Predictive Analytics QBUS2820 Exam notes.

Charles Christopher Hyland

Semester 2 2017

Abstract

Welcome! Hopefully these notes help you to ace QBUS2820!

Contents

1 Statistical Decision Theory

1.1 Introduction and notation

We focus on 2 different topics in this unit, **cross sectional prediction**, whereby we work with subjects and wish to predict values for new subjects, and **forecasting** whereby we want to predict the value of a response variable in the future based on past and current information.

Supervised Learning is when we attempt to learn a function to predict Y based on input $x_1, ..., x_p$. We develop methods to help us learn this function based on labelled data $\{(y_i, \boldsymbol{x_i})\}_{i=1}^{N}$ aka *training data.*

In this unit, we have 6 stages:

1) Problem formulation

2) Data collection and preparation

3) EDA

4) Model building, estimation, and selection

5) Model evaluation

6) Communication of results

Recall that Y is a random variable whilst y are realised observed values of a random variable. We use i for observations and j for predictors. Therefore, x_{ij} refers to the *value* of the particular predictor j for observation i whilst x_i just refers to observation i with all its predictors and finally, x_j refers to the values of predictor j for each observation. We use hat notation $\hat{\beta}$ to represent estimators or estimates. Recall that estimators are attempting to estimate random variables whilst the estimate is an actual value computed. You can think of estimators(method to compute mean) being *rules* to estimate estimates(sample mean) in order to figure out what the population average is. Finally, we put vectors in lower case bold $\mathbf{x}$ and matrices in uppercase bold $\mathbf{X}$. Recall that a design matrix $\mathbf{X}$ is a NxP matrix (where N is number of observations and is on the row axis, whilst P is the number of predictors which is represented by the columns).

1.2 Statistical Decision Theory

We have 2 steps to prediction, first we need to train a predictive function $f(\hat{\mathbf{x}})$ (which is an estimate) using the training data set D = $\{(y_i, \boldsymbol{x_i})\}_{i=1}^{N}$. Currently, the data has a model of Y = f(x) + ϵ which we are trying to estimate. However, we don't know what f(x) is. After learning this predictive function, we want to take in a new observation point $\boldsymbol{x_0}$, and from that, make the prediction $\hat{f}(\boldsymbol{x_0})$. This is simlpy the predictive function evaluated at the observed new point $\boldsymbol{x_0}$. To go about doing this, we need to use **decision theory!**.

1.2.1 Loss Functions

Loss/cost functions $L(y, f(\boldsymbol{x}))$ simply measures how bad/cost of our prediction for f(**x**) is, when in actual fact it is y. A loss function people tend to use is the **squared loss** which is defined as:

$$L(y, f(\boldsymbol{x})) = (y - f(\boldsymbol{x}))^2$$

A concrete example would be lets say we predicted the price of a house to be \$1 million but it turns out to be \$1.2 million. We have a prediction error of \$200,000 but this is not necessarily the cost/loss since we need to factor in and penalise errors in prediction. The loss function will take this number and scale it in order to tell us how bad is it to make errors in prediction. Keep in mind for later sections that the loss function can be thought of as the loss for observed values.

So a framework we can think of, is in 4 steps: 1) We have a **state** s whereby which is unknown. This could be whether is an email spam or not.

2) We are given observations in which we do know.

3) From this, we take an action a based on the observation. This can be in the form of how we classify the observation.

4) Resultantly, we get a loss L(s,a) from this.

From this, given x, we wish to minimise the loss L(y,$\hat{y}$) except that we don't know what y is. Furthermore, we also wanted to pick a f(x) in order to minimise L(y,f(x)), but we don't know what x we will be given, alongside the y in wish we get (if we knew y, no point in prediction!). Therefore, we need to use probability to help us. Instead we now focus on the random variables Y and X instead. An idea is that we want to have a small loss **on average**. To further extend things, we can call this the concept of **expected loss**. What we can then do is that for any given x in which we wish to predict, we have that:

$$E(L(Y, \hat{y})|X = x) = \sum_{y \in Y} L(y, \hat{y}) P(y|x)$$

and we now have a better problem in which we can minimise over.

We also need to brush up on some probability theory. Let the random variables Y and X have a joint probability distribution P(X,Y) (recall that joint probability distribution is looking at the simultaneous behaviour of multiple random variables, or in other words, how likely certain values are for each random variable to take). With decision theory, we want to take the action that minimises the **expected loss/frequentist risk**. We wish to minimise:

$$R(f) = E[L(Y, f(X))]$$

whereby we are taking the expectation over P(X,Y) or the joint probability distribution (since we don't know what either x or y will be). Here, we need to choose a f(x) that generalises well over new unseen data, which is why we take the expectation over the joint distribution of P(X,Y). From this, the risk is for a given function f(.) Since the data is a random variable, we can use **law of iterated expectations** to rewrite the expected loss as:

$$R(f) = E[E[L(Y, f(X))]|X]$$

and then using the squared loss where L(Y,f(X)) = $(Y - f(x))^2$, we can get:

$$R(f) = E[E(Y - f(X))^2|X]$$

whereby the expected value of random variable is the sum of expected random variables. From all this, we are interested in computing the optimal prediction, so in order to do this, we should be minimising the expected loss. Therefore, the optimal action is to choose a prediction function $\delta(.)$ that minimises the expected loss (which is equivalent to minimising the expected loss at every input **x**).

$$\delta(\boldsymbol{x}) = argmin_{f(.)} = E(L(Y, f(x))|X = (x))$$

whereby the solution to the squared loss is the conditional expectation (taking the expectation over X and then minimising it by differentiating it):

$$\delta(x) = E(Y|X = x)$$

Since under the squared error loss, the optimal prediction of Y at any point X=x is the conditional mean $E(Y|X = x)$. We can also prove this:

$$E(L(Y, \delta)|X = x) = \int L(y, \delta)p(y|x)dy$$

since P(X,Y) is a density, p(y|x) is the conditional density of y given x, and we integrate over all values of y. We then plug in our squared loss.

$$= \int (y - \delta)^2 p(y|x)dy$$

We can then assume p(y|x) is a smooth function to allow for ease of differentiation so that we can minimise with respect to δ. This assumption allows for differentiation under the integral sign. (We swap y and δ around for ease of algebraic manipulation and the square gets rid of the negative sign anways).

$$= \int \frac{\partial}{\partial\delta}(\delta - y)^2 p(y|x)dy$$

$$= \int 2(\delta - y)p(y|x)dy$$

and then expanding it we get:

$$= 2\delta \int p(y|x)dy - 2\int yp(y|x)dy$$

Now note that in the first term, $\int p(y|x)dy$ is just a conditional probability distribution where we integrate over all values of y and just evaluates to 1. Then $\int yp(y|x)dy$ is just the definition of the conditional expectation of y given x so then we get:

$$= 2\delta(1) - 2E(Y|X = x)$$

Then we can set this equal to 0 in order to solve.

$$\delta = E(Y|X = x)$$

We can prove this is a minimum by taking the 2nd derivative of the equation to then get:

$$\frac{\partial}{\partial\delta} = 2\delta \int p(y|x)dy - 2\int yp(y|x)dy$$

$$\frac{\partial^2}{\partial\delta^2} = 2 > 0$$

therefore this is the minimum. This makes sense since if X and Y are related, and we are trying to predict Y, we need the help of X! Additionally, it is proven that using this $E(Y|X = x)$ has a mean squared error compared to other functions X can take on (no need to prove this). From this, what we want to do is to estimate the conditional expectation function $E(Y|X = \boldsymbol{x})$. In order to learn this though, we need to introduce assumptions which then leads us to statistical models. In the case of the linear regression model, it is assumed that that the conditional expectation function $E(Y|X = \boldsymbol{x})$ is linear:

$$E(Y|X = \boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{\beta}$$

When attempting to train a prediction function $\hat{f}(x)$, using data $\{(y_i, \boldsymbol{x_i})\}_{i=1}^N$, we have the true form (and unknown to us) being the **additive error model**.

$$Y = f(X) + \epsilon$$

whereby we **assume** this is the relationship between X and Y. We call f(.) as the unknown *regression function* which we are trying to figure out, so that we can then predict Y. Additionally, there is a random error term ϵ whose mean is $E(\epsilon) = 0$. Even if we plug in a value X into this model, there is still a random term ϵ and is also known as the **irreducible error**. If you think about it, it makes sense since if X = years of education and Y = salary, even if we had the exact number of years of education, we can't get the exact salary because there are still other factors outside the model. From this, we want to choose a prediction function $\delta(.)$ that minimises the expected loss for this model. When we select the squared loss as the loss function to use, and upon minimising the expected loss of the squared loss, we have that the optimal prediction of Y here is simply the conditional mean/expectation which is expressed as:

$$E(Y|X = \boldsymbol{x}) = E(f(\boldsymbol{x}) + \epsilon) = f(\boldsymbol{x})$$

With our new frameworks in mind, we can apply this to **linear regression model**. Note that linear regression is just a technique to analyse the linear relationship between variables, whilst the model form contains all the Gauss-Markov assumptions + normality of error terms. From this, **we assume that** the regression function that captures the population is:

$$f(X) = \beta_0 + \beta_1 X_1 + ... + \beta_P X_P$$

which then gives us the model when subbing this function into the error additive model:

$$Y = \beta_0 + \beta_1 X_1 + ... + \beta_P X_P + \epsilon$$

We attempt to **estimate the regression function** f(X) with $\hat{f}(\boldsymbol{x})$ with:

$$\hat{f}(\boldsymbol{x}) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + ... + \hat{\beta}_P X_P$$

whereby the vector $\hat{\boldsymbol{\beta}} = (\hat{\beta}_0, ..., \hat{\beta}_p)$ contains the *least squares estimates* of the model parameters.

From all this talk regarding statistical decision theory, we now want to **evaluate model performance** by estimating the *expected loss of a trained model* and to choose a **learning method** that finds and estimates an appropriate model such that we minimise our expected loss (which is just the conditional expectation from what we said earlier).

Model evaluation is simply estimating the expected loss of a trained model and in order to do so, we split the dataset into 2 parts. A **training set** for EDA, model building, model estimation, model selection (where we are aiming to estimate the regression function with $\hat{f}(\boldsymbol{x})$) and a **test set** which is used for *model evaluation* (whereby we estimate the expected loss of the model we selected from the training set). Resultantly, we need to ensure that the test data is never used until the end of the analysis and furthermore, we cannot tweak the model based on the results from the test dataset. The rationale is that we are trying to see how well our model will do for future data and if we tweak based on the test data, we are cheating as we are simply fitting the test data, but in no ways does it mean our model can then generalise and perform well for unseen data. Resultantly, we place 50-80% of data to the training sample whereby having a higher training sample will lead to more accurate model estimation but also higher variance in estimating the expected loss. This means that whatever expected loss we predicted, it does not mean that it'll be an accurate indication of the expected loss. Furthermore, we normally shuffle the data before partioning it to reduce bias.

1.2.2 Model Evaluation

Suppose we have M test observations $\{(\tilde{y}_i, \tilde{\boldsymbol{x}}_{\boldsymbol{i}})\}_{i=1}^{M}$ and predictions we generated from our model, which are a function of our *estimated* regression function $\hat{f}(\tilde{\boldsymbol{x}}_i)$ for i = 1,...,M. To evaluate the model's performance by using the **empirical risk/error** for the test set:

$$\hat{R}_{\text{test}} = \frac{1}{M}\sum_{i=1}^{M} L(\tilde{y}_i, \hat{f}(\tilde{x}_i))$$

We compute the sample risk and then compute the average risk. This is similar to the expected loss, where the expected loss can be thought of as the average loss. Here in the empirical risk, we take in our prediction and the true value(from the test data) and compute the loss. We now drop the   to make it easier to read.

For the loss function L(.), different choices of this loss function leads to different meaures of predictive accuracy. We can use the **squared** loss function whereby if we had **n** observations y_i and predictions from our regression function $\hat{y}_i = \hat{f}(\boldsymbol{x}_i)$. This means that our empirical risk becomes the **test mean squared error** for our test dataset which is:

$$\text{Test MSE} = \frac{1}{m}\sum_{i=1}^{M}(y_i - \hat{y}_i)^2$$

We can also extend this concept of mean squared error to any arbitrary sample of size n:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Then for any arbitrary sample, we can then also derive the **root mean-squared error** and the prediction R^2:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$$\text{Prediction } R^2 = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Note that the prediction R^2 is different to the usual R^2.

If we decided to use the absolute error loss function, we then have our **emprical risk/error** becoming the **mean absolute error** (MAE).

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} y_i - \hat{y}_i$$

However, this method is less easy to solve, which is why we tend to use the squared error loss. Here, the optimal prediction for δ is the **conditional median**, not the mean.

However, when evaluating algorithms with these metrics, they may be sensitive to **sampling error**, whereby the samples in which we evaluate over are not representative of the population. Resultantly, we should also report the sample uncertainty for every important estimate in our analysis. We can compute the standard error for the test MSE (since test MSE is just a sample average). The formula for this is:

$$SE(MSE) = \frac{1}{\sqrt(n)}\sqrt{\sum_{i=1}^{n}\frac{((y_i - \hat{f}((x)_i)) - MSE)^2}{n-1}}$$

However, we want to be able to generalise this. The **test/generalisation** error is the expected loss for the model estimated with the training data D.

$$\text{Err} = E_{X,Y}[L(Y, \hat{f}(X))|D]$$

whereby the expectation is over P(X,Y). Here, this says that, what is given the training observations we got, what would be the average loss with our regression function $\hat{f}(X)$ over the entire population of X and Y. The *test MSE* estimates the test error under the squared error loss. NOte that the empirical risk/error is a random variable since it'll change depending on the data set D whilst the generalisation error is over all possible datasets.

1.3 Bias-Variance Tradeoff

First, we consider the additive error model again with homoskedasticity for the error ϵ:

$$Y = f(X) + \epsilon$$

Prior to this, we treated $\hat{f}(.)$ as a given since our objective was to estimate the test error. Now we want to choose a method to learn the **predictive function** $\hat{f}(.)$. We are now interested in actually deriving this predictive function.

We have the **expected prediction error** for a new input point $X = x_0$ to be:

$$Err(\boldsymbol{x}_0) = E[(Y_0 - \hat{f}(x_0))^2 | X = \boldsymbol{x}_0]$$

whereby $Y_0 = f(x_0) + \epsilon_0$. The expectation is over both the ϵ_0 and sampling distribution of $\hat{f}(.)$. $\hat{f}\boldsymbol{x}_0$ changes each time depending on the dataset. The EPE is an expected loss. **This is different to generalisation error**

$$\text{Err} = E_{X,Y}[L(Y, \hat{f}(X))|D]$$

The generalisation has $\hat{f}(\boldsymbol{x}_0)$ as an **estimate** and the expectation is over the entire population P(X,Y), this makes the prediction function not dependent on a dataset. In this case, we assumed we have a fixed prediction function and we wish to generalise it. The EPE involves a random variable prediction function since the prediction function will change depending on what the dataset being used to train the prediction function is.

We can decompose EPE into **EPE = Irreducible Error + Reducible Error**. We can never reduce the irreducible error and places an upper bound on our predictions' accuracy. In choosing a method to learn, we only worry about the reducible error and wish to minimise this reducible error.

From this, we can see that the reducible error then is made up of two parts:

$$\text{Reducible Error} = \text{Bias}^2 + \text{Variance}$$

and this is known as the famous bias-variance tradeoff. We want our model to be flexible so that we can approximate complext relationships between Y and X. More complex models means lower bias but then it will be difficult to generalise since more effective number of parameters to estimate, which leads to a higher variance. Increasing model complexity always reduces the **training error** but there is an optimal level of complexity that minimises the **test/generalisation error**. All models are approximations and having models that are more realistic will not make it's predictive capabilities better, in fact, if too many parameters we can have a high variance and therefore not do so well. High variance can lead to the model modelling the noise rather than the intended output. This is also known as **overfitting** whereby the estimated model is too flexible. This is the case in which the model has memorised the training set.

We use the process of **model selection** which will allow us to choose the right model among options of different complexity on the training data. Here, model selection is also concerned with estimating the generalisation error similar to model evaluation. So note that model evaluation is interested in predicting the expected loss of a trained model whilst model selection is making sure we pick the optimal level of complexity.

A **parametric model** has a fixed number of parameters. These are fast to use and easy to interpret but requires strong assumptions. On the other hand, **non-parametric model** has the case that the number of parameters grow with size of training data. THese have a higher variance and can be infeasible for large datasets and are also somewhat infeasible. Furthermore, this can also be considered to also be a tradeoff between interpretability agaisnt accuracy. Note that there is no single model that works well for ALL problems, this is known as the **free lunch theorem**. From this, we need to be careful of speed-accuracy-complexity tradeoffs and data driven considerations.

1.4 Review Questions for Module 1

What is predictive modelling? Predictive Modelling is using methods to detect patterns and using patterns to predict future observations and aid decision making

What is the difference between cross-sectional prediction and forecasting? Cross sectional involves using collecting data on subjects and predicting response for new observations. Time series is forecasting future response variables based on past and current information. Can only be done for response variable only.

What is supervised learning? Supervised learning: You give it both response and features. Learn a function to predict Y based off training dataset X. We have labelled data.

What is a loss function? Loss function is the cost of predicting f(x) when the actual value is y

What do we learn from statistical decision theory for regression problems? Statistical decision theory allows us to evaluate model performance by estimating the expected loss. We can also find a learning method which will minimise expected loss.

How do we evaluate model performance? We can compute the empirical risk which is for the test dataset where actual value is y and predicted value. We have a loss function which measures the cost of inaccurate predictions on the test data and take the average of that. However this is specific to only particular datasets, so we can take the expectation of the empirical risk to get us the test/generalisation error. The empirical risk is an estimate of the generalisation error.

What is the difference between the generalisation error and the expected prediction error? Generalisation error is taking the expectation of the empirical risk/error. In the generalisation error, we suppose that the prediction function $\hat{f}(x)$ is a given/estimate. In the expected prediction error, the prediction function is now a random variable. Furthermore, the generalisation error is conditioned over the population whilst expected prediction error is conditioned over training sample.

What is the bias-variance trade-off and why is it important for predictive modelling? Bias-variance tradeoff is how if we reduce bias, we increase variance. More complex model means bias is reduced but variance is increased (since more parameters to estimate). We want to find optimal model complexity to minimise expected loss.

What is model selection? How is it different from model evaluation? Model selection is choosing a model under different complexity. Model selection is concerned with estimating the generalisation error whilst model evaluation is estimating the generalisation error too (expected loss of a trained model). Model selection is choosing and finding a model that minimises expected loss. Model evaluation is estimating expected loss of a trained model

What is overfitting? Model is too flexible and incorporates noise. Poor prediction since it has memorised the training set.

What is the difference between parametric and nonparametric models? What are the advantages and disadvantages of each approach? Parametric: Fixed number of parameters. Advantages are: Easy to interpret, fast computationally, and low variance. Disadvantages are: High bias and not flexible enough. Lots of assumptions.

Non-parametric: No fixed number of parameters. Grows with size of training data. Advantages are: Low bias, flexible. Disadvantages are: High variance, difficult to interpret, and can be computationally infeasible.

2 Linear Regression

For linear regression, we consider a regression function of the form:

$$f(\boldsymbol{x}) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p$$

whereby we learn the prediction coefficients $\hat{\beta}$ by fitting the model to the training data using the *least squares method.* Here, we are assuming the regression function is the form stated above.

Using $\{(y_i, \boldsymbol{x_i})\}_{i=1}^{N}$as the training data, we can use define RSS as a *function* of β.

$$RSS(\boldsymbol{\beta}) = \sum_{i=1}^{N}(y_i - f(\boldsymbol{x}_i; \boldsymbol{\beta}))^2$$

Here, **ordinary least squares** gives the maximum likelihood estimate for β under all the assumptions (note that the generalized least squares allows for heteroskedasticity). With this method, our loss function is the *squared error loss* whereby OLS is now minimizing the *empirical loss/risk* for our choice of predictive function (from the β parameters we choose).

For an invertible matrix (so no perfect multicollinearity) with a unique solution, we have that:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

The vector of fitted values for entire sample is:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

We can then define the **hat matrix** H to be:

$$H = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$$

Then vector of residuals are:

$$\begin{aligned}\boldsymbol{e} &= \boldsymbol{y} - \hat{\boldsymbol{y}} \\ &= \boldsymbol{y} - \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \\ &= (\boldsymbol{I} - \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T)\boldsymbol{y}\end{aligned}$$

Statistical Models are description of a *data generating process* based on a set of mathematical assumptions about the population and sampling process. In particular, a **regression model** is a description of the relationship of response Y and predictors $\boldsymbol{X}$ and is of the form: $P(y|\boldsymbol{x}, \theta)$.

2.1 Statistical Properties

Population parameter β is fixed and data is random sample. We apply an **estimator** $\hat{\beta}(D)$ to the data (where the estimator is the method and in this case, it is the OLS algorithm).

The sampling distribution of an estimator is when imagine we had many dataset $D^{(s)} = (1, ..., S)$ from the true model $p(y|x, \beta)$ whereby each has size N. Imagine applying our estimator (OLS algorithm) and obtaining a set of estimates $\hat{\beta}$. The sampling distribution is the induced distribution on $\hat{\boldsymbol{\beta}}(.)$ as S $\rightarrow \infty$. This is the sampling distribution of our estimator.

A **consistent** estimator is when it recovers the true parameter that generated the data when sample size goes to infinity.

An estimator is **unbiased** $\hat{\theta}(D)$ when E$(\hat{\theta}) = \theta$, whereby θ is the true parameter.

An estimator is **efficient** if it has the minimum variance.

2.2 Gaussian MLR Model

We now make distributional assumptions regarding the error terms. From the last part, we know some information such as the mean and variance regarding the estimators but if we assumed Gaussian error terms, then we can say more things about it. From this, the full form of the conditional distribution of Y is:

$$Y|X = \boldsymbol{x} \sim N(\beta_0 + \sum_{j=1}^{p} \beta_j x_j, \sigma^2)$$

2.3 Maximum Likelihood Estimation

This is fine when specify a full probabilistic model for the population. ML chooses parameters that maximises the likelihood of observed data under the model. **Likelihood function** is the joint PDF of data evaluated at sample values. We multiply PDFs for each observation. From that, we can take the log-likelihood and differentiate. This technique is equivalent to OLS except that now for statistical deicsion theory, we are now able to approximate the optimal prediction for any loss, since it estimates the full conditional distribution $P(Y|X = \boldsymbol{x})$

One final note is that for **causal analysis**, we are saying that E(Y|do X=x) which is an explicit intervention whilst we should be doing predictive modelling which tells us if we observe X=x, then we predict E(Y|X=x). Causal analysis requires appropriate **study design**.

2.4 Review Questions for Module 2

How do we obtain the OLS estimates? Go through the full process. Coefficient values which minimizes sum of square errors. OLS minimizes squared the empirical loss whereby the loss function is the squared error loss.

What is a sampling distribution? Dataset is random and estimator is the OLS algorithm. We apply OLS to numerous datasets. The distribution of beta-hat that is the induced distribution, which is the sampling distribution.

What is maximum likelihood estimation? What type of model is it applicable to? MLE chooses parameters that maximizes the likelihood of the data we observe. Useful for Gaussian error models. MLE is equivalent to OLS estimator. We look at Gaussian distributed errors when trying to estimate parameters for beta.

What is the difference between predictive and causal analysis? Casual: If we do X=x, then we expected E(Y|X=x). Predictive: If we observe X=x, then we predict E(Y|X=x)

3 K Nearest Neighbours

We define KNN as:

$$\hat{y}(\boldsymbol{x}) = \frac{1}{k} \sum_{\boldsymbol{x} \in N_k(\boldsymbol{x}, D)} y_i$$

for a training sample D= $\{(y_i, \boldsymbol{x_i})\}_{i=1}^{N}$. This is the sample average of the response values for the k training observations which are closest to the query point $\mathbf{x}$. For k=1, E($\hat{y}(\boldsymbol{x})$) = f($\mathbf{x}$) so KNN can approximate any regression functions without any assumptions. This will lead to a low bias method. Here, we need to specify 3 things for KNN: distance metric, k, predictors.

3.1 Distance

Normally we use Euclidean distance or the ℓ^2 norm. We can use the Mahalanobis distance if predictors on different scales since it takes covariance into account.

3.2 k neighbours

k is a hyperparameter and needs to be specified prior to learning process. We choose it based on bias-variance tradeoff. We can use model selection (whereby we choose optimal level of complexity) and estimate test error for each candidate value of k. We then select a k based of this criterion.

Bias-variance tradeoff for KNN is:

$$Err(\boldsymbol{x}_0) = \sigma^2 + [f(\boldsymbol{x}_0 - \frac{1}{k}\sum_{\ell=0}^{k} f(\boldsymbol{x}_\ell))]^2 + \frac{\sigma^2}{k}$$

whereby increasing k causes variance to fall but bias to rise and vice versa. KNN is subject to curse of dimensionality whereby it breaks down for high-dimensional inputs since difficult to find training observations close to prediction point $\mathbf{x}$. KNN is memory intensive and computationally costly.

Here, linear regression outperforms KNN is true form for regression function is correctly specified or close to it.

3.3 Review Questions for Module 3

How does the KNN method compute predictions? It takes the K nearest neighbors to it and compute the average of those values.

What is a hyperparameter? Hyperparameter are parameters we specify prior to the learning process. We can use cross validation on the training set to determine which hyperparameters to use.

What is the Euclidean distance between two points? The l2 norm of 2 vectors.

What is the curse of dimensionality? High dimensional inputs/features compared to data. Things become more complicated as the number of dimensions increase. Increasing number of features means more difficult to find training data to fit it.

Write and interpret the bias-variance decomposition for a KNN prediction. $\text{Err}(\mathbf{x}_0) = \sigma^2 + (f(x_0 - \frac{1}{k}\sum_{\ell=0}^{k} f(x)_\ell) + \frac{\sigma^2}{k}$ Increase k, variance falls but bias rises.

What are the advantages and disadvantages of the KNN method? Advantages: Makes no structural assumptions and therefore low bias. 1) Disadvantages: Computationally intense. High variance. Memory intensive. Very sensitive to values. Curse of dimensionality can occur.

4 Model Selection

Model selection methods estimates the generalisation performance of a model based on training data, which then helps us in choosing between models of different degrees of complexity. We select the model that is estimated to have the best predictive ability. There are 3 ways to do model selection:

4.1 Validation Set

Randomly split data into 2 and choose model with best predictive performance in validation set. Then re-estimate selected model by combining training and validation set and predict the test data (model that has been kept away this whole time) with the selected model.

4.2 Resampling Method/Cross Validation

We split data into k folds and estimate the folds on k-1 folds to predict the kth fold. Cross validation error is average error across K validaiton sets. Setting K=N is same as leave one out cross validation LOOCV. LOOCV has low bias but computationally expensive and high variance since training on nearly identical data each time. K-fold has more bias since less data but runs faster and less variance.

For this, we can use **one standard deviation rule** whereby we pick simplest model within one standard deviation of the model with lowest cross validated score. For the choice of K, it depends on trade-off between bias, variance, computational cost etc. Do note that LOOCV has a shortcut whereby we can automatically compute the MSE automatically based on the diagonals of the hat matrix **H**. From this, we have **generalised cross validation** which approximates leave one out MSE by using the trace of H matrix or any other matrix S. It is therefore computationally convenient.

4.3 Optimism

The **training error** is the empirical loss for the training data:

$$\bar{\text{err}}_D = \frac{1}{N}\sum_{i=1}^{N} L(Y_i, \hat{f}(\boldsymbol{x}_i))$$

and from this, using the squared loss, we get:

$$\bar{\mathrm{err}}_D = \frac{1}{N}\sum_{i=1}^{N} L((Y_i - \hat{f}(\boldsymbol{x}_i))^2)$$

If we used least squares for linear regression, this is equivalent to the RSS/N . This is just the error on the training sample we have.

The **expected prediction error** (EPE) is:

$$EPE(\boldsymbol{x}_0) = \sigma^2 + E_D[(f(\boldsymbol{x}_0 - \hat{f}(\boldsymbol{x})_0))^2]$$

Recall that EPE is the test/generalisation error but now averaged over all things that are random (so now our error term doesn't depend on a particular training dataset).

Now for a particular observation i within our sample, we can define the estimation error for i as:

$$E_D(Y_i - \hat{f}(\boldsymbol{x}_i))^2 = \sigma^2 + E_D[(f(\boldsymbol{x}_i - \hat{f}(\boldsymbol{x})_i))^2] - 2Cov(\hat{f}(\boldsymbol{x}_i), \epsilon_i)$$

This is analysing for just a single observation in our sample, what is the expected prediction error if we sum up over infinite datasets. Note that the reason we have covariance is that the estimator $\hat{f}(\boldsymbol{x}_i)$ is a function of D, which includes training case i itself. So therefore, there is some covariance in the estimation error. For OLS, we are aiming to minimise the difference between prediction and actual y value, so there is alot of dependence. We can then take the summation and then average of the training data set and compute estimation error for each observation to get us:

$$E_D(\mathrm{err}_d) = \sigma^2 + \frac{1}{N}\sum_{i=1}^{N} E_D[(f(\boldsymbol{x}_i - \hat{f}(\boldsymbol{x})_i))^2] - \frac{2}{N}\sum_{i=1}^{N} Cov(\hat{f}(\boldsymbol{x}_i), \epsilon_i)$$

This final term shows that the training error is not a good estimate of the expected prediction error. Since if the model complexity increases, the training error falls. Models that are too complex, overfits and underestimates the EPE since last term will fall dramatically.

We define **out of sample error** as:

$$\bar{Err}_{\mathrm{out}} = \frac{1}{N}\sum_{i=1}^{N} E[(Y_i^0 - \hat{f}(x_i))^2]$$

$$\bar{Err}_{\mathrm{out}} = \sigma^2 + \frac{1}{N}\sum_{i=1}^{N} E_D[(f(x_i) - \hat{f}(x_i))^2]$$

whereby $Y_i^0 = f(\boldsymbol{x}_i) + \epsilon_i^0$ is an independent case for a given $\boldsymbol{x}_i$. This is simply the same term as the expected prediction error for the training data, but now since $\hat{f}(x_i)$ is not a function of data D, there is no covariance term any longer. Error term doesn't decrease just because of increase in model complexity.

Optimism of the training error is therefore:

$$\bar{Err}_{\text{out}} - E(\bar{Err}_D) = \frac{2}{N}\sum_{i=1}^{N} Cov(\hat{f}(x_i), \epsilon_i)$$

The out of sample error $\bar{Err}_{\text{out}}$ is always greater than expected value of training error $E(\bar{Err}_D)$ since the EPE of training error has a downward bias. This means that, the higher the model complexity, the greater the degrees of freedom, the higher the covariance in the EPE training error, and the higher the optimism. This means we are overfitting dramatically and the EPE of training error will be much smaller relative to the out of sample error. This means we will generalise badly since the error on training error is too low as the same data was used to fit the model and also to assess its error.

For the next section, we then have that:

Out of sample error = training loss + penalty for number of parameters

For linear regression, the optimism term is:

$$\text{Optimism} = \frac{2}{N}\sigma^2(p+1)$$

whereby the larger sample size N means it is harder to overfit, more parameters p means more overfitting, or a larger variance of errors mean more overfitting.

4.4 Analytical Criteria

These estimate the **test/generalisation error** so they refer to making a single prediction out of sample. They have the form:

Criterion = training loss + penalty for number of parameters

4.4.1 Mallow's C_p statistic

This applies to linear regression whereby:

$$C_p = \frac{\text{RSS}}{N} + \frac{2\hat{\sigma}^2}{N}(p+1)$$

$\hat{\sigma}^2$ is an estimate of σ^2 based on *largest model* under consideration.

4.4.2 Akaike Information Criterion

AIC applies to models estimated by maximum likelihood.

$$AIC = -2L(\hat{\theta}) + 2d$$

whereby $L(\hat{\theta})$ is the maximized log-likelihood and d is the number of parameters. Minimizing the negative loglikelihood is equivalent to maximizing log-likelihood. We select the model with the lowest AIC. This follows the analytical criterion of a training loss + a penalty for number of parameters. AIC is an asymptotic approximation as $N \to \infty$.

4.4.3 Bayesian Information Criterion

BIC also is for maximum likelihood.

$$BIC = -2L(\hat{\theta}) + log(N)d$$

BIC penalises copmlexity more heavily when N $\geq$ 8. AIC gives a penalty of 2 but BIC gives a penalty of log(N).

4.5 Comparison of Model Selection

Consistency: In collection of models which includes correct model, probability model selection criterion chooses correct one approaches one when $N \to \infty$. This includes BIC. Often chooses models too simple due to heavy penalty on complexity.

Efficiency: Selected model predicts as well as theoretically best model under considerations in terms of expected loss when $N \to \infty$. This includes LOOCV, AIC, and Mallow's C_p. These are equivalent as $N \to \infty$ and that AIC and C_p are theoretical approxiations to LOOCV. This occurs since they select unbiased estimators but then select models that are too complex as $N \to \infty$ since penalty is too small. AIC and C_p are computationally faster than LOOCV but if assumptions are wrong for maximum likelihood, then LOOCV would be a better choice. LOOCV has no assumption requirements and therefore is a better choice.

These 2 properties are mutually exclusive.

4.6 Complete Subset Regression

CSR predicts response by taking simple average of predictions produced by linear regression. We specify a subset size S for a subset of predictors k, and then for all possible combinations of $k < p$ predictors whereby k is just a size we iterate through. Then we just take the average of the S predictions generated with k predictors.

A final note on all this is that statistical inference is no longer valid after model selection (estimating the performance of different models in order to choose the best one). Inference assumes a fixed model but model selection will pick the model that best fits the sample. This leads to overly optimistic estimates of sample variation based on chosen model. One way we can fix is this is by splitting data for model selection and inference.

4.7 Review Questions for Module 4

How does model selection relate to the bias-variance trade-off ? More complex model, lower bias but higher variance. Need to find right level.

What is a validation set? How is it different from a test set? Training set is used to fit the models. Validation set is used to estimate the prediction error for the model selection and to choose between right level of complexity for our model. Test set is used to asses the generalise/test error. It is used to see how well will model do on future data.

What is K-Fold cross validation? Describe how it works. Split data up into k folds. Use k-1 to train model and then predict last fold. Then save the mean squared error. Repeat each process for each fold. Then take average.

What is the one standard deviation rule? Choose the simplest model within one standard deviation of model with lowest cross-validated error

What is the Akaike Information Criterion? Has log-likelihood function. Minimising the negative of likelihood whilst at same time, penalising the complexity of the model. Has in-sample performance + penalty.

Why is it incorrect to conduct statistical inference after model selection (using the same data)? Statistical inference assumes a fixed model, but model selection selects the best model. Optimistic estimates of the model.

5 Resampling Methods

The **empirical distribution** is a discrete distribution since it defines a random variables that takes a value in the finite set D = $\{y_1, ..., y_n\}$. We note that we no longer have any distributional assumptions for inferences and therefore resampling methods are non-parameteric. We use these techniques since we don't require assumptions, allow for more accurate inferences, and is useful to check if large sample properties are accurate. However, resampling can still fail if observations are not independent whilst at the same time, being computationally expensive to run.

We have 3 resampling tests to look at: permutation tests, bootstrapping, and cross-validation.

5.1 Permutation Test

We have a treatment group m and a control group n. To run this, we look at the difference between 2 groups. Then, we compute all different permutations of the dataset ACROSS the 2 groups. We then note the number of samples that are greater than the difference we initially computed. We can then calculate the probability to be the fraction of permutations that is greater/less than the initial difference we calculated. We make the assumption under the null that all of these datapoints come from the same data generating process and therefore are indistinguishable. However, if if sample size too larger, then too many differences to compute and therefore not feasible. From this, we turn to another technique.

5.1.1 Monte Carlo Sampling

As before, we assume the data is from the same distribution under the null hypothesis. Here, we base our results on several random permutations of the data. We sample m observations *without replacement* from the pooled data of m+n observations. We label m observations as the treatment group and assign remaining n to the control group. We then compute permutation test as usual for a certain number of replications R. Here, we are only doing R permutations rather than all permutations like before. We can multiply test statistic by 2 if 2-sided test. Luckily, this test is still robust to deviations from this assumption.

These techniques may cause results to be different to traditional ones because of the skewness in the data. If the data is too small, then large sample theorems won't help us. These permutation distributions helps to reflect the impact of the skewness.

5.2 Bootstrapping

Here, we sample **WITH replacement**. Computational method to estimate sampling distribution of methods. This method is good for small sample sizes. We approximate the sampling distribution of a statistic by resampling the data and recomputing the statistic several times.

More formally, we draw many different datasets of size n from population p(y;θ) and then apply the estimator $\hat{\theta}(.)$ to each dataset and obtain a set of estimates. The sampling distribution of the estimator $\hat{\theta}(D)$ is the induced distribution on $\hat{\theta}_s$ as $S \to \infty$.

The bootstrap's sample approximates the population in which it was drawn from since drawing samples from sample, is similar to drawing samples from population. Therefore, distribution of statistics over bootstrap sample approximates sampling distribution of the statistic.

Bootstrap standard error of statistic is just standard deviation of the bootstrap distribution of that statistic. **Bootstrap bias estimate** is the mean of bootstrap distribution less the original sample statistic.

Bootstrap distribution is not accurate approximation to center of sampling distribution, instead it is approximate to original value of statistic. The bias of bootstrap is a noisy estimate of bias of sampling distribution.

Spread and skewness of bootstrap reflects spread of sampling distritbuion.

Bootstrap percentile intervals means to get all values from bootstrap and pick the value corresponding to the interval in which we are estimating. Bad idea since it'll be too small.

We can also compute test-statistic and then have a t-distribution distribution. We can compute **bootstrap t confidence interval** by using our original's sample t-statistic and standard error alongside the quantiles from t-distribution. These are the most accurate (more than standard ones). Performs poorly for statistics depending on small number of observations.

5.3 Bootstrapping Regression Models

We can bootstrap observations (predictors are random) or residuals (predictors are fixed).

5.3.1 Bootstrapping Observations

Directly sample as usual. Then compute regression estimator for each of bootstrap samples. Then we have our estimates from induced distribution.

5.3.2 Bootstrapping Residuals

Treat predicts as fixed and resample only residuals. Estimates sampling distribution conditioned on predictors.

1) Estimate regressor coefficients on data to get fitted model and residuals.

2) Generate bootstrap samples of residuals. Now our bootstrapped response vavlues are the fitted values + bootstrapped sample residuals.

3) For each bootstrap sample, regress bootstrapped response variable on fixed design matrix X.

4) Bootstrapped coefficients from step 3 are the results of what we want. We then have an induced distribution after repeating it numerous times.

Results from bootstrapping are subjected to **Monte Carlo error** whereby the more replications, the better but then this is computationally expensive.

5.4 Review Questions for Module 5

What is sampling with and without replacement? Sampling with replacement: Replacing observation. Without replacement: no replacement.

What is resampling? Resampling are non parametric methods to statistical inference based on computations rather than assumptions and large sample sizes.

What is a permutation test? Compute many resamples without replacement for certain number of samples. Compute the differences between them. Calculate how many observations above the observed test statistics. Times 2 if 2 sided test.

What is the key idea of the bootstrap method? Population to sample is the sample to the bootstrap sample. Sample approximates population its drawn from. Sampling distribution of bootstrap approximate sampling distribution of sample.

How do we obtain bootstrap confidence intervals? Bootstrap percentile intervals: Calculate bootstrap samples. Then use percentile of them from the quantiles. For the Bootstrap t CI: Compute numerous t test statistics (bootstrap – sample stat)/SE(bootstrap sample SE). use t-distribution as empirical quantiles for standard error original sample

What are advantages and disadvantages of the bootstrap relative to large sample approximations? Advantages: No assumptions and nonparametric. Variance, skew, spread of distribution is good approximation. Also reflects bias of original sample but still noisy bias. More accurate inference since no distribution assumption. Validity check for n>=30 assumptions. Disadv: Large computation cost. Not accurate centre approximation to sampling distribution.

6 Estimation Methods

Relative straightforward parts of the course.

7 Variable Selection

8 Regularisation Methods

9 Dimensional Reduction

What are dimension reduction methods for regression Reduce the number of predictors p. Build M linear combinations of the predictors. *What is PCA?* PCA: Looking at variance and constructing components

*What is PCR*Regressing on 1:M PCR components and picking best model based on CV

Difference between PCR and PLS PCR is unsupervised and uses only variance. PLS is supervised and uses correlation with y too.

*Purpose of robust regression*Estimation based on squared loss does badly when outliers present. Poor fit when outliers present. We can use LAD or Huber Loss.

10 Classification

Classification is the scenario where we have discrete Y response variable as a **Qualitative or Categorical** variable. This means it can take a value in the **finite unordered set** γ = **{1,...,C}** where C is the number of classes. We want to be able to predict which class does a subject belong to based on input variables we have aka X. More formally, we want to derive a **classifier** $\hat{Y}(X)$ which maps an input vector **x** to an instance of {1,...,C}. We can think of a classifier as a prediction rule, whereby based on this rule, we can assign subjects to a class based on **x**. For example, if we had fruits, our Y value can take on the values of Y={Apple, Oranges, Bananas}. From this, if we are given a mysterious fruit, we want to be able to classify which class/fruit it belongs to. Furthermore, we estimate the probability that the subject belongs to the class C. Note that the labels assigned tends to be arbitrary if we use numbers to represent classes such as letting 1 = sunny, 2 = rainy etc.

From here, we let capital P denotes probability. So for the probability of A, we can denote this as P(A). Lowercase p(A) would mean the **probability mass function** (if A is a discrete r.v) or a **probability density function** (if A is a continuous r.v). Do note that there are differences! In probability theory and statistics, a probability mass function (pmf) is a function that gives the probability that a discrete random variable is exactly equal to some value. In probability theory, a probability density function (PDF), or density of a continuous random variable, is a function that describes the relative likelihood for this random variable to take on a given value. Probability mass function and probability density function would look similar except that in case of a continuous variable the probability that the variable is exactly equal to some value is zero. The probability density function/mass function is **different** to probability. The PDF is something we integrate over to find the probability. p(A) is a 'density function' f(X) whilst probability is a value realised over the range of [0,1].

10.1 Probability Theory Review

10.1.1 Sum rule, Product rule, and Bayes Theorem

Discrete r.v X takes on a value in finite/countably infinite set $\chi = \{1, 2, 3, 4, ..\}$. We denote that probability of an event X=x by P(X=x) or P(x). **We will be working with discrete random variables for the next section**, hence why we sum up rather than integrate the random variables. This is the **probability mass function** we mentioned earlier!. Here, p(x) is: 0≤ p(x) ≤ 1 and the total summation of p(x) gives us 1.

$$\text{Recall: } P(X \cup Y) = P(X) + P(Y) - P(X \cap Y)$$

If X and Y are mutually exclusive, P(X ∪ Y) = P(X) + P(Y)

$$\text{Product rule (probability of joint event):} P(X \cap Y) = P(X, Y) = P(X|Y)P(Y)$$

This tells us what the probability of the joint event of X and Y occurring. Also, we can also compute marginal probabilities when given the joint distribution P(X, Y).

$$\text{Marginal Probability: } P(X = x) = \sum_{y \in Y} P(X = x, Y = y)$$

and then by subbing in: $P(X = x, Y = y) = P(X = x|Y = y)P(Y = y)$, we can get:

$$P(X = x) = \sum_{y \in Y} P(X = x|Y = y)P(Y = y)$$

This is known as the **Sum rule**. This calculates the probability of x taking on a certain value (so we hold it fix since we set X=x) over all possible values of Y (as we sum up over all possible values of y $\sum\limits_{y \in Y} Y = y)$. Y here stands for a discrete set of values in which y can take. Calculating the marginal probability is based on the joint probability and marginalise all possible values/outcomes that Y can take. We are asking, over all possible values of Y, what is the probability X=x.
It then follows from the product rule that:

$$\text{Product rule: } P(X, Y) = P(X \cap Y) = P(X|Y)P(X)$$

Can be rewritten as:

$$P(X|Y) = \frac{P(X, Y)}{P(Y)}$$

or

$$P(Y|X) = \frac{P(Y, X)}{P(X)}$$

Where denominator P() > 0. This is known as the **conditional probability**, which tells us how likely is X going to happen given that Y has happened already.

Recall that we are only considering the case where X and Y are discrete random variables, we can now define **Bayes Rule** (which is super important for finals).

$$\text{Bayes Rule} = P(Y = y|X = x) = \frac{P(Y = y, X = x)}{P(X = x)}$$

Then we can sub in:

$$P(Y = y, X = x) = P(X = x|Y = y)P(Y = y)$$

We want it this way around because Bayes rule already has P(Y=y|X=x). We can also sub in:

$$P(X = x) = \sum_{y \in Y} P(X = x|Y = y)P(Y = y)$$

Which then gives us our final form of:

$$\text{Bayes Rule} = P(Y = y|X = x) = \frac{P(X = x|Y = y)P(Y = y)}{\sum\limits_{y \in Y} P(X = x|Y = y')P(Y = y')}$$

This again tells us, what is the probability of Y=y, given that X is realised as x.

Now, in the case that X is continuous, we have that:

$$P(Y = y|X = x) = \frac{p_x(x|Y = y)P(Y = y)}{p_x(x)} = \frac{p_x(x|Y = y)P(Y = y)}{\sum\limits_{y' \in Y} p_x(x|Y = y')P(Y = y')}$$

whereby $p_x(.)$ and $p_x(.|Y = y)$ are density functions.

This helps prevents us from the **base rate fallacy** which is where basic probability does account for how prevalent or likely is Y=y. So for the cancer example in lecture, the reason why the probability in the end is so small, is because the probability of getting cancer is so small to begin with, and therefore if the mammogram says you have cancer, it is more likely the mammogram is incorrect in detecting cancer rather than you actually having cancer.

10.1.2 Multinomial Distribution

Multinomial distribution is a generalisation of Bernoulli distribution whereby it now accounts for multiple outcomes rather than 2. A **multinomial random variable** can take on a single value from y $\in$ {1,...,C}.

The probability of an outcome is then given by:

$$\text{P(Y=c)} = \pi_c$$

for c = 1,...,C. The summation of probabilities must obviously equal 1: $\sum\limits_{c=1}^{C} \pi_c = 1$.

From this, we can then derive the **multinomial distribution**

First we denote a multinomial random variable as Y $\sim$ Cat$(\pi_1, \pi_2, ..., \pi_C)$

This means that unlike the Bernoulli distribution, we have C categories where only one is the actual outcome/success.

The **Probability Mass Function** is defined as:

$$p(y) = \prod_{c=1}^{C} \pi_c^{I(y=c)}$$

Where I(y=c) is an **indicator function**. This means that p(y) = π_c when y = c.
From this, we get the multinomial distribution which is defined as:

For n independent trials each of which leads to a success for exactly one of k categories, with each category having a given fixed success probability, the multinomial distribution gives the probability of any particular combination of numbers of successes for the various categories.

10.2 Decision Theory Classification

10.2.1 Loss Function for Classification

As you should recall, the loss function simply tells us the cost of making a wrong prediction. It is a function of the actual value and value we predicted. We can represent this in classification problems with a $C \times C$ **loss matrix L**. Each entry of the loss matrix, specifies the loss of classifying/predicting an observation in a class l when in fact the actual class was k. In mathematical notation, we say that $L_{k,l} = Loss(k,l)$ represents the loss of classifying an observation in class l, when in actual fact it belongs to class k. From matrix notation, recall that the first letter in subscript refers to rows and second one refers to column. So in our loss matrix, the rows denote the actual class whilst the columns denote the predicted class.

In this course, we focus on the **0-1 Loss Function** whereby if we classify a class correctly, the loss is 0 (so the cost is 0). If we classified a class incorrectly, we set the loss of that misclassification to be 1. Therefore our loss matrix L will just comprise of 0's and 1's. For every classification outcome, we have a predicted label and an actual label.

$$L(y, \hat{y}) = \begin{cases} 1 & \text{if} \quad \hat{y} \neq y \\ 0 & \text{if} \quad \hat{y} = y \end{cases}$$

From this our loss matrix looks like:

	$\hat{A}$	$\hat{B}$
A	0	1
B	1	0

From this table, we can see that if we classify A correctly then we get a cost of 0 by looking at the 0,0 entry in our matrix. Likewise we can see the cost is 0 for the 1,1 entry in our table.

10.2.2 Classification Risk

As we can recall, risk is also known as the expected value of the loss function. In essence, we take the average of the loss function. From this, we seek to minimise the risk or expected loss of our classifier:

$$\text{R}(\hat{Y}(\text{X})) = E[L(Y, \hat{Y}(X))]$$

Notice that it is uppercase Y and X, which denotes a r.v. This means that we are taking the average loss across all subjects in the population whereby each subject in the population has the pair of X and Y respectively.
We can then condition it on the predictors for us to get:

$$\text{R}(\hat{Y}(\text{X})) = E_X[\sum_{c=1}^{C} E[L(c, \hat{Y}(X))]P(Y = c|x)]$$

10.2.3 Bayes Classifier

We want to minimise the loss for any given input.

$$\hat{Y} = \arg\min_{k \in y} \sum_{c=1}^{C} L(c, k)P(Y = c|X = x)$$

For 0-1 Loss function, the problem of choosing optimal classifier now becomes trying to minimise the probability of a misclassification.

$$\hat{Y} = \arg\min_{c \in y}[1 - P(Y = c|X = x)]$$

This is known as the **Bayes Classifier**. A Bayes Classifier minimises the probability of misclassification and therefire classifies the subject to the most probable class. We choose a class which minimises the expected loss. So if P(Y=1|X=x) = 0.6, then the expected loss is (1-0.6) since that is the probability of us being wrong. So from this, we are trying to choose a prediction k which minimises the likelihood of us making a mistake.

10.2.4 Bayes Error Rate

We can rewrite the Bayes Classifier from the last part so that we can view it as us picking a class that maximizes the probability:

$$\hat{Y} = \arg\min_{c \in Y}[P(Y = c|X = x)]$$

such that

$$P(Y = \hat{Y}|X = x) \geq P(Y = c|X = x) \forall c \in Y$$

We classify the observation into the class with the highest probability. The **Bayes error rate** is the expected 0-1 loss under the Bayes classifier. It is the lowest possible error rate. It is the minimal misclassification rate we can get from this boundary.

Here, we are choosing the Y that maximises the probability (what is the most likely class given x).

10.2.5 Bayes Decision Boundary

The decision boundary tells us the boundary between 2 classes. In other words, it shows us the boundary of when the probability of an observation belonging to class k is equal to the probability of the observation belonging to class l. So for classes k and l, it is the set of **x**

$$\{\boldsymbol{x} : P(Y = k|X = \boldsymbol{x}) = P(Y = l|X = \boldsymbol{x})\}$$

So the Bayes error rate can also be known as the probability of getting an observation on being on the wrong side of the Bayes decision boundary.

However in practice, we don't know what the actual probabilities are, so we need to estimate the probabilities ourselves. Therefore, we get the **estimated conditional probabilities** where $\hat{P}(Y = c|X = x)$ for c = 1,..,C. From this, we classify an observation into the class with the highest estimated probability.

In the case of binary classifications, we classify a subject to be 1 for $\hat{y} = 1$ if $\hat{P}(Y = 1|X = x) \geq 0.5$. From all this, we can now evaluate the **misclassifcation** or **error rate** for the test data as:

$$\bar{Err}_{test} = \frac{1}{n}\sum_{i=1}^{n}\frac{I(y_{i,0} \neq \hat{y}_{i,0})}{n}$$

10.3 KNN Classifier

KNN classifier estimates conditional probability for class c by:

$$\hat{P}(Y = c|X = x) = \frac{1}{K}\sum_{x_i \in N_k} I(y_i = c)$$

for a training sample D = $\{(y_i, \boldsymbol{x_i})\}_{i=1}^{N}$.

Again, we need to select the distance metric used and the number of neighbours k. The indicator function switches to 1 when the observation's response value is the same class as we are trying to compute the class' conditional probability. We find the k nearest training points which are closest to **x** and compute the conditional probability as the fraction of points that belongs to class c. In essence, we carry out majority voting, whereby we classify an observation to be the class that is most frequent around that observation. KNN here is a direct nonparametric approximation to the Bayes classifier. Again, as we lower K, we have a more flexible decision boundary so we have a lower bias but higher variance. If K is too low, we will create little islands and the decision boundary will be too flexible. However if K is too large, we won't have enough flexibility and the decision boundary will not be so accurate. We can use cross validation in order to select K. From this, KNN tends to perform well for nonlinear decision boundaries.

10.4 Discriminative vs Generative Models

Generative models examines joint probabilities whilst discriminant looks at conditional probabilities.

Generative: Given class c and observed data D, we then attempt to maximimize the joint distribution P(c,D) by adjusting the relative frequencies. This is used in Naive Bayes and other models. We can think of the class c generating the data D (since we have a prior probability P(c) in our models too).

Discriminative: Given class c and observed data D, we then attempt to maximize the conditional probability of P(c|D), that is, given the data D we have just observed, what class c is the most likely. This is used in methods such as logistic regressions, KNN, classification trees and so on. Discriminative classifiers directly estimates the class conditional probabilities p(y=c|x). The distribution of the inputs are arbitrary.

10.5 Naive Bayes Classifier

Generative classifiers: These are models that specifies how to generate the data given the **class conditional densities** $p(x|y = c)$ and the **prior class probabilities** p(y=c). This is a model for the joint distribution p(y,x). We can think of it as there is a model y, that is generating observations x. We can compute the conditional probabilities for classification using Bayes' theorem:

$$p(y = c|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y = c))y = c}{\sum\limits_{c' \in Y} p(\boldsymbol{x}|y = c')p(y = c')}$$

By taking the prior probability into account, we classify an observation into the highest input density. We can rewrite this by letting the prior $\pi_c = p(y = c)$, and the density function of **X** being $f_c(X) = p(x|y = c)$. We then get Bayes theorem as:

$$p(y = c|\boldsymbol{x}) = \frac{f_k \pi_c}{\sum\limits_{c=1}^{k} f_c(x)\pi_c}$$

The prior class probability p(y=c) tells us the probability that a given observation is associated with the class c, whilst p(x|y=c) tells us the density function for an observation that comes from class c. p(x|y=c) will be high if its likely that the observation x comes from class c. Our estimate of π_c is simply the fraction of the training observations that belong to class c. For $f_c(X)$, we need to make some assumptions to do so and to estimate $f_c(X)$ will be in the section of LDA/QDA.

Now moving on to the actual concept of Naive Bayes! The **Naive Bayes classifier** is a simple generative model based on the assumption that predictors are **conditionally independent** given the class label. It is "naive" because we do not in fact believe that features are not conditionally independent. However, Naive Bayes makes it hard to overfit, which is quite good, which is useful for when we have a large number of features. This relates back to bias-variance tradeoff since the assumption of class-conditional independence leads to more biased probabilities but also reduces the variance, which can be substantial if we have numerous features that are correlated. Assumptions of conditional independence makes it easy to mix and match different predictor types. From this assumption, the class conditional density becomes:

$$p(\boldsymbol{x}|y = c) = \Pi_{j=1}^{p} p(x_j|y = c)$$

since they are independent.

Now we can specify functional forms for the predictors. For real valued predictors, we can now assume that:

$$X_j|Y = c \sim N(\mu_{jc}, \sigma_{jc}^2)$$

whereby μ_{jc} and σ_{jc}^2 are the mean and variance of the predictor j, conditional on the class c. From this, if we assume Gaussian form. we get:

$$p(x_j|y = c) = \frac{1}{\sqrt{2\pi\sigma_{jc}}} exp(-\frac{(x_j - \mu_{jc})}{2\sigma_{jc}^2})$$

If we had binary predictors whereby $x_j \in \{0, 1\}$, we can use the Bernoulli distribution:

$$X_j|Y = c \sim Bernoulli(\theta_{jc})$$

whereby θ_{jc} is the probability that feature j occurs in class c.

Furthermore, in the case of categorical features, $x_j \in \{1, ..., K\}$, we can use the Multinoulli distribution:

$$X_j|Y = c \sim Cat(\boldsymbol{\theta}_{jc})$$

whereby $bm\theta_{jc}$ is a K-vector of probabilities conditional on class c.

Often we transform the predictors in order to make the variable approximately normal or symmetric. We can use techniques such as the Box-Cox transformation. We can also use other distributional assumptions or use nonparameteric approaches to estimate the class conditional densities by using kernel densitity estimates. We can easily combine parametrics and nonparametric univariate models for predictors. For Naive Bayes, LDA, and QDA, we need to normalise the variables first.

So recall from Bayes' theorem we had that:

$$P(y|x_1, ..., x_p) = \frac{P(x_1, ..., x_p|y)P(y)}{P(x_1, ..., x_p)}$$

and from the Naive independence assumption, we get that $P(x_j|y, x_1, ..., x_p) = P(x_j|y)$. Therefore, we get:

$$P(y|x_1, ..., x_p) = \frac{\Pi_{i=1}^{p} P(x_j|y)P(y)}{P(x_1, ..., x_p)}$$

We can now apply Naive Bayes to document classification. Here, we can represent each document as a vector of binary variables, whereby if a word j appears in document i, we then set $x_{ij} = 1$, else if it doesn't appear, we set $x_{ij} = 0$. This is known as the **bag of words** model. From this, we can get the *class conditional probability mass function* as:

$$p(\boldsymbol{x_i}|y_i = c, \boldsymbol{\theta}) = \Pi_{j=1}^{p} \theta_{jc}^{x_{ij}} (1 - \theta_{jc})^{1 - x_{ij}}$$

whereby p is the number of words in the dictionary, θ_{jc} is the probability that the word j appears in a document of type c.

We can estimate the Naive Bayes model via MLE as seen next:

1) We estimate the prior class probabilities by computing the sample proportions of each class in the training data.

2) We fit univariate models separately for each predictor within each class.

Look at lecture slides for derivation of MLE for Naive Bayes.

Essentially, Naive Bayes is just MLE. The class priors after maximizing log likelihood is π_c = $\frac{N_c}{N}$. Naive Bayes works really well in practice in comparison to more complex alternatives. This again is because of its relatively low variance yet high bias approach.

Learn how to compute things for Naive Bayes including if we had 2 features for exam. Also make sure you know how to compute bag of words with Naive Bayes..

10.6 Decision Theory for Binary Classification

Normally, there are distinct losses associated with each classification outcome. Sometimes a false positive is worse and more costly than a false negative etc. We can denote the possible terminologies as:

	$\hat{Y} = 0$	$\hat{Y} = 1$
Y = 0	True Negative	False Positive
Y = 1	False Negative	True Positive

We always refer to the prediction made. So if we classify it as 0, it is a negative, else it is a positive since we classify $\hat{y}$ as 1. Then whether it's false or true, depends on the actual outcome. So for the 0,0 index, we classified an observation as 0, so that makes it a negative. Then to determine whether it is true or false, we look at the actual value. The actual value is 0, and therefore since we classified it as 0, and it IS 0, this makes it a true negative.

We can further extend upon this to create a **loss matrix**.

	$\hat{Y} = 0$	$\hat{Y} = 1$
Y = 0	L_{TN}	L_{FP}
Y = 1	L_{FN}	L_{TP}

Here, the initials also stand for whether it is true/false and positive/negative.

From this, we can now classify a subject as positive or negative based on a new **decision rule**.

$$\delta(x) = \begin{cases} 1 & \text{if P(Y=1|X=x)} > \tau \\ 0 & \text{if P(Y=1|X=x)} \leq \tau \end{cases}$$

Where τ is a decision threshold parameter. What this means is that if we set the threshold to be 0.6, and if the probability of the subject being in the class is greater than 0.6, then we classify it to be in the class. For example if we are classifying whether a Pokemon was a fire type or not, if the probability of the pokemon being a fire type was greater than 60%, then we classify that Pokemon to be a fire type.

From this, it is easy to see that the value of τ we set is what determines and influences the classifications we make. Therefore, our decision problem is to select an optimal threshold τ.

$$\tau^* = \arg\min_{0<\tau<1} E[L(Y, \delta_\tau(x))|X = x]$$

In basic English, we are trying to minimise the expected loss. The expected loss depends on the difference between the actual value Y and the classification made depending on the value of δ_τ we use, conditioned on the subject having the observed values X=x.

Furthermore, we can simplify the notation by setting: π = P(Y=1|X=x). We can break the expected loss up in the above equation into 2 parts: the expected loss if we classified the subject as positive (which include true positive and false positive rate) and the expected loss if we classified the subject as negative (which involves the true negative and false negative rate). From this, we get the expected loss to be:

$$E[L(Y, \delta_\tau(x))|X=x] =$$

$$\pi L_{TP} + (1-\pi) L_{FP} \text{ if } \delta_\tau(x) = 1$$

$$\pi L_{FN} + (1-\pi) L_{TN} \text{ if } \delta_\tau(x) = 0$$

Therefore, intuitively, our expected loss is the loss from classifying a subject as a postive and classifying the subject as a negative. Remember that π is the probability that the subject actually belongs to class Y=1 (in our example, it is a fire pokemon). So the first line says: the expected loss if we classified the Pokemon as a fire type, is the probability that the Pokemon is a fire type multiplied by the True Positive Loss (we've correctly identified a fire type Pokemon) + the probability that the Pokemon is not a fire type Pokemon multiplied by the false positive type error (so we said a Pokemon was a fire type but it actually is not a fire type). **Our decision is therefore picking the option which minimises the expected loss and that is our final classification**. We pick the line (and classification) which will giveu us a lower expected loss.

Optimal decision threshold corresponds to the probability value π such that the loss from a positive or negative classification is equal.

$$\tau^* L_{TP} + (1-\tau^*) L_{FP} = \tau^* L_{FN} + (1-\tau^*) L_{TN}$$

From this, we can solve for our optimal threshold τ^*:

$$\tau^* = \frac{L_{FP} - L_{TN}}{L_{FP} + L_{FN} - L_{TP} - L_{TN}} \tag{1}$$

Note that the level of τ^* depends on the levels of losses for the false or true positives/negatives. If a false negative is very painful (therefore L_{FN} is very large), then the decision threshold τ^* will be relatively low, because we rather classify subjects as 1 instead of 0 in order to avoid false negatives (saying it is 0 when in fact it is 1). This is done to minimise the loss.

10.7 Model evaluation for Binary Classification

We can evaluate the model by computing the **misclassification** or error rate for the test data as:

$$\bar{Err}_{test} = \frac{1}{n}\sum_{i=1}^{n} \frac{I(y_{i,0} \neq \hat{y_{i,0}})}{n}$$

Here, we count the number of misclassifications we have and then take the average of it. It is an indicator function whereby it takes on 1 when our actual classification does not equal the true value.

We can also compute the **confusion matrix**.

	$\hat{Y} = 0$	$\hat{Y} = 1$
Y = 0	True Negative	False Positives
Y = 1	False Negative	True Positives
Total	Negative Predictions	Positive Predictions

Summation of the top row gives us the total number of negative N. Summation of middle row gives us the total number of positive P.

From all this, we can estimate the **generalisation error** using the loss and confusion matrices. We need to also quantify the uncertainty in the estimate by reporting the standard error.

We can compute the test error (which is an estimate of the expected loss for the classifier by):

$$TestError = \frac{T_N L_{TN} + F_P L_{FP} + F_N L_{FN} + T_P L_{TP}}{\text{Size of test data N}} \tag{2}$$

Sensitivity/Recall/True Positive Rate: This tells us how accurate we are in estimating the positive values. Formula is:

$$P(\hat{Y} = 1|Y = 1) = \frac{TP}{TP + FN} = \frac{TruePositives}{ActualPositives} \tag{3}$$

Recall is looking at out of how many positives we classified, how many are actually positive (which is the same definition as sensitivity).

Specificity/True Negative Rate: This tells us how accurate we are in estimating the negative values. Formula is:

$$P(\hat{Y} = 0|Y = 0) = \frac{TN}{TN + FP} = \frac{FalseNegatives}{ActualNegativs} \tag{4}$$

To explain the denominator for the true postive rate, the total number of positives are the observations that are positive we correctly identified (true positives) + the observations we didn't correclty identify as positive (false negatives since we identified those positive values as negative instead).

False Positive Rate (FPR)/ Fallout: This tells us out of all the negative values, how many were incorrect positive classifications.

$$P(\hat{Y}=1|Y=0)=\frac{FP}{TN+FP}=\frac{FalsePositives}{ActualNegatives}=1-\text{Specificity} \tag{5}$$

False Negative Rate (FNR): This tells us that out of all the positive values, how many were incorrect negative classifications.

$$P(\hat{Y}=0|Y=1)=\frac{FN}{TP+FN}=\frac{FalseNegatives}{ActualPositives}=1-\text{Sensitivity} \tag{6}$$

Additionally, we have something called **precision** which is:

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

This measures out of everything we classified as positive, what was actually positive.

From all this, we have an important concept that there is a **trade-off between sensitivity and specificity**. A classifier can obtain maximum sensitivity by setting $\tau = 0$, since then the classifier will classify all observations as positive and therefore, using our formula for sensitivity, we will not have any false negatives (since we never classified anything as negative). This means that the sensitivity will be 1. However, we can also think of there being a trade-off between **sensitivity** and achieving a **lower false positive rate**. Furthermore, this holds in vice versa whereby if we set $\tau = 1$, we will get maximum specificity, since we always classify an observation as negative. Thus, if we want to get a better sensitivity, we should lower the threshold τ and if we want a better specificity, we should raise the threshold τ.

$$\delta(x)=\begin{cases} 1 & \text{if P(Y=1|X=x)} > \tau \\ 0 & \text{if P(Y=1|X=x)} \leq \tau \end{cases}$$

Recalling our decision rule, by lowering τ, then we will classify more things as positive.

Receiver Operating Characteristic/ROC curve: This plots the sensitivity agaisnt the specificity or false positive rate (which is 1-specificity) for a range of threshold values τ. From this, this tells us the false positive rate we need, in order to obtain a certain sensitivity or true positive rate we want to get. We can summarise the ROC curve as a single number known as the **Area Under Curve AUC**. A higher AUC score is better and is maximized at hte value 1. A diagonal straight line indicates a 50-50 chance in classifying a positive accurately. The closer the curve is to the top left corner, then the better the tradeoff between the sensitivity and the specificity rate.

An issue we haven't touched on is **class imabalances**. Sometimes, the frequency that positive values may appear much more frequently than the negative values. Therefore, the specificity will not be informative since it will be high regardless of quality of classifier. Link on more information: http://www.chioka.in/class-imbalance-problem/.

Precision: For imbalanced classes, we are more interested in the proportion of detections that are acutally positive. Therefore we define precision as:

$$P(Y=1|\hat{Y}=1) = \frac{TP}{\text{TP + FP}} = \frac{\text{True Positives}}{\text{Positive Classifcations}} \tag{7}$$

This tells us out of things we classified as positive, how accurate are we and are we getting those positive classifications correctly.

False discovery rate: This is one minus the precision, which tells us how bad are our **positive classifcations**.

$$P(Y=0|\hat{Y}=1) = \frac{FP}{\text{TP + FP}} = \frac{\text{False Positives}}{\text{Positive Classifcations}} \tag{8}$$

Precision Recall Curve: This curve plots the precision agaisnt the recall(sensitivity) as we vary the threshold τ. This is looking at how many correct positive classifications we are making out of everything we classify as positive agaisnt how many correct positive identications we are making out of all the values that are actually positive. The mean precision (averaging over recall/sensitivity values) approximates the area under the precision recall curve.

From all this, we have the following terminologies:

Sensitivity/Recall/True Positive Rate $= \frac{TP}{TP+FN} = \frac{\text{Correct Positives}}{\text{All positives}}$

Specificity $= \frac{TN}{TN+FP} = \frac{\text{Correct Negative}}{\text{All Negative}}$

False Positive Rate $= \frac{FP}{FP+TN} = \frac{\text{False Positives}}{\text{Potential Candidates for FP}}$

False Negative Rate $= \frac{FN}{FN+TP} = \frac{\text{False Negatives}}{\text{Potential Candidates for FN}}$

ROC: Plots sensitivty vs specificity or FPR. The score is based on the **AUC**.

Precision $= \frac{TP}{TP+FP} = \frac{\text{Correct Positives}}{\text{All positive classification}}$

False Discovery Rate $= \frac{FP}{FP+TP} = \frac{\text{Wrong Positives}}{\text{All positive classification}}$

Precision-recall curve: Mean precision approximates area under curve.

10.8 Review Questions for Module 9

What is classification? Mapping an observation to a class c via a classifier (function that maps input to a class).

What is zero-one loss? Loss function where 1 if wrong classification else 0.

What is the Bayes classifiers? Attempting to maximize correct classification through placcing the observations into a class where the probability is the highest. This is:

$$\hat{y} = \arg\max_{c \in Y} P(Y = c | X = \boldsymbol{x})$$

whereby

$$P(Y = \hat{y} | X = \boldsymbol{x}) \geq P(Y = c | X = \boldsymbol{x}) \forall c \in Y$$

What is the misclassification rate? It is the error rate for the *test data* whereby:

$$\bar{\text{Err}}_{\text{test}} = \frac{1}{n} \sum_{i=1}^{n} \frac{I(y_{i,0} \neq \hat{y}_{i,0})}{n}$$

Explain the KNN classifier? This estimates the conditional probability for class c as:

$$\hat{P}(Y = c | X = \boldsymbol{x}) = \frac{1}{K} \sum_{\boldsymbol{x}_i \in N_k} I(y_i = c)$$

whereby the probability is simply the number of neighbours within k of the observation we are looking at.

What is a loss matrix? Also known as a cost-benefit matrix, the loss matrix tells us the loss for whether we make a TP, TN, FN, FP classification.

How do we formulate a decision rule for binary classification? The decision rule is to classify a subject based on whether if the $P(Y = 1 | X = \boldsymbol{x})$ is above or below a threshold τ. If $P(Y = 1 | X = \boldsymbol{x}) > \tau$, then we classify the observation as positive.

What is a confusion matrix? This tells us the actual numbers for the TP, TN, FN, FP classifications.

What are sensitivity, specificity, and precision? Sensitivity = (TP)/(TP+FN). Specificity = (TN)/(TN+FP). Precision = (TP)/(TP+FP).

Why is there a trade-off between sensitivity and specificity? This depends on the threshold value set. We can maximize sensitivity by setting the threshold as $\tau = 0$ so that we always return positive classifcations.

What is the difference between discriminative and generative classifiers? Generative models $P(Y|X)$ via modelling prior $P(Y)$ and class conditional density $P(X|Y)$. This then helps us to model the *joint distribution* $P(X,Y)$ since $P(X,Y) = P(X|Y)P(Y)$ or $P(X,Y) = P(Y|X)P(X)$. We use Bayes' theorem to do so. Discriminative model directly estimates p(y=c|$\boldsymbol{x}$) directly. We aren't concerned with distribution of inputs.

What is the key assumption of the Naive Bayes classifier? Predictors are **conditionally independent** given the class label. p($\boldsymbol{x}|y=c$) = $\prod\limits_{j=1}^{p} p(x_j|y=c)$.

10.9 Logistic Regression

10.9.1 Binary Logistic Regression

Logistic regression is a **discriminative classifier** since attempts to model conditional distribution P(Y=1|X=**x**) directly. Baseline model to use and used in Google (logistic regression, permutation test, and bootstrapping are frequently used there). Here, we assume that the response Y follows the Bernoulli distribution. We therefore specify Y as either 1 with probability P(Y=1|X=x) or 0 with probability 1-P(Y=1|X=x). We want to know how to model the conditional probability P(Y=1|X=x) as a function of the predictors.

Since P(Y=1|X=x) = E(Y|X=**x**) (or the conditional mean), we can specify a **linear probability model** as a model for the conditional mean in order to estimate the conditional probability. Mathematically, this can be written by the usual linear regression formula of Y = $\beta_0 + \sum_{j=1}^{p} \beta_j x_j + \epsilon$. The response is a linear function of the predictors and a random error. We don't use it directly since we need to consider the distribution of the response variable. It is based on restrictive assumptions such as Gaussian distribution. There are quite a few reasons why we want to move beyond this framework though.

1) First of all, there is no guarantee that LPM will generate probabilities between 0 and 1. This is because the regression function is unconstrained. Resultantly, linearity assumption does not hold. There is no guarantee probability will be an actual probability.

2) Bernoulli distribution has a variance p(x)(1-p(x)). Linear probability model now violates assumption of homoskedasticity. It becomes inefficient OLS estimators.

3) LPM does not generalise to multi-classes as well, which is an issue since alot of problems require this.

We can define the **logistic regression model** as:

$$Y|X = \boldsymbol{x} \sim Ber(p(x))$$

whereby the probability

$$p(x) = \frac{exp(\beta_0 + \sum_{j=1}^{p} \beta_j x_j)}{1 + exp(\beta_0 + \sum_{j=1}^{p} \beta_j x_j)}$$

Parameter of this distribution is **p(x)**, which is the probability of it being 1, as given by the above formula. We do a transformation so that the probabilities we get, will be between 0 and 1 by taking the exponential.

$$\text{Logistic Function} = \frac{exp(a)}{1 + exp(a)} = \frac{1}{1 + exp(-a)}$$

This is known as the **logistic function**. The Logistic function takes a number between -$infty$ and $infty$, and outputs a number between 0 and 1. This satisfies our requirement of all probabilities being between 0 and 1.

We can also define the **odds ratio** as:

$$\frac{p(x)}{1 - p(x)}$$

Furthermore, we can show that:

$$\frac{p(x)}{1 - p(x)} = exp(\beta_0 + \sum_{j=1}^{p} \beta_j x_j)$$

This is based on the odds. The odds ratio looks at the ratio of the probability of success over probability of failure. Therefore, a larger odds means that there is a higher probability of success relative to the probability of failure. From this, the unrestricted linear regression looks at the effect on the **log odds ratio**.

$$log(\frac{p(x)}{1 - p(x)}) = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$$

The left hand side is known as the **logit transformation**.

We can estimate the logistic regression by MLE. Look at the slides for derivation of the MLE since this is stuff from QBUS2810. Intuitively though, the likelihood function looks at, given the observations we got, what parameters are the most likely ones in order to give rise to the values we got. Therefore, the parameters which maximizes the likelihood function, means that these parameters are the ones that are most likely for generating the data we currently have. The likelihood function is simply the dot product of the probability mass function for each training case i. Do note that we can take log transformations in order to maximise the likelihood. Negative of the Log-likelihood as a loss function is known as the **cross-entropy loss function** or log loss. Convenient in many settings since minimizing the cross-entropy loss function is the equivalent to maximizing the log-likelihood. After all the derivations, we should end up with the MLE for logistic regression model being:

$$\hat{\beta} = \arg\max_{\beta} L(\beta)$$

whereby L(β) is:

$$L(\beta) = \sum_{i=1}^{N} y_i(\beta_0 + \sum_{j=1}^{p} \beta_j x_i) - log(1 + exp(\beta_0 + \sum_{j=1}^{p} \beta_j x_i))$$

Taking partial derivatives, we have to use numerical optimisation techniques to solve the system of nonlinear equations when maximizing the $\hat{\beta}$ parameters since we get estimation equations that are nonlinear in the coefficients. We can conduct statistical inferences for the logistic regression model using large sample theory for ML estimation or bootstrap method.

We can generate predictions for logistic regression.

$$\hat{p(x)} = \frac{exp(\hat{\beta_0}) + \sum_{j=1}^{p} \hat{\beta}_j x_j}{1 + exp(\hat{\beta_0} + \sum_{j=1}^{p} \hat{\beta}_j x_j)}$$

where $\hat{\beta}$ are the maximum likelihood estimates. We can do a logistic regression plot to see the predicted probabilities. Referring to graph on slide 21, if expenses is 110, then the probability will be over 80%. We can run the logit model in python and see the output. However, recall that we can't make casual inferences since we did not design the dataset in a way to allow for us to determine causality. If we want to compute the probability of success for an individual, we just plug in values into our predicted probability equation from above.

10.9.2 Regularised Logistic Regression

We can further extend such that we can also use the **regularised logistic regression**. Here, we can also incorporate regularisaion methods from the LASSO. Here, we can have an ℓ_1 penalty just like in the LASSO such that if a coefficient in the logistic regression model is below the threshold ℓ_1, we will set it to 0, In effect, we are doing variable selection alongside our logistic regression. We now have the following minimisation problem:

$$\arg\min_{\beta} = -L(\beta) + \lambda \sum_{j=1}^{p} |\beta_j|$$

whereby our loss function (which is the cross-entropy loss function) is the same as before:

$$L(\beta) = \sum_{i=1}^{N} y_i(\beta_0 + \sum_{j=1}^{p} \beta_j x_j) - log(1 + exp(\beta_0 + \sum_{j=1}^{p} \beta_j x_j))$$

We can also use subset selection, dimension reduction with principal components as well for our logistic regression.

10.9.3 Multinomial Logistic Regression

The multinomial logistic regression is a generalisation of the logistic regression to multiple classes. The model specifies that we have:

$$p(y = c|x) = \frac{exp(\beta_{0c} + \beta_c^T x)}{\sum_{c'=1}^{C} exp(\beta_{0c'} + \beta_{c'}^T x}$$

whereby β_c is the vector of coefficients for the class c. We look each classes prediction over all the predictions. From this, we can then further define the **softmax function**:

$$S_c(a_1, ..., a_C) = \frac{exp(a_c)}{\sum_{c'=1}^{C} exp(a_{c'})}$$

The softmax function ensures that the conditional class probabilities are in the (0,1) interval and that the total probability adds up to one. What we do is that we take a_c and exponentiate it and divide it by the sum of exponentials of a_c.

Furthermore, to ensure we don't have redundancies in the parameters, we specify intercepts in base class C = 0, $\beta_{0C} = 0$ and $\boldsymbol{\beta_C} = 0$. We now have a baseline label (similar to how we have a base case for dummy variables). From this, we get:

$$p(y = c|\boldsymbol{x}) = \frac{exp(\beta_{0c} + \boldsymbol{\beta_c^T x})}{1 + \sum\limits_{c=1}^{C-1} exp(\beta_{0c} + \boldsymbol{\beta_c^T x})}$$

for c=1,...,C-1. From this, we have C-1 log odds for all categories. Then, to estimate the probability for the base class C, we have:

$$p(y = C|\boldsymbol{x}) = \frac{1}{1 + \sum\limits_{c=1}^{C-1} exp(\beta_{0c} + \boldsymbol{\beta_c^T x})}$$

10.10 Gaussian Discriminant Analysis

This is a **generative** method (not obvious from the name!). We know that what we need to do is to specify a conditional probability. From this, we can assume that predictors are normally distributed conditional on the class:

$$X|Y = c \sim N(\boldsymbol{\mu_c}, \boldsymbol{\Sigma_c})$$

whereby μ_c is the mean of the predictors in class c and Σ_c is the covariance matrix (since we have multiple predictors, we have a covariance matrix to analyse the variance and covariance with other variables). Σ_c^{-1} refers to the inverse of this covariance matrix. Recall that a Gaussian probability density function *for one variable*, we have that:

$$p(x_j|y=c) = \frac{1}{\sqrt{2\pi\sigma_{jc}}} exp(-\frac{(x_j - \mu_{jc})}{2\sigma_{jc}^2})$$

From this, we can extend it to a multivariate Gaussian by replacing σ_{jc} with Σ_c. From this, the class conditional density is:

$$p(\boldsymbol{x}|y=c) = \frac{1}{\sqrt{(2\pi)^p|\boldsymbol{\Sigma_c}|}} exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu_c})^T \Sigma_c^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c))$$

We then let $\pi_c = P(Y = c)$ be the prior probability for class c. Then using Bayes' theorem, the conditonal probability is:

$$p(y = c|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=c)\pi_c}{\sum\limits_{c=1}^{C} \pi_{c'} p(\boldsymbol{x}|y=c')}$$

10.10.1 Quadratic Discriminant Analysis

Then subbing in the class conditional density from earlier, we get:

$$p(y = c|\boldsymbol{x}) = \frac{\pi_c(\frac{1}{\sqrt{(2\pi)^p|\boldsymbol{\Sigma_c}|}})exp[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu_c})^T \Sigma_c^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c)]}{\sum\limits_{c'=1}^{C} \pi_{c'}(\frac{1}{\sqrt{(2\pi)^p|\boldsymbol{\Sigma_c}|}})exp[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu_c})^T \Sigma_c^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c)]}$$

or

$$p(y = c|\boldsymbol{x}) = \frac{\pi_c(2\pi|\boldsymbol{\Sigma_c}|)^{-\frac{1}{2}} exp[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu_c})^T \Sigma_c^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c)]}{\sum\limits_{c'=1}^{C} \pi_{c'}(2\pi|\boldsymbol{\Sigma_c}|)^{-\frac{1}{2}} exp[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu_c})^T \Sigma_c^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c)]}$$

This is known as **quadratic discriminant analysis** (QDA). If we assume that Σ_c is diagonal (a matrix where non-zero elements are only in the diagonal), then QDA is the same as Naive Bayes. We can now maximize $\pi_k f_k(c)$ over class c. Recall that we only require the numerator as for any class we look at, the denominator will always be the same. Therefore, we are only interested in maximizing the numerator. We can compute the numerator for all potential classes, and whichever one has the highest score, we classify it into such class. Dropping the denominator and taking the logarithm gives us the **discriminant function**:

$$\delta_c(\boldsymbol{x}) = log\pi_c - \frac{1}{2}log|\boldsymbol{\Sigma}_c| - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_c^T)\boldsymbol{\Sigma}_c^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c)$$

We can now rewrite the conditional probability in terms of the discriminant function as:

$$p(y = c|\boldsymbol{x}) = \frac{exp[\delta_c(\boldsymbol{x})]}{\sum\limits_{c'=1}^{C} exp[\delta_{c'}(\boldsymbol{x})]}$$

Disriminant function will give us sufficient information to compute conditional probability based on Bayes theorem. We can now describe any decision rule in terms of the discriminant score. One example is that under the zero-one loss, the decision rule is:

$$\hat{Y}(\boldsymbol{x}) = \arg\max_c \delta_c(\boldsymbol{x})$$

whereby we decide to classify observation **x** into class c. THe log-odds between two classes is:

$$log\frac{P(Y = c|X = \boldsymbol{x})}{P(Y = c'|X = \boldsymbol{x})} = \delta_c(\boldsymbol{x}) - \delta_{c'}(\boldsymbol{x})$$

Decision boundary is an equation that is described by the discriminant function. This formula will be quadratic in terms of predictors since we have the transpose of x multiplied by the x. Therefore, we get power terms of features and also interaction between features. Resultantly, **QDA** will give us nonlinear decision boundaries. More specifically, we get quadratic decision boundaries when classifying.

From this, we want actually compute stuff. First, we let $N_c = \sum\limits_{i=1}^{N} I(y_i = c)$. From this, we can estimate the model parameters as:

$$\hat{\pi}_c = \frac{N_c}{N}$$

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{N}\sum_{i:y_i=c} \boldsymbol{x}_i$$

$$\hat{\boldsymbol{\Sigma}}_c = \frac{1}{N_c} \sum_{i:y_i=c} (x_i - \hat{\mu}_c)(x_i - \hat{\mu}_c)^T$$

for c = 1,...,C. $\hat{\pi}_c$ is the frequency of class over total sample. $\hat{\boldsymbol{\mu}}_c$ is the average of the values for that class. $\hat{\boldsymbol{\Sigma}}_c$ is multiplying out the matrices differenced by the mean to get covariance matrix. We then compute the discriminant score using:

$$\delta_c(\boldsymbol{x}) = log\pi_c - \frac{1}{2}log|\boldsymbol{\Sigma}_c| - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_c^T)\boldsymbol{\Sigma}_c^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c)$$

and whichever class has the highest discriminant score, we classify observation **x** into that class. Note that we can't use QDA if there are dummy variables present since we assume Gaussian distributions.

10.10.2 Linear Discriminant Analysis

In LDA, we assume that the classes have a comman covariance matrix, $\boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}$ for c=1,...,C. This assumptions gives us the **linear discriminant function**:

$$\delta_c(\boldsymbol{x}) = log(\pi_c) + \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c - \frac{1}{2}\boldsymbol{\mu}_c^T\boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c$$

which now gives us *linear decision boundaries* (so straight lines between classes). It is a similar process in estimating LDA except that we now compute the *pooled covariance matrix*:

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N} \sum_{c=1,..,C} \sum_{i:y_i=c} (x_i - \hat{\mu_c})(x_i - \hat{\mu}_c)^T$$

for c=1,...,C.

We can additionally model LDA in a similar manner to the logistic regression. We can rewrite the LDA function from:

$$\delta_c(\boldsymbol{x}) = log(\pi_c) + \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c - \frac{1}{2}\boldsymbol{\mu}_c^T\boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c$$

into:

$$\delta_c(\boldsymbol{x}) = \alpha_{0c} + \boldsymbol{\alpha}_c^T\boldsymbol{x}$$

whereby:

$$\alpha_{0c} = log(\pi_c) - \frac{1}{2}\boldsymbol{\mu}_c^T\boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c$$

$$\boldsymbol{\alpha}_c = \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c$$

from this, the conditional probability under the LDA is:

$$p(y = c|\boldsymbol{x}) = \frac{exp(\delta_c(\boldsymbol{x}))}{\sum\limits_{c'=1}^{C} exp[\delta_{c'}(\boldsymbol{x})]}$$

$$p(y = c|\boldsymbol{x}) = \frac{exp(\alpha_{0c} + \boldsymbol{\alpha}_c^T\boldsymbol{x})}{\sum\limits_{c'=1}^{C} exp[\alpha_{0c} + \boldsymbol{\alpha}_c^T\boldsymbol{x}]}$$

This gives us the exact same specification for the conditional probability like the logistic regression. However, they differ in the manner in which they estimate the coefficients. Logistic model only specifies the distribution of $P(Y = c|X = \boldsymbol{x})$ with no assumptions regarding distribution of predictors whilst the LDA specifies the joint distribution of X and class. It is also a mixture of Gaussians. For MLE, we maximize the likelihood of $P(Y = c|X = x)$ for logistic regression whilst for LDA, we maximize the joint likelihood of $P(X = x, Y = c)$. LDA estimates the parameters more efficiently since it uses more information about the data ONLY if these assumptions are correct.

10.10.3 Regularised Gaussian Discriminant Analysis

From this, we can comprimise between QDA and LDA by computing the covariance matrix in a different manner (refer to slide 41 of module 10). We compute the covariance matrix based on a linear combination of assuming an identical matrix or not. We choose the level of α based on cross validation. We can also shrink the covariance matrices $\boldsymbol{\Sigma}$ towards diagonal matrices by achievieving compromises between QDA/LDA/Naive Bayes. Diagnonal matrices are a case of Naive Bayes since in that scenario, we assume predictors are independent from one another.

Further extensions are on slide 42 whereby we can use multivariate t-distributions in order to be robust agaisnt outliers. Furthermore, the **mixture discriminant analysis** assumes that the class conditional distributions are mixtures of distributions and can lead to highly flexible models, allowing for approximation of complex decision boundaries.

10.11 Comparison of classification methods

If the assumptions of Gaussian discriminant analysis are correct, the models will require less training than logistic regression to achieve a decent level of performance (since the assumptions made improves estimation efficiency). However, if our assumptions are wrong, the logistic regression will do better.

Generative models model the distribution of the individual classes themselves. They are easy to fit and do not need to be retrained if we use new classes. They can handle missing features (since has the distribution of random variables for features), and be used in semi-supervised learning (partially labelled data).

Discriminative models learns the hard/soft **boundaries** between classes. These can handle constructed predictors and generally have better calibrated probability estimates.

10.12 Review Questions for Module 10

What is the logistic regression model? Logistic regression model assumes that $Y|X = x \sim$ Bernoulli distribution, whereby the probability p(x) is given by the logistic function transformation of the linear probability model.

What is Gaussian Discriminant analysis? Gaussian discriminant analysis is a generative model whereby we assume that the predictors are normally distributed conditional on the class $X|Y = c \sim N(\mu_c, \Sigma_c)$. From this, we can get the class conditional density based on the Gaussian formula. We then compute discriminant functions/scores in order to classify an observation into a class.

$$\delta_c(\boldsymbol{x}) = log\pi_c - \frac{1}{2}log|\boldsymbol{\Sigma}_c| - \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_c^T)\boldsymbol{\Sigma}_c^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_c)$$

What is the difference between LDA and QDA? LDA assumes that the classes have a common variance: $\Sigma_c = \Sigma$ for all classes. We get a linear discriminant function.

$$\delta_c(\boldsymbol{x}) = log(\pi_c) + \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_c - \frac{1}{2}\boldsymbol{\mu}_c^T\boldsymbol{\Sigma}_c^{-1}\boldsymbol{\mu}_c$$

Identify whether each classification method that we studied so far is generative or discriminative? Gaussian Discriminant Analysis - Generative

Naive Bayes - Generative

Logistic Regression/Multinomial - Discriminative

KNN - Discriminative

What are the relative advantages of generative vs discriminative classifers?

Generative: Easy to fit, no need to retrain for new classes, can easily handle missing features, can be used in semi-supervised learning.

Discriminative: Can easily handle constructed predictors and generally have better calibrated probability estimates.

In what situations do each of the classification methods that we have studied so far tend to be most useful?

Naive Bayes good for large feature space.

Gaussian Discriminant is good if our assumptions of normality and covariances are correct.

Logistic regressions are more robust.

KNN is good for many instances but low dimensionality and with complex decision boundaries.

11 Forecasting

There isn't as much detailed notes in the time series section as there is a good resource at: https://www.otexts.org/fpp/7/1

First we define a **time series** as a set of observations $y_1, ..., y_t$ *ordered* in time. From this, we now have a nxt matrix (with each row being a time series). In this unit we only look at rows of time series.

We are only interested in quantitative (data based) forecasting for this unit. Here, predictions are based on the history of the time series to predict future prices. We don't use any information from predictors, as we are only concerned with the time dimension and therefore only use past values to predict future values.

Looking at the graph on slide 4 of module 11, we note that there is seasonality, size of oscillation is increasing, and upward drift. From this, we can be reasonably confident in number of customers coming in each season

Definition 11.1. A **forecast** is a prediction about future events and conditions given all current information (including historical data). The act of making such predictions is what is known as **forecasting**.

11.1 Problem Definition

We have to predict y at the point t+h, where t is the current period. Furthermore, we can use information prior to t as well in order to make a forecast for the future. Do note that it is a lower case y since we have realised values up to y_t, including $y_1, ..., y_{t-1}$. From this, we want to predict from Y_{t+1} onwards to Y_{t+h}, whereby h is the **forecast horizon**. If we wanted to, we can also include predictors $\boldsymbol{x_1}, ..., \boldsymbol{x_p}$, leading to a **dynamic regression problem**.

From this, we need some decision theory in order to structure the problem more formally. First, we let a **point forecast** be denoted as $\hat{Y}_t$, which is a function of f($Y_{1:t}$) (here we have a capital letter since it is decision theory so we are talking about random variables.). $Y_{1:t}$ denotes a time series. We want to learn a function f(.) in order to make good forecasts. $\hat{Y}_t$ is a function of the predictors as we defined it. As in the cross-sectional case, we can assume a squared error loss function:

$$L(Y_{t+h}, f(Y_{1:t}) = (Y_{t+h} - f(Y_{1:t}))^2)$$

whereby it is the square of the actual value less our prediction. $Y_{1:t}$ is the *slice notation* which is a compact way to write $Y_1, ..., Y_t$.

Recall just like in the case of cross sectional data, the optimal point forecast under the squared error loss in the **conditional expectation**.

$$f(Y_{1:t}) = E(Y_{t+h}|Y_{1:t})$$

In other words, the optimal forecast is the expected value of Y_{t+h} given the values of $Y_{1:t}$ (kinda obvious but we just needed to specify it formally). We wish to approximate the conditonal expectation of Y_{t+h} given the historical data for multiple values of h. We want to minimize expected value of the loss function so the optimal prediction is the expectation of random variable conditioned on variables realised

So we have: $E(Y_{t+h}|Y_{1:t} = y_1, \dots, y_t)$

We are also interested in **density forecasts** which contains the uncertainty quantitification. We want to generate a **point forecast** but we also need the density forecast. So instead of giving just a number, we want to know what is the probability that the future value will be this value x. Mathematically, a density forecast is:

$$\hat{p}(Y_{t+h}|y_1, ..., y_t)$$

which aims to estimate

$$P(Y_{t+h}|y_1, ..., y_t)$$

More formally, we can define it as:
Definition 11.2. A **density forecast** of the realisation of a random variable at some future time is an estimate of the *probability distribution* of the possible future values of that variable. In other words, what are the probabilities of the random variable taking on certain values.

From the density forecast, we can derive the *interval forecast.*
Definition 11.3. An **interval forecast** consists of an upper and lower limit between which a future value is expected to lie with a prescribed proability. It is an interval $(\hat{y}_{t+h,L}, \hat{y}_{t+h,U})$ such that $\hat{P}(\hat{y}_{t+h,L} < Y_{t+h} < y_{t+h,U}) = 1 - \alpha$.

Density forecast estimates the probability distribution whilst **interval forecast** gives an interval for which a future value may lie. A density forecast is estimating the whole probability distribution of a variable in the future conditioned on the observed values of the time series. Interval forecast is derived from density forecast but gives us more specific information. We want to have an interval probability that it is within the interval 1-α. So if $\alpha = 0.1$, then we have an interval forecast of 0.9 or 90% probability. This is different to confidence interval since prediction interval depends upon on the density forecast and values whilst confidence interval depends upon estimators/paramters. Recall confidence interval does not talk about probability (we talk about 90% confidence). From this, confidence $\neq$ probability which then $\rightarrow$ interval forecast $\neq$ confidence interval.

An interval forecast is an interval associated with a random variable yet to be observed, with a specified probability of the random variable lying within the interval. For example, I might give an 80% interval for the forecast of GDP in 2014. The actual GDP in 2014 should lie within the interval with probability 0.8.

A confidence interval is an interval associated with a parameter and is a frequentist concept. The parameter is assumed to be non-random but unknown, and the confidence interval is computed from data. Because the data are random, the interval is random. A 95% confidence interval will contain the true parameter with probability 0.95. That is, with a large number of repeated samples, 95% of the intervals would contain the true parameter.

An interval forecast is also known as a prediction interval.

11.1.1 Fan Chart

For consecutive forecast horizons, we can construct prediction/interval forecasts using different probability levels. What this means is that we construct prediction intervals/interval forecasts for different probability levels and then plot it using different shades. The intervals typically get wider with the horizon, which represents the uncertainty about future values. White area in the graph is observed series. Line is the present. Then the shaded area is the future. Lighter colour means a higher probability for the interval (which makes sense since there is a wider range of values for future value to take). The darker the colour, the narrower the interval and a lower probability level. Ultimate goal is to find best model to generate interval forecasts.

11.2 Time series Pattern

We can think of time series as a regression problem but set up in a different manner.

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \epsilon$$

This is an AR(2) model which uses 2 lagged terms of Y_t. However, we assume that there are patterns that are present which we would want to pick up on directly rather than indirectly like the AR model. We can say that Y_t is a function of the trend, seasonality, cyclicality, and error component. We have models to estimate each of the components in the model and then use it to forecast time series in the future. From this, we can interpret a time series as:

$$Y_t = f(T_t, S_t, C_t, E_t)$$

Trend T_t is the trend component. Here, this is the systematic long term increase/decrease in the series.

Seasonal S_t is the seasonal component. This is a systematic chance in the mean of the series due to seasonal factors such as the month, day of the week etc.

Cycle: C_t is the cyclical component. These are not fixed periods. Here, these patterns exists when there are medium/long run fluctuations in the time series that are not of a fixed period. One example can be a business cycle on page 18. Some periods we have recessions and some we have growth. These periods of expansions and recessions are not of a fixed term.

Irregular/Error: E_t are short term fluctuations and noise.

We then estimate a model that takes into account every component. So if the time series has a trend but not a seasonal/cyclical component and therefore we omit the trend to get:

$$Y_t = (T_t, I_t)$$

Furthermore, we have 2 different models to use where one is additive and the other is multiplicative.

Additive: Time series is sum of individual component. $Y_t = T_t + S_t + E_t$

Multiplicative: Time series is multiplicative. $Y_t = T_t \cdot S_t \cdot E_t$. However, we can take the log transformation to make it easier to work with: $logY_t = logT_t + logS_t + logE_t$.

We need to choose which model to use. We don't add cycles in general models since it is very challenging to do so and therefore we either ignore the cycles as unpredictable or we say say that the trend component T_t is absorbing the cyclical component.

If seasonal component is not proportional (related to a trend), then we need to use an additive model to account for the seasonality component. This is because if we believe that that the seasonality is unique and distinct from the trend, then using an additive model will capture this whilst a multiplicative will contain the seasonality alongisde the trend. For the seasonal pattern, we should take the average of each component in seasonal variation. So we get average for each month and see how it changes for each type of data.

General use of **time series decomposition** is used for interpretation whereby we can seasonally adjust data. If there are seasonal trends but we are interested in long run values, we can remove the seasonality in order to get a true trend. Addtionally, we can have seasonally adjusted values. Decompositions are really useful in visualising patterns.

11.3 Forecasting Methods

Random Walk is forecasting the series using the value of the last available observation

$$\hat{y}_{t+h} = y_t$$

From this, we can extend it to a **seasonal random walk** whereby we forecast the series with the value of the last available observation *in the same season.*

$$\hat{y}_{t+h} = y_{t+h-m} \quad (if h \leq m)$$

whereby m denotes the seasonal period. If m = 12, we are looking at monthly data. If m = 4, we are looking at quarterly data. From this, the general formula is

$$\hat{y}_{t+h} = y_{t+h-km} \quad k = \frac{h-1}{m+1}$$

Drift method forecasts the series as the sum of the most recent values (similar to the Naive method) and the average change over time.

$$\hat{y}_{t+1} = y_t + \sum_{i=2}^{t} \frac{y_i - y_{i-1}}{t-1}$$

for one period in the future. If we extend it further to h periods:

$$\hat{y}_{t+h} = y_t + h \cdot \sum_{i=2}^{t} \frac{y_i - y_{i-1}}{t-1}$$

In essence, we are taking the last observation and adding on to it, the average change in values over the **entire** time frame.

11.4 Model Diagnostic

We want to check for autocorrelation to check if the model is capturing patterns adequately. Furthermore, we want to do density forecasting so we need to check if the assumptions we are building for these forecast intervals are appropriate for the models we are estimating.

Autocorrelation of time series process is to see the correlation with itself.

$$\rho_k = \frac{E[(Y_t - \mu)(Y_{t-k} - \mu)]}{\sigma^2} = Corr(Y_t, Y_{t-k})$$

whereby k is the lag, $\mu\sigma^2$ are the mean and variance of the time series.

We can then calculate the **sample autocorrelation**.

$$\tau_k = \frac{\sum_{t=1}^{T-k} (y_{t-k} - \bar{y})(y_t - \bar{y})}{\sum_{t=1}^{T} (y - \bar{y})^2}$$

Corr$(Y_t, Y_{t-k}) = \frac{Cov(Y_t, Y_{t-k})}{\sqrt{Y_t}\sqrt{Y_{t-k}}}$

We make the assumption that the variance of the time series is constant. From all this, the **autocorrelation function (ACF)** plot tells us the autocorrelation for a range of lags.

White Noise Process Is a sequence of iid random variables $\sim$ $(0,\sigma^2)$. We assume that the error terms are white noises.

If the model is well specified and captures the time series patterns well, the residuals should behave like a white noise process.

$$Y_t = f(Y_{1:t}) + \epsilon_t$$
$$\epsilon_t = f(Y_{1:t}) - Y_t$$

so if the error term is autocorrelated, then our model we set up is missing time series pattern and therefore should adjust the model accordingly to get an uncorrelated series of residuals.

3 diagnostic plots we should use:

Residual Plot: The presence of patterns in time series of residuals may suggest assumption violations and need for alternative models.

Residual ACF Plot: This tells us the autocorrelation for different lag lengths. If substantial autocorrelation, we have an issue. This is for us to get accurate point forecast. Well specified models should lead to small and insignificant sample autocorrelation, which is consistent with a white noise process.

Residual distribution plots: This is useful for density forecast. We can use normal assumption for density forecasting but if not, we need to use bootstrapping and other techniques. This allows us to check the appropriate assumptions for interval forecasting. We can check these via histograms, KDE, Q-Q plots.

LOOK AT TUTORIAL ON HOW TO DO ALL OF THESE

Time series analysis steps

1) Problem definition

2) EDA

3) Modelling

4) Model Diagnostic

5) Model Evaluation

6) Forecast

11.5 Model Validation

Recall that model validation looks at whether models we are using are suitable for our needs. We score a given model with a measure of how well it serves its purpose. On the other hand, model selection answers which model is most useful for our needs.

We incorporate model validation in forecasting by setting aside a validation sample to estimate and compare performances of different models (normally the last 20-50% of the data). We can refer to the training sample as the *in sample data* and the validation set as the *out of sample data.* Here in time series, there is no concept of test set for time series since by definition, they won't be independent of the training set. So instead we have a validation data and this is called validation for 2 reasons:

1) We use this for model selection in the end since we then use it to forecast.

2) We are still evaluating the model.

Recall that model evaluation refers to checking the performance of our final predictions on the test dataset. Model validation generally refers to the hyperparameter tuning of the model on the cross validation set in order to get the best results.

We validate forecasts by following a real time approach. What that means is that at every period t, we use all available data at present toe estimate the model and predict the future value of the series. So when forecasting, we actually combine the training data and the validation set in order to forecast the next value. We use all the previous observations to estimate the model.

So for validation period t+1, we use the training sample $y_{1:t}$ and forecast it via $f^1(y_{1:t})$. Then for t+2, we use the training sample and append t+1 observation $y_{1:t+1}$ and forecast it via $f^2(y_{1:t+1})$.

The test data is considered the random values in the future in which we forecast for whilst the validation set in the data we splitted up in half.

So the steps are:

1) Starting at t=n, use observations at time 1,2,...,t to estimate the forecasting model. Then use the estimated model to forecast the observation at time t+1.

2) From this, we repeat step 1 for t=n+1,...,T-1.

3) We then compute the forecast accuracy measures based on the prediction errors $y_{n+1} - \hat{y}_{n+1}, ..., y_T - \hat{y}_T$.

We can use either an expanding window or rolling window for this validation period.

Expanding Window At each step, add the latest observation to the estimation sample.

Rolling Window At each step, use only the most recent n observations for estimation. This implicitly assumes that the dynamics of the series has a changing nature and therefore observations in the past are less relevant in forecasting future values.

From all this, we can measure forecast accuracy. We use the squared error loss and compute out-of-sample MSE to measure forecast accuracy. We can also use other measures such as **percentage errors** and **scaled errors** as well for business forecasting.

Percentage Errors are scale independent as denoted by the formula percentage error = $p_t = 100 \cdot \frac{y_t - \hat{y}_t}{y_t}$. From this, we can then compute the **mean absolute percentage error** by: $MAPE = mean(|p_t|)$. However, there is the disadvantage that the metric breaks down to ∞ if $y_t =$ for any t periods or that values will be extreme if $y_t \approx 0$. However, if 0 is meaningful, then percentage errors are valid.

Scaled Errors Here, we can scale the errors based on the training MAE/MSE from a benchmark model. For non-seasonal time series, we can use naive forecasts to define scaled errors q_t. The formula for so is:

$$q_t = \frac{y_t - \hat{y}_t}{\frac{1}{T-1}\sum\limits_{i=2}^{T} |y_i - y_{i-1}|}$$

Hence, we get the error $y_t - \hat{y}_t$ and then scale it via the naive forecast. Since both numerator and denominator involves values on the scale of the original data, q_t is independent of the scale of the data again.

Mean Absolute Error is then defined to be MASE = mean($|q_t|$). This scaled error < 1 if the errors are less than the errors of a naive/random walk method evaluated on the training data. This is because the residuals from our model is less than the residuals than the Niave method, thereby saying that our model performs better.

11.6 Random Walk

Random walk suggests things are unpredictable. For any forecast horizon, we use the last observed value. We make the assumptions of white nosie error terms in order to generate the *point and interval forecasts*. (But recall that interval forecasts requires the density forecasts which we don't have yet). We have the model of:

$$Y_t = Y_{t-1} + \epsilon_t$$

whereby so far, we just assume that the $\epsilon_t \sim i.i.d(0, \sigma_2)$.

Since $Y_t = Y_{t-1} + \epsilon_t$, we can use back substitution to show that:

$$Y_{t+h} = Y_t + \epsilon_{t+1} + ... + \epsilon_{t+h}$$

$$Y_{t+h} = Y_t + \sum_{i=1}^{h} \epsilon_{t+i}$$

From this, when obtaining the point forecast for any horizon as

$$\hat{y}_{t+h} = E(Y_{t+h}|y_{1:t})$$

$$= E(Y_t + \sum_{i=1}^{h} \epsilon_{t+1}|y_{1:t})$$

$$= y_t$$

The conditional variance is then:

$$Var(Y_{t+h}|y_{1:t}) = Var(y_t + \sum_{i=1}^{h} \epsilon_{t+i}|y_{1:t})$$

$$= h\sigma^2$$

The conditional variance increases as the time horizon increases. So we need to also make the assumption of normal distribution of the error terms to then to be able to make density forecasts (in order for us to then compute interval forecasts). We assume Gaussian errors so that $\epsilon_t \sim N(0, \sigma^2)$ to then get:

$$Y_{t+h}|y_{1:t} \sim N(y_t, h\sigma^2)$$

Forecast interval has a similar style to be computed like the confidence interval.

$$y_t \pm z_{\frac{\alpha}{2}} x \times \sqrt{h\hat{\sigma}^2}$$

$z_{\frac{\alpha}{2}}$ is the appropriate critical value from the normal distribution and that:

$$\hat{\sigma}^2 = \frac{\sum_{t=2}^{T}(y_t - y_{t-1})^2}{T-1}$$

This is also known as the **plug-in method** since we replace the unknown σ^2 with an estimate. However, this ignores to parameter uncertainty and leads to prediction intervals that are too narrow.

Normally, distribution of errors are not Gaussian and therefore our forecast interval is flawed. So there is no central limit theorem for this model. If error is not Gaussian, then no way for prediction interval to be Gaussian. From this, we use the Bootstrap algorithm. We use the empirical distibution of the residuals to approximate the error distribution. We use the quantiles of the bootstrap rather than the error distribution. Read up more on slide 47. Even though we don't require Gaussian errors, we do require i.i.d errors for the Bootstrap algorithm. The residual may be too small for complex models that are subject to large optimism. The prediction interval will be too narrow in this case.

11.7 Review Questions for Module 11

What is point and interval forecasting? Point forecast is just a single value whilst an interval provides a $1-\alpha$ interval of which that point forecast may fall.

What are the four time series components? Trend, seasonality, cycles, and error terms.

Which diagnostics do we use for univariate time series models and why? Residual plot: check assumptions of model are okay. May need to use other models.

ACF: See if residuals are white noise process.

Residual distribution: make sure interval forecasts are okay to see if residuals are Gaussian.

How do we conduct model validation for forecasting? We do real time forecasting. We can use expanding or rolling window, whereby we forecast with 1-n observations, include the next observation, predict again, and so on. We can then compute MSE, scaled, and percentage errors (MASE and MAPE).

How do we compute forecasts and prediction intervals for the random walk model? Assume Gaussian error terms that are white noise processes. Then we go:

$$y_t \pm z_\alpha \times \sqrt{(h\hat{\sigma}^2)}$$

whereby $\hat{\sigma}$ is a forecast based on variance of observations. IF not, then use bootstrapping techniques.

11.8 Exponential Smoothing

Here, we forecast future observations as an average of past observations. The more recent the data, the higher weight we give it in our forecast This allows for things like seasonal pattern or trends to vary from year to year. These models are useful for when time series components are changing over time. Goes all the way back to y_1 and ℓ_0. We want to construct models whereby the forecasts are exponential weightings of previous observations:

$$\hat{y}_{T+1|T} = \alpha y_t + \alpha(1-\alpha)y_{T-1} + \alpha(1-\alpha)^2 y)y_{T-2} + ...$$

11.8.1 Simple Exponential Smoothing/Exponentially Weighted Moving Average

We can represent our forecasts in component form. The intuition behind this models is that we are assuming there are no trends or seasonal patterns in the data, and therefore we just simply take averages of previous observations. This is recursive in nature. We use auxillary equations to set up our model. **Smoothing equation** is a moving average of the time series and estimating the level. We have the forecasting rule of:

$$\hat{y}_{t+1} = \ell_t \quad \text{(forecast equation)}$$

$$\text{whereby } \ell_t = \alpha y_t + (1-\alpha)\ell_{t-1} \quad \text{(smoothing equation)}$$

whereby for an initial value ℓ_0 and $0 \leq \alpha \leq 1$. This forecast equation ℓ_t is a weighted average whereby one of the terms in the equation is the most recent observation and then we give weight to period ℓ_{t-1}. α is constant throughout the equations. The smoothing equation for the level, gives the estimated level of the series at each period. We need to initialise it such that at level 0 ℓ_0, we set the value to be y_1.

$$\ell_1 = \alpha y_1 + (1-\alpha)\ell_0$$
$$\ell_2 = \alpha y_2 + (1-\alpha)\ell_1$$
$$\ell_2 = \alpha y_2 + (1-\alpha)\alpha y_1 + (1-\alpha)^2\ell_0$$

and repeat so on...

The final term in the equation for ℓ_0 are a residual term which is based on the initialisation of ℓ_0.

From this, it follows that:

$$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}$$
$$\ell_t = \alpha y_t + (1-\alpha)\alpha y_{t-1} + (1-\alpha)^2\alpha y_{t-2} + ... + (1-\alpha)^{t-1}\alpha y_1 + (1-\alpha)^t\ell_0$$

(notice how there is no α term in front of ℓ_0). $(1-\alpha)^t$ converges to 0 as $t \to \infty$. Weights decrease exponentially which is why its called a *exponentially weighted smoothing average*. A closed form would be:

$$\ell_t = \alpha y_t + \sum_{i=1}^{t} (1-\alpha)^i \alpha y_{t-i} + (1-\alpha)^t \ell_0$$

Useful for forecasting time series without seasonal components but with changing levels. We can interpret the model in terms of α. Higher values of α means a higher weighting to recent values. This makes the model more adaptive. Lower α means larger weights for past observations and therefore smoother forecasts. We set $\ell_0 = y_1$ since t=1 is the start of the series. (NOT 0). In general, not worth to estimate ℓ_0 as a parameter.

Disadvantage of this method is that it is not responsive whereby it takes a while for the model to respond to dramatic changes so we can raise α to be higher in order to be able to capture dramatic changes quickly. Average is less smooth.

We can use *least squares method* in order choose the level of $\hat{\alpha}$. We now have a minimisation problem whereby we have nonlinear equations to solve for via empirical risk minimisation.

$$\hat{\alpha} = \arg\min_{\alpha} \sum_{t=1}^{N} (y_t - \ell_{t-1})^2$$

Each ℓ_t is a nonlinear function of α so we need to use numerical optimisation methods to obtain a solution.

11.8.2 Statistical Modeling of Exponential Smoothing

This only gives us a forecasting algorithm from previous sections. How can we forecast for multiple time horizons rather than just 1 period ahead and how do we include forecast intervals as well. From this, we need a more formal statistical model. First, going back to our original rule:

$$\hat{y}_{t+1} = \ell_t$$

$$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}$$

So now, we convert it into a statistical model. We assume random errors ϵ_t are i.i.d with constant variance σ^2. We have the case that:

$$Y_t = \ell_{t-1} + \epsilon_t$$

$$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}$$

whereby it is now for a random variable Y with an error term and ℓ is fore the same period. In forecasting, we are interested in computing the point forecasts and interval forecasts for multiple forecasting horizons h.

Point forecasts if you recall are defined as $\mathrm{E}(Y_{t+h}|y_{1:t})$. We also get the distribution of $P(y_{t+h}|y_{1:h})$ which we can get the variance $Var(Y_{t+h}|y_{1:t})$. The first step we need to do is *derive the error correction form* in order to allow for us to include uncertainty in the model. We obtain the error correction form as:

$$\ell_t = \alpha y_t + (1-\alpha)\ell_{t-1}$$

$$\ell_t = \ell_{t-1} + \alpha(Y_t - \ell_{t-1})$$

and since $Y_t = \ell_{t-1} + \epsilon$

$$\ell_t = \ell_{t-1} + \alpha\epsilon_t$$

From this, we can rewrite the model for Y_{t+1} as:

$$Y_{t+1} = \ell_t + \epsilon_{t+1}$$

and then subbing in:$\ell_t = \ell_{t-1} + \alpha\epsilon_t$

$$Y_{t+1} = \ell_{t-1} + \alpha\epsilon_t + \epsilon_{t+1}$$

We can generalise this by using $\ell_t = \ell_{t-1} + \alpha\epsilon_t$ to get that:

$$\ell_{t+1} = \ell_t + \alpha\epsilon_{t+1}$$

$$\ell_{t+2} = \ell_{t+1} + \alpha\epsilon_{t+2}$$

$$= \ell_t + \alpha\epsilon_{t+1} + \alpha\epsilon_{t+2}$$

and that:

$$\ell_{t+3} = \ell_t + \alpha\epsilon_{t+1} + \alpha\epsilon_{t+2} + \alpha\epsilon_{t+3}$$

which then generalises to:

$$\ell_{t+h} = \ell_t + \sum_{i=1}^{h} \alpha\epsilon_{t+i}$$

From this, this allows us to model:

$$Y_{t+1} = \ell_t + \epsilon_{t+1}$$

$$Y_{t+2} = \ell_{t+1} + \epsilon_{t+2}$$

$$= \ell_t + \alpha\epsilon_{t+1} + \epsilon_{t+2}$$

which then generalises to:

$$Y_{t+h} = \ell_{t+h-1} + \epsilon_{t+h}$$

giving us a general formula of:

$$Y_{t+h} = \ell_t + \sum_{i=1}^{h-1} \alpha\epsilon_{t+i} + \epsilon_{t+h}$$

(since ℓ is for period t-1, we have h-1 $\alpha\epsilon$ terms whilst we then have a final ϵ term on its own for period h).

$$Var(\ell_t) = \alpha^2 Var(y_t) + (1-\alpha)Var(\ell_{t-1}) + \alpha(1-\alpha)Cov(y_t, \ell + t - 1)$$

Whereby it is very hard to model the covariance term here. The second step we need to do is *derive the constant plus noise representation of future observation.* This rewrites random variables as the sum of 2 terms whereby the first term is a constant and the second term is a sum of independent random errors, which will be easier to work with due to linearity of expectation (expectation of sum is sum of expectation) and the variance of the sum, will be the sum of the variance due to the errors being i.i.d. Finally, we can then *calculate point forecasts and variances.*

With this, the levels are now formualted in terms of an error term ϵ_t rather than other levels ℓ. Then from this, we can get a new error correction form, which now expresses the equation in terms of a level ℓ_t and the summation of a random variablce ϵ_t.

$$Y_{t+1} = \ell_t + \epsilon_{t+1}$$

$$E(Y_{t+1}|y_{1:t}) = E(\ell_t + \epsilon_{t+1}|y_{1:t})$$
$$= \ell_t + E(\epsilon_{t+1}|y_{1:t}) = \ell_t$$

and in addition to that, the forecasting variance will be:

$$Var(Y_{t+1}|y_{1:t}) = Var(\ell_t + \epsilon_{t+1}|y_{1:t})$$

$$Var(\ell_t) + Var(\epsilon_{t+1}) + 2Cov(\ell_t, \epsilon_{t+1})$$
$$= 0 + \sigma^2 + 0$$

When we forecast for Y_{t+2}, we have the case whereby ℓ_{t+1} is now a random variable and depends on the expectation of this. This makes computing the expected forecasted value quite challenging now. The previous example we did, ℓ_t was known which wasn't an issue. Additionally, when computing the variance for 2 points into the future, we have the same issue whereby we don't know what the variance of the random future level ℓ_{t+1}. From this, we need to be able to set up a way to get the expectation and variance of this unknown variable ℓ_{t+h}. We now have a linear combination of a constant plus noise representation of future observations:

$$Y_{t+h} = \ell_t + \sum_{i=1}^{h-1} \alpha\epsilon_{t+i} + \epsilon_{t+h}$$

From the linearity of expectations, the point forecast for any horizon h is:

$$\hat{y}_{t+h} = E(Y_{t+h}|y_{1:t})$$

$$= E(\ell_t + \sum_{i=1}^{h-1} \alpha\epsilon_{t+i} + \epsilon_{t+h}|y_{1:t})$$

$$\hat{y}_{t+h} = \ell_t$$

since expectation of error term is 0. We therefore now have a point forecast for $\hat{y}_{t+h}$. For the variance:

$$Var(Y_{t+h}|y_{1:t}) = \sigma^2(1 + (h-1)\alpha^2)$$

whereby the variance increases as the forecast horizon h increases. Simply put, we have an error term for the current period + an error times scaled by alpha for previous periods.

From this, we get the point forecast and getting the variance will allow us to be able to compute the distribution of the forecast. To get the distribution for forecast, we need to make more assumptions that error terms have a Gaussian distribution. Due to the fact that a random variables composed of a linear combination of a Gaussian distribution, it itself a Gaussian distribution. This means the future value random variable is of a Gaussian distribution. From this, we can now get an interval forecast by estimating $\hat{\alpha}$ and $\hat{\sigma}$ via maximum likelihood. From assuming Gaussian random errors, we get that the point forecast is:

$$Y_{t+h}|y_{1:t} \sim N(\ell_t, \sigma^2[1 + (h-1)\alpha^2])$$

where we set $\chi = \sigma^2[1 + (h-1)\alpha^2$. To compute interval forecast, we estimate values of α and σ^2 to get that:

$$\hat{\ell}_t \pm z_{crit} \times \sqrt{\chi}$$

whereby $\hat{\sigma}^2$ is just the variance of the residuals. If errors aren't normal, use bootstrap method, similar to the random walk but with additional steps. We can also make other distributional assumptions.

11.8.3 Trend Corrected/Holt Exponential Smoothing

This allows for a time varying trend:

$$\hat{y}_{t+1} = \ell_t + b_t \quad \text{forecast equation}$$

whereby:

$$\ell_t = \alpha(y_t) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \quad \text{smoothing equation}$$

$$b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1} \quad \text{trend equation}$$

whereby we initialise ℓ_0 and b_0 with both $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$.

b_t is a local slope which changes the slope. This is a trend which we can now include in the model and allow for this trend to change over time.

We need to derive an error correction form for the trend equation and remove ℓ_t from the equation. Alot of stuff is just a repeat of the SES model, so I shall omit that and you can read up on it in the lecture slides for module 12.

Note that when we go forward in time, we have this new term telling us the trend slope. This allows for b_t to change over time but when we forecast a time series, we use the most recent slope to help project for future forecasts. From this, future forecasts are the most recent level + a linear trend scaled by the forecast horizon.

We have a point forecast for any horizon h (expectation of $Y_{t+h}|y_{1:t}$) as:

$$\hat{y}_{t+h} = \ell_t + hb_t$$

and

$$Var(Y_{t+h}|y_{1:t}) = \sigma^2[1 + \alpha^2 \sum_{i=1}^{h-1}(1+i\beta)^2]$$

We can compute interval forecasts and such as before.

11.8.4 Holt-Winters Smoothing

3 components now: level, time change component, and now we add for time changing seasonal patterns over time. If seasonal variation is constant, we use an additive model else we use a multiplicative model.

11.8.5 Additive Holt-Winters Smoothing

$$\hat{y}_{t+1} = \ell_t + b_t + S_{t+1-L} \quad \text{forecast equation}$$

whereby:

$$\ell_t = \alpha(y_t - S_{t-L}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \quad \text{level}$$

$$b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1} \quad \text{trend equation}$$

$$S_t = \delta(y_t - \ell_t) + (1-\delta)S_{t-L} \quad \text{seasonal indices}$$

We now have a seasonal frequency L, and 3 initial values for i = 1,...,L. The range of α, β, δ are all [0,1]. The seasonal index is updated only depending on what value we set L. If L=12, then update it every 12 months.

The forecasts now have a seasonal pattern and is no longer just a straight line like in previous models.

11.8.6 Multiplicative Holt-Winters Smoothing

$$\hat{y}_{t+1} = ()\ell_t + b_t) \times S_{t+1-L} \quad \text{forecast equation}$$

whereby:

$$\ell_t = \alpha(\frac{y_t}{S_{t-L}}) + (1-\alpha)(\ell_{t-1} + b_{t-1}) \quad \text{level}$$

$$b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)b_{t-1} \quad \text{trend equation}$$

$$S_t = \delta(\frac{y_t}{\ell_t}) + (1-\delta)S_{t-L} \quad \text{seasonal indices}$$

We now have a seasonal frequency L, and 3 initial values for i = 1,...,L. The range of α, β, δ are all [0,1]. The seasonal index is updated only depending on what value we set L. If L=12, then update it every 12 months.

11.8.7 Damped Trend Exponential Smoothing

If forecast for time series for large number of periods, then kinda unreasonable to assume today's trend will help project future's values. We put a penalty on the trend and that this trend line flattens out over time. We accomplish this by adding a damping parameter and flattens out the trend as we project it out. We have a *phi* damping paramter which is $\in [0, 1]$.

$$y_{t+1} = ell_t + \phi(b_t) + \epsilon_{t+1}$$

$$\ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + \phi(b_{t-1}))$$

$$b_t = \beta(\ell_t - \ell_{t-1}) + (1-\beta)\phi b_{t-1}$$

We have a forecasting equation:

$$\hat{y}_{t+h} = \ell_t + \phi(b_t) + \phi^2(b_t) + ... + \phi^h(b_t)$$

11.9 Review Questions for Module 12

What is exponential smoothing?

What is the difference between simple, trend corrected, and Holt-Winters exponential smoothing methods?

Derive point forecasts and variance forecast or the SES and trend corrected methods, starting from the model equations. Write and justify every step.

Explain how to compute forecast intevals based on the SES and trend corrected methods.

11.10 Introduction to ARIMA Models

ARIMA models are extremely popular for forecasting. These are alternatives to exponential smoothing. Before, we attempted to directly model the time series components in one equation (level, trend, seasonal) but now with ARIMA models, we aim to analyse the serial dependence in the data. We aim to model the autocorrelation in the data and then describe this dependence via an ARIMA model. However though, this method is not a replacement to exponential smoothing(where weights given to past data decays exponentially) but rather a complement to it.

11.11 Stationarity and Box-Jenkins methodology

We require the data to be **stationary**, whereby the statistical properties of the data do not change over time. ARIMA attempts to separate the trend, seasonal, and irregularity components. We aim to make a transformation of the data to remove the trend and then another transformation to remove the seasonality. Time series with trend and seasonality are not stationary, since these patterns affect the mean of the series over time. What we end up with is an irregularity component which we can describe with models. We want to do these removals because we want a transformed time series with *stable properties.* We need stationarity so that when we compute the mean, we can have convergence.
Definition 11.4. A time series is **strictly stationary** when the joint distribution of $Y_t, Y_{t-1}, ..., Y_{t-j}$ does not depend on t. The joint density $p(y_t, y_{t-1}, ..., y_{t-k})$ does not depend on t.

The joint distribution does not depend on time means that the values and such aren't changing due to time factor. If time series are stable, it is easy to estimate since the components are not changing over time. This means that the probability distribution function is the same across any index. However, strict stationarity is a very strong assumption so we have a simpler condition that is sufficient for ARIMA models so now we use **weakly stationarity/covariance stationary**. A weakly stationary series is one whereby the mean is constant, variance is constant, and its covariance is *not* a function of time. In the literature, stationary refers to weakly stationary unless specified otherwise.

$$E(Y_t) = \mu$$

$$Var(Y_t) = \sigma^2$$

$$Cov(Y_t, Y_{t-k}) = f(k) \neq g(t)$$

Note that strictly stationary implies weakly stationary data. Do not get this confused with **weakly dependent time series** which states that the correlation for:

$$Corr(X_t, X_{t+h}) \to 0 \quad h \to \infty$$

This means that X_t becomes less correlated with values that are further away from the future. Note that weakly stationary data requires the Covariance to be a function of k and not time, whilst weakly dependence specifies the behaviour of this correlation/covariance needs to decrease to 0 as a function of k. A **strongly dependent/highly persistent** time series is the case in which this does not hold and is the opposite to a weakly dependent series.

From all this, we can use **Box-Jenkins** approach in (a) finding a stationary transformation of the data and (b) modelling the autocorrelations in the transformed data (recall that autocorrelation is the correlation between elements of a series and itself). We can see this is different to exponential smoothing, where for that technique, we explicitly model the different time series components via additive/multiplicative specifications.

We can stabilise the variance of a series via log or Box-Cox transformations. From this, we can also **difference** the data in order to achieve stationarity in the mean of the data (which removes the changes in the level of the series due to things such as trends or seasonality). Finally, ACF and PACF plots helps us to assess stationarity and identify suitable specification for the number of lag terms for the stationary transformation of the series. Conclusively, take log/box-cox to get a constant variance and difference the data to get a constant mean.

PACF is the correlation of different time points once already taking into account of the time series and the effects of the time periods between t and t-k.
Definition 11.5. The **partial autocorrelation of order k** ρ_{kk} is the correlation between Y_t and Y_{t-k} net of effects at times t-1, t-2,...,t-k+1. r_{kk} estimates ρ_{kk}.

We can think of this as a linear regression whereby the partial autocorrelation ρ is the regression coefficient on Y_{t-k}, so now we look at the effect of Y_{t-k} on Y_t after taking into account of the effects of $Y_{t-1}, .., Y_{t-k+1}$. So here we have ρ_{kk} where the first k lets us know what the order of the partial autocorrelation and the second k is which autocorrelation in particular we are looking at. We can see some examples below.

$$Y_t = \rho_{1,0} + \rho_{1,1}Y_{t-1} + a_t$$

$$Y_t = \rho_{2,0} + \rho_{2,1}Y_{t-1} + \rho_{2,2}Y_{t-2} + a_t$$

$$Y_t = \rho_{k,0} + \rho_{k,1}Y_{t-1} + ... + \rho_{k,k}Y_{t-k} + a_t$$

Here, in the second equation, $\rho_{2,1}$ looks at the autocorrelation of $(Y_t, Y_{t-1}$ net of the effect of the correlation with Y_{t-2}

11.12 Differencing

The result time series from differencing is a new transformation. The **first difference** of a time series is:

$$\Delta Y_t = Y_t - Y_{t-1}$$

We try this on a *random walk* model

$$Y_t = Y_{t-1} + \epsilon_t$$

whereby taking the first difference leads to stationary white noise series:

$$\Delta Y_t = Y_t - Y_{t-1} = \epsilon_t$$

We said that $\epsilon_t \sim iid(0, \sigma^2)$. We have a white noise process. From this, we get $\text{E}(\epsilon_t) = 0$, $\text{var}(\epsilon_t) = \sigma^2$, and $\text{cov}(\epsilon_t, \epsilon_{t-k}) = 0$. We have stationarity!

Sometimes we need to take 2 differences to achieve stationarity.

$$\Delta^2 Y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = Y_t - 2Y_{t-1} + Y_{t-1}$$

Whether a time series needs differencing, we can look at the ACF and PACF. In previous component models, ACF plots should see correlation falling slowly over time but now for our differenced data, these autocorrelations should fall quickly, which then indicates stationary data. The ACF of a non-stationary data will decrease slowly whilst the ACF of a stationary series should drop to zero relatively quickly.

Unit root tests are common tests to check for stationarity and whether we need to difference the data but are very sensitive to assumptions so conclusions can change easily. Additionally, hypothesis testing approach isn't quite appropriate since we are using forecasting and thus, we want to minimise the expected loss which we do via model selection. We don't really care if there's a unit root since we just care about the forecast. From this, we should use model selection for model selection and not hypothesis testing.

11.12.1 Unit Root Tests

Unit root tests checks for whether is the data non-stationary and whether does it contain an unit root or not. The null hypothesis is defined as the presence of an unit root. A Dickey-Fuller test is just for an AR(1) model whereby we different the AR(1) model to get:

$$y_t = \rho y_t + u_t$$
$$\Delta y_t = (\rho - 1)y_{t-1} + u_t = \delta y_{t-1} + u_t$$

We then test whether $\delta = 1$ and if we reject the null hypothesis, there is no unit root. However, we can further extend on using more complicated models by utilising the **augmented Dickey-Fuller** test. Now we choose numerous lags to ensure residuals are not *serially correlated*, which can then choose by AIC, BIC, etc. We have that:

$$\Delta Y_t = \alpha + \beta_t + \gamma y_{t-1} + \delta_1 \Delta y_{t-1} + ... + \delta_{p-1} \Delta y_{t-p+1} + \epsilon_t$$

whereby this is the case in which we add a constant and linear trend (We can also remove them). We then test the term on y_{t-1} whereby the null is that $\gamma = 0$ vs $\gamma < 0$. If we reject the null, just like the Dickey-Fuller test, then there is **no unit root**.

Note that all unit roots have *serial correlation* but not all serial correlations have unit roots.

11.12.2 Seasonal Difference

We can also take seasonal differences of the data to address non-stationarity caused by seasonality. The general case is:

$$\Delta_m Y_t = Y_t - Y_{t-m} \quad \text{m is number of seasons}$$

$\delta_{12} Y_t = Y_t - Y_{t-12}$ means that we take the differences over a whole year. We compare the level of time series today in November with the level of time series last November. Then all seasonalities across the year is now not going to have a effect since we now have a time series which looks at only one point per year. We can identify a need for seasonal difference by looking at the ACF in the seasonal lags (how the dependencies look for periods of m). So if monthly data, we compare at points a year ago etc. The ACF of a series that needs seasonal differencing will decrease slowly at the seasonal lags m, 2m, 3m, ...

11.12.3 Combining everything

Here, we can combine both first and seasonal differencing in order to achieve stationarity in our time series data. There may be the case that there is a changing level and seasonal pattern. So now we have a seasonal difference, and then take the first difference of that.

$$\Delta_m(\Delta Y_t) = (Y_t - Y_{t-1}) - (Y_{t-m} - Y_{t-m-1})$$

Doesn't matter which order we difference the data. $\delta_{12}\delta Y_t = (Y_{06/2017} - Y_{05/2017}) - (Y_{06/2016} - Y_{05/2016})$.

11.13 Backshift Operators

These are useful notational devices for ARIMA.

$$BY_t = Y_{t-1}$$

From this, we can then get for 2 lags:

$$B^2 Y_t = BY_{t-1} = Y_{t=2}$$

We can then generalise this to:

$$B^k Y_t = Y_{t-k}$$

When we wish to difference data, we can use the notation (1-B) as well.

$$(1-B)Y_t = Y_t - BY_t = Y_t - Y_{t-1} = \Delta Y_t$$

A D^{th} order difference can be represented by (remember to expand the brackets out first):

$$(1-B)^D Y_t$$

For seasonal differences, we have $(1 - B^m)$

$$(1 - B^m)Y_t = Y_t - B^m Y_t = Y_t - Y_{t-m}$$

And if we wished to do both seasonal and first differences:

$$\begin{aligned}(1-B)(1-B^m)Y_t &= (1 - B - B^m + B^{m+1})Y_t \\ &= (Y_t - BY_t - B^m Y_t + B^{m+1} Y_t) \\ &= Y_t - Y_{t-1} - Y_{t-m} + Y_{t-m-1}\end{aligned}$$

11.14 Models for stationary series

We now have stationary data after all those techniques which is quite useful. There will still be some dependence in the time series over time (so if they are statistically significant in correlation, we can use that information to make forecasts).

11.14.1 AR Models

Autoregressive models of order p, AR(P), are:

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \epsilon_t$$

whereby ϵ_t is a white noise series. Here, the predictors are lag values of the time series.

Let us analyse the AR(1) model $Y_t = c+\phi_1 Y_{t-1}+\epsilon_t$ whereby $\epsilon_t \sim N(0, \sigma^2)$. From this:

$$E(Y_t|y_1, ..., y_{t-1}) = E(Y_t|y_{t-1}) = c + \phi_1 y_{t-1}$$

$$Var(Y_t|y_1, ..., y_{t-1}) = Var(Y_t|y_{t-1}) = \sigma^2$$

We can represent AR(1) model with backshift operator by going: $(1 - \phi\beta)Y_t^* = c + \epsilon_t$. We can then further express this as first difference by going: (1 - $\phi\beta$)$(1 - \beta)Y_t^* = c + \epsilon_t$. So we now can read this as the AR(1) model for the first difference.

If we see a falling ACF, we can assume it is an AR model. If AR model has a negative coefficient, we can see a falling ACF again in absolute terms but it alternates due to the sign of the coefficient. In AR(p) processes, the autocorrelation decreases exponentially. We can identify the autoregressive process by looking at the PACF (see at which lag does it drop). So if it drops dramatically after lag 1, we know it is an AR(1) model. The p^{th} autocorrelation $\rho_{pp} = \phi_p$.

From the linearity of expectations for AR(P) models:

$$E(Y_{t+h}|y_{1:t}) = c + \phi_1 E(Y_{t+h-1}|y1 : t) + ... + \phi_p E(Y_{t+h-p}|y1 : t)$$

whereby the expectation if $\hat{y}_{t+h-i} \text{if} h > 1$ or the forecasted value if we are looking at the future else if h $\leq$ i = y_{t+h-i} or a past value of y.

From this, as h gets larger, both the point forecast and the conditional variance converge exponentially to a constant. We need some restrictions on AR coefficients to ensure stationary models. For AR(1): $-1 < \phi_1 < 1$. For AR(2): $-1 < \phi_2 < 1, (\phi_1 + \phi_2 < 1), (\phi_2 - \phi_1 < 1)$.

11.14.2 MA Models

Instead of having a regression on lag of time series, we now have that the predictors are the lagged values of the random errors.

$$Y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q}$$

whereby ϵ_t is a white noise series.

For a MA(1) process:

$$Y_t = c + \epsilon_t + \theta_1 \epsilon_{t-1}$$

$$E(Y_t | y_{t-1}) = c + \theta_1 \epsilon_{t-1}$$

$$Var(Y_t | y_{t-1}) = \sigma^2$$

These MA models leave clear patterns in the PACF and ACF plots. The patterns will be the reverse of the autoregressive process. The autocorrelation ρ_k cuts off after lag q whilst the partial autocorrelation ρ_{kk} decreases exponentially.

MA(q) processes are **invertible** when we can write them as a linear combination of past values (an AR(∞) process) plus the contemporaneous error term. We require invertibility for estimation/forecasting methods. Therefore we have restrictions on the MA coefficients to achieve this. So if we had

$$X_t = \epsilon_t - \theta \epsilon_{t-1} = (1 - \theta B)\epsilon$$

$$\frac{X_t}{(1 - \theta B)} = \epsilon$$

Recall that the sum of geometric series:

$$S_\infty = a + ra + r^2 a + ... = \frac{a}{1 - r}$$

so if $|\theta| < 1$, this is the restriction required to allow for inversion. So our geometric series when comparing to LHS of our equation and setting a = X_t and a = θB is:

$$X_t + \theta B X_t + \theta^2 B^2 X_t + ...$$

Subbing that into our equation and using the lag operator, we have that:

$$X_t + \theta B X_t + \theta^2 B^2 X_t + \theta^3 B^3 X_t ... = \epsilon$$

$$X_t = -\theta B X_t - \theta^2 B^2 X_t - \theta^3 B^3 X_t - ... + \epsilon$$

$$X_t = -\theta X_{t-1} - \theta^2 X_{t-2} - \theta^3 X_{t-3} ... + \epsilon$$

From this, we have converted a MA(1) process into a AR(∞) process.

We can compare the differences in correlogram behaviour for these two models.: AR(P) $\rightarrow$ ACF decreases exponentially and the PACF is 0 after lag P.

MA(Q) $\rightarrow$ ACF is zero after lag Q and the PACF decreases exponentially.

11.14.3 ARMA Models

We can combine both AR and MA models into an ARMA(P,Q) model by using both types of predictors.

$$Y_t = C + \phi_1 Y_{t-1} + ... + \phi_p Y_{t-p} + \theta_1 \epsilon_{t-1} + ... + \theta_q \epsilon_{t-q} + \epsilon_t$$

whereby ϵ_t is a white noise series. Using *backshift notation*

$$(1 - \sum_{i=1}^{p} \phi_i B^i) Y_t = c + (1 + \sum_{i=1}^{p} \theta_i B^i) \epsilon_t$$

whereby the LHS is the AR(P) process, which we then move over to the other side. ARMA(P,Q) $\rightarrow$ Both ACF and PACF decreases exponentially.

For ARMA(1,1):

$$Y_t = c + \phi_1 Y_{t-1} + \theta_1 \epsilon_{t-1} + \epsilon_t$$

$$Y_t = c + \phi_1 B Y_t + \theta_1 B \epsilon_t + \epsilon_t$$

$$(1 - \phi B) Y_t = c + (1_{\theta_1} B) \epsilon_t$$

where the LHS is the AR(1) model and the RHS is the MA(1) model.

11.15 ARIMA Models

ARIMA(P,d,Q) whereby d is the first difference.

$$(1 - \sum_{i=1}^{p} \phi_i B^i)(1 - B)^d Y_t = c + (1 + \sum_{i=1}^{p} \theta_i B^i) \epsilon_t$$

p is AR order, d is degree of first differencing, and q is moving average order. Note that you can rewrite ARIMA models into simple exponential smoothing models. c is just an intercept we include in the model for linear trend. To include linear trend, we can just add a drift term into the equation. Our forecasts is going to have a linear trend trend c. The **random walk plus drift model**

$$Y_t = c + Y_{t-1} + \epsilon_t$$

whereby

$$Y_{t+h} = Y_t + \sum_{i=1}^{h} (c + \epsilon_{t+i})$$

$$\hat{y}_{t+h} = y_t + c \times h$$

$$Var(Y_{t+h}|y_{1:t}) = h\sigma^2$$

We can estimate the model via maximum likelihood. To choose the order (p,q), we can use visual identification from ACF/PACF plot, AIC, and model validation. We can use model selection to also decide whether to include an intercept to model permanent trends.

11.16 Seasonal ARIMA Models

In seasonal ARIMA, the first 3 terms describe the non-seasonal parts whilst the last 3 describes the seasonal components.

$$ARIMA(p, d, q)(P, D, Q)_m$$

where D is the order of seasonal differencing. P and Q are the order of the seasonal AR,MA component and m is the number of seasons/period.

ARIMA(0,0,0)(P,0,0) $\rightarrow$ sample autocorrelation decreases exponentially for lags m, 2m, 3m. Sample partial autocorrelation cuts off at lag Pm.

ARIMA(0,0,0)(0,0,Q) $\rightarrow$ sample partial autocorrelation decreases exponentially for lags m, 2m, 3m. Sample autocorrelation cuts off at lag Qm.

Usually, we only require one seasonal AR or MA term.

11.17 Review Questions for Module 13

What is stationarity and why is it a fundamental concept in ARIMA modelling? Stationarity is when we have mean, variance constant and covariance doesn't depend on time. We are modelling the serial correlation of dependent variable.

What transformation do we apply to a time series to make it stationary? We difference it to make it stationary for the mean whilst log/box-cox is for variance.

How do we identify AR vs MA processes from ACF and PACF plots? AR for ACF will decreases exponentially and the PACF cuts off after lag p. For MA, ACF cuts off after lag p whilst PACF falls exponentially.

What is an ARIMA model? ARMA model for differenced series.

Write the equation for a seasonal ARIMA model using backshift notation.

$$(1 - \sum_{i=1}^{p} \phi_i \beta^i)(1 - \beta)^d Y_t = c + (1 + \sum_{i=1}^{p} \theta_i \beta^i)\epsilon_t$$