Machine Learning

Semester 1, 2024

Dr Simon Denman

Tarang Janawalkar

Contents

1 Introduction

Machine learning is a field of computer science concerned with the development of statistical algorithms that enable computer systems to learn insights from data. Machine learning is a multi-disciplinary field that is related to statistics, pattern recognition, data mining, and artificial intelligence.

1.1 Machine Learning Process

Broadly speaking, there are three steps to enable machines to learn.

1. **Input Data**: A model must be provided with input data. This data consists of audio, images, text, or any other form of data, which enables us to build a model. Ideally, it is desirable to have a large amount of data, as this produces a more accurate model, and removes the possibility of overfitting when many features are present.
2. **Data Abstraction**: To effectively train a model on this data, it needs to be abstracted into a consistent format that can be understood by a model. This includes pre-processing the data, removing any noise or irrelevant features, and ensuring that the dimensionality (or the number of features) is the same across all samples.
3. **Generalisation**: When data is ready, we can use a machine learning technique to determine whether a model generalises well to new data.

1.2 Machine Learning Paradigms

There are three main types of machine learning paradigms:

- **Supervised Learning**: In this type of learning, a model is provided both input and expected output data. The purpose here is to learn a mapping from input to output, to predict an output for new unseen data.

  For example, we might provide a model a video of pedestrians with a person count in each frame. The model then regresses features such as size, texture, and shape to predict the number of people in unseen frames.

  Another related example may include object tracking, where a model learns to track objects in a video, given a pre-labelled set of frames with tracked objects.

- **Unsupervised Learning**: In this type of learning, a model is provided only input data, and is mainly used for knowledge discovery—to find order and characteristics in data.

  An example of this is clustering, where a model might group people who are walking together, based on their motion characteristics.

  Another example is the detection of anomalies or abnormal behaviour in data. For example, a model may be trained on pedestrian motion, and use unsupervised learning to track vehicles and bicycles in a pedestrian zone.

- **Reinforcement Learning**: In this type of learning, a model learns to make decisions by interacting and responding to an environment. The model is rewarded or penalised based on its actions, and its goal is to learn the best sequence of actions to maximise its reward.

An example of this may be a robot learning to walk. The objective might be to maximise the distance travelled in one direction.

1.3 Importance of Data

Data is the most important aspect of machine learning, both in terms of **quality and quantity**. A high-quality dataset that is well prepared and which covers all possible cases considered in a model, leads to better outcomes.

- a model cannot produce predictions that are better than the data it is trained on
- a model cannot compensate for errors in the annotation of data

It is also important to consider **data diversity** to ensure that a model does not learn **biases** present in the data. For example, a male dominated dataset may lead to a model that is biased against women.

Data also suffers from the *curse of dimensionality*. Datasets often have:

- a large number of dimensions or features (i.e., columns, observed variables)
- but a small number of samples (i.e., rows, observations)

In such cases, the feature space is sparsely populated and the model may **overfit**. Every new feature increases the complexity of the model, and also necessitates more data, to produce an accurate model—this number depends on the type of classifier or learning algorithm used, and therefore a model should be kept simple.

1.4 Data Splitting

To efficiently train and evaluate a model, we can split the data into three sets:

- **Training Set**: This set is used to train the model.
- **Validation Set**: This set is used to tune model hyperparameters to evaluate a model's performance. The training set will be trained using different hyperparameters to evaluate which set of hyperparameters produce the best model.
- **Test Set**: This set is used to evaluate the model's performance on unseen data.

This approach uses *holdout validation* where data is *held out* for validation and/or testing, so that the training, validation, and testing stages are completely separate. Note that this requires datasets to be large enough to ensure that a model has enough data for training.

When **insufficient data** is available, *cross-validation* can be used to dynamically split the dataset into training and validation sets (i.e., a 80% training/20% validation split). This may be repeated 5 times, so that each split is used as a validation set. The best model is then selected based on the average performance across all splits.

In some cases, machine learning may not be possible if the dataset is too small. This may be because the data does not contain the patterns and relationships that we are trying to learn. For some problems, this threshold may be a few hundred samples, while for others, it may be a few million. Extrapolation beyond the data may also be problematic in such cases.

1.5 Traditional Machine Learning to Deep Learning

1.5.1 Traditional Machine Learning

The traditional machine learning pipeline typically consists of:

- **Pre-processing**: Preparing data for a model using normalisation, scaling, noise reduction, etc.
- **Feature Extraction**: Extracting features from data; MFCC (audio), HOG (image/video), and bag-of-words (text).

  This step is often informed by the task. For example, if we need to recognise shapes in images, we might use a technique that captures edge detection. In the case of audio, we might use a technique that captures frequency information.
- **Machine Learning**: Passing the features to a model using a machine learning technique.

In this model, features are **hand-crafted**, and based on domain-specific knowledge. Choosing the optimal features for a given problem can be a tedious and iterative process. This leads to different formulations for different tasks, and may significantly increase the complexity of a problem, especially when we need to extract multiple sets of features.

1.5.2 An Aside on Neural Networks

A neural network is modelled after the human brain and nervous system. It is built on a single unit or **node** called a *neuron*, in which data is passed through a series of **layers**, each of which is connected to other neurons through **edges**. Neural networks typically consist of a large number of parameters and many interconnections.

Choosing the appropriate number of layers in a neural network (or the *depth* of a neural network), is problem dependent. A higher depth allows us to learn *higher level features*, and contains more parameters, but this is often harder to learn and also increases the likelihood of overfitting. Larger networks also require more data, and therefore more memory and computational resources.

1.5.3 Deep Learning

Deep learning is a subset of machine learning that uses neural networks to learn features from data. It differs from the traditional machine learning pipeline in that it does not require manual feature extraction, but instead learns it's own representation from pre-processed data. This makes deep learning more **adaptable** to different tasks, and also reduces the complexity of a problem.

A downside of deep learning is that models are often extremely large compared to traditional approaches;

- a traditional machine learning model may have a few hundred parameters
- a deep learning model may have millions of parameters

This leads to an increase in computational resources required to train and evaluate a model, and also makes a model difficult to interpret.

In deep learning, feature engineering is replaced with **network engineering**, where we must carefully choose the appropriate architecture for a problem. This includes:

- the number of layers,
- the number of neurons in each layer, and
- the type of activation function used.

This is often an iterative process, and it may not be feasible to find the optimal architecture for a given problem, through exhaustive testing.

2 Linear Regression

A simple model for predicting the relationship between a dependent variable y (**response**), and an independent variable x (**predictor**), is a straight line. Given a set of n input-output pairs $\left(x^{(i)},\, y^{(i)}\right)$, we can model a linear relationship between x and y as:

$$y^{(i)} = \beta_0 + \beta_1 x^{(i)}$$

where β_0 is the y-**intercept** and β_1 is the **gradient** of the line. In practice, data rarely fits a model perfectly and therefore, we will assume the relationship:

$$\hat{y}\,(x) = \hat{\beta}_0 + \hat{\beta}_1 x$$

where $\hat{y}$ is known as the **hypothesis function** which **predicts** value of y, and $\hat{\beta}_0$ and $\hat{\beta}_1$ are the **estimated** values of the **parameters** β_0 and β_1, respectively. This model adds a **residual** term ϵ to account for the error in each observation:

$$\begin{aligned} y^{(i)} &= \hat{y}\left(x^{(i)}\right) + \epsilon^{(i)} \\ &= \hat{\beta}_0 + \hat{\beta}_1 x^{(i)} + \epsilon^{(i)} \end{aligned}$$

here ϵ is assumed to be normally distributed with zero mean and σ^2 variance:

$$\epsilon \overset{\text{iid}}{\sim} \mathrm{N}\left(0,\, \sigma^2\right).$$

As y depends on ϵ, we can also show that

$$y \sim \mathrm{N}\left(\beta_0 + \beta_1 x,\, \sigma^2\right)$$

where y is normally distributed for fixed values of x. The values of $\hat{\beta}_0$ and $\hat{\beta}_1$ arise when we consider the **loss function**:

$$L\left(\boldsymbol{\beta}\right) = \frac{1}{2}\epsilon^2 = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2}(y - (\beta_0 + \beta_1 x))^2.$$

which we wish to minimise. As these estimators apply to the entire training set, we will consider the **cost function**:

$$J\left(\boldsymbol{\beta}\right) = \frac{1}{n}\sum_{i=1}^{n} L\left(\beta\right).$$

which averages the loss across all observations. The goal of this cost function is to find the best parameters $\hat{\boldsymbol{\beta}}$ that minimise the loss across all observations:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} J\left(\boldsymbol{\beta}\right).$$

In this model, we must ensure that there are at least as many observations as there are parameters to estimate.

2.1 Correlation

Correlation is a statistical relationship between two variables. This measure allows us to determine the strength of a linear relationship between two variables. One common measure of correlation is the **Pearson correlation coefficient**, which is given by:

$$\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

where σ_{xy} is the **covariance** between x and y, and σ_x and σ_y are the variances of x and y, respectively. Using the sample covariance[1]:

$$r_{xy} = \frac{s_{xy}}{s_x s_y}.$$

The sample covariance s_{xy} has the following characteristics:

- $s_{xy} > 0$: y increases as x increases
- $s_{xy} < 0$: y decreases as x increases
- $s_{xy} \approx 0$: no linear relationship between x and y

To generalise this measure across datasets, we often consider the *normalised* correlation coefficient r_{xy}, which tells use the strength of a linear relationship between two variables. The following diagram shows various datasets and their correlation coefficients:

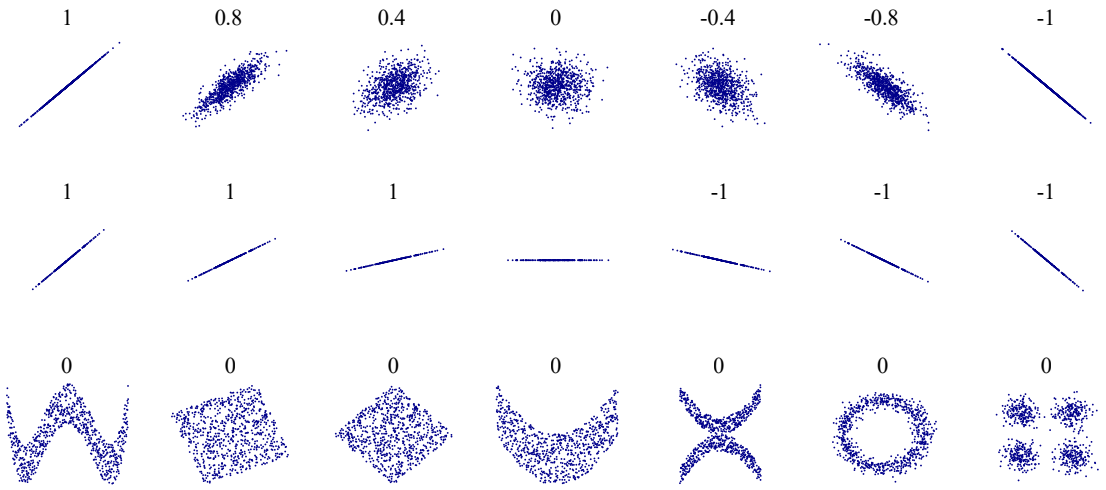


Figure 1: Pearson correlation coefficient of x and y for several sets of points.

There are a number of important points to note about the correlation coefficient:

[1] See Appendix A for more details.

- The correlation coefficient does not contain any information about the gradient, as is seen in the second row.
- A correlation coefficient of 0 indicates *no linear relationship*, it does not imply that there is *no relationship*, as is seen in the third row.
- The correlation coefficient does not imply *causation*. Care must be taken when assuming that a relationship between two variables is causal.

To minimise redundancy in a model, it is crucial for predictors to be correlated with response variables, and uncorrelated to other predictors.

2.2 Simple Linear Regression

Simple linear regression is a special case of linear regression, where we have only one input variable x_1, and we can use the method of maximum likelihood estimation to estimate the parameters β_0 and β_1:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} \left(x^{(i)} - \bar{x}\right)\left(y^{(i)} - \bar{y}\right)}{\sum_{i=1}^{n} \left(x^{(i)} - \bar{x}\right)^2}$$
$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

where $\bar{x}$ and $\bar{y}$ are the sample means of x and y, respectively.

2.3 Multiple Linear Regression

In multiple linear regression, we consider a p-dimensional feature space, or p input variables, with $p+1$ parameters to estimate. Here hypothesis function generalises to:

$$\hat{y}\left(x\right) = \hat{\beta}_0 x_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p = \sum_{i=0}^{p} \hat{\beta}_i x_i = \boldsymbol{\beta}^\top \mathbf{x}.$$

where we have introduced the feature $x_0 \coloneqq 1$ to simplify our notation. There are several ways to estimate $\hat{\boldsymbol{\beta}}$, and here we will use matrix calculus. The cost function $J\left(\boldsymbol{\beta}\right)$ can be written as:

$$J\left(\boldsymbol{\beta}\right) = \frac{1}{2n} \sum_{i=1}^{n} \left(y^{(i)} - \boldsymbol{\beta}^\top \mathbf{x}^{(i)}\right)^2 = \frac{1}{2n} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 = \frac{1}{2n} \boldsymbol{\epsilon}^\top \boldsymbol{\epsilon}.$$

where $\mathbf{X}$ is an $n \times (p+1)$ matrix of input variables:

$$\mathbf{x} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_p^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_p^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \cdots & x_p^{(n)} \end{bmatrix}$$

Consider the vector derivative of the cost function with respect to $\boldsymbol{\beta}$:

$$\begin{aligned}
\frac{\partial J(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \frac{1}{2n}\frac{\partial}{\partial \boldsymbol{\beta}}\left[\boldsymbol{\epsilon}^\top \boldsymbol{\epsilon}\right] \\
&= \frac{1}{2n}\frac{\partial}{\partial \boldsymbol{\beta}}\left[(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^\top (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\right] \\
&= \frac{1}{2n}\frac{\partial}{\partial \boldsymbol{\beta}}\left[(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^\top \mathbf{y} - (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^\top \mathbf{X}\boldsymbol{\beta}\right] \\
&= \frac{1}{2n}\frac{\partial}{\partial \boldsymbol{\beta}}\left[\mathbf{y}^\top \mathbf{y} - (\mathbf{X}\boldsymbol{\beta})^\top \mathbf{y} - \mathbf{y}^\top \mathbf{X}\boldsymbol{\beta} + (\mathbf{X}\boldsymbol{\beta})^\top \mathbf{X}\boldsymbol{\beta}\right] \\
&= \frac{1}{2n}\frac{\partial}{\partial \boldsymbol{\beta}}\left[\mathbf{y}^\top \mathbf{y} - \boldsymbol{\beta}^\top (\mathbf{X}^\top \mathbf{y}) - (\mathbf{y}^\top \mathbf{X})\boldsymbol{\beta} + \boldsymbol{\beta}^\top (\mathbf{X}^\top \mathbf{X})\boldsymbol{\beta}\right] \\
&= \frac{1}{2n}\left(-\mathbf{X}^\top \mathbf{y} - (\mathbf{y}^\top \mathbf{X})^\top + 2\mathbf{X}^\top \mathbf{X}\boldsymbol{\beta}\right) \\
&= \frac{1}{n}\left(\mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} - \mathbf{X}^\top \mathbf{y}\right).
\end{aligned}$$

This implies that the optimal parameters $\boldsymbol{\beta}$ are given by:

$$\begin{aligned}
\frac{\partial J(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \mathbf{0} \\
\mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} - \mathbf{X}^\top \mathbf{y} &= \mathbf{0} \\
\mathbf{X}^\top \mathbf{X}\boldsymbol{\beta} &= \mathbf{X}^\top \mathbf{y} \\
\boldsymbol{\beta} &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.
\end{aligned}$$

2.4 Assumptions of Linear Regression

There are several assumptions that must be satisfied for linear regression to be valid:

1. **Linearity**: The relationship between the response and predictors is linear.
2. **No Multicollinearity**: Predictors are not linearly dependent.
3. **Independence**: Responses are linearly independent.
4. **Normality**: Residuals are normally distributed.
5. **Homoscedasticity**: Residuals have constant variance.
6. **Exogeneity**: There is no correlation between predictors and residuals.

2.5 Testing the Assumptions of Linear Regression

To test the above assumptions, we can use the following techniques:

- Plots of predictions and actual values: Compare the regression with actual values
- Regression outputs: R^2, adjusted R^2, F-statistic, RMSE (root mean squared error)
- Correlation: Correlation between pairs of predictors, and between the response and predictors
- Analysis of residuals: Testing the normality and variance of residuals

2.5.1 Linearity

- **Scatter plot**: Plot observations with the predicted model to visually check for linearity
- **Residual plot**: Plot residuals against predictors. If the residuals are randomly distributed about 0, then the model suggests linearity

2.5.2 No Multicollinearity

- **Correlation matrix**: Check for high correlation between predictors, i.e., $\left|r_{x_1,x_2}\right| > 0.7$

High correlation between predictors may result in unreliable p-values. In this case, it might be necessary to remove one of the predictors from the model.

2.5.3 Independence

It is important to verify that responses are independent of each other. In time series data, this assumption can be tested using the **Durbin-Watson** test, which tests for the presence of **autocorrelation** in the residuals. A value close to 2 suggests no autocorrelation, while a value close to 0 or 4 suggests positive or negative autocorrelation, respectively.

2.5.4 Normality

- **Q-Q plot**: A quantile-quantile plot can be used to compare the distribution of residuals to a normal distribution. In this plot, we should expect the residuals to lie on a straight line.
- **Histogram**: A histogram of residuals can also be used to check for normality.

2.5.5 Homoscedasticity

- **Residual plot**: Plot residuals against predicted values. A constant spread of residuals about 0 suggests homoscedasticity.

2.5.6 Exogeneity

- **Residual plot**: Plot residuals against predictors. If the residuals are randomly distributed about 0, then the model suggests exogeneity.

2.6 Testing Overal Model Fit

The OLS (Ordinary Least Squares) model in the `statsmodels` package can be used to produce regression outputs from a linear regression model. The following table is an example of the output from a linear regression model:

Dep. Variable:	y	**R-squared:**	R^2
Model:	OLS	**Adj. R-squared:**	$\bar{R}^2$
Method:	Least Squares	**F-statistic:**	F_{test}
Date:	Date	**Prob (F-statistic):**	$\Pr\left(F_{p,n-p} > F_{\text{test}}\right)$
Time:	Time	**Log-Likelihood:**	$\ln\left(\hat{L}\right)$
No. Observations:	n	**AIC:**	AIC
Df Residuals:	$n-p-1$	**BIC:**	BIC
Df Model:	p		
Covariance Type:	nonrobust		

Table 1: OLS Regression Results

	coef	**std err**	t	$p > \|t\|$	[0.025, 0.975]
x_0	$\hat{\beta}_0$	$\text{SE}\left(\hat{\beta}_0\right)$	$t_{\text{test}}\left(\hat{\beta}_0\right)$	$\Pr\left(t_{\text{df}_{\text{error}}} > \left\|t_{\text{test}}\left(\hat{\beta}_0\right)\right\|\right)$	$\hat{\beta}_0 \pm t_{\frac{\alpha}{2},\text{test}}\left(\hat{\beta}_0\right)\text{SE}\left(\hat{\beta}_0\right)$
x_1	$\hat{\beta}_1$	$\text{SE}\left(\hat{\beta}_1\right)$	$t_{\text{test}}\left(\hat{\beta}_1\right)$	$\Pr\left(t_{\text{df}_{\text{error}}} > \left\|t_{\text{test}}\left(\hat{\beta}_1\right)\right\|\right)$	$\hat{\beta}_1 \pm t_{\frac{\alpha}{2},\text{test}}\left(\hat{\beta}_1\right)\text{SE}\left(\hat{\beta}_1\right)$
$\vdots$					
x_p	$\hat{\beta}_p$	$\text{SE}\left(\hat{\beta}_p\right)$	$t_{\text{test}}\left(\hat{\beta}_p\right)$	$\Pr\left(t_{\text{df}_{\text{error}}} > \left\|t_{\text{test}}\left(\hat{\beta}_p\right)\right\|\right)$	$\hat{\beta}_p \pm t_{\frac{\alpha}{2},\text{test}}\left(\hat{\beta}_p\right)\text{SE}\left(\hat{\beta}_p\right)$

Table 2: OLS Regression Coefficients

The following quantities will be used in the following discussion:

- Total sum of squares (SST): Total variation in the response variable

$$\text{SST} = \sum_{i=1}^{n}\left(y^{(i)} - \bar{y}\right)^2$$

- Error sum of squares (SSE): Total variation in the response variable not explained by the model

$$\text{SSE} = \sum_{i=1}^{n}\left(y^{(i)} - \hat{y}\left(x^{(i)}\right)\right)^2$$

- Regression sum of squares (SSR): Variation in the response variable explained by the model

$$\text{SSR} = \sum_{i=1}^{n}\left(\hat{y}\left(x^{(i)}\right) - \bar{y}\right)^2$$

The following relationship holds between the sum of squares:

$$\text{SST} = \text{SSE} + \text{SSR}.$$

The quantities in the regression results are described below:

- The **coefficient of determination** R^2 measures the proportion of the variance in the dependent variable explained by the predictors.

$$R^2 = 1 - \frac{\text{SSE}}{\text{SST}} = \frac{\text{SSR}}{\text{SST}}$$

- The **adjusted coefficient of determination** $\bar{R}^2$ takes into account the number of predictors in the model.

$$\bar{R}^2 = 1 - \frac{\text{SSE}/\text{df}_{\text{error}}}{\text{SST}/\text{df}_{\text{total}}}$$

  where $\text{df}_{\text{error}} = n - p - 1$ and $\text{df}_{\text{total}} = n - 1$ are the degrees of freedom for the error and total sum of squares, respectively.

- The **F-statistic** is used to test the overall significance of the model. It is given by:

$$F_{\text{test}} = \frac{(\text{SSR}/p)}{(\text{SSE}/\text{df}_{\text{error}})}$$

- The probability $\Pr\left(F_{p,n-p} > F_{\text{test}}\right)$ is the probability of observing an F-statistic greater than F_{test} under the null hypothesis that the model is statistically insignificant. A value less than 0.05 suggests that the model is statistically significant.

- The **Log Likelihood** $\ln\left(\hat{L}\right)$ is a measure of how well the model fits the data.

$$\ln\left(\hat{L}\right) = -\frac{n}{2}\log\left(2\pi\right) - \frac{n}{2}\log\left(\frac{\text{SSE}}{n}\right) - \frac{n}{2}$$

  where $\hat{L}$ is the maximum value of the likelihood function for the model.

- The **Akaike Information Criterion** (AIC) and **Bayesian Information Criterion** (BIC) are used to compare models. They are given by:

$$\text{AIC} = 2\left(p+1\right) - 2\ln\left(\hat{L}\right)$$
$$\text{BIC} = \left(p+1\right)\ln\left(n\right) - 2\ln\left(\hat{L}\right)$$

  Smaller values of AIC and BIC suggest a better model fit.

The quantities in the coefficient table are described below:

- The **coefficient** β_i is the estimated value of the parameter β_i.

- The **standard error** $\text{SE}\left(\hat{\boldsymbol{\beta}}\right)$ is the standard deviation of the sampling distribution of the coefficient. It can be computed as:

$$\text{SE}\left(\hat{\boldsymbol{\beta}}\right) = \sqrt{\frac{\text{SSE}}{\text{df}_{\text{error}}}\,\text{diag}\left(\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\right)}$$

  The results is a $p+1$ column vector of standard errors for each coefficient.

- The **t-statistic** $t_{\text{test}}\left(\hat{\boldsymbol{\beta}}\right)$ is used to test the null hypothesis that the coefficient is equal to 0. It is given by:

$$t_{\text{test}}\left(\hat{\boldsymbol{\beta}}\right) = \frac{\hat{\boldsymbol{\beta}}}{\text{SE}\left(\hat{\boldsymbol{\beta}}\right)}$$

- The **p-value** $p > |t|$ is the probability of observing a t-statistic greater than $|t|$ under the null hypothesis that the coefficient is equal to 0. A value less than 0.05 suggests that the coefficient is statistically significant.

$$\Pr\left(t_{\text{df}_{\text{error}}} > \left|t_{\text{test}}\left(\hat{\boldsymbol{\beta}}\right)\right|\right) = 2\Pr\left(t_{\text{df}_{\text{error}}} > t_{\text{test}}\left(\hat{\boldsymbol{\beta}}\right)\right)$$

- The **confidence interval** $[0.025, 0.975]$ is the range of values that the coefficient is likely to lie in. It is given by:

$$[0.025, 0.975] = \hat{\boldsymbol{\beta}} \pm t_{\frac{\alpha}{2},\text{df}_{\text{error}}} \text{SE}\left(\hat{\boldsymbol{\beta}}\right)$$

  where $t_{\frac{\alpha}{2},\text{df}_{\text{error}}}$ is the critical value of the t-distribution for a given significance level α. In this case, $\alpha = 0.05$.

Another measure of error of a model is the **root mean squared error** (RMSE), which is given by:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}\left(x^{(i)}\right)\right)^2} = \sqrt{\frac{\text{SSE}}{n}}$$

2.6.1 p-Values

The p-values obtained from the F-statistic and t-statistic are used to test the significance of the model and the coefficients, respectively. The p-value is the probability of observing a statistic greater than the observed value under the null hypothesis. A small p-value suggests that the null hypothesis is unlikely to be true, and therefore, the coefficient is statistically significant.

It is important to note that p-values can be misleading when model assumptions are violated.

3 Regularisation

Regularisation is a technique used to prevent overfitting in a model. Including additional predictors or higher order terms in a model may increase the accuracy of the model on the training set, but will often result in poor performance on the test set. In such cases, the model will tend to fit noise in the training set, rather than the underlying relationship between the predictors and the response.

Very complex models are also difficult to inspect or tune due to their size, and often contain many redundant predictors. To improve this model, we must use a validation set to tune the model such that it generalises well to new unseen data.

3.1 Bias and Variance

Bias and variance are two factors that contribute to the total error of a model.

- The **bias** of a model is the error introduced by erroneous assumptions in the model. A model with **high bias** will *underfit* the training data. Increasing the complexity of a model will reduce bias.

- The **variance** of a model is the error introduced by the model's sensitivity to fluctuations in the training set. A model with **high variance** will *overfit* the training data. Reducing the complexity of a model will reduce variance.

The goal of regularisation is to find a model that minimises both bias and variance. This is known as the **bias-variance trade-off**. To formally state the bias-variance trade-off, consider a dataset which was generated by

$$y^{(i)} = y^{\star}\left(x^{(i)}\right) + \xi^{(i)}$$

where $y^{\star}\left(x^{(i)}\right)$ is some underlying function of the predictors, and $\xi^{(i)}$ is observation noise with distribution $\xi \sim \mathrm{N}\left(0, \sigma^2\right)$. Our goal is to predict the underlying function $y^{\star}(x)$ from the observations $\left(x^{(i)}, y^{(i)}\right)$.

If we take a test example $(x,\ y)$ such that $y = y^{\star}(x) + \xi$, and measure the expected test error (averaged over the training set and randomness of ξ), we find that:

$$\mathrm{MSE}(x) = \mathrm{E}\left[\left(y - \hat{y}(x)\right)^2\right].$$

Assuming independence between $y^{\star}$ and the noise ξ, we can decompose the expected test error into three components:

$$\begin{aligned}
\mathrm{MSE}(x) &= \mathrm{E}\left[\left(y - \hat{y}(x)\right)^2\right] \\
&= \mathrm{E}\left[\left(\xi + \left(y^{\star}(x) - \hat{y}(x)\right)\right)^2\right] \\
&= \mathrm{E}\left[\xi^2\right] + \mathrm{E}\left[\left(y^{\star}(x) - \hat{y}(x)\right)^2\right] \\
&= \sigma^2 + \mathrm{E}\left[\left(y^{\star}(x) - \hat{y}(x)\right)^2\right]
\end{aligned}$$

Let us define $\hat{y}_{\mathrm{avg}}(x) = \mathrm{E}\left[\hat{y}(x)\right]$ as the average model, obtained by drawing an infinite number of datasets, training on them, and averaging their predictions on x. It can be shown that $\hat{y}_{\mathrm{avg}}(x)$ is approximately equal the model obtained by training on a *single* dataset with an infinite number of samples. Let us introduce the constant $c = \hat{y}_{\mathrm{avg}}(x) - y^{\star}(x)$, so that:

$$\begin{aligned}
\mathrm{MSE}(x) &= \sigma^2 + \mathrm{E}\left[\left(y^{\star}(x) - \hat{y}(x)\right)^2\right] \\
&= \sigma^2 + \mathrm{E}\left[\left(\left(y^{\star}(x) - \hat{y}_{\mathrm{avg}}(x)\right) + \left(\hat{y}_{\mathrm{avg}}(x) - \hat{y}(x)\right)\right)^2\right] \\
&= \sigma^2 + \mathrm{E}\left[\left(\hat{y}_{\mathrm{avg}}(x) - y^{\star}(x)\right)^2\right] + \mathrm{E}\left[\left(\hat{y}(x) - \hat{y}_{\mathrm{avg}}(x)\right)^2\right] \\
&= \sigma^2 + \left(\hat{y}_{\mathrm{avg}}(x) - y^{\star}(x)\right)^2 + \mathrm{E}\left[\left(\hat{y}(x) - \hat{y}_{\mathrm{avg}}(x)\right)^2\right] \\
&= \underbrace{\sigma^2}_{\text{Irreducible Error}} + \underbrace{\left(\hat{y}_{\mathrm{avg}}(x) - y^{\star}(x)\right)^2}_{\text{Bias}^2} + \underbrace{\mathrm{Var}\left[\hat{y}(x)\right]}_{\text{Variance}}.
\end{aligned}$$

The constant c is the **bias** of $\hat{y}(x)$, relative to the average model $y^{\star}(x)$. Regularisation therefore aims to minimise the total error of a model.

3.2 Regularisation Techniques

Regularisation adds a penalty term to the cost function of a model to penalise large coefficients, reducing the complexity and variance of a model. Two common regularisation techniques are:

- L1 LASSO Regularisation
- L2 Ridge Regularisation

The new regularised cost function is given by:

$$J_\lambda(\boldsymbol{\beta}) = J(\boldsymbol{\beta}) + \lambda R(\boldsymbol{\beta})$$

where $J(\boldsymbol{\beta})$ is the original cost function, $\lambda \leqslant 0$ is the regularisation parameter that dictates the strength of the regularisation, and $R(\boldsymbol{\beta})$ is the regulariser. The regulariser has the form

$$R(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_l^l = \frac{1}{l}\sum_{i=1}^{p}|\beta_i|^l$$

where $\|\cdot\|_l$ represents a L^p-norm, denoted l here to avoid confusion with the number of predictors p. Note that the regulariser is applied to all coefficients except the intercept term β_0.

3.3 LASSO Regularisation

LASSO (Least Absolute Shrinkage and Selection Operator) regularisation is a regularisation technique used for feature selection as it can shrink coefficients to 0. The regulariser penalises the absolute size of each coefficient through the L^1-norm:

$$R_{\text{LASSO}}(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_1 = \sum_{i=1}^{p}|\beta_i| = \mathbf{1}^\top|\boldsymbol{\beta}|$$

where $\mathbf{1}$ is a p-dimensional vector of ones and $|\boldsymbol{\beta}|$ is the element-wise absolute value of $\boldsymbol{\beta}$. The regularised cost function is therefore given by:

$$\begin{aligned} J_\lambda(\boldsymbol{\beta}) &= J(\boldsymbol{\beta}) + \lambda R_{\text{LASSO}}(\boldsymbol{\beta}) \\ &= \frac{1}{2n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 + \lambda\|\boldsymbol{\beta}\|_1. \end{aligned}$$

Attempting to use the same derivation as in the previous section reveals that it is no longer possible to analytically solve for the optimal parameters $\boldsymbol{\beta}$. This is shown below:

$$\begin{aligned} \frac{\partial J_\lambda(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \frac{\partial J(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} + \lambda\frac{\partial R_{\text{LASSO}}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} \\ &= \frac{1}{n}\left(\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} - \mathbf{X}^\top\mathbf{y}\right) + \lambda\,\text{sgn}(\boldsymbol{\beta}) \end{aligned}$$

If we set this equal to 0, we find an implicit relationship between $\boldsymbol{\beta}$ and λ:

$$\begin{aligned} \frac{\partial J_\lambda(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \mathbf{0} \\ \mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} - \mathbf{X}^\top\mathbf{y} + n\lambda\,\text{sgn}(\boldsymbol{\beta}) &= \mathbf{0} \\ \mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} &= \mathbf{X}^\top\mathbf{y} - n\lambda\,\text{sgn}(\boldsymbol{\beta}) \\ \boldsymbol{\beta} &= \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\left(\mathbf{X}^\top\mathbf{y} - n\lambda\,\text{sgn}(\boldsymbol{\beta})\right). \end{aligned}$$

This implicit relationship must be solved using an iterative method.

3.4 Ridge Regularisation

Ridge regularisation is a regularisation technique that is used to reduce large coefficients. This is done using the L^2-norm, which penalises the squared size of each coefficient:

$$R_{\text{Ridge}}(\boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 = \frac{1}{2}\sum_{i=1}^{p}\beta_i^2 = \frac{1}{2}\boldsymbol{\beta}^\top\boldsymbol{\beta}$$

The regularised cost function for this technique becomes:

$$\begin{aligned} J_\lambda(\boldsymbol{\beta}) &= J(\boldsymbol{\beta}) + \lambda R_{\text{Ridge}}(\boldsymbol{\beta}) \\ &= \frac{1}{2n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2 + \lambda\frac{1}{2}\|\boldsymbol{\beta}\|_2^2. \end{aligned}$$

Unlike LASSO regularisation, it is possible to analytically solve for the optimal parameters $\boldsymbol{\beta}$ using the following derivation:

$$\begin{aligned} \frac{\partial J_\lambda(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} &= \frac{\partial J(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} + \lambda\frac{\partial R_{\text{Ridge}}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} \\ &= \frac{1}{n}\left(\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} - \mathbf{X}^\top\mathbf{y}\right) + \lambda\boldsymbol{\beta} \end{aligned}$$

Setting this equal to 0 we find:

$$\begin{aligned} \frac{\partial J_\lambda(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}} &= \mathbf{0} \\ \mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} - \mathbf{X}^\top\mathbf{y} + n\lambda\boldsymbol{\beta} &= \mathbf{0} \\ \mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} + n\lambda\boldsymbol{\beta} &= \mathbf{X}^\top\mathbf{y} \\ \mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} + n\lambda\boldsymbol{\beta} &= \mathbf{X}^\top\mathbf{y} \\ \left(\mathbf{X}^\top\mathbf{X} + n\lambda\mathbf{I}\right)\boldsymbol{\beta} &= \mathbf{X}^\top\mathbf{y} \\ \boldsymbol{\beta} &= \left(\mathbf{X}^\top\mathbf{X} + n\lambda\mathbf{I}\right)^{-1}\mathbf{X}^\top\mathbf{y}. \end{aligned}$$

The term "ridge" refers to the ill-conditioned nature of the matrix inverse performed on the covariance matrix $\mathbf{X}^\top\mathbf{X}$ due to multicollinearity. Ridge regularisation aims to smooth the cost function near the ridge, by adding a penalty term to the diagonal of the covariance matrix. This reduces the likelihood of the matrix being singular when it's columns are linearly dependent.

3.5 Ridge vs LASSO Regularisation

When choosing between ridge and LASSO regularisation, it is important to understand their advantages and disadvantages.

- Ridge regression is typically faster to train than LASSO regression, as it has a closed-form solution.
- LASSO regression produces a simpler model than ridge regression, as it can shrink coefficients to 0, effectively performing feature selection.

- LASSO regression performs faster during the testing phase, as the model has fewer coefficients to compute.

To understand why LASSO regression can eliminate coefficients, consider the optimisation problem where the objective function is $J(\boldsymbol{\beta})$, with the constraint $R(\boldsymbol{\beta}) \leqslant t$:

$$\min_{\boldsymbol{\beta}} J(\boldsymbol{\beta}) \quad \text{subject to} \quad R(\boldsymbol{\beta}) \leqslant t$$

where t is a constant that is inversely proportional to λ. If we imagine a problem where $p = 2$, and plot the level curves of J on the feature plane, we can see that the constraints for LASSO and ridge regularisation have different characteristics.

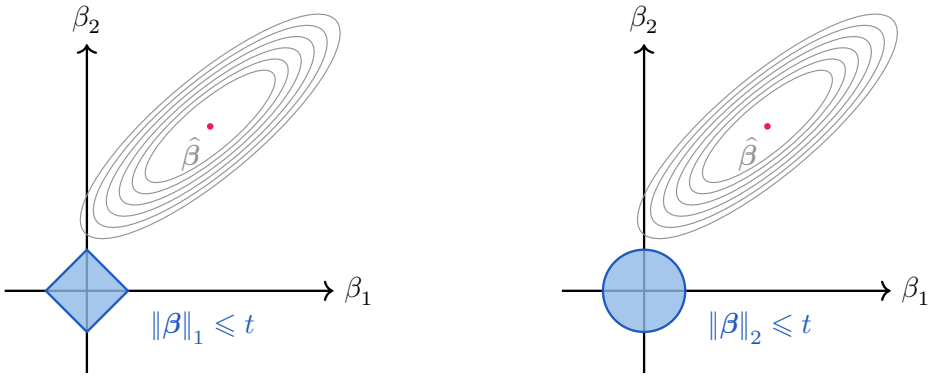


Figure 2: Comparison of constrain region $R \leqslant t$ for LASSO (L1) and ridge (L2) regression.

The sharp corners of the LASSO constraint region cause the level curves of J to intersect the constraint region at the axes, leading to a sparse solution. Because of it's constant radius, ridge regression tends to produce a solution with small coefficients that are not necessarily zero.

4 Classification

Classification is a supervised learning task that predicts classes of categorical data. There are two main types of classification:

- **Binary classification:** The response variable has two classes.
- **Multiclass classification:** The response variable has more than N classes.

4.1 Support Vector Machines

Support Vector Machines (SVM) are a supervised learning model used for binary classification. The model works by finding a hyperplane that separates classes with the maximum margin. Consider a dataset of n points of the form:

$$\left\{\left(\mathbf{x}^{(i)}, y^{(i)}\right)\right\}$$

where $\mathbf{x}^{(i)} \in \mathbb{R}^p$ is a p-dimensional and $y^{(i)} \in \{-1, +1\}$ indicates the class of the ith point. We can model a p-dimensional hyperplane by the equation:

$$\mathbf{w}^\top \mathbf{x} - b = 0$$

where $\mathbf{w} \in \mathbb{R}^p$ is a vector normal to the hyperplane and $b \in \mathbb{R}$ is the bias term. In this model, the distance between the hyperplane and the origin is given by $|b|/\|\mathbf{w}\|$.

4.1.1 Hard Margin SVM

If a dataset is linearly separable, we can select two parallel hyperplanes that separate the classes, such that the distance between them is maximised.We start by defining the **functional margin** of a point $\mathbf{x}^{(i)}$ as

$$\hat{\gamma}^{(i)} = y^{(i)} \left(\mathbf{w}^\top \mathbf{x}^{(i)} - b\right)$$

and the **functional margin** of the optimal hyperplane as

$$\hat{\gamma} = \min_{i=1,\,\ldots,\,n} \hat{\gamma}^{(i)}$$

This value represents the **correctness** and **confidence** of the classification of a point for a given hyperplane:

- A hyperplane correctly classifies a point when $\hat{\gamma}^{(i)} > 0$.
- A hyperplane confidentally classifies a point when $\left|\hat{\gamma}^{(i)}\right| \gg 0$.

Using this definition, we also define the **geometric margin** of a point $\mathbf{x}^{(i)}$, to allow us to compare the confidence of points across different hyperplanes:

$$\gamma^{(i)} = \frac{\hat{\gamma}^{(i)}}{\|\mathbf{w}\|}.$$

In a similar manner, the **geometric margin** of the optimal hyperplane is defined:

$$\gamma = \frac{\hat{\gamma}}{\|\mathbf{w}\|}$$

To understand why we define the functional margin in this way, consider the points $\mathbf{x}_+$ that are classified as $+1$, and the points $\mathbf{x}_-$ that are classified as -1. We wish to constrain the two classes by the hyperplanes:

$$\begin{aligned} \mathbf{w}^\top \mathbf{x}_+ - b &\geqslant +\hat{\gamma} \\ \mathbf{w}^\top \mathbf{x}_- - b &\leqslant -\hat{\gamma} \end{aligned}$$

If we multiply both equations by y, which for $\mathbf{x}_+$ is $+1$, and for $\mathbf{x}_-$ is -1, we find:

$$\begin{cases} y\left(\mathbf{w}^\top \mathbf{x}_+ - b\right) \geqslant \hat{\gamma} \\ y\left(\mathbf{w}^\top \mathbf{x}_- - b\right) \geqslant \hat{\gamma} \end{cases}$$

$$\therefore\ y^{(i)} \left(\mathbf{w}^\top \mathbf{x}^{(i)} - b\right) \geqslant \hat{\gamma}.$$

We can then pose the following optimisation problem:

$$\begin{aligned} \max_{\hat{\gamma},\, \mathbf{w},\, b} \quad & \frac{\hat{\gamma}}{\|\mathbf{w}\|} \\ \text{subject to} \quad & y^{(i)}\left(\mathbf{w}^\top \mathbf{x}^{(i)} - b\right) \geqslant \hat{\gamma}, \quad i = 1, \ldots, n \end{aligned}$$

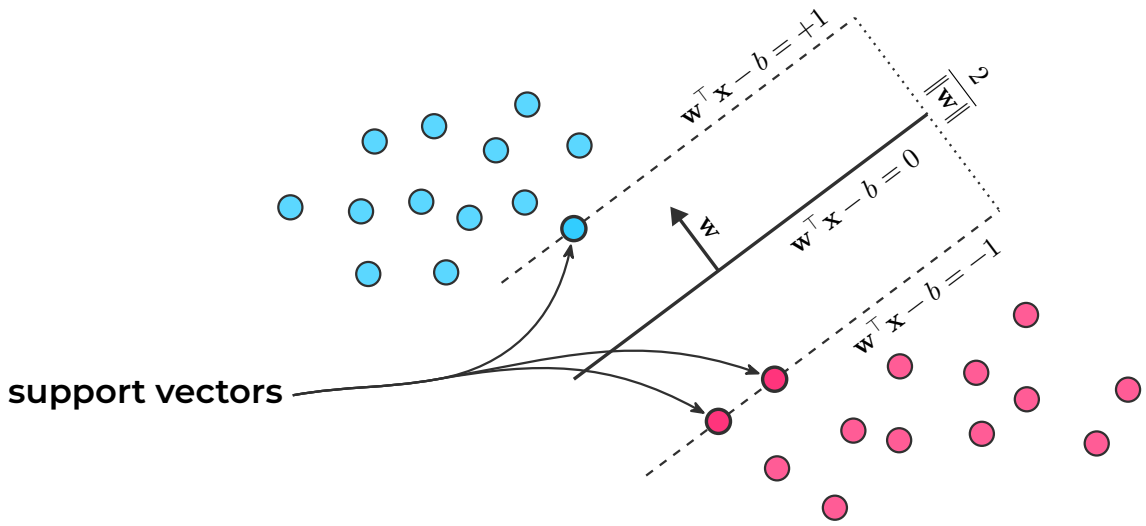
This is a non-convex problem, as the objective function contains the nonlinear term $\|\mathbf{w}\|$. If we impose the constraint

$$\hat{\gamma} = 1$$

and note that maximising $\hat{\gamma}/\|\mathbf{w}\| = 1/\|\mathbf{w}\|$ is the same as minimising $\|\mathbf{w}\|^2$, we can solve the equivalent problem:

$$\begin{aligned} \min_{\mathbf{w},\, b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y^{(i)}\left(\mathbf{w}^\top \mathbf{x}^{(i)} - b\right) \geqslant 1, \quad i = 1, \ldots, n. \end{aligned}$$

This is an optimisation problem with a convex quadratic objective function and linear constraints whose solution gives us the **optimal margin classifier**. The following figure shows a geometric interpretation of the optimal margin classifier:



The **decision boundary** is completely determined by the points that lie nearest to it. These points are called **support vectors**.

4.1.2 The Dual Problem

We can reformulate the optimisation problem for the optimal margin classifier using **Lagrange multipliers**:

$$\begin{aligned}\mathcal{L}\left(\mathbf{w},\, b,\, \boldsymbol{\alpha}\right) &= \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \alpha_i \left(1 - y^{(i)}\left(\mathbf{w}^\top \mathbf{x}^{(i)} - b\right)\right) \\ &= \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \alpha_i \left(1 - y^{(i)}\mathbf{w}^\top \mathbf{x}^{(i)} + y^{(i)} b\right)\end{aligned}$$

where $\boldsymbol{\alpha} = (\alpha_1,\, \ldots,\, \alpha_n)$ is a vector of Lagrange multipliers. The **dual problem** is then given by:

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad & \min_{\mathbf{w},\, b} \mathcal{L}\left(\mathbf{w},\, b,\, \boldsymbol{\alpha}\right) \\ \text{subject to} \quad & \alpha_i \geqslant 0, \quad i = 1,\, \ldots,\, n \\ & \frac{\partial \mathcal{L}\left(\mathbf{w},\, b,\, \boldsymbol{\alpha}\right)}{\partial \mathbf{w}} = 0 \\ & \frac{\partial \mathcal{L}\left(\mathbf{w},\, b,\, \boldsymbol{\alpha}\right)}{\partial b} = 0.\end{aligned}$$

The two partial derivatives are given by:

$$\begin{aligned}&\frac{\partial \mathcal{L}\left(\mathbf{w},\, b,\, \boldsymbol{\alpha}\right)}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y^{(i)} \mathbf{x}^{(i)} = \mathbf{0} \quad \implies \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y^{(i)} \mathbf{x}^{(i)} \\ &\frac{\partial \mathcal{L}\left(\mathbf{w},\, b,\, \boldsymbol{\alpha}\right)}{\partial b} = \sum_{i=1}^{n} \alpha_i y^{(i)} = 0.\end{aligned}$$

The Lagrangian then simplifies to:

$$\begin{aligned}\mathcal{L}\left(\mathbf{w},\, b,\, \boldsymbol{\alpha}\right) &= \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{n} \alpha_i \left(1 - y^{(i)}\mathbf{w}^\top \mathbf{x}^{(i)} + y^{(i)} b\right) \\ &= \frac{1}{2}\left(\sum_{i=1}^{n} \alpha_i y^{(i)} \mathbf{x}^{(i)}\right)^\top \left(\sum_{i=1}^{n} \alpha_i y^{(i)} \mathbf{x}^{(i)}\right) \\ &\quad + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha^{(j)} y^{(i)} y^{(j)} \left\langle \mathbf{x}^{(i)},\, \mathbf{x}^{(j)} \right\rangle + b \sum_{i=1}^{n} \alpha_i y^{(i)} \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha^{(j)} y^{(i)} y^{(j)} \left\langle \mathbf{x}^{(i)},\, \mathbf{x}^{(j)} \right\rangle.\end{aligned}$$

Therefore, in this form, we find that our problem only depends on the inner product of points in the dataset:

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha^{(j)} y^{(i)} y^{(j)} \left\langle \mathbf{x}^{(i)},\, \mathbf{x}^{(j)} \right\rangle \\ \text{subject to} \quad & \alpha_i \geqslant 0, \quad i = 1,\, \ldots,\, n \\ & \sum_{i=1}^{n} \alpha_i y^{(i)} = 0.\end{aligned}$$

By examining the dual form of the optimisation problem, we can see that our algorithm is in terms of inner products between input vectors. This will be important when we consider kernel methods in the next section.

4.1.3 Kernel Methods

Consider a dataset where a linear model is not sufficient. We can transform the input vectors $\mathbf{x}$ into a higher-dimensional space using the **feature map** $\phi : \mathbb{R}^p \to \mathbb{R}^d$ such that

$$\mathbf{x} \mapsto \phi(\mathbf{x}).$$

Let us define a **kernel function** corresponding to the feature map ϕ as a function $K : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ which satisfies:

$$K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) = \left\langle \phi\left(\mathbf{x}^{(i)}\right), \phi\left(\mathbf{x}^{(j)}\right)\right\rangle_V.$$

K is a symmetric positive semi-definite kernel because it satisfies Mercer's theorem, that is,

$$\sum_{i=1}^{p}\sum_{j=1}^{p} c_i c_j K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right) \geqslant 0.$$

Alternatively, we can construct the Gram matrix $\mathbf{K}$ where $\mathbf{K}_{ij} = K\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)$, demonstrating that $\mathbf{K}$ is positive semi-definite.

Our goal is then to find a method to compute the inner product $\left\langle \phi\left(\mathbf{x}^{(i)}\right), \phi\left(\mathbf{x}^{(j)}\right)\right\rangle_V$ without explicitly computing ϕ, so that we can replace all inner products involving the input vector with the kernel function K. This allows us to work with ϕ, which can be infinite in dimension, without knowing its explicit form.

This is the idea behind the **kernel trick**, which allows us to solve the dual form of the optimisation problem using a kernel function.

4.1.4 Common Kernels

In this section, we will use the symbols $\mathbf{x}$ and $\mathbf{z}$ to represent input vectors $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ for brevity.

Consider the function $K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}$. Through some algebraic manipulation, we can show that this is a valid kernel function:

$$\begin{aligned} K(\mathbf{x}, \mathbf{z}) &= \mathbf{x}^\top \mathbf{z} \\ &= \langle \mathbf{x}, \mathbf{z} \rangle_V. \end{aligned}$$

This forms our first trivial kernel function, the **linear kernel**, where $\phi(\mathbf{x}) = \mathbf{x}$. Now consider the kernel function $K(\mathbf{x}, \mathbf{z}) = f(\mathbf{x}) f(\mathbf{z})$, for some function $f : \mathbb{R}^p \to \mathbb{R}$. We can show that this is also a valid kernel function by letting $\phi(\mathbf{x}) = f(\mathbf{x})$:

$$\begin{aligned} K(\mathbf{x}, \mathbf{z}) &= f(\mathbf{x}) f(\mathbf{z}) \\ &= \langle f(\mathbf{x}), f(\mathbf{z}) \rangle_V. \end{aligned}$$

Let K_1 and K_2 be valid kernel functions, and let $\alpha_1, \alpha_2 \geqslant 0$, then, the following transformations also produce valid kernel functions:

- Scalar multiplication: $K(\mathbf{x}, \mathbf{z}) = \alpha K_1(\mathbf{x}, \mathbf{z})$.

- Adding a constant: $K(\mathbf{x},\, \mathbf{z}) = K_1(\mathbf{x},\, \mathbf{z}) + \alpha$.
- Linear combination: $K(\mathbf{x},\, \mathbf{z}) = \alpha_1 K_1(\mathbf{x},\, \mathbf{z}) + \alpha_2 K_2(\mathbf{x},\, \mathbf{z})$.
- Element-wise product: $K(\mathbf{x},\, \mathbf{z}) = K_1(\mathbf{x},\, \mathbf{z})\, K_2(\mathbf{x},\, \mathbf{z})$.
- Polynomial map: $K(\mathbf{x},\, \mathbf{z}) = P_d(K_1(\mathbf{x},\, \mathbf{z}))$ where P_d is a polynomial of degree d with positive coefficients.
- Exponential map: $K(\mathbf{x},\, \mathbf{z}) = \exp(K_1(\mathbf{x},\, \mathbf{z}))$.

These transformations allow us to construct a wide variety of kernel functions that satisfy the conditions defined above. Some common kernel functions are:

- Polynomial kernel: $K(\mathbf{x},\, \mathbf{z}) = (\gamma \mathbf{x}^\top \mathbf{z} + r)^d$ for $d \in \mathbb{N}$.
- Gaussian kernel: $K(\mathbf{x},\, \mathbf{z}) = \exp\left(-\gamma \|\mathbf{x} - \mathbf{z}\|^2\right)$.
- Sigmoid kernel: $K(\mathbf{x},\, \mathbf{z}) = \tanh(\gamma \mathbf{x}^\top \mathbf{z} + r)$.

where γ controls the influence of each training example on the decision boundary, and r is a bias term.

The choice of kernel is problem dependent, but generally, points that are similar should produce a large value of K, and points that are dissimilar should produce a small value of K.

A good starting point is the linear kernel, which allows us to visualise the decision boundary and understand the problem. If this kernel does not perform well, the Gaussian kernel should be used, as it only has one hyperparameter that needs to be tuned.

- A small value of γ extends influence of a point and leads to a smooth decision boundary.
- A large value of γ results in a more complex decision boundary. This can lead to overfitting.

The polynomial kernel is often more difficult to tune, as the degree of the polynomial can have a large effect on the decision boundary and the sigmoid kernel should only be used in specific cases.

4.1.5 Soft Margin SVM

Our discussion so far has only considered a linearly separable dataset. When a dataset is not linearly separable, the optimal margin classifier algorithm may fail to converge or return a suboptimal solution. This is due to the fact that the optimal margin classifier is sensitive to outliers, and may not generalise well to new data.

To prevent this, we will introduce the **slack variable** $\xi^{(i)}$ for each point $\mathbf{x}^{(i)}$, and reformulate the dual problem as follows:

$$\begin{aligned} \min_{\mathbf{w},\, b,\, \xi} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} \xi^{(i)} \\ \text{subject to} \quad & y^{(i)}\left(\mathbf{w}^\top \mathbf{x}^{(i)} - b\right) \geqslant 1 - \xi^{(i)}, \quad i = 1,\, \ldots,\, n \\ & \xi^{(i)} \geqslant 0, \quad i = 1,\, \ldots,\, n \end{aligned}$$

This modification now allows points to be misclassified with a penalty $C\xi^{(i)}$. C is a regularisation parameter (sometimes called a box constraint) that controls the trade-off between the margin and the slack variables.

- A small value of C will penalise misclassifications less and produce a wider margin.
- A large value of C will penalise misclassifications more and produce a smaller margin.

$C = \infty$ is equivalent to the hard margin SVM, as it forces $\xi^{(i)} = 0$ for all i.

4.2 K-Nearest Neighbours

The k-nearest neighbours (k-NN) algorithm is a simple classification algorithm that classifies a point based on the majority class of its k nearest neighbours. The algorithm is non-parametric, meaning that it does not make any assumptions about the underlying distribution of the data. In this algorithm:

- The training stage only consists of storing the feature vectors and their corresponding class labels.
- The testing stage then finds the k nearest neighbours of a given point and assigns it the majority class of those neighbours.

When classifying a point in the test set, we must choose an appropriate number of neighbours k.

- A small value of k may lead to poor performance on noisy data, as it will be sensitive to outliers.
- A large value of k may lead to misclassification when points are close to the decision boundary, or when classes are imbalanced.

The distance metric used to find the nearest neighbour is problem dependent, but the following are common choices:

- **Manhattan distance:** $\left\|\mathbf{x}^{(i)} - \mathbf{x}\right\|_1$.
- **Euclidean distance:** $\left\|\mathbf{x}^{(i)} - \mathbf{x}\right\|_2$.
- **Minkowski distance:** $\left\|\mathbf{x}^{(i)} - \mathbf{x}\right\|_p$ for $p \in \mathbb{N}$.
- **Cosine similarity distance:** $1 - \frac{\mathbf{x}^\top \mathbf{z}}{\|\mathbf{x}\|_2 \|\mathbf{z}\|_2}$ measures the angle between vectors.
- **Mahalanobis distance:** $\sqrt{\left(\mathbf{x}^{(i)} - \mathbf{x}\right)^\top \mathbf{S}^{-1} \left(\mathbf{x}^{(i)} - \mathbf{x}\right)}$ assumes $\mathbf{x}$ are distributed according to some distribution, with covariance matrix $\mathbf{S}$.
- **Hamming distance:** Used for binary data, it counts the number of positions at which corresponding symbols are different.

4.3 Random Forest

A random forest is an ensemble learning method that is used for both classification and regression that constructs multiple decision trees at training time to predict an examples class. In classification tasks, the output of the random tree is the class selected by the most trees. It's aim is to produce a diverse set of trees that are uncorrelated, and therefore, when combined, produce a more accurate prediction than any individual tree.

4.3.1 Decision Trees

Decision trees are a classical learning method that iteratively split data based on some criterion. Decision trees consists of two types of nodes:

- **Decision nodes:** Represent a decision that splits data entering that node.
- **End nodes:** Represent a class label with some probability.

A tree is constructed using the entire training set, until one of the following conditions are met:

- The tree reaches a maximum depth.
- The previous decision splits the data into classes that are pure, or nearly pure (i.e., consisting of a single class). This is determined by the **Gini impurity**.
- The information gain from a split is below a certain threshold.

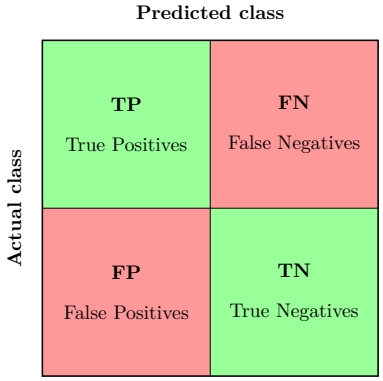
4.3.2 Random Forests

In the random forest algorithm, a number of decision trees are constructed using a random subset of the training data. This can be done in one of two ways:

- **Bootstrap aggregation (bagging)**: Sampling with replacement to create a new training set of the same size. This is repeated B times to create B new training sets, for B *different* decision trees.
- **Feature bagging**: Randomly select a subset of

4.4 Classification Metrics

In binary classification, the confusion matrix is a table that is used to describe the performance of a classification model. It has the following metrics:



The following metrics are used to assess the performance of binary classification models.

Metric	**Formula**	**Interpretation**
Accuracy	$\frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$	Overall performance of model
Precision	$\frac{\text{TP}}{\text{TP} + \text{FP}}$	Accuracy of positive predictions
Recall Sensitivity	$\frac{\text{TP}}{\text{TP} + \text{FN}}$	Coverage of actual positive samples
Sensitivity	$\frac{\text{TN}}{\text{TN} + \text{FP}}$	Coverage of actual negative samples
F1 Score	$\frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$	Harmonic mean of precision and recall

Compared to accuracy, precision and recall provide a more detailed analysis of how a model performs. Choosing the right metric depends on what is important for a problem:

- If we wish to minimise false negatives: Use recall.
- If we wish to minimise false positives: Use precision.
- For overall performance: Use accuracy.

These metrics can also be generalised to multiclass classification by computing them for each class. Here we must consider datasets with class imbalance, as the metrics may be biased towards the majority class.

A Numerical Summaries of Data

A.1 Mean

The mean of a set of n observations $\left(x^{(i)}\right)$ is given by:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)}$$

A.2 Variance

The variance of a set of n observations $\left(x^{(i)}\right)$ is given by:

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} \left(x^{(i)} - \bar{x}\right)^2$$

For a sample, the variance is given by:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} \left(x^{(i)} - \bar{x}\right)^2$$

A.3 Covariance

The covariance between two sets of n observations $\left(x^{(i)}\right)$ and $\left(y^{(i)}\right)$ is given by:

$$\sigma_{xy} = \frac{1}{n} \sum_{i=1}^{n} \left(x^{(i)} - \bar{x}\right) \left(y^{(i)} - \bar{y}\right)$$

For a sample, the covariance is given by:

$$s_{xy} = \frac{1}{n-1} \sum_{i=1}^{n} \left(x^{(i)} - \bar{x}\right) \left(y^{(i)} - \bar{y}\right)$$