ is the same as the OLS solution and the MLE for the varince of the Gaussian white noise process is equal to the average squared loss incurred by setting $\boldsymbol{w}$ to be equal to its MLE.

Proposition 2. *The OLS estimator is an unbiased estimator.*

Proof.

$$\begin{aligned} \mathbb{E}(\hat{\boldsymbol{w}}^{OLS}) &= \mathbb{E}((\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}^T\boldsymbol{t}) \\ &= \mathbb{E}((\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon})) \\ &= \mathbb{E}((\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}) + \mathbb{E}((\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}^T\boldsymbol{\epsilon}) \\ &= \mathbb{E}(\boldsymbol{w}) + (\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}^T\mathbb{E}(\boldsymbol{\epsilon}) \\ &= \boldsymbol{w}. \end{aligned}$$

□

Proposition 3. *The Ridge estimator is a biased estimator.*

Proof. Let $\boldsymbol{R} = \boldsymbol{X}\boldsymbol{X}^T$ then,

$$\begin{aligned}
\mathbb{E}(\hat{\boldsymbol{w}}^{Ridge}) &= \mathbb{E}((\boldsymbol{X}\boldsymbol{X}^T + N\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{t}) \\
&= \mathbb{E}((\boldsymbol{R} + N\lambda\boldsymbol{I})^{-1}\boldsymbol{R}(\boldsymbol{R}^{-1}\boldsymbol{X}^T\boldsymbol{t})) \\
&= \mathbb{E}(\boldsymbol{R}(\boldsymbol{I} + N\lambda\boldsymbol{I}\boldsymbol{R}^{-1})^{-1}\boldsymbol{R}((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{t}) \\
&= \mathbb{E}(\boldsymbol{R}(\boldsymbol{I} + N\lambda\boldsymbol{I}\boldsymbol{R}^{-1})^{-1}\boldsymbol{R}\hat{\boldsymbol{w}}^{OLS}) \\
&= \mathbb{E}((\boldsymbol{I} + N\lambda\boldsymbol{I}\boldsymbol{R}^{-1})^{-1}\boldsymbol{R}^{-1}\boldsymbol{R}\hat{\boldsymbol{w}}^{OLS}) \\
&= \mathbb{E}((\boldsymbol{I} + N\lambda\boldsymbol{I}\boldsymbol{R}^{-1})^{-1}\hat{\boldsymbol{w}}^{OLS}) \\
&= (\boldsymbol{I} + N\lambda\boldsymbol{I}\boldsymbol{R}^{-1})^{-1}\mathbb{E}(\hat{\boldsymbol{w}}^{OLS}) \\
&\neq \boldsymbol{w}.
\end{aligned}$$

However,

$$\lim_{\lambda \to 0} \mathbb{E}(\hat{\boldsymbol{w}}^{Ridge}) = \mathbb{E}(\hat{\boldsymbol{w}}^{OLS}) = \boldsymbol{w}.$$

That is, with $\lambda = 0$, the Ridge estimator becomes unbiased. □

Predictive Variability

In this framework, it is possible to take into account the variability of the estimates using the covariance matrix associated to the estimators.

Proposition 4. *The covariance matrix of the OLS estimator is a function of the true variance σ^2 and the data.*

$$Var(\hat{\boldsymbol{w}}^{OLS}) = \sigma^2(\boldsymbol{X}\boldsymbol{X}^T)^{-1}$$

Proof.

$$\begin{aligned}
Var(\hat{\boldsymbol{w}}^{OLS}) &= Var((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{t}) \\
&= Var((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon})) \\
&= Var((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{\epsilon})) \\
&= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T Var(\boldsymbol{\epsilon})((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T)^T \\
&= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\sigma^2((\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T)^T \\
&= \sigma^2(\boldsymbol{X}\boldsymbol{X}^T)^{-1}.
\end{aligned}$$

□

We can use this covariance matrix to take into account the variability of the estimates. In the OLS case, we would have that:

$$Var(\boldsymbol{X}\hat{\boldsymbol{w}}^{OLS}) = \boldsymbol{X}Var(\hat{\boldsymbol{w}}^{OLS})\boldsymbol{X}^T = \boldsymbol{X}\sigma^2(\boldsymbol{X}\boldsymbol{X}^T)^{-1}\boldsymbol{X}^T.$$

Note: since the variance of the prediction is a function of the true variance, we have to approximate it with the estimator of the variance, however, it can be shown that the variance estimator $\hat{\sigma}^2$ is a biased estimator. In particular, $\hat{\sigma}^2$ underestimates the true variance σ^2 and as a consequence, the predictions will be overconfident.

Overall, adopting the MLE framework, has allowed us to express our uncertainty in the predictions as well as in the estimators.

Chapter 5

Bayesian Framework

5.1 General Theory

Theorem 1 (Bayes' Theorem)**.** *Let $p(\cdot)$ denote a probability function. Then,*

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)},$$

provided $p(y) > 0$.

If we apply this framework to our prediction problem, we have that

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{t}) = \frac{p(\boldsymbol{t}|\boldsymbol{w}, \boldsymbol{X})p(\boldsymbol{w})}{p(\boldsymbol{t}|\boldsymbol{X})}$$

where

$$p(\boldsymbol{t}|\boldsymbol{X}) = \int_{\mathcal{W}} p(\boldsymbol{t}|\boldsymbol{w}, \boldsymbol{X})d\boldsymbol{w}.$$

Naming conventions:

- $p(\boldsymbol{w})$: Prior distribution. This distribution should encapsulate our prior (before seeing any evidence) beliefs about the parameters of our model.
- $p(\boldsymbol{t}|\boldsymbol{w}, \boldsymbol{X})$: Likelihood. This function conveys how likely it is to have observed the targets $\boldsymbol{t}$ given a set of parameters $\boldsymbol{w}$ and data $\boldsymbol{X}$.
- $p(\boldsymbol{t}|\boldsymbol{X})$: Marginal likelihood. This function conveys how likely it is to have observed the targets $\boldsymbol{t}$ given some data $\boldsymbol{X}$ regardless of what the parameters are. This quantity is a normalising constant as in it ensures that the posterior distributions integrates/sums to 1.
- $p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{t})$: Posterior distribution. This distribution encapsulates our posterior (after seeing evidence) beliefs about $\boldsymbol{w}$.

Remark 7. *Remarks regarding the Bayesian framework:*

1. *As we accumulate more evidence, the effect of the prior will diminish.*
2. *When we do not have a lot of evidence, the effects of the prior are very strong. Hence, we must choose our prior carefully.*

3. *The prior beliefs can come from a variety of sources such as experts, data type, and, computational consideration (conjugacy).*

Definition 23 (Naive Bayes Assumption)**.** *Let $\boldsymbol{X} = (X_1, \ldots, X_m)$ be a random vector and Y a random variable. Then, the assumption that*

$$\mathbb{P}(\boldsymbol{X} = x|Y) = \prod_{i=1}^{m} \mathbb{P}(X_i = x_i|Y)$$

is called the Naive Bayes Assumption. That is, we assume that the components of the random vector $\boldsymbol{X} = (X_1, \ldots, X_m)$ are independent given the random variable Y written

$$\coprod_{j=1}^{m} X_j|Y.$$

Remark 8. *There are several advantages as well as disadvantage to the Bayesian framework:*

Advantages *:*

1. *Coherent framework to reason under uncertainty.*
2. *Outputs are densities.*
3. *Outputs are not sensitive to small perturbations.*
4. *More difficult to overfit thanks to the regularising power of the prior.*
5. *Uncertainty can be quantified via the covariance of the posterior density.*

Disadvantages *:*

1. *High computational complexity*
2. *We are not guaranteed to make better predictions as we gather more evidence.*

5.2 Posterior Approximations

Definition 24 (Closed Under Sampling: Conjugate Prior)**.** *A family of distributions $\mathcal{P} = \{p(\boldsymbol{x}|\boldsymbol{y}), \boldsymbol{y} \in \mathbb{R}^D\}$ is said to be closed under sampling for a likelihood $L(\boldsymbol{x}|\boldsymbol{z})$, if for a prior $p(\boldsymbol{x}) \in \mathcal{P}$, $L(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{x}) \in \mathcal{P}$, that is, the posterior distribution is in the same family as the prior one. In this case, the prior is said to be a conjugate prior with respect to the posterior density.*

Definition 25 (Maximum-A-Posteriori)**.** *The maximum a posterior is defined to be*

$$\boldsymbol{w}^{\hat{MAP}} = \arg\max_{\boldsymbol{w} \in \mathcal{W}} p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{t}) = \arg\max_{\boldsymbol{w} \in \mathcal{W}} p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w})$$

Definition 26 (Laplace Approximation)**.** *The Laplace approximation of a posterior density*

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{t}) \propto p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w})$$

is defined as

$$q(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{t}) := \mathcal{N}(\mu, \Sigma)$$

where

$$\mu = \boldsymbol{w}^{\hat{MAP}}, \quad \Sigma^{-1} = \frac{\partial^2 \log p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}) p(\boldsymbol{w})}{\partial \boldsymbol{w}^T \partial \boldsymbol{w}}\Big|_{\boldsymbol{w}=\boldsymbol{w}^{MAP}}$$

That is, we approximate the posterior density by a Gaussian centred at the maximum-at-posterior estimate of the posterior distribution and with a covariance matrix based on the curvature around the the MAP estimate.

5.3 Bayesian Regression

5.3.1 Bayesian Linear Regression

Let the likelihood function be as defined before:

$$p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{x}_n \boldsymbol{w}, \sigma^2 | \boldsymbol{x}_n) = \mathcal{N}(\boldsymbol{X}\boldsymbol{w}, \sigma^2 \boldsymbol{I} | \boldsymbol{X}).$$

Further, let the prior over the parameters be Gaussian as well:

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$$

Then, the posterior can be shown to be proportional to

$$p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{t}, \sigma^2) \propto p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \sigma^2) p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

where the posterior mean and covariance matrix are equal to

$$\begin{aligned}
\boldsymbol{\Sigma} &= \Big(\frac{1}{\sigma^2}\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{\Sigma}_0^{-1}\Big)^{-1} \\
\boldsymbol{\mu} &= \boldsymbol{\Sigma}\Big(\frac{1}{\sigma^2}\boldsymbol{X}^T\boldsymbol{t} + \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\Big)
\end{aligned}$$

Now, depending on the mean and covariance matrix of the prior, we can recover the OLS and Ridge estimators.

Proposition 5. *In the Bayesian Linear Regression model with i.i.d. responses whose likelihood is given by*

$$p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{w}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(\boldsymbol{x}_n \boldsymbol{w}, \sigma^2 | \boldsymbol{x}_n) = \mathcal{N}(\boldsymbol{X}\boldsymbol{w}, \sigma^2 \boldsymbol{I} | \boldsymbol{X}),$$

a prior over the parameters equal to $\mathcal{N}(0, \frac{\sigma^2}{N\lambda}\boldsymbol{I})$ gives rise to a posterior mean equal to the Ridge estimate when $\lambda > 0$ and to the OLS estimate when $\lambda >= 0$.

Proof.

$$\begin{aligned}
\boldsymbol{\mu} &= \boldsymbol{\Sigma}\Big(\frac{1}{\sigma^2}\boldsymbol{X}^T\boldsymbol{t} + \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\Big) \\
&= \Big(\frac{1}{\sigma^2}\boldsymbol{X}^T\boldsymbol{X} + \boldsymbol{\Sigma}_0^{-1}\Big)^{-1}\Big(\frac{1}{\sigma^2}\boldsymbol{X}^T\boldsymbol{t} + \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\Big) \\
&= \Big(\frac{1}{\sigma^2}\boldsymbol{X}^T\boldsymbol{X} + \frac{N\lambda}{\sigma^2}\boldsymbol{I}\Big)^{-1}\Big(\frac{1}{\sigma^2}\boldsymbol{X}^T\boldsymbol{t}\Big) \\
&= \sigma^2(\boldsymbol{X}^T\boldsymbol{X} + N\lambda\boldsymbol{I})^{-1}\frac{1}{\sigma^2}\Big(\boldsymbol{X}^T\boldsymbol{t}\Big) \\
&= (\boldsymbol{X}^T\boldsymbol{X} + N\lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{t},
\end{aligned}$$

which we recognise as the RIDGE estimate. Aetting $\lambda = 0$ yields the OLS estimate:

$$\boldsymbol{\mu} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{t}.$$

□

Making Predictions

In order to compute predictions, we need to calculate the predictive density:

$$p(\boldsymbol{t}^*|\boldsymbol{X}^*, \sigma^2) = \int_{\mathcal{W}} p(\boldsymbol{t}^*|\boldsymbol{X}^*, \boldsymbol{w}, \sigma^2)p(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{X}, \sigma^2)d\boldsymbol{w}$$

5.4 Bayesian Classification

5.4.1 Naive Bayes Classifier

Let there be K different classes $\{C_k\}_{k=1}^K$, and denote by $\boldsymbol{x}^*$ a new single D dimensional observation and by t^* its label. Then, we shall model the probability that $\boldsymbol{x}^*$ belongs to class C_k, $t^* = k$ as

$$\mathbb{P}(t^* = k|\boldsymbol{x}^*, \boldsymbol{t}, \boldsymbol{X}) = \frac{\mathbb{P}(\boldsymbol{x}^*|t^* = k, \boldsymbol{t}, \boldsymbol{X})\mathbb{P}(t^* = k)}{\sum_{j=1}^K \mathbb{P}(\boldsymbol{x}^*|t^* = j, \boldsymbol{t}, \boldsymbol{X})\mathbb{P}(t^* = j)}$$

Further, by assuming that the features/attributes of the each observation are i.i.d. Gaussians given the class, we can write down the likelihood of the data as

$$\begin{aligned}\mathbb{P}(\boldsymbol{x}^*|t^* = k, \boldsymbol{t}, \boldsymbol{X}) &= \prod_{d=1}^D \mathbb{P}(\boldsymbol{x}_d^*|t^* = k, \boldsymbol{t}, \boldsymbol{X}) \\ &= \prod_{d=1}^D \mathcal{N}(\boldsymbol{x}_d^*|\mu_{k,d}, \sigma_{k,d})\end{aligned}$$

That is, each attribute of the observation is independently and identically distributed according to a class dependent Gaussian distribution. If this assumption does not suit the task at hand, we can model the likelihood as a single D dimensional, class dependent Gaussian distribution

$$\mathbb{P}(\boldsymbol{x}^*|t^* = k, \boldsymbol{t}, \boldsymbol{X}) = \mathcal{N}(\boldsymbol{x}^*|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Prediction

For each data point, we can evaluate the K posterior distributions $\{\mathbb{P}(t^* = k|\boldsymbol{x}^*, \boldsymbol{t}, \boldsymbol{X})\}_{k=1}^K$ and decided to let $t^* = i$ whenever

$$i = \underset{k\in\{1,2,\ldots,K\}}{\arg\max}\ \mathbb{P}(t^* = k|\boldsymbol{x}^*, \boldsymbol{t}, \boldsymbol{X})$$

Remark 9. *This is an example of a generative model because we are modelling the data generating process for each class.*

Maximum Likelihood Estimates

Under the Maximum likelihood framework, we can compute the point estimate of $\mu_{k,d}, \sigma_{k,d}$ from the training data $(\boldsymbol{t}, \boldsymbol{X})$ by solving

$$(\hat{\mu}_{k,d}, \hat{\sigma}_{k,d}) = \underset{(\mu_{k,d},\sigma_{k,d})\in\mathbb{R}\times\mathbb{R}_+}{\arg\max} \mathcal{N}(\mu_{k,d}, \sigma_{k,d}|\boldsymbol{x}^k_{:,d}),$$

where $\boldsymbol{x}^k_{:,d}$ is the vector corresponding to the d^{th} column of $\boldsymbol{X}$ where $t = k$, which yields

$$\hat{\mu}_{k,d} = \frac{1}{N_k}\sum_{n=1}^{N_k} x^k_{n,d}$$
$$\hat{\sigma}_{k,d} = \frac{1}{N_k}\sum_{n=1}^{N_k}(x^k_{n,d} - \hat{\mu}_{k,d})^2$$

where N_k is the length of $\boldsymbol{x}^k_{:,d}$.

5.4.2 Logistic Regression

This is an example of a discriminative model in which the decision boundaries that separate the classes are modelled directly.

Consider the case binary-class case for simplicity. Assume that the log-odds ratios of class membership can be modelled as a linear function:

$$\begin{aligned}
\log\frac{\mathbb{P}(t=1|\boldsymbol{w},\boldsymbol{x}_n)}{\mathbb{P}(t=0|\boldsymbol{w},\boldsymbol{x}_n)} = \boldsymbol{x}_n\boldsymbol{w} &\implies \\
\frac{\mathbb{P}(t=1|\boldsymbol{w},\boldsymbol{x}_n)}{\mathbb{P}(t=0|\boldsymbol{w},\boldsymbol{x}_n)} = \exp(\boldsymbol{x}_n\boldsymbol{w}) &\implies \\
\mathbb{P}(t=1|\boldsymbol{w},\boldsymbol{x}_n) = \exp(\boldsymbol{x}_n\boldsymbol{w})(1-\mathbb{P}(t=1|\boldsymbol{w},\boldsymbol{x}_n)) &\implies \\
\mathbb{P}(t=1|\boldsymbol{w},\boldsymbol{x}_n) = \frac{\exp(\boldsymbol{x}_n\boldsymbol{w})}{(1+\exp(\boldsymbol{x}_n\boldsymbol{w}))} &\implies \\
\mathbb{P}(t=1|\boldsymbol{w},\boldsymbol{x}_n) = \frac{1}{(1+\exp(-\boldsymbol{x}_n\boldsymbol{w}))},&
\end{aligned}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is known as the logistic or sigmoid function. This can be extended to the multi-class case where number of classes is k. Doing so, yields the so called soft-max function:

$$\mathbb{P}(t=k|\boldsymbol{w}_k,\boldsymbol{x}_n) = \frac{\exp(\boldsymbol{x}_n\boldsymbol{w}_k)}{\sum_{j=1}^K \exp(\boldsymbol{x}_n\boldsymbol{w}_j)}$$

where $\boldsymbol{w}_k$ is the vector of coefficients associated to the k^{th} decision boundary, and $\boldsymbol{\sigma}(\boldsymbol{x}) = \left(\frac{\exp(x_1)}{\sum_{j=1}^K \exp(x_j)}, \ldots, \frac{\exp(x_K)}{\sum_{j=1}^K \exp(x_j)}\right)$ is the soft-max function which squashes a real valued, K dimensional vector $\boldsymbol{x}$ into a real valued, K dimensional vector $\boldsymbol{\sigma}(\boldsymbol{x})$ whose components add up to 1.

In Logistic Regression, we assume that the log-odds rations of class membership can be modelled as a linear function. As we have seen, this leads to having

$$\mathbb{P}(t=k|\boldsymbol{w}_k,\boldsymbol{x}_n) = \frac{\exp(\boldsymbol{x}_n\boldsymbol{w}_k)}{\sum_{j=1}^K \exp(\boldsymbol{x}_n\boldsymbol{w}_j)}.$$

Now, let the posterior distribution over the matrix of parameters $\boldsymbol{W}$ which define the decision boundaries be given by

$$p(\boldsymbol{W}|\boldsymbol{X}, \boldsymbol{t}) = \frac{p(\boldsymbol{t}|\boldsymbol{W}, \boldsymbol{X})p(\boldsymbol{W})}{p(\boldsymbol{t}|\boldsymbol{X})}$$

For simplicity, let us consider the binary-classification problem once again, and look at the form of the likelihood function under the i.i.d assumption.

$$\begin{aligned} p(\boldsymbol{t}|\boldsymbol{w}, \boldsymbol{X}) &= \prod_{n=1}^{N} p(t_n|\boldsymbol{w}, \boldsymbol{x}_n) \\ &= \prod_{n=1}^{N} \mathbb{P}(t_n = 1|\boldsymbol{w}, \boldsymbol{x}_n)^{t_n} (1 - \mathbb{P}(t_n = 1|\boldsymbol{w}, \boldsymbol{x}_n))^{1-t_n} \end{aligned}$$

from which we can observe that it follows a Bernoulli distribution with probability of success $\mathbb{P}(t_n = 1|\boldsymbol{w}, \boldsymbol{x}_n)$.

Depending on the choice of the prior distribution $p(\boldsymbol{t})$, we might not have conjugacy. As a consequence, we might want to resort to point estimates of the parameters of the posterior distribution, for example the Maximum-A-Posteriori estimates, approximations like the Laplace approximation, or sampling method such as MCMC.

Chapter 6

Sparse Kernel Machines

6.1 Support Vector Machines For Binary Classification

Support Vector Machines (SVMs) belong to a class of non-probabilistic classification models called maximum margin classifiers.

Definition 27 (Margin)**.** *We define the margin as the smallest distance between the decision boundary and any of the samples.*

6.1.1 Hard Margin SVMs

Suppose we want to find a linear decision boundary of the form

$$\boldsymbol{x_n w} + \boldsymbol{b} = 0$$

when having linearly separable train examples $\{\boldsymbol{x}_n, t_n\}_{n=1}^N$ such that $t_n \in \{-1, 1\}$, such that $\forall n$

$$\begin{aligned} \boldsymbol{x_n w} + b > 0 &\implies t_n = 1 \\ \boldsymbol{x_n w} + b < 0 &\implies t_n = -1 \end{aligned}$$

so that the margin between the two classes is maximised. Now, by recalling that the perpendicular distance of a point $\boldsymbol{x}_n$ to a hyper pane defined by $y(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{w} + b$ is given by

$$\frac{|y(\boldsymbol{x}_n)|}{||\boldsymbol{w}||_2}.$$

Furthermore, we are interested in an allocation such that all the points are correctly classified, that is, such that

$$t_n y(\boldsymbol{x}_n) > 0, \quad \forall n.$$

Thus, the distance from the point $\boldsymbol{x}_n$ to the decision hyperplane can be expressed as

$$\frac{t_n y(\boldsymbol{x}_n)}{||\boldsymbol{w}||_2} = \frac{t_n(\boldsymbol{x_n w} + b)}{||\boldsymbol{w}||_2}$$

The margin, is the perpendicular distance to the closest point $\boldsymbol{x}_n$ from decision boundary. Thus, the maximum margin solution is found by solving

$$\arg\max_{\boldsymbol{w},b} \Big\{ \frac{1}{||\boldsymbol{w}||_2} \min_n \{t_n(\boldsymbol{x_n}\boldsymbol{w} + b)\} \Big\}.$$

Direct solutions to this optimisation problem would be very complex, and so we shall convert it to an equivalent problem which is much easier to solve by noting that rescaling $\boldsymbol{w} \to k\boldsymbol{w}$, and $b \to kb$ will not change the perpendicular distance from any point to the decision boundary. This freedom, allows us to set

$$t_n(\boldsymbol{x_n}\boldsymbol{w} + b) = 1,$$

so that

$$\begin{aligned} \boldsymbol{x_n}\boldsymbol{w} + b > 1 &\implies t_n = 1 \\ \boldsymbol{x_n}\boldsymbol{w} + b < -1 &\implies t_n = -1, \end{aligned}$$

which in turns our optimisation problem into

$$\begin{aligned} &\arg\max_{\boldsymbol{w},b} \frac{1}{||\boldsymbol{w}||_2} \\ &\quad \text{s.t. } t_n(\boldsymbol{x_n}\boldsymbol{w} + b) \geq 1, \quad \forall n. \end{aligned}$$

This in turn is equivalent to

$$\begin{aligned} &\arg\min_{\boldsymbol{w},b} \frac{1}{2}||\boldsymbol{w}||_2^2 \\ &\quad \text{s.t. } t_n(\boldsymbol{x_n}\boldsymbol{w} + b) \geq 1, \quad \forall n. \end{aligned}$$

in order to solve this quadratic optimisation problem, we introduce Lagrange multipliers $\{\alpha_n \geq 0\}_{n=1}^N$, giving the Lagrange function

$$L(\boldsymbol{w}, \boldsymbol{\alpha}, b) = \frac{1}{2}||\boldsymbol{w}||_2^2 - \sum_{n=1}^{N} \alpha_n (t_n(\boldsymbol{x_n}\boldsymbol{w} + b) - 1)$$

Note the minus sign is in front of the Lagrangian multipliers because we are minimising w.r.t. $\boldsymbol{w}$ and b, but we are maximising with respect to α since we want as the classifications to be correct, that is we want $t_n(\boldsymbol{x_n}\boldsymbol{w} + b) \geq 1$. By setting to zero the partial derivatives w.r.t $\boldsymbol{w}$ and b we get the following two conditions:

$$\begin{aligned} \frac{\partial L(\boldsymbol{w}, \boldsymbol{\alpha}, b)}{\partial \boldsymbol{w}} = 0 &\implies \boldsymbol{w} = \sum_{n=1}^{N} \alpha_n t_n \boldsymbol{x}_n \\ \frac{\partial L(\boldsymbol{w}, \boldsymbol{\alpha}, b)}{\partial b} = 0 &\implies \sum_{n=1}^{N} \alpha_n t_n = 0. \end{aligned}$$

Using the conditions above to get rid of the dependencies on $\boldsymbol{w}$ and b from $L(\boldsymbol{w}, \boldsymbol{\alpha}, b)$ leads to the dual formulation of the problem:

$$\begin{aligned} &\arg\max_{\boldsymbol{\alpha}} \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m t_n t_m \boldsymbol{x}_n \boldsymbol{x}_m^T \\ &\quad \text{s.t. } \alpha_n \geq 0, \quad \sum_{n=1}^{N} \alpha_n t_n = 0, \quad \forall n. \end{aligned}$$

6.1.2 Soft Margin SVMs

Let us introduce the slack variables $\xi_n \geq 0$ to allow for misclassification errors to occur. That is, allow non linearly separable dataset to be dealt with. The resulting optimisation problem and the associated Lagrange function are

$$\begin{aligned} &\arg\min_{\boldsymbol{w},b} ||\boldsymbol{w}||_2^2 + C\sum_{n=1}^{N}\xi_n \\ &\quad \text{s.t. } t_n(\boldsymbol{x_n}\boldsymbol{w}+b) \geq 1-\xi_n, \quad \xi_n \geq 0 \quad \forall n \end{aligned}$$

$$L(\boldsymbol{w},\boldsymbol{\alpha},b) = ||\boldsymbol{w}||_2^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n(t_n(\boldsymbol{x_n}\boldsymbol{w}+b)-1) - \sum_{n=1}^{N}\mu_n\xi_n,$$

where $\{\mu_n \geq 0\}_{n=1}^{N}$ are the Lagrange multipliers associated to $\{\xi_n \geq 0\}_{n=1}^{N}$.

By setting to zero the partial derivatives w.r.t $\boldsymbol{w}$,ξ_n, and b we get the following three conditions:

$$\begin{aligned} \frac{\partial L(\boldsymbol{w},\boldsymbol{\alpha},b)}{\partial \boldsymbol{w}} = 0 &\implies \boldsymbol{w} = \sum_{n=1}^{N}\alpha_n t_n \boldsymbol{x}_n \\ \frac{\partial L(\boldsymbol{w},\boldsymbol{\alpha},b)}{\partial b} = 0 &\implies \sum_{n=1}^{N}\alpha_n t_n = 0. \\ \frac{\partial L(\boldsymbol{w},\boldsymbol{\alpha},b)}{\partial \xi_n} = 0 &\implies \alpha_n = C - \mu_n. \end{aligned}$$

Using the conditions above to get rid of the dependencies on $\boldsymbol{w}$ and b from $L(\boldsymbol{w},\boldsymbol{\alpha},b)$ leads to the dual formulation of the problem:

$$\begin{aligned} &\arg\max_{\boldsymbol{\alpha}} \sum_{n=1}^{N}\alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m t_n t_m \boldsymbol{x}_n\boldsymbol{x}_m^T \\ &\quad \text{s.t. } 0 \leq \alpha_n \leq C, \quad \sum_{n=1}^{N}\alpha_n t_n = 0, \quad \forall n. \end{aligned}$$

6.1.3 Kernel SVMs

If one looks at the dual formulation of the optimisation problem in both the hard and soft margin SVM, it is possible to notice that the data enters the loss function via an inner product. Thus, we can substitute $\boldsymbol{x}_n\boldsymbol{x}_m^T$ with $\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_m)^T = k(\boldsymbol{x}_n\boldsymbol{x}_m^T)$ in the hope that in this new space, the data gets closer to being linearly separable.

We call $k(\boldsymbol{x}_n\boldsymbol{x}_m^T)$ the kernel function and popular choices for such a function are:

Linear Kernel : $\boldsymbol{x}_n^T\boldsymbol{x}_m$

Radial Basis Function Kernel (RBF) : $\exp\{-\beta(\boldsymbol{x}_n-\boldsymbol{x}_m)(\boldsymbol{x}_n-\boldsymbol{x}_m)^T\}$

Polynomial Kernel (RBF) : $(1+\boldsymbol{x}_n\boldsymbol{x}_m^T)^\beta$

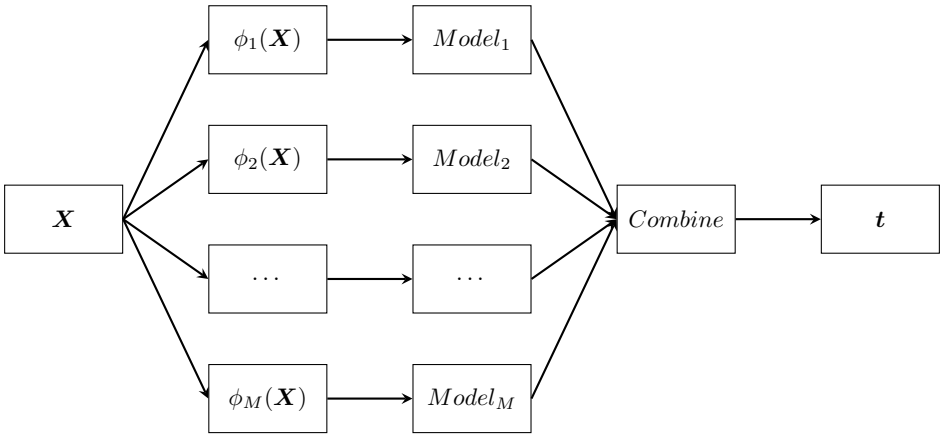
Chapter 7

Ensemble learning

Ensemble learning consists of combining multiple models to improve predictive performance. Here we present four kinds of ensembles: (i) Output combination; (ii) Staking; (iii) Boosting; (iv) Bagging. Throughout this chapter, let $\phi(\boldsymbol{X})$ denote a feature space.

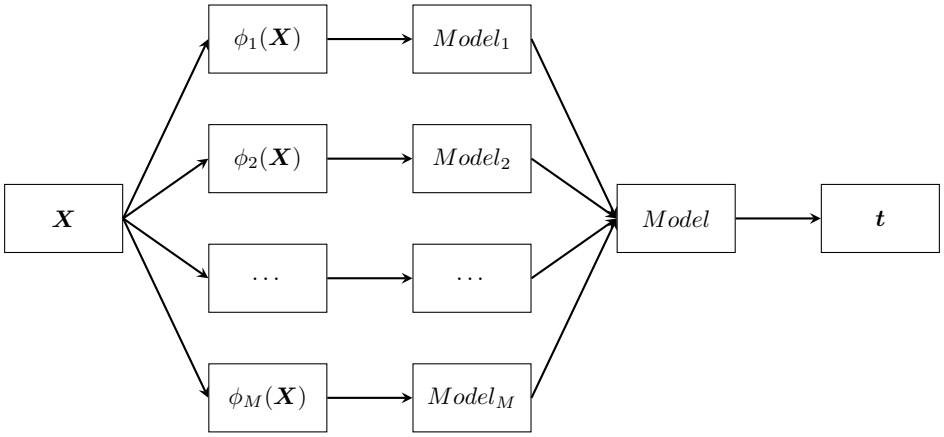
7.1 Combination Of Model Output

In this kind of ensemble, M base models, possibly from different families, are trained on M, possibly different, dataset generated from different feature engineering techniques. Consequently, the output of these models is combined via some sort of voting scheme.



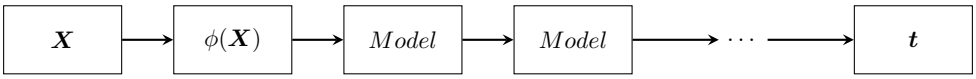
7.2 Stacking

Stacking also called Blending, and Meta learning, consists of training M base models, possibly from different families, on M, possibly different, dataset generated from different feature engineering techniques. Consequently, the output of these models is used to train a model, sometimes called the meta learner which will learns weighted combinations of the base models' outputs.



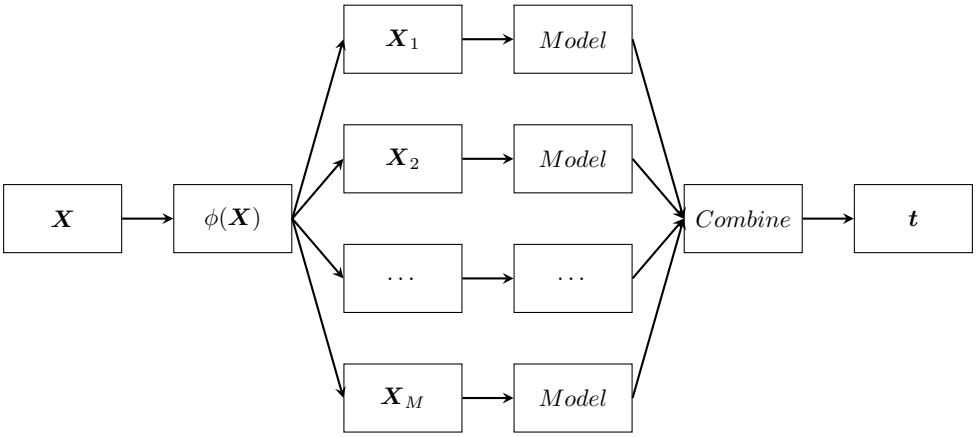
7.3 Boosting

In this kind of ensemble, models are trained sequentially on the output of the previous model. Usually, the models involved in this kind of ensembles are weak learners - models which tend to underfit.



7.4 Bagging

Bagging also called bootstrap aggregating, consists of training M models on M different bootstrap-with-replacement samples. The output of these models is then combined via some sort of voting scheme.



Bagging is particularly effective when the base models are high variance models, so that the final estimate has a lower overall variance because of the "averaging" that takes place at the end.

7.5 Combinations Schemes

7.5.1 Probabilistic Output

Classification

Sum Rule : $p_e(t^* = k|\boldsymbol{x}^*, \dots) = \frac{1}{M}\sum_{m=1}^{M} p_m(t^* = k|\boldsymbol{x}^*, \dots)$

Product Rule : $p_e(t^* = k|\boldsymbol{x}^*, \dots) = \frac{1}{M}\prod_{m=1}^{M} p_m(t^* = k|\boldsymbol{x}^*, \dots)$. Note the product rule assumes independence of base models.

7.5.2 Non-Probabilistic Output

Classification

Voting : for example, $(\boldsymbol{t}_e)_i = majority\{(\boldsymbol{t}_1)_i, \dots, (\boldsymbol{t}_M)_i\}$.

Regression

Linear Combination : $\boldsymbol{t}_e = \sum_{m=1}^{M} \boldsymbol{w}_m \boldsymbol{t}_m$. Note, if we let $\boldsymbol{w}_m$ be the OLS or Ridge coefficients then this becomes an example of Staking.

7.6 Mean Square Error Of An Ensemble Models

In a previous section, we have introduced the notion of Mean Squared Error of an estimator and we saw how it can be decomposed in terms of bias and variance. When an estimator is an ensemble, this decomposition is no longer valid as we need to factor in the possible correlations between the various base models.

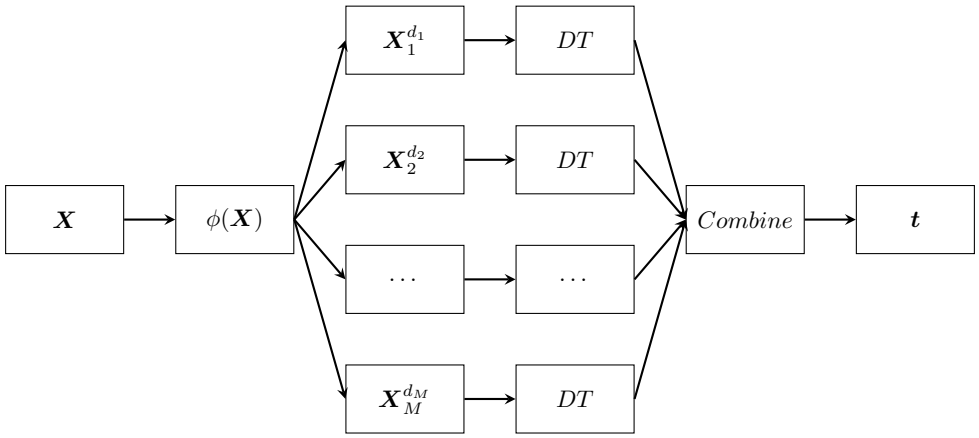
Proposition 6. *The MSE of an ensemble $\hat{\theta}^e$, composed of M base models, with respect to an unknown parameter θ can be decomposed in terms of Variance, Bias, and Covariance:*

$$\begin{aligned}
MSE(\hat{\theta}^e) =& \mathbb{E}((\hat{\theta}^e - \theta)^2) \\
=& \Big(\frac{1}{M}\sum_{m=1}^{M} Bias(\hat{\theta}_m)\Big)^2 + \frac{1}{M}\sum_{m=1}^{M} Var(\hat{\theta}_m) + \Big(1 - \frac{1}{M}\Big) Cov(\hat{\theta}^e) \\
=& \Big(\sum_{m=1}^{M} \mathbb{E}(\hat{\theta}_m - \theta)\Big)^2 + \\
& \frac{1}{M}\sum_{m=1}^{M} \mathbb{E}((\hat{\theta}_m - \theta)^2) + \\
& \Big(1 - \frac{1}{M}\Big) \sum_{m=1}^{M}\sum_{l \neq m}^{M} \mathbb{E}((\hat{\theta}_m - \theta)(\hat{\theta}_l - \theta))
\end{aligned}$$

where $\hat{\theta}^e$ is the overall ensemble model and $\hat{\theta}_k i, k \in \{m, l\}$ are the base models.

7.7 An Ensemble Model: Random Forests

One example of an ensemble model is the Random Forests model. This model is based on Bagging, and it consists of training M decision trees on M samples of the data where not only the observations but also the features are bootstrap samples. That is the M decision trees are trained on $\boldsymbol{X}_j^{d_i}$ where j denotes the j^{th} bootstrap sample of the data and d_i denotes the i^{th} sampled subset of attributes.



This ensemble model has proven itself to be very effective because it tries to reduce the variance, bias and covariance of the final estimator. Decision trees are high variance low bias estimators hence, we are already assured a low bias score. Now, the fact that we are not training a single decision tree but M of them will reduce the overall variance of the estimator because of the aggregation of the base models output taking place at the end. Further, we hope that sampling both the observations and their attributes will generate fairly uncorrelated base models and as a consequence a low covariance.

Chapter 8

Supervised Learning Performance Metrics

8.1 Regression

Definition 28 ($RMSE$)**.** *The Root Mean Squared Error (RMSE) is defined as*

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(t_n - f(\boldsymbol{x}_n; \boldsymbol{w}))^2}.$$

The lower the $RMSE$ is the better.

Definition 29 (R^2)**.** *The coefficient of determination R^2 is defined as*

$$R^2 = 1 - \frac{\sum_{n=1}^{N}(t_n - f(\boldsymbol{x}_n; \boldsymbol{w}))^2}{\sum_{n=1}^{N}(t_n - \bar{t})^2}$$

where $\bar{t} = \frac{1}{N}\sum_{n=1}^{N} t_n$. The higher this number the better. Note how the R^2 is bounded above by 1, a scenario which occures when the predicted targets are equal to actual ones.

Definition 30 (Pearson sample correlation coefficient)**.** *The Pearson sample correlation coefficient $r(\boldsymbol{t}, \hat{\boldsymbol{t}})$ is defined as*

$$r(\boldsymbol{t}, \hat{\boldsymbol{t}}) = \sqrt{\frac{\sum_{n=1}^{N}(\hat{t}_n - \bar{t})^2}{\sum_{n=1}^{N}(t_n - \bar{t})^2}}$$

where $\bar{t} = \frac{1}{N}\sum_{n=1}^{N} t_n$. This quantity captures the percentage of the variation explained by the model. higher this number the better.

8.2 Classification

8.2.1 Binary Classification

There exists several performance metrics derived form the binary confusion matrix.

Definition 31 (Binary Confusion Matrix)**.** *The binary confusion matrix is constructed in the following manner:*

	True label +	True label −
Predicted +	True Positives (TP)	False Positives (FP)
Predicted −	False Negatives (FN)	True Negatives (TN)

Definition 32 (Precision)**.** *Precision is defined as*

$$Precision = \frac{TP}{TP + FP}$$

Definition 33 (Recall)**.** *Recall is denied as*

$$Recall = \frac{TP}{TP + FN}$$

Definition 34 (F1 Score)**.** *The F1 score is defined as*

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = 2\frac{Precision \times Recall}{Precision + Recall}.$$

The higher this number the better. An $F1$ score equal to 1 indicates perfect classification this is because $F1 = 1$

$$F1 = 1 \implies \frac{1}{Precision} + \frac{1}{Recall} = 2 \implies Precision + Recall = 2$$
$$Precision = 1 \implies FP = 0, Recall = 1 \implies FN = 0$$

which in turn implies that we have classified all the observations correctly.

Definition 35 (Accuracy)**.** *Accuracy is defined as*

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Definition 36 (ROC Curve)**.** *Define $Specificity$ and $Sensitivity$ as*

$$Specificity = \frac{TN}{TN + FP}$$
$$Sensitivity = Recall = \frac{TP}{TP + FN}$$

Then, the Receiver Operating Characteristic (ROC) curve of a model M is defined as all the points

$$\{(1 - Specificity(M_\alpha), Sensitivity(M_\alpha)) \in [0,1]^2 | \alpha \text{ being a decison threshold for } M\}$$

The best model would have a very high $Specificity$ and $Sensitivity$ thus living in the upper left corner of the plot. Anything above the 45 degree line is better than random guessing, anything below is worst than random guessing.

8.2.2 Multi-Class Classification

Definition 37 (Multi-Class Confusion Matrix)**.** *The multi-class confusion matrix is constructed in the following manner:*

	True Class 1	True Class 2	...	True Class K
Predicted Class 1	$p_{1,1}$	$p_{1,2}$	...	$p_{1,K}$
Predicted Class 2	$p_{2,1}$	$p_{2,2}$	...	$p_{2,K}$
...	...	...	...	
Predicted Class K	$p_{K,1}$	$p_{K,2}$	...	$p_{K,K}$

Part II

Unsupervised Learning

Chapter 9

Clustering

The goal of clustering is to extract patterns from data without explicit supervision. It is worth making a few remarks on clustering:

1. Clustering is an ill-defined problem because there might not be unique solution to the problem. Most of the time, it depends on the context in which the data is used.
2. Distance and similarity metrics play a key role and as a consequence, the output of a model can vary greatly depending on the chosen metric.
3. Clustering can be used before some supervised methods. For example, it could be used to determine the number of clusters to use in a classification problem.

9.1 Hierarchical Clustering

We distinguish between two kinds of Hierarchical Clustering:

Agglomerative : at the first iteration each observation is considered as a separate cluster. Then clusters are successively merged based on similarity until only one cluster remains

Divisive : at the first iteration all the observations belong to the same cluster. Then, clusters are successively divided based on similarity until every observation makes up its own cluster.

Algorithm 4 Agglomerative Hierarchical Clustering

1: Choose the distance metric and the linkage method.
2: Create one cluster for each observation, and create an $N \times N$ distance matrix.
3: **for** $i \in \{1, 2, \ldots, N-1\}$ **do**
4: Find the two most similar cluster and merge them.
5: Update the distance matrix by computing the distance between the merged cluster and all other clusters.
6: **end for**

This kind of clustering method requires computing distances between clusters, which is not trivial when clusters have more than one element. The method to compute the distance between two clusters is called the linkage method. Four common choices are:

Single linkage : the distance between two clusters is defined to the be distance between the two nearest elements belonging to those clusters. Single linkage is unaffected by monotonic changes in the input distances. This linkage method can produce very elongated clusters where some elements are very far apart/different from each other.

Complete linkage : the distance between two clusters is defined to the be distance between the two farthest elements belonging to those clusters. Complete linkage is unaffected by monotonic changes in the input distances. This linkage method forces clusters where no two elements are very far away/different from each other.

Average linkage : the distance between two clusters is defined to be the average of all pairwise distances between those two clusters. Average linkage is affected by non-linear changes in the input distances. This linkage method enforces clusters where most elements are nearby, and penalises clusters where two elements are far apart.

Ward linkage : the distance between two clusters is defined to be the increase in within-cluster sum-of-squares. That is, the two clusters to be merged are those minimising the increase in within-cluster sum-of-squares. Average linkage is affected by non-linear changes in the input distances. This linkage method enforces spherical clusters.

Advantages of Hierarchical Clustering

1. When coupled with a dendrogram, it gives an idea about the number of clusters in the data.
2. Only requires inputs and distances.

Disadvantages of Hierarchical Clustering

1. It can be very sensitive to the choice of distance metric and linkage method.
2. Will not-reallocate elements assigned to the wrong cluster in early iterations - no backtracking.
3. It has not probabilistic interpretations. All the assignments are hard in assignments.

9.2 K-Means Algorithm

9.2.1 Vanilla K-Means Algorithm

Algorithm 5 K-Means

1: Choose the number of clusters K
2: Initialise the centroids $\{\boldsymbol{\mu}_k\}_{k=1}^K$
3: Assign each $\boldsymbol{x}_n$ to its closest $\boldsymbol{\mu}_k$ according to the chosen distance metric.
4: **if** the closest centroid to $\boldsymbol{x}_n$ is $\boldsymbol{\mu}_k$ **then**
5: let $z_{n,k} = 1$
6: **else**
7: let $z_{n,k} = 0$
8: **end if**
9: update the centroids such that

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^N z_{n,k}\boldsymbol{x}_n}{\sum_{n=1}^N z_{n,k}}$$

10: return to 2 until the assignments do not change.

9.2.2 Kernel K-Means Algorithm

Algorithm 6 K-Means

1: Choose the number of clusters K
2: Choose a kernel function $k(\boldsymbol{x}_m\boldsymbol{x}_n^T)$ and its parameters
3: Initialise the assignments $z_{n,k}$
4: **for** each $\boldsymbol{x}_n$ **do**
5: assign it to the nearest centroid, where distance is defined as

$$k(\boldsymbol{x}_m\boldsymbol{x}_n^T) - 2N_k^{-1}\sum_{m=1}^N z_{m,k}k(\boldsymbol{x}_m\boldsymbol{x}_n^T) + N_k^{-2}\sum_{m=1}^N\sum_{l=1}^N z_{m,k}z_{l,k}k(\boldsymbol{x}_m\boldsymbol{x}_l^T)$$

6: **end for**
7: return to 4 until the assignments do not change.

Note: the kernalised distance formula is obtained by substituting the centroid $\boldsymbol{\mu}_k$ as calculated in the vanilla K-Means Algorithm into the euclidean distance between $\boldsymbol{x}_n$, and $\boldsymbol{\mu}_k$.

Advantages of K-Means

1. Flexibility: it can reallocate observations at each iteration.
2. Has nice connections with model-based clustering.

Disadvantages of K-Means

1. The number of clusters needs to be specified a-priori

2. Computing the centroids requires coordinated therefore, a distance matrix is not enough.

3. It is sensitive to the choice of distance metric.

9.3 Gaussian Mixture Models

We start by assuming that the data can be described by a mixture of K distributions, where K is the chosen number of clusters, so that

$$p(\boldsymbol{x}_n) = \sum_{k=1}^{K} p(\boldsymbol{x}_n | z_{n,k} = 1)\mathbb{P}(z_{n,k} = 1)$$

where once again $z_{n,k} = 1$, if $\boldsymbol{x}_n$ is assigned to the k^th cluster and 0 otherwise. Now, by assuming i.i.d. data, the joint likelihood is given by

$$p(\boldsymbol{X}) = \prod_{n=1}^{N} \sum_{k=1}^{K} p(\boldsymbol{x}_n | z_{n,k} = 1)\mathbb{P}(z_{n,k} = 1)$$

Then, by assuming that data was generated form a mixture of Gaussian distributions each having a responsibility weight $\pi_k = \mathbb{P}(z_{n,k} = 1)$, the joint likelihood of the data can be written as

$$p(\boldsymbol{X}) = \prod_{n=1}^{N} \sum_{k=1}^{K} \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\mathbb{P}(z_{n,k} = 1)$$

We can then choose $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ so that the likelihood of the data is maximised via the Expectation-Maximisation algorithm.

Advantages of Model-Based Clustering

1. Cluster probabilities measure uncertainty in cluster allocations.

2. The number of clusters is a parameter of the model and it can be selected using standard model choice criteria.

3. Model assumptions can be checked.

Disadvantages of Model-Based Clustering

1. Computational complexity.

Chapter 10

Dimensionality Reduction

10.1 Principal Component Analysis

The goal of Principal Component Analysis (PCA) is to transform the samples $\boldsymbol{X}$ into $\boldsymbol{Z}$. Such that:

- $\boldsymbol{Z}$ is a linear combination of $\boldsymbol{X}$, i.e. $\boldsymbol{Z} = \boldsymbol{W}\boldsymbol{X}$.
- $\boldsymbol{z}_d$ is orthogonal to $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{d-1}$
- $\boldsymbol{z}_1$ preserves as much variance in $\boldsymbol{X}$ as possible. Conditional on $\boldsymbol{z}_1$, $\boldsymbol{z}_2$ preserves as much variance in $\boldsymbol{X}$ as possible. Conditional on $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, $\boldsymbol{z}_3$ preserves as much variance in $\boldsymbol{X}$ as possible and so on.

where $\boldsymbol{z}_d$ is a row vector denoting the d^{th} column of $\boldsymbol{Z}$

We call the $\{\boldsymbol{z}_d\}_{d=1}^D$ the principal components of $\boldsymbol{X}$ and they can be shown to be equal to $\{\boldsymbol{e}_d^T\boldsymbol{X}\}_{d=1}^D$ where $\{\boldsymbol{e}_d\}_{d=1}^D$ are the unit eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$. Furthermore, the amount of variance explained by $\boldsymbol{z}_d$ is equal to λ_d where $\{\lambda_d\}_{d=1}^D$ are the eigenvalues associated to $\{\boldsymbol{e}_d\}_{d=1}^D$. Note, we order the PCs in descending order based on the size of the associated eigenvalue.

10.2 Multidimensional Scaling

The goal of multidimensional scaling is to find a low dimensional representation of the data in which pairwise distances are good approximations of the pairwise distances in the original space.

10.2.1 Classical Multidimensional Scaling

Let $d_{i,j}$, and $\delta_{i,j}$ denote the distance between i and j in the new space and the original space respectively. Then, in order to find an arrangement which preserves pairwise distances in the new space as well as possible, we minimise a loss function which measures the stress S of the allocations:

$$\begin{aligned} S &= \frac{\sum_{i,j}(d_{i,j} - \delta_{i,j})^2}{\sum_{i,j} d_{i,j}} \\ &= \frac{\sum_{i<j}(d_{i,j} - \delta_{i,j})^2}{\sum_{i<j} d_{i,j}} \end{aligned}$$

where the denominator is a normalising constant which makes the solution scale invariant, and the last equality is due to the fact that distances are symmetric,

10.2.2 Metric Multidimensional Scaling

In metric multidimensional scaling, we allow the pairwise distances in the new space to be a function of the distances in the original space. That is, we let

$$d_{i,j} \approx f(\delta_{i,j})$$

for some function f. Then, the stress function which needs minimising becomes

$$S = \frac{\sum_{i<j}(d_{i,j} - f(\delta_{i,j}))^2}{\sum_{i<j} d_{i,j}}$$

10.2.3 Ordinal Multidimensional Scaling

Ordinal multidimensional scaling deals with cases where the data can only be ranked. As a consequence, we need to impose that

$$d_{i,j} \approx f(\delta_{i,j}), \quad \text{s.t.} \quad d_{i_1,j_1} \leq d_{i_2,j_2} \leq \cdots \leq d_{i_N,j_N}.$$

Chapter 11

Unsupervised Learning Performance Metrics

11.1 Clustering

Definition 38 (Within Cluster Sums Of Squares)**.** *Let N_k be the number of observations in cluster C_k and denote by $\boldsymbol{m}_k$ the mean of cluster C_k then the within cluster sums of squares id defined by*

$$WSS = \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in C_k} ||\boldsymbol{x} - \boldsymbol{m}_k||_2^2,$$

where

$$\boldsymbol{m}_k = \frac{1}{N_k} \sum_{\boldsymbol{x} \in C_k} \boldsymbol{x}$$

Definition 39 (Between Cluster Sums Of Squares)**.** *Let N_k be the number of observations in cluster C_k and denote by $\boldsymbol{m}_k$ the mean of cluster C_k then the between clusters sums of squares id defined by*

$$BSS = \sum_{k=1}^{K} N_k ||\bar{\boldsymbol{x}} - \boldsymbol{m}_k||_2^2,$$

where

$$\boldsymbol{m}_k = \frac{1}{N_k} \sum_{\boldsymbol{x} \in C_k} \boldsymbol{x}$$

$$\bar{\boldsymbol{x}} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n$$

Definition 40 (Silhouette Score)**.** *Define $a(\boldsymbol{x}_n)$ to be the average distance between $\boldsymbol{x}_n$ and the points in its cluster, and let $b(\boldsymbol{x}_n)$ be the minimum of the average distances between $\boldsymbol{x}_n$ and any other point not in its cluster. Then, the Silhouette Score for data point $b(\boldsymbol{x}_n)$ is defined as*

$$S(\boldsymbol{x}_n) = \frac{b(\boldsymbol{x}_n) - a(\boldsymbol{x}_n)}{\max\{b(\boldsymbol{x}_n), a(\boldsymbol{x}_n)\}},$$

where $-1 \leq S(\boldsymbol{x}_n) \leq 1$ by construction. The higher this number the better.

Appendix A

Mathematical Identities

A.1 Vector Differentiation

Table A.1:

$f(\boldsymbol{w})$	$\partial f(\boldsymbol{w})/\partial \boldsymbol{w}$
$\boldsymbol{w}^T\boldsymbol{a}$	$\boldsymbol{a}$
$\boldsymbol{a}^T\boldsymbol{w}$	$\boldsymbol{a}$
$\boldsymbol{w}^T\boldsymbol{w}$	$2\boldsymbol{w}$
$\boldsymbol{w}^T\boldsymbol{C}\boldsymbol{w}$	$2\boldsymbol{C}\boldsymbol{w}$