Deanship of Graduate Studies

Al-Quds University

Multivariate Time Series with Application On Recurrent Neural Networks

Safa Nader Mustafa Shanaa

M.Sc. Thesis

Jerusalem-Palestine

1441/2020

Multivariate Time Series with Application On Recurrent Neural Networks

Prepared by:

Safa Nader Mustafa Shanaa

B.Sc:Mathematics-Al-Quds University/ Palestine

Supervisor: Dr. Khalid Salah

A thesis submitted in partial fulfillment of requirements for the degree of Master of Mathematics, Department of Mathematics / Graduate Studies / Al-Quds University.

1441\2020

Thesis Approval

Multivariate Time Series with Application On Recurrent Neural Networks

Prepared by: Safa N. Shanaa

Registeration No.: 21510045

Supervisor: Dr. Khalid Salah

Master Thesis submitted and accepted, Date: 22 / 6 / 2020

The names and the signatures of the examining committee members are as

1- Head of Committee: **Dr. Khalid Salah.** Signature:

2- Internal Examiner: **Dr. Jamil Jamal** Signature: Jamil Jamal

3- External Examiner: **Dr. Mahmoud AlManassra** Signature:

Jerusalem-Palestine

1441/2020

Dedication

To my mother, my friends, and to all those who is interested in mathematics especially statistics.

Safa Shanaa

Declaration

I certify that this submitted for the degree of master is the result of my own research, except where otherwise acknowledge. And that this (or any part of the same) has not been submitted to a higher degree to any other university or institution.

Signature:

Student's name: Safa Nader Mustafa Shanaa

Date: 22 / 6 / 2020

Acknowledgement

Thanks for GOD, the Cherisher and Sustainer of the words. Peace and blessing upon our first teacher and educator, Prophet Mohammed, who has taught the whole world.

I am thankful to my supervisor Dr. Khalid Salah for all his support, guidance and encouragement to finish this thesis successfully, also my great thanks goes to the examiners, who contributed to improving the message through their constructive comments and advice.

My big thanks are presenting to my university presents by its head president Dr. Emad Abu Keshek for giving me all support.

Abstract

Multivariate time series data in practical applications, such as health care, geosciences, engineering, and biology. This thesis introduces a survey study of time series analysis to recurrent neural networks research, an analytic domain that has been essential for understanding and predicting the behavior of variables across many diverse fields, in this research the following were investigated.

First, the characteristics and preliminaries of time series data are investigated and discussed, including various time series models, specially, Autoregressive Models such as, AR, MA, ARMA, and ARIMA. Frequently one wishes to fit a parametric model to time-series data and determine accurate values of the parameters and reliable estimates for the uncertainties in those parameters. It is important to gain a thorough understanding of the noise and develop appropriate methods for parameter estimation, so that various approaches of parameter estimates will be considered in this thesis, such as, yules walker method, least square method, method of moments and maximum likelihood approach.

Second, different time series modeling techniques are surveyed that can address various topics of interest to artificial neural networks researchers, including describing the pattern of change in a variable, modeling seasonal effects, assessing the immediate and long-term impact of a salient event, and forecasting future values. The structure of the artificial neural networks especially the recurrent neural networks were discussed in details in this research, concerning on GRUs and LSTMs, and their properties, also some difficulties that arises in recurrent neural networks such as vanished gradient and the overfitting were discussed.

To illustrate these approaches, an illustrative application based on Monte Carlo and bootstrapping methods is used throughout the research, constructing a one layer hidden recurrent neural networks and applied back-propagation, for the purpose of comparison, the variance of error in each method was estimated.

Contents *page*

Declaration *i*

Acknowledgment *ii*

Abstract *iii*

Table of Contents *iv*

List of Tables *vi*

List of Figures *vii*

Chapter one 1

Introduction

Chapter two 4

Time Series

2.1 Preliminaries of Time Series 4

2.2 Autoregressive model AR(p) 12

2.3 Moving Average model 26

2.4 Autoregressive Moving Average model 39

2.5 ARIMA model (p, q, d) 44

Chapter three 46

Parameter estimation approaches in Time Series model

3.1 Yule Walker equations 47

3.2 The Method of Moments 49

3.3 The Least Square method (LSE) 55

3.4 The Maximum Likelihood Estimation 59

3.5 Box Jenkins method for ARIMA (p, q, d) 63

3.6 Monte Carlo method 65

3.7 Bootstrapping in ARIMA models 67

Chapter four 69

Recurrent Neural Networks

4.1 Artificial Neural Network structure 69

4.2 The activation functions 71

4.3 The perceptron 74

4.4 The learning procedure 76

4.5 Training 84

4.6 Recurrent Neural Networks 84

Chapter five 93

Recurrent Neural Networks Applications

5.1 Using the bootstrapping method in minimizing the error 94

5.2 Using the back-propagation method to update the weights 101

5.3 Using Monte Carlo method in estimating the error 102

5.4 Discussion and conclusion 107

References 111

Arabic abstract 115

List of Tables

5.1 Table of weights 95

5.2 Table of bootstrap samples 96

5.3 Table of the minimum of least square error 98

5.4 Table of the average and the estimated standard error of the i^{th} predictive values 99

5.5 Table of the weights update w_{ij} 102

5.6 Table of the Monte Carlo inputs generated randomly . . 103

5.7 Table of the Monte Carlo estimates 104

5.8 Table of the average and the estimated standard error of the estimated outputs 104

5.9 Table of the skewness and the kurtosis of the estimated outputs 105

5.10 Table of comparison between the Monte Carlo method and the bootstrapping 107

List of Figures

The figure's number	the figure's name	the page
2.1	AR(1) process when $\phi = 0.9$	17
2.2	AR(1) process when $\phi = 1.5$	17
2.3	AR(1) process when $\phi = -0.2$	18
2.4	AR(1) process when $\phi = 0$	18
2.5	AR(1) process when $\phi = 1$	19
2.6	ACF of AR(1) model when $\phi = 0.2$	20
2.7	ACF of AR(1) model when $\phi = -0.2$	20
2.8	ACF of AR(1) model when $\phi = 0.9$	21
2.9	MA(1) process when $\theta = 0.8$	30
2.10	MA(1) process when $\theta = 1.6$	30
2.11	MA(1) process when $\theta = -0.3$	31
2.12	MA(1) process when $\theta = 0$	31
2.13	MA(1) process when $\theta = 1$	32
2.14	ACF of MA(1) model when $\theta = 0.3$	37
2.15	ACF of MA(1) model when $\theta = -0.3$	38
2.16	ACF of MA(1) model when $\theta = 0.8$	38
5.1	Estimation of y_1 by the bootstrapping method	100
5.2	Estimation of y_1 by the bootstrapping method	100
5.3	Estimation of y_1 by the Monte Carlo method	106
5.4	Estimation of y_2 by the Monte Carlo method	106
5.5	Estimation of y_3 by the Monte Carlo method	107

Chapter 1

Introduction

One of the most beautiful data analysis performed in many fields is: The time series. We fed you with a brief introduction to the time series analysis, in the field of data science; to have a good taste of the time series. Time series analysis is increasingly very important because it's needed for a large number of applications with both statistical and machine learning techniques.

There are many things in our life that completely stopped when their sequence is stopped such as language and to use them with a reasonable output, we need a network that use the previous knowledge about the data to understand them completely. For this reason; the recurrent neural networks are invented.

The recurrent neural networks are feedback artificial neural networks, that are a branch of the nonlinear time series. They have an internal memory, hence they are used for machine learning problems that have sequential data, and are used also widely in advances of network architectures, optimization techniques, and parallel computation.

The contents of the thesis are five chapters as follow: In the next chapter, we gave the characteristics of time series, how to change a non-stationary time series to a stationary one by using differencing method. Then discussed the characteristics of the ARIMA models and gave examples of AR(1) and MA(1) and sketched them and their autocorrelation function and analyzed their graphs.

In the third chapter, we estimated the parameters of the ARIMA models by many methods such as the Yule Walker equations that estimate the AR(p) parametres, the method of moments, the least square method and the maximum likelihood estimation. And discussed the procedure of the Box Jenkins method, the Monte Carlo method and the bootstrapping method.

Chapter 4 is about the structure of the recurrent neural networks, how they work and their activation functions, then discussed the learning procedure, the algorithm of the backpropagation, and the learning methods. We explain how to solve some problems that the recurrent neural network process faced such that the vanishing gradient problem and the overfitting. Then showed the structure of the long short term memory and the gated recurrent neural networks, and what are the differences between them.

Finally in chapter 5, we made an example for the recurrent neural network with one hidden layer, and we updated the weights by the backpropagation algorithm and then applied the bootstrapping method and the Monte Carlo method on it to estimate the predictive value of the output, sketched the estimated predictive values and compared the results in the two methods.

Chapter 2

Time Series

2.1 Preliminaries of Time series

Definition 1. *Time series* $(X_t)_{t=1}^{\infty}$ *is a set of observations that occurred sequentially over time. [29]*

Time series data most often gathered in regular intervals, it isn't only about the observations that happened in chronological order but the Time series analysis can be applied to any variable that changes over time. The serial dependence happened when the value of a data point at one time is statistically dependent on another data point in another time and the ordering of the time points is important hence it often shows serial dependence and makes the time series analysis unique.

Trends are a simple and effective means for incorporating a steady upward or downward movement over time into the behavior of a time series.[2]

Seasonability happened when short term movements occurred in the data because of seasonal factors.

Definition 2. *A multivariate stochastic process $\{X_t; t \in T\}$ is a collection of vector-valued random variables*

$$X_t = \begin{bmatrix} X_{t1} \\ X_{t2} \\ . \\ . \\ . \\ X_{tm} \end{bmatrix} \tag{2.1}$$

Where T is an index set for which the random variables $\{X_t; t \in T\}$ are defined on the same sample space. When T represents time, then $\{X_t; t \in T\}$ is a multivariate time series. [32]

Definition 3. *The multivariate time series $\{X_t\}$ is a linear process if it has the representation*

$$X_t = \sum_{j=-\infty}^{\infty} C_j Z_{t-j},\ Z_t \sim \mathcal{WN}(0, \sigma^2).$$

Where $\{C_t\}$ is a sequence of $m \times m$ matrices whose components are absolutely summable. [4]

Stationarity: A time series $(X_t)_{t=1}^{\infty}$ is stationary if it has statistical properties similar to those time shifted series $(X_{t+h})_{t=1}^{\infty}, \forall h \in \mathbb{Z}$ [4]

Definition 4. *A time series $(X_t)_{t=1}^{\infty}$ is strict stationary if the joint distribution of any collection of k values is time invariant, that means* $\forall k > 1 \quad and \quad \forall s > 0, k, s \in \mathbb{Z}$

$p(X_{t_1}, ..., X_{t_k}) = p(X_{t_1+s}, ..., X_{t_k+s})$ *that's the mean and the variance of the time series are constants over time and the* $cov(X_t, X_{t-k})$ *doesn't depend on the value of t and depend only on k. [25]*

Definition 5. *A time series $(X_t)_{t=1}^{\infty}$ is weakly stationary or second order stationary if the mean, variance and covariances are time invariant. That's for the integers* $t > 0$ *and* $s < t$,

$E(X_t) = \mu.$

$var(X_t) = \sigma^2 \;\; that's \;\; \sum_{t=0}^{\infty} \mid X_t \mid < \infty.$

$cov([X_{t-s}, X_s]) = \gamma_s$ *so* $cov([X_{t-s}, X_s])$ *depends on s only. [25]*

The sample autocovariance function (ACVF), γ_k, for some lag k can be given as [15]

$$\gamma_k = \frac{1}{n}\sum_{t=1}^{n-k}(X_t - \bar{X})(X_{t+k} - \bar{X}) \tag{2.2}$$

We substitute n-k by n since the time series here is stationary, see [15] that's

$$\gamma_k = cov(X_t, X_{t-k}) = E((X_t - \mu)(X_{t-k} - \mu)). \tag{2.3}$$

Note that the sample autocovariance of X_t at lag 0, γ_0, is the sample variance of X_t that's

$$\gamma_0 = cov(X_t, X_t) = E((X_t - \mu)^2) = \sigma^2 \tag{2.4}$$

The autocorrelation shows the relation between the time series values in different time, the coefficient of correlation between two values in the time series is called the autocorrelation function (ACF). The ACF describes the autocorrelation between an observation and another observation at a previous time step that includes direct and indirect dependence information.

Let $(X_t)_{t=1}^{\infty}$ be a stationary time series, then the sample autocorrelation function (ACF) is

$$\begin{aligned} \rho_k &= Corr(X_t, X_{t+k}) = \frac{cov(X_t, X_{t+k})}{\sqrt{\sigma_{x_t}\sigma_{x_{t-k}}}} = \frac{cov(X_t, X_{t+k})}{\sigma_{x_t}} \\ &= \frac{\gamma_k}{\gamma_0}[4] \end{aligned} \tag{2.5}$$

Where $k \in \mathbb{N}$.

$\sigma_{X_t} = \sigma_{X_{t+k}}$; when the process is stationary (the variance is time invariant in the stationary process).

This value of k shows the amount of the time passing previously that's called the lag.

For a stationary time series, the ACF drops to zero quickly. While the

ACF of nonstationary data decreases slowly. As a result, the autocorrelation $\rho_k = \frac{\gamma_k}{\gamma_0}$ is also independent of t.

example 2.1.1. *White noise [29] $(X_t)_{t=1}^{\infty}$ is a sequence of independent and identically distributed random variables with finite mean and variance, all the ACFs for white noise are zero. That's*

$$Corr[\epsilon_t, \epsilon_{t-j}] = 0, \forall j, j \neq 0. \tag{2.6}$$

If $(X_t)_{t=1}^{\infty}$ is normally distributed with mean zero and finite variance, then it's called Gaussian white noise, and we can then denote it as (if mean and var is known) $\epsilon_t \sim \mathcal{WN}(0, \sigma^2)$.

remark 2.1.1. *The white noise process is stationary.*

Proof. Let ϵ_t be a white noise series,that's ϵ_t consists of iid serially uncorrelated random variables with $E(\epsilon_t) = 0, var(\epsilon_t) = \sigma^2$, both constants are free of t.
$cov(\epsilon_t, \epsilon_t) = E((\epsilon_t - \mu)^2) = \sigma^2$.
$$\gamma_k = cov(\epsilon_t, \epsilon_{t-k}) = \begin{cases} \sigma^2 & k = 0 \\ 0 & k \neq 0 \end{cases}$$
which is free of time (i.e. depends only on k).
So the white noise process is stationary [29]. □

remark 2.1.2. *The random walk process is nonstationary.*

Proof. Let y_t be a random walk series, that's $y_t = y_{t-1} + \epsilon_t$, where ϵ_t is white noise series, with $E(\epsilon_t) = 0, var(\epsilon_t) = \sigma^2$,

so $y_{t-1} = y_{t-2} + \epsilon_{t-1}$.

Similarly $y_{t-2} = y_{t-3} + \epsilon_{t-2}$,

so by recursive substitutions, we get

$y_t = y(0) + \epsilon_t + \epsilon_{t-1} + \epsilon_{t-2} + \epsilon_{t-3} + ... + \epsilon_1$,

so $E(y_t) = E(y(0)) + E(\epsilon_t + \epsilon_{t-1} + \epsilon_{t-2} + \epsilon_{t-3} + ... + \epsilon_1)$,

but $E(\epsilon_j) = 0$, for any j $\in \mathbb{N}$,

so $E(y_t) = y(0)$,

so $E(y_t)$ is constant, $\forall t$, which is free of t,

that's the mean of the random walk series is time invariant,

also $cov(y_t, y_{t-k}) = cov(\epsilon_t + \epsilon_{t-1} + ... + \epsilon_1, \epsilon_{t-k} + \epsilon_{t-k-1} + ... + \epsilon_1)$, $k \in \mathbb{N}$.

$$
\begin{aligned}
cov(y_t, y_{t-k}) &= cov(\epsilon_t + \epsilon_{t-1} + ... + \epsilon_1, \epsilon_t + \epsilon_{t-1} + ... + \epsilon_1) + cov(\epsilon_t + \\
&\quad \epsilon_{t-1} + \epsilon_{t-2} + \epsilon_{t-3} + ... + \epsilon_1, \epsilon_{t-1} + \epsilon_{t-2} + ... + \epsilon_{t-k}). \\
&= \sum_{i=1}^{t-k} cov(\epsilon_i, \epsilon_i) + \sum_{1 \leq i} \sum_{\neq j \leq t-k} cov(\epsilon_i, \epsilon_j). \\
&= \sum_{i=1}^{t-k} var(\epsilon_i). \\
&= (t-k)\sigma^2.
\end{aligned}
$$

We see that $cov(y_t, y_{t-k})$ depends on time t.

Thus the random walk process is nonstationary [29]. $\square$

remark 2.1.3. *The deterministic trend process is nonstationary.*

Proof. Let y_t be a deterministic trend series where deterministic trend is nonrandom function of t, that's $y_t = f(t) + c_t$.

Where c_t is a stationary ARMA(p,q) process and f(t) is function of time.

But c_t is stationary so $E(c_t) = u$, where u is a constant. So

$$E(y_t) = E(f(t)) + E(c_t).$$
$$= E(f(t)) + u_t.$$

We see that $E(y_t)$ changes with time and it's not constant so the deterministic trend process isn't stationary. □

remark 2.1.4. *The deterministic linear trend process is nonstationary.*

Proof. Let y_t be a deterministic linear trend series where deterministic trend is nonrandom function of t, that's $y_t = at + \epsilon$.

where, a is a constant and ϵ is a white noise series, with $E(\epsilon_t) = 0, var(\epsilon_t) = \sigma^2, so \;\; E(y_t) = E(at) + E(\epsilon_t)$, but $E(\epsilon_t) = 0$.

so $E(y_t) = at$ which changes with time and not constant so the deterministic trend process is not stationary. □

Differencing method

Differencing [25] is computing differences between consecutive observations, differencing stabilizes the mean and the variance of the time series by removing changes in the level of the time series and also eliminating trend and seasonability.

A time series $(x_t)_{t=1}^{\infty}$ has a constant drift in trend that may be transformed to a stationary time series (no mean drift) by taking first differences $w_t = x_t - x_{t-1}$ that's $w_t = (1-B)x_t$, where B is the back shift operator.

Higher order differencings are computed to remove polynomial trends, e.g. The 2nd order differencing $w_t = (1-B)^2 x_t$ removes a constant growing drift in trend.

For any time series $(X_t)_{t=0}^{n}$, the first difference process ∇X_t of $(X_t)_{t=0}^{n}$ is $\nabla X_t = X_t - X_{t-1}$ for all t =1, 2, ..., n.

In many situations, a nonstationary process can be transformed into a stationary process by taking first difference.

The first difference $X_t - X_{t-1} = \epsilon_t$ is white noise which is stationary.

The second difference process $\nabla^2 X_t$ is

$$
\begin{aligned}
\nabla^2 X_t &= \nabla X_t - \nabla X_{t-1}. \\
&= X_t - X_{t-1} - (X_{t-1} - X_{t-2}). \\
&= X_t - 2X_{t-1} + X_{t-2}.
\end{aligned}
$$

So the d^{th} difference process $\nabla^d X_t$ is

$$\begin{aligned}\nabla^d X_t &= \nabla(\nabla^{d-1} X_t). \\ &= \nabla^{d-1} X_t - \nabla^{d-1} X_{t-1}. \\ &= \sum_{j=0}^{d} (-1)^d \binom{n}{d} X_{t-d}.\end{aligned}$$

Where $d \in \mathbb{N}$, we have $X_{t-1} = BX_t$, where B is shift back operator when it's applied to the time series, it shifts the time by one unit, that's

$$X_{t-d} = B^d X_t. \tag{2.7}$$

also $\nabla X_t = X_t - X_{t-1} = X_t - BX_t = (1 - B)X_t$,
so $\nabla^d X_t = (1 - B)^d X_t$.

2.2 Autoregressive Model AR(p)

The autoregressive model AR(p) [25] is a linear invertible time series model that's written as a function of previous values of the series $(y_t)_{t=0}^{\infty}$, where AR(p) is of order p, p positive integer number.

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t \tag{2.8}$$

Where ϕ_i's are regression parameters, $\forall i \in \mathbb{N}$, ϵ_t is a white noise that's independent of all previous values $y_{t-1}, y_{t-2}, ..., y_{t-p}$.

The order of an autoregression p is the number of immediately preceding values in the series that is used to predict the value at the present time. More generally, a k^{th} order autoregression, it's written as AR(k), it is a multiple linear regression in which the value of the series at any time t is a (linear) function of the values at times t-1, t-2, ... , t-k.

Substitute $y_t = y_t - \mu$ in (2.8) then

$y_t - \mu = \phi_1(y_{t-1} - \mu) + \phi_2(y_{t-2} - \mu) + ... + \phi_p(y_{t-p} - \mu) + \epsilon_t.$

$$\begin{aligned} y_t &= \mu + \phi_1(y_{t-1} - \mu) + \phi_2(y_{t-2} - \mu) + ... + \phi_p(y_{t-p} - \mu) + \epsilon_t. \\ &= \mu + \phi_1 y_{t-1} - \phi_1\mu + \phi_2 y_{t-2} - \phi_2\mu + ... + \phi_p y_{t-p} - \phi_p\mu + \epsilon_t. \\ &= \mu(1 - \phi_1 - \phi_2 - ... - \phi_p) + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t. \\ &= \alpha + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t. \end{aligned} \tag{2.9}$$

Where $\alpha = \mu(1 - \phi_1 - \phi_2 - ... - \phi_p)$.

(2.9) is similar to the regression model so we can use regression parameter estimation methods to estimate $\phi_i's$, i=1, 2, ..., p.

Let B be a shift operater that's

$$(y_t)B = y_{t-1}. \tag{2.10}$$

Substitute (2.10) in (2.8), we get

$$y_t = \phi_1 B y_t + \phi_2 B^2 y_t + ... + \phi_p B^p y_t + \epsilon_t. \tag{2.11}$$

Substitute $y_t - \mu$ instead of y_t in (2.10) results

$$(1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^P)(y_t - \mu) = \epsilon_t. \tag{2.12}$$

or

$$\phi(B)(y_t - \mu) = \epsilon_t. \tag{2.13}$$

Where

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p \tag{2.14}$$

is the nonseasonal AR operator of order p.

The equation

$$\phi(B) = 0 \tag{2.15}$$

is the characteristic equation for the autoregressive process.

remark 2.2.1. *For p=1, AR(1) is written as*

$$y_t = \phi y_{t-1} + \epsilon_t. \tag{2.16}$$

Assume that AR(1) is stationary, then the ACF for AR(1) decays exponentially as k increases. [4]

Proof. Let y_t be a stationary time series, so AR(1) is written as

$y_t = \phi y_{t-1} + \epsilon_t$.

$var(y_t) = var(\phi y_{t-1} + \epsilon_t) = \phi^2 var(y_{t-1}) + var(\epsilon_t) + 2\phi cov(y_{t-1}, \epsilon_t)$,

but y_{t-1} *and* ϵ_t *are independent so* $cov(y_{t-1}, \epsilon_t) = 0$.

So $var(y_t) = \phi^2 var(y_{t-1}) + \sigma^2$. (Since $var(\epsilon_t) = \sigma^2$).

But $var(y_t) = var(y_{t-1}) = \gamma_0$. (Since y_t is a stationary time series).

So $\gamma_0 = \frac{\sigma^2}{1-\phi^2}$.

But $\gamma_0 > 0$ $\ so\ $ $0 < \phi^2 < 1$ $\ that's\ $ $-1 < \phi < 1$.

To find γ_k, *multiply both sides of* (2.16) *by* y_{t-k} *to get*

$y_t y_{t-k} = \phi y_{t-1} y_{t-k} + \epsilon_t y_{t-k}$.

Take expectation of both sides of the last equation, we get

$E(y_t y_{t-k}) = \phi E(y_{t-1} y_{t-k}) + E(\epsilon_t y_{t-k})$.

But y_{t-1} *and* ϵ_t *are independent since* $AR(1)$ *is stationary.*

So $E(\epsilon_t y_{t-k}) = E(\epsilon_t)E(y_{t-k}) = 0$. (Since $E(\epsilon_t) = 0$).

But $E(y_t) = 0$, $\forall t$ since AR(1) is stationary, hence we have

$\gamma_k = cov(y_t, y_{t-k}) = E(y_t y_{t-k}) - E(y_t)E(y_{t-k}) = E(y_t y_{t-k})$.

$\gamma_{k-1} = cov(y_{t-1}, y_{t-k}) = E(y_{t-1} y_{t-k}) - E(y_{t-1})E(y_{t-k}) = E(y_{t-1} y_{t-k})$.

We let $\gamma_k = \phi\gamma_{k-1}$.

When k=1, $\gamma_1 = \phi\gamma_0 = \phi\frac{\sigma^2}{1-\phi^2}$.

When k=2, $\gamma_2 = \phi\gamma_1 = \phi^2\frac{\sigma^2}{1-\phi^2}$.

For $k > 2, \gamma_k = \phi\gamma_{k-1} = \phi^k\frac{\sigma^2}{1-\phi^2}, \quad for\ k = 1, 2, \ldots$.

$$\rho_k = \frac{\gamma_k}{\gamma_0} = \frac{\frac{\phi^k\sigma^2}{1-\phi^2}}{\frac{\sigma^2}{1-\phi^2}} = \phi^k, for\ k = 1, 2, ... \qquad (2.17)$$

The ACF for AR(1) decays exponentially as k increases since

$-1 < \phi < 1$.

So the ACF $\rightarrow 0$ fast enough (like a geometric series) as k$\rightarrow \infty$ (but never be truncated). [5] $\square$

example 2.2.1. *We draw some different AR(1) processes:* $y_t = \phi y_{t-1} + \epsilon_t$

with n= 15, and notice that:

- $\phi = 0.9$ *in figure 2.1, AR(1) process is stationary.*
- $\phi = 1.5$ *n figure 2.2, AR(1) process isn't stationary, it's trend.*
- $\phi = -0.2$ *in figure 2.3, AR(1) process is stationary.*
- $\phi = 0$ *in figure 2.4, AR(1) process is* $y_t = \epsilon_t$ *is a white noise.*
- $\phi = 1$ *in figure 2.5, AR(1) process is* $y_t = \epsilon_t$ *is a random walk.*

remark 2.2.2. *AR(1) process is stationary iff* $\mid \phi \mid < 1$.

For bigger values and with more values, when $\phi > 0$, *then the ACF simulation is smooth, and the adjacent values of* y_t *are positively correlated, but when* $\phi < 0$, *then the ACF simulation is violent and rapid oscillations, and the adjacent values of* y_t *are negatively correlated.*

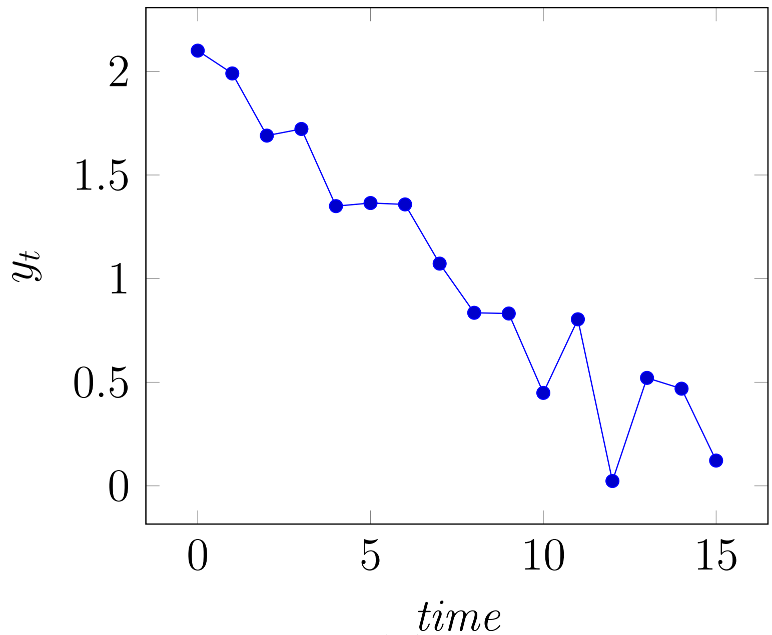

Figure 2.1: AR(1) process when $\phi = 0.9$

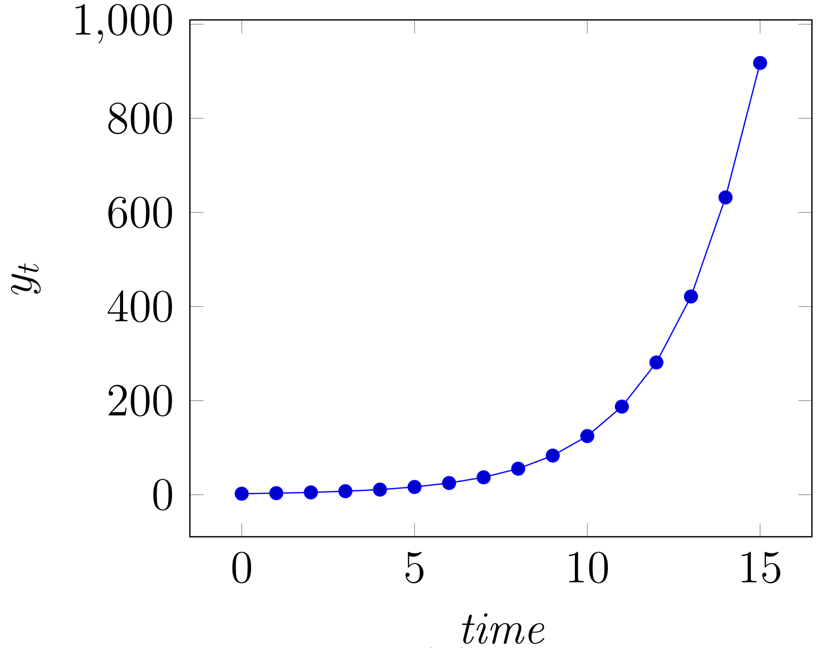

Figure 2.2: AR(1) process when $\phi = 1.5$

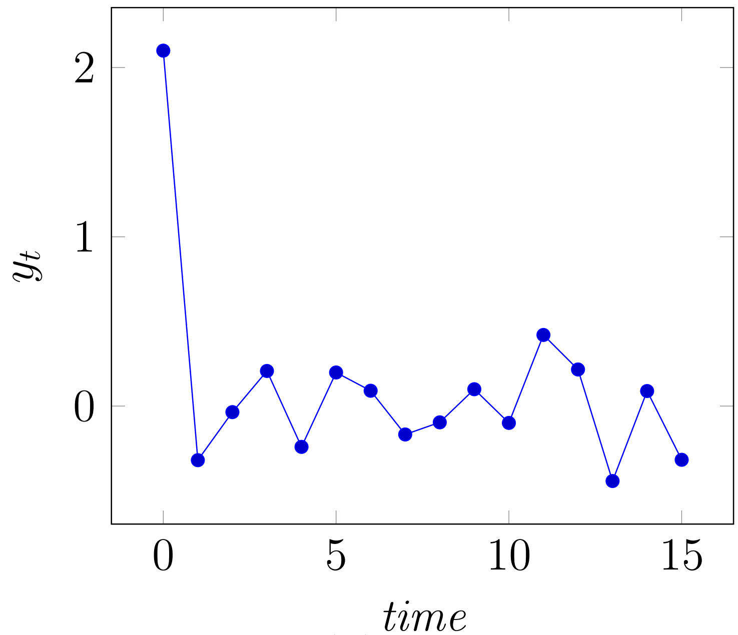

Figure 2.3: AR(1) process when $\phi = -0.2$

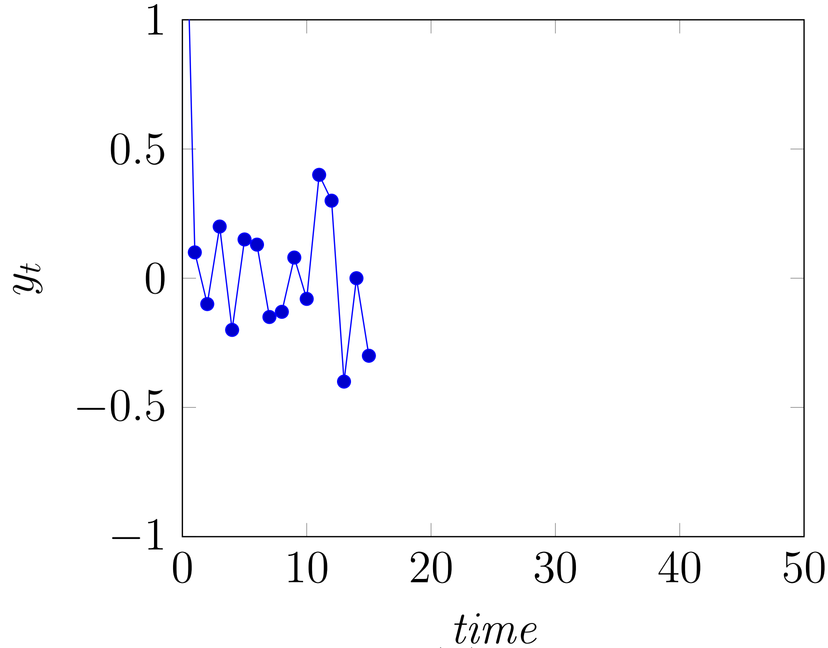

Figure 2.4: AR(1) process when $\phi = 0$

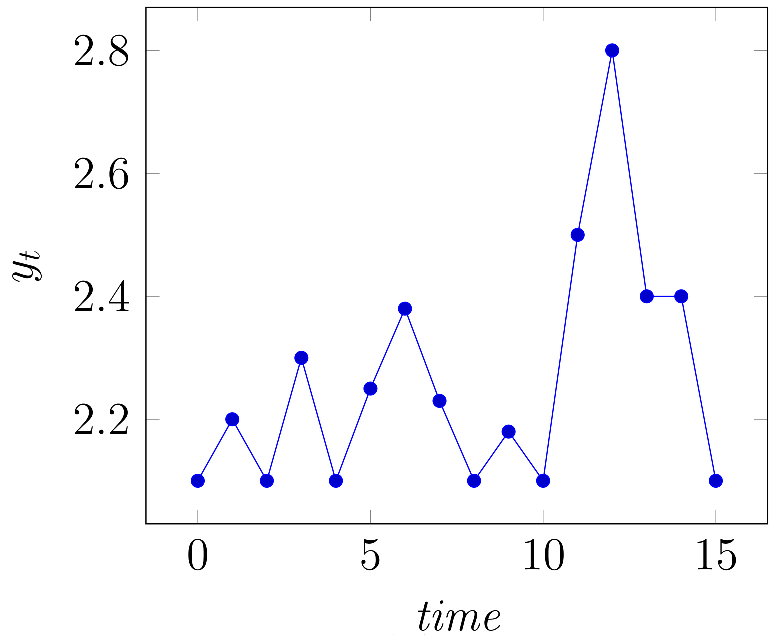

Figure 2.5:AR(1) process when $\phi = 1$

We draw the ACFs for some different AR(1) processes, and notice that:

- $\phi = 0.2$ *in figure 2.6, ACF decays rapidly, all* $\rho_k > 0$.
- $\phi = -0.2$ *in figure 2.7, ACF decays rapidly but all* ρ_k *is alternating.*
- $\phi = 0.9$ *in figure 2.8, ACF decays slowly, all* $\rho_k > 0$.

remark 2.2.3. *When $\phi < 0$, the autocorrelations tends to oscillate between positive(k is even) and negative values(k is odd).*

When $\phi > 0$, the autocorrelations will be positive, decays exponentially to zero.

If ϕ is close to ± 1, then the decay of ACF will be more slowly.

If ϕ isn't close to ± 1, then the decay of ACF will decrease rapidly.

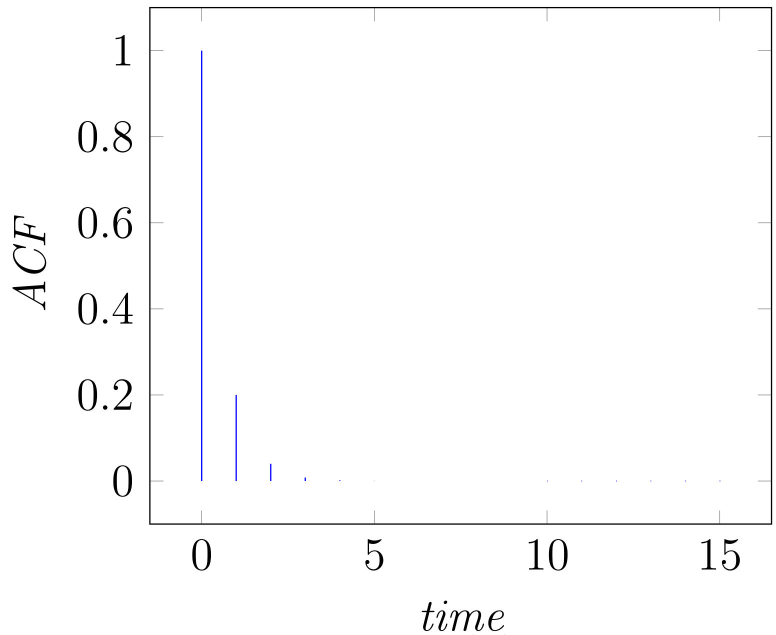

Figure 2.6:ACF of AR(1) model when $\phi = 0.2$

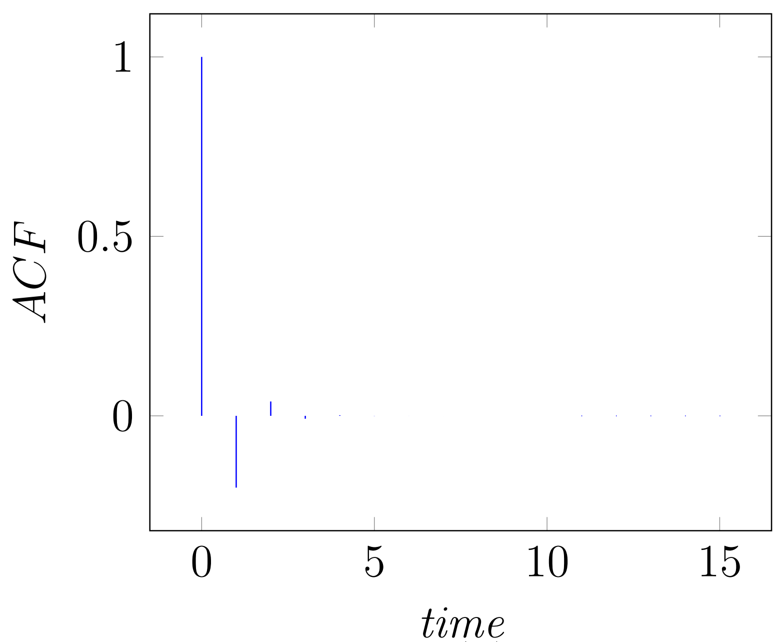

Figure 2.7: ACF of AR(1) model when $\phi = -0.2$

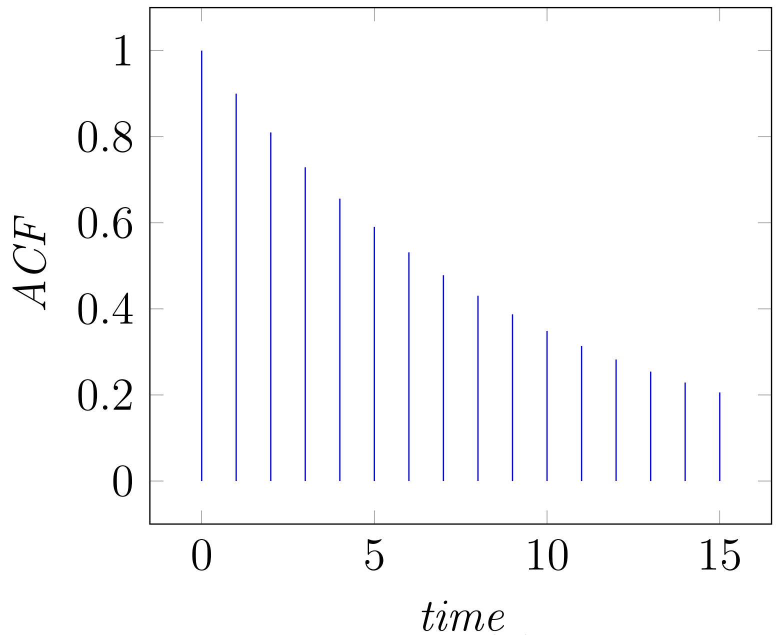

Figure 2.8: ACF of AR(1) model when $\phi = .9$

remark 2.2.4. *The AR(1) is stationary when $|\phi| < 1$. [10]*

Proof. The AR(1) process is written as $y_t = \phi y_{t-1} + \epsilon_t$ or substitute k=1 in (2.12) to get $(1 - \phi B)y_t = \epsilon$.

Where B is shift back operator that's $y_t B = y_{t-1}$.

The characteristic polynomial for AR(1) is

$$1 - \phi B = 0 \tag{2.18}$$

So the root of (2.18) is $\phi = \frac{1}{B}$.

But AR(1) is stationary so $|\frac{1}{\phi}| > 1$.

So AR(1) is stationary when $|\phi| < 1$. [10] $\square$

remark 2.2.5. *The AR(2) is stationary if and if $|\phi_2| < 1$, $\phi_1 + \phi_2 < 1$, $\phi_2 - \phi_1 < 1$. [5]*

Proof. The AR(2) process is written as $y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \epsilon_t$,

or substitute k=2 in (2.12) to get $(1 - \phi_1 B - \phi_2 B^2)y_t = \epsilon$.

Where B is shift operator that's $y_t B = y_{t-1}$.

The characteristic polynomial for AR(2) is $\phi(y) = 1 - \phi_1 y - \phi_2 y^2 = 0$.

So $y^{-2} - \phi_1 y^{-1} - \phi_2 = 0$.

AR(2) is stationary when the roots of the characteristic equation lies outside the unit circle, that's $\mid y \mid > 1$, that's when the modulus of the roots of the characteristic equation greater than 1.

Let $\lambda = y^{-1}$. So

$$\lambda^2 - \phi_1 \lambda - \phi_2 = 0. \tag{2.19}$$

When $\mid y \mid > 1, then \mid \lambda \mid = \mid y^{-1} \mid < 1$.

So the root of (2.19) is $\lambda_{1,2} = \frac{\phi_1 \pm \sqrt{\phi_1^2 + 4\phi_2}}{2}$.

When AR(2) is stationary and λ_1 and λ_2 are real then

$\mid \frac{\phi_1 \pm \sqrt{\phi_1^2 + 4\phi_2}}{2} \mid < 1$.

$\Rightarrow -1 < \frac{\phi_1 \pm \sqrt{\phi_1^2 + 4\phi_2}}{2} < 1$.

$\Rightarrow -2 < \phi_1 \pm \sqrt{\phi_1^2 + 4\phi_2} < 2$.

So the larger bound of the roots is bounded by $\phi_1 + \sqrt{\phi_1^2 + 4\phi_2} < 2$.

$\Rightarrow \sqrt{\phi_1^2 + 4\phi_2} < 2 - \phi_1$.

$\Rightarrow \phi_1^2 + 4\phi_2 < 2 - {\phi_1}^2$.

$\Rightarrow \phi_1^2 + 4\phi_2 < 4 - 4\phi_1 + \phi_1^2$.

$\Rightarrow \phi_2 < 1 - \phi_1$ that's $\phi_2 + \phi_1 < 1$.

When λ_1 and λ_2 are complex, then $\sqrt{\phi_1^2 + 4\phi_2} < 0$ and

$$\lambda_{1,2} = \frac{\phi_1}{2} \pm i\frac{\sqrt{-\phi_1^2 - 4\phi_2}}{2}$$

$$\lambda_{1,2}^2 = \frac{\phi_1}{2}^2 \pm i\frac{\sqrt{-\phi_1^2 - 4\phi_2}}{2}^2.$$
$$= \frac{{\phi_1}^2 - {\phi_1}^2 - 4\phi_2}{4} = -\phi_2.$$

But AR(2) is stationary, so $| \lambda |< 1$ that's $-\phi_2 < 1$, hence $\phi_2 > -1$.

We have also $\phi_2 < 1 - \phi_1$ and $\phi_2 < 1 + \phi_1$, so $\phi_2^2 < 1$, so $\phi_2 < 1$, also $\phi_2 > -1$, so $| \phi_2 |< 1$.

The smaller bound of the roots is bounded by $\phi_1 - \sqrt{\phi_1^2 + 4\phi_2} > -2$.

$\Rightarrow -\sqrt{\phi_1^2 + 4\phi_2} > -2 - \phi_1$.

$\Rightarrow \sqrt{\phi_1^2 + 4\phi_2} < 2 + \phi_1$.

$\Rightarrow \phi_1^2 + 4\phi_2 < 2 + {\phi_1}^2$.

$\Rightarrow \phi_1^2 + 4\phi_2 < 4 + 4\phi_1 + \phi_1^2$.

$\Rightarrow \phi_2 < 1 + \phi_1$ that's $\phi_2 - \phi_1 < 1$.

So AR(1) is stationary if and if $| \phi_2 |< 1$, $\phi_1 + \phi_2 < 1$,

$\phi_2 - \phi_1 < 1$. [5] □

remark 2.2.6. *The AR process is stationary when the roots of the characteristic equation fall outside the unit circle.[3]*

Proof. The AR equation written as

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t.$$

That's

$$y_t = \sum_{j=1}^{p} \phi_j y_{t-j} + \epsilon_t.$$

Or

$$\phi(B)(y_t - \mu) = \epsilon_t.$$

When $\mu = 0$ then $y_t = \phi^{-1}(B)\epsilon_t \equiv \psi(B).\epsilon_t = \sum_{j=0}^{\infty} \psi_j \epsilon_{t-j}$, for some function ψ given the right side is convergent,

Let $\phi^{-1}(B) = \prod_{i=1}^{p}(1 - G_i B)$.

Where $G_1^{-1}, G_2^{-1}, ..., G_p^{-1}$ are the roots of $\phi(B) = 0$.

Expanding $\phi^{-1}(B)$ in partial fractions results

$y_t = \phi^{-1}(B)\epsilon_t = \sum_{i=0}^{p} \frac{K_i \epsilon_t}{1 - G_i B}$, where K_i is a constant $\forall i \in \mathbb{N}$.

When AR(p) is stationary,

then $\psi(B) = \phi^{-1}(B)$ is convergent series for $\mid B \mid < 1$.

That's $\psi_j = \sum_{i=1}^{p} K_i G_i^j$ are absolutely summable,

So $\mid G_i \mid < 1$, for i=1, 2, ..., p.

Hence the roots of $\phi(B) = 0$ must lie outside the unit circle.

So AR(p) is stationary when the roots of the characteristic equation fall outside the unit circle.[3] $\square$

The autocorrelation of AR(p) at k=0:

Consider the AR(p) process:

$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t$.

But the autocorrelation of AR(p) at k=0 is $\gamma_0 = var(Y_t)$, and $var(\epsilon_t) =$

${\sigma_\epsilon}^2$, when k=0, $\gamma_0 = \phi_1\gamma_{-1} + \phi_2\gamma_{-2} + ... + \phi_p\gamma_{-p} + {\sigma_\epsilon}^2$

substituting $\gamma_{-k} = \gamma_k$ for k=1, 2, ..., p,$\gamma_k = \gamma_0\rho_k$, we get

$\gamma_0 = \rho_1\phi_1\gamma_0 + \phi_2\rho_2\gamma_0 + ... + \phi_p\rho_p\gamma_0 + {\sigma_\epsilon}^2$, so

$\gamma_0 = \gamma_0(\phi_1\rho_1 + \phi_2\rho_2 + ... + \phi_p\rho_p) + {\sigma_\epsilon}^2$.

So $\gamma_0((1 - \phi_1\rho_1 - \phi_2\rho_2 - ... - \phi_p\rho_p) = {\sigma_\epsilon}^2$. So

$$\gamma_0 = \frac{{\sigma_\epsilon}^2}{(1 - \phi_1\rho_1 - \phi_2\rho_2 - ... - \phi_p\rho_p)}. \tag{2.20}$$

remark 2.2.7. *The ACF for the AR(p) process forms damped exponential decays to 0 as k increases. [3]*

Proof. Multiply (2.8) by $y_{t-k} - \mu$ to get $(y_{t-k} - \mu)(y_t - \mu) =$

$$(y_{t-k}-\mu)\phi_1(y_{t-1}-\mu)+(y_{t-k}-\mu)\phi_2(y_{t-2}-\mu)+...+(y_{t-k}-\mu)\phi_p(y_{t-p}-\mu)+(y_{t-k}-\mu)\epsilon_t. \tag{2.21}$$

By taking expected values for (2.21), we get the difference equation for the autocovariance function the AR(p) process is given by

$$\gamma_k = \phi_1\gamma_{k-1} + \phi_2\gamma_{k-2} + ... + \phi_p\gamma_{k-p}, \qquad \forall k > 0. \tag{2.22}$$

Divide (2.22) by γ_0 to get the ACF for the AR(p) process is given by

$$\rho_k = \phi_1\rho_{k-1} + \phi_2\rho_{k-2} + ... + \phi_p\rho_{k-p}, \qquad \forall k > 0. \tag{2.23}$$

Let B be a shift operater that's given by

$$\rho_t B = \rho_{t-1}. \tag{2.24}$$

So

$$(1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^P)\rho_k = \phi(B)\rho_k = 0, \qquad \forall k > 0. \quad (2.25)$$

The general solution for the difference equation is given by [15]

$$\rho_k = A_1 {G_1}^k + A_2 {G_2}^k + ... + A_p {G_p}^k, \qquad \forall k > 0. \quad (2.26)$$

Where ${G_1}^{-1}, {G_2}^{-1}, ..., {G_p}^{-1}$ are distinct roots of the characteristic equation $\phi(B) = 0$, and A_i's are constants.
If a root ${G_i}^{-1}$'s is real then $\mid {G_i}^{-1} \mid > 1$ due to the stationary conditions.
So $\mid G_i \mid < 1$ and $A_i {G_i}^k$ forms a damped exponential which geometrically decays to 0 as k increases.
Complex roots forms a damped sine wave to ACF.
So ACF for a stationary AR process will consists of a combination of damped exponential and damped sine waves.[3] □

2.3 Moving average Model

The Moving average model MA(q) [25] is a linear stationary time series model that's written as a function of previous values of the white noise and the mean of the previous values of the series $(y_t)_{t=0}^{\infty}$, where

MA(q) is of order q (q positive integer number) is given by

$$y_t - \mu = \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - ... - \theta_q\epsilon_{t-q}. \tag{2.27}$$

where θ_i's are regression parameter $\forall i \in \mathbb{N}$, ϵ is a white noise.

Remember that the moving average process is an autoregression model of the time series of residual errors from prior predictions. Also the moving average model corrects future forecasts based on errors made on recent forecasts.

Let B be a shift operator that's given by

$$(\epsilon_t)B = \epsilon_{t-1}. \tag{2.28}$$

Substitute (2.28) into (2.27), we get

$$\begin{aligned} y_t - \mu &= \epsilon_t - \theta_1 B\epsilon_t - \theta_2 B^2\epsilon_{t-2} - ... - \theta_q B^q\epsilon_{t-q}. \\ &= (1 - \theta_1 B - \theta_2 B^2 - ... - \theta_q B^q)\epsilon_t. \end{aligned} \tag{2.29}$$

Or

$$\theta(B)\epsilon_t = y_t - \mu. \tag{2.30}$$

Where

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - ... - \theta_q B^q. \tag{2.31}$$

is the nonseasonal MA operator of order q.

remark 2.3.1. *For q =1, MA(1) is written as* $y_t - \mu = \epsilon_t - \theta_1 \epsilon_{t-1}$*, then the ACF is given by*

$$\rho_k = \begin{cases} 1 & k = 0 \\ \frac{-\theta_1}{1+{\theta_1}^2} & k = 1 \\ 0 & k > 1 \end{cases} \tag{2.32}$$

Proof. So $E(y_t) = E(\epsilon_t - \theta_1\epsilon_{t-1}) + \mu = \mu$, since ϵ_t is white noise i.e. $E(\epsilon_t) = 0, \forall t$.

$$\begin{aligned} var(y_t) = \gamma_0 &= var(\epsilon_t - \theta_1\epsilon_{t-1}). \\ &= var(\epsilon_t) - var(\theta_1\epsilon_{t-1}). \\ &= \sigma^2 + {\theta_1}^2 var(\epsilon_{t-1}) - 2\theta_1 cov(\epsilon_t, \epsilon_{t-1}). \\ &= \sigma^2 + {\theta_1}^2\sigma^2. \\ &= \sigma^2(1 + {\theta_1}^2). \end{aligned}$$

The autocovariance at k=1 is given by

$$\begin{aligned} &\gamma_1 = cov(y_t, y_{t-1}). \\ &\quad = cov(\epsilon_t - \theta_1\epsilon_{t-1}, \epsilon_{t-1} - \theta_1\epsilon_{t-2}). \\ &\quad = cov(\epsilon_t, \epsilon_{t-1}) - \theta_1 cov(\epsilon_t, \epsilon_{t-2}) - \theta_1 cov(\epsilon_{t-1}, \epsilon_{t-1}) + {\theta_1}^2 cov(\epsilon_{t-1}, \epsilon_{t-2}). \end{aligned}$$

But $cov(\epsilon_t, \epsilon_{t-k}) = 0$.

$So\ \ \gamma_1 = -\theta_1 var(\epsilon_{t-1}) = -\theta_1\sigma^2.$

for $k > 1, \gamma_k = cov(y_t, y_{t-k}) = 0.$

$$So\ \ \gamma_k = \begin{cases} \sigma^2(1+{\theta_1}^2) & k=0 \\ -\theta_1\sigma^2 & k=1 \\ 0 & k>1 \end{cases}$$

$$The\ ACF\ \ is\ \ \rho_k = \frac{\gamma_k}{\gamma_0} \begin{cases} 1 & k=0 \\ \frac{-\theta_1}{1+{\theta_1}^2} & k=1 \\ 0 & k>1 \end{cases}$$ □

Theorem 2.3.1. *The MA process is stationary .*

Proof. The y_t's are finite linear combination of the previous values of the white noise, and the white noise process is stationary.

So the MA process is stationary whatever were the values of the MA parameters. □

example 2.3.1. *We draw some different MA(1) processes,*

$y_t = \theta\epsilon_{t-1} + \epsilon_t$*, with n= 15 and five different values for* θ *and notice that:*

MA(1) is stationary for all values of θ*.*

remark 2.3.2. *When* $\theta_1 = 0$*, MA(1) is white noise.*

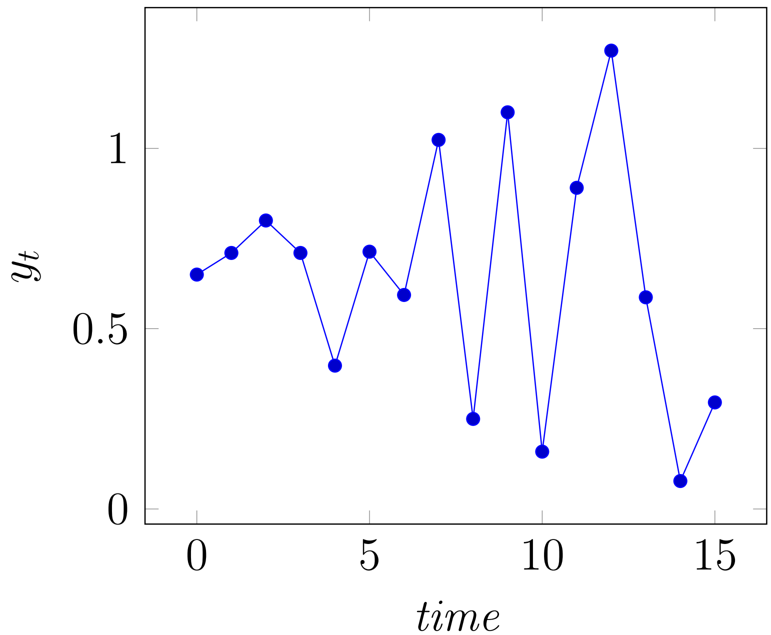

Figure 2.9: MA(1) process when $\theta = 0.8$

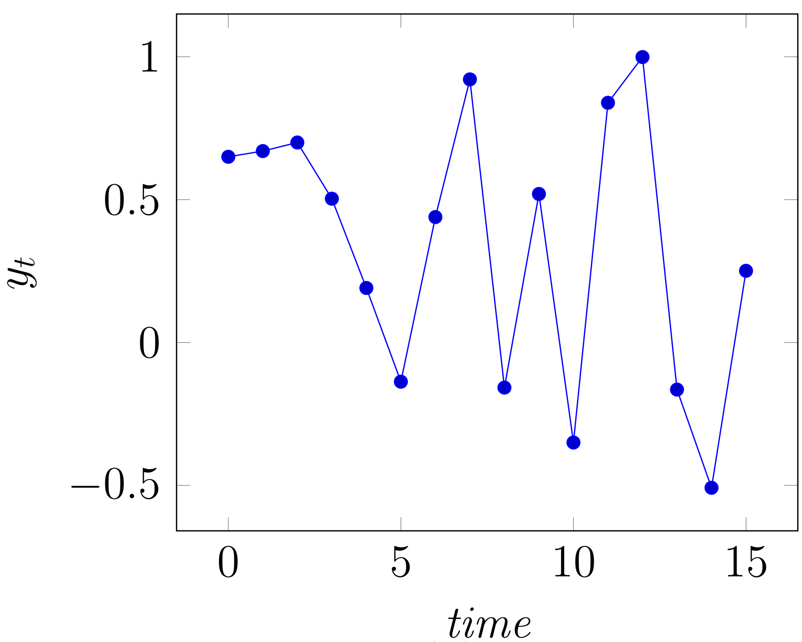

Figure 2.10: MA(1) process when $\theta = 1.6$

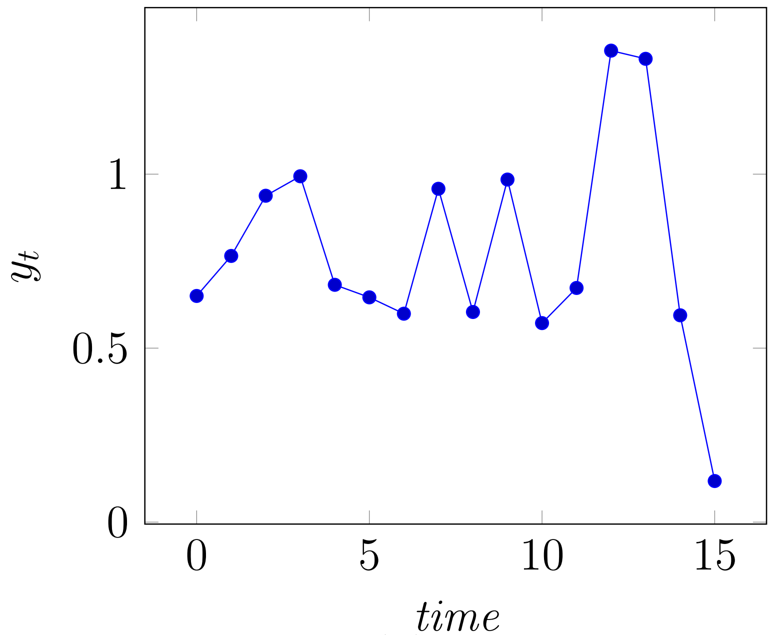

Figure 2.11: MA(1) process when $\theta = -0.3$

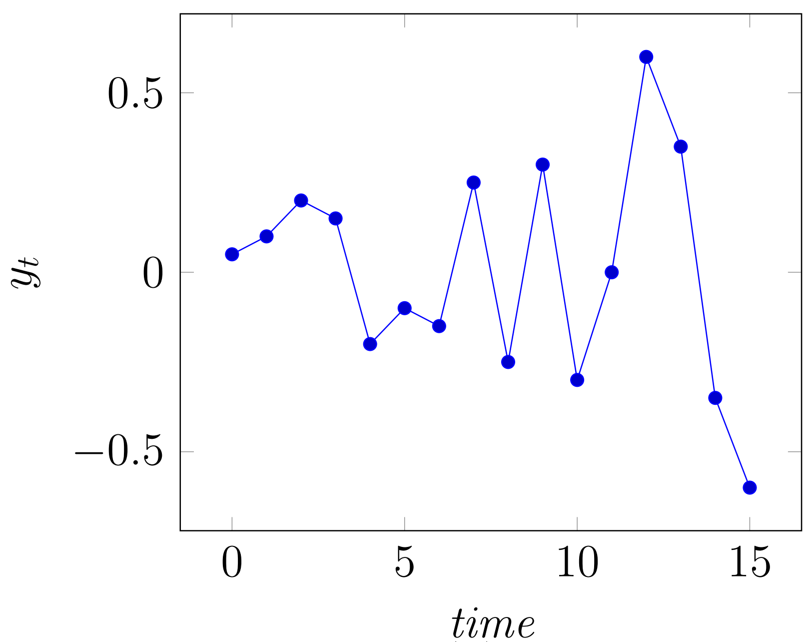

Figure 2.12: MA(1) process when $\theta = 0$

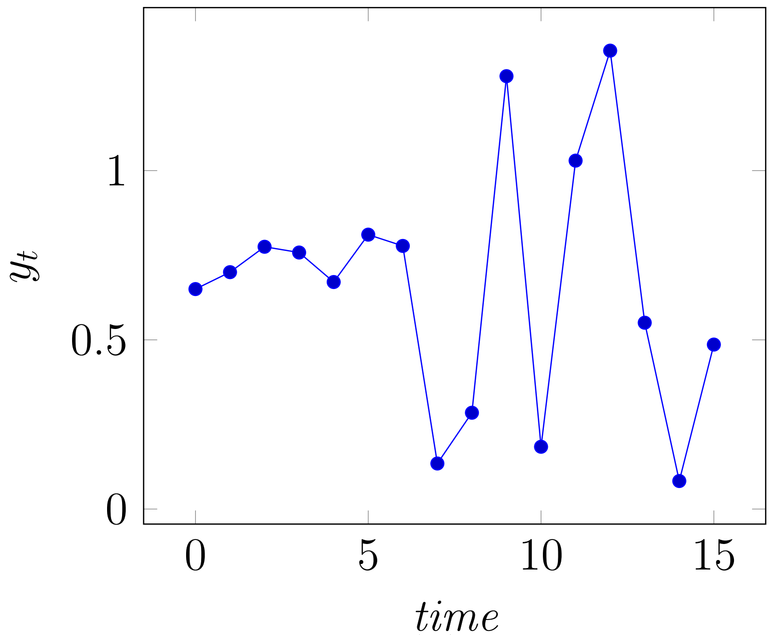

Figure 2.13: MA(1) process when $\theta = 1$

Definition 6. *The equation*

$$\theta(B) = 0 \tag{2.33}$$

is the characteristic equation for the MA process.

remark 2.3.3. *The MA process is invertible when the roots of the characteristic equation fall outside the unit circle. [3]*

Proof. The MA equation written as

$$y_t - \mu = \epsilon_t - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2} - ... - \theta_q \epsilon_{t-q},$$

that's

$$y_t = \mu + \epsilon_t - \sum_{j=1}^{q} \theta_j \epsilon_{t-j}.$$

or

$$\theta(B)\epsilon_t = (y_t - \mu).$$

When $\mu = 0$ then $\epsilon_t = \theta^{-1}(B)y_t \equiv \pi(B)y_t$, for some function π.

Let $\theta(B) = \prod_{i=1}^{q}(1 - H_i B)$.

Where $H_1^{-1}, H_2^{-1}, ..., H_q^{-1}$ are the roots of $\theta(B) = 0$,

expanding $\theta^{-1}(B)$ in partial fractions results,

$\pi(B) = \theta^{-1}(B) = \sum_{i=1}^{q} \frac{M_i}{1 - H_i B}$.

When MA(q) is stationary,

then $\pi(B) = \theta^{-1}(B)$ is convergent series,

that's $\pi_j = -\sum_{i=1}^{q} M_i H_i^j$ are absolutely summable,

so $\mid H_i \mid < 1$, for i=1, 2, ..., q.

Hence the roots of $\theta(B) = 0$ must lie outside the unit circle.

So MA(q) is stationary when the roots of the characteristic equation fall outside the unit circle.[3] $\square$

remark 2.3.4. *$MA(\infty) = AR(1)$.*

Proof. The AR(1) process is given by

$$y_t = \phi_1 y_{t-1} + \epsilon_t. \tag{2.34}$$

Similarly

$$y_{t-1} = \phi_1 y_{t-2} + \epsilon_{t-1}. \tag{2.35}$$

Then we substitute y_{t-1} from (2.35) into (2.34) to get

$$
\begin{aligned}
y_t &= \phi_1(\phi_1 y_{t-2} + \epsilon_{t-1}) + \epsilon_t. \\
&= {\phi_1}^2 y_{t-2} + \epsilon_{t-1}\phi_1 + \epsilon_t.
\end{aligned} \tag{2.36}
$$

Similarly substitute $y_{t-2} = \phi_1 y_{t-3} + \epsilon_{t-2}$ into (2.36) to get

$$
\begin{aligned}
y_t &= {\phi_1}^2(\phi_1 y_{t-3} + \epsilon_{t-2}) + \epsilon_{t-1}\phi_1 + \epsilon_t. \\
&= {\phi_1}^3 y_{t-3} + {\phi_1}^2\epsilon_{t-2} + \epsilon_{t-1}\phi_1 + \epsilon_t.
\end{aligned}
$$

Continuing this type of substitution indefinitely, we get

$y_t = \epsilon_t + \epsilon_{t-1}\phi_1 + {\phi_1}^2\epsilon_{t-2} + {\phi_1}^3\epsilon_{t-3} + ...$.

Let $\theta_i = {\phi_1}^i, \quad \forall i \in \mathbb{N}$, we get

$y_t = \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + \theta_3\epsilon_{t-3} + ...$.

Which is an equation for $MA(\infty)$ so $AR(1) = MA(\infty)$. $\square$

remark 2.3.5. *The* $AR(\infty) = MA(1)$.

Proof. The MA(1) process is given by

$$y_t = \epsilon_t - \theta_1\epsilon_{t-1}, \tag{2.37}$$

that's

$$\epsilon_t = y_t + \theta_1\epsilon_{t-1}. \tag{2.38}$$

Similarly

$$\epsilon_{t-1} = y_{t-1} + \theta_1\epsilon_{t-2}. \tag{2.39}$$

Then we substitute ϵ_{t-1} from (2.39) into (2.38) to get

$$\begin{aligned}\epsilon_t &= y_t + \theta_1(y_{t-1} + \theta_1\epsilon_{t-2}). \\ &= y_t + \theta_1 y_{t-1} + {\theta_1}^2\epsilon_{t-2}.\end{aligned} \tag{2.40}$$

Similarly substitute $\epsilon_{t-2} = y_{t-2} + \theta_1\epsilon_{t-3}$ into (2.40) to get

$$\begin{aligned}\epsilon_t &= y_t + \theta_1 y_{t-1} + {\theta_1}^2(y_{t-2} + \theta_1\epsilon_{t-3}). \\ &= y_t + \theta_1 y_{t-1} + {\theta_1}^2 y_{t-2} + {\theta_1}^3\epsilon_{t-3}.\end{aligned}$$

Continuing this type of substitution indefinitely, we get

$$\epsilon_t = y_t + \theta_1 y_{t-1} + {\theta_1}^2 y_{t-2} + {\theta_1}^3 y_{t-3} + \dots \,.$$

Let $\phi_i = -{\theta_1}^i, \quad \forall i \in \mathbb{N}$, we get

$$y_t = \epsilon_t + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots \,.$$

which is an equation for$AR(\infty)$ so $MA(1) = AR(\infty)$. $\square$

The difference equation for the autocovariance function for the MA(q) process is given by

$$\gamma_k = E[(y_{t-k} - \mu)(y_t - \mu)]. \tag{2.41}$$

By using (2.27), we get

$\gamma_k = E[(\epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - \dots - \theta_q\epsilon_{t-q})(\epsilon_{t-k} - \theta_1\epsilon_{t-k-1} - \theta_2\epsilon_{t-k-2} - \dots - \theta_q\epsilon_{t-k-q})].$

After multiplication and taking expected values for the last equation, the autocovariance function is given by

$$\gamma_k = \begin{cases} (-\theta_k + \theta_1\theta_{k+1} + \theta_2\theta_{k+2} + ... + \theta_{q-k}\theta_q){\sigma_\epsilon}^2 & k = 1, 2, ..., q-1 \\ (-\theta_q){\sigma_\epsilon}^2 & k = q \\ 0 & k > q \end{cases} \tag{2.42}$$

Where $\theta_0 = 1$ and $\theta_{-k} = 0$, $\forall k > 0$. When k= 0 in (2.41) then the variance is given by

$$\gamma_0 = (1 + {\theta_1}^2 + {\theta_2}^2 + ... + {\theta_k}^2){\sigma_\epsilon}^2. \tag{2.43}$$

By dividing the autocovariance function by the variance, we get ACF for MA(q) process is given by

$$\rho_k = \begin{cases} 1 & k = 0 \\ \frac{-\theta_k + \theta_1\theta_{k+1} + \theta_2\theta_{k+2} + ... + \theta_{q-k}\theta_q}{1 + {\theta_1}^2 + {\theta_2}^2 + ... + {\theta_k}^2} & k = 1, 2, ..., q-1 \\ \frac{-\theta_q}{1 + {\theta_1}^2 + {\theta_2}^2 + ... + {\theta_k}^2} & k = q \\ 0 & k > q \end{cases} \tag{2.44}$$

We see that the ACF of MA(q) cuts off after lag q.

example 2.3.2. *We draw the the ACF of different MA(1) processes and notice that: ACF of MA(1) process cuts off after lag 1 in all values of* θ.

remark 2.3.6. *When k=1,*

$$\rho_1 = \begin{cases} 1 & k=0 \\ \frac{-\theta_1}{1+{\theta_1}^2} & k=1 \\ 0 & k>1 \end{cases} \tag{2.45}$$

when $\theta_1 = 0$, MA(1) process became a white noise.

As θ_1 ranges from -1 to 1, the population lag 1 autocorrelation ρ_1 ranges from the largest $\rho_1 = 0.5$ to the smallest $\rho_1 = -0.5$.

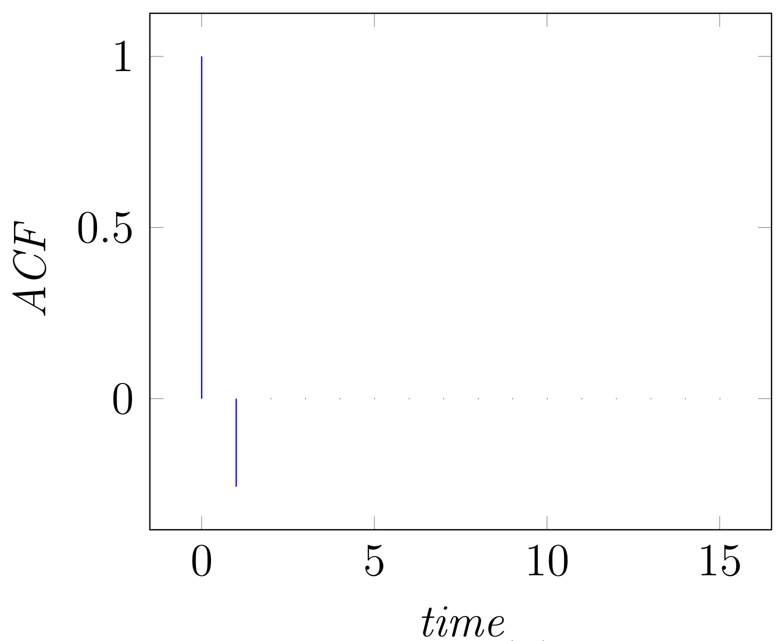

Figure 2.14: ACF of MA(1) model when $\theta = .3$

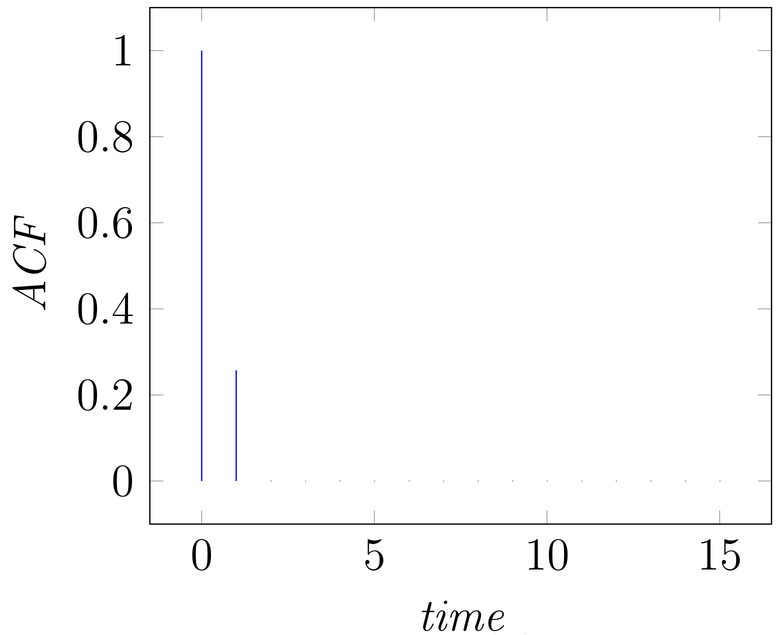

Figure 2.15: ACF of MA(1) model when $\theta = -.3$

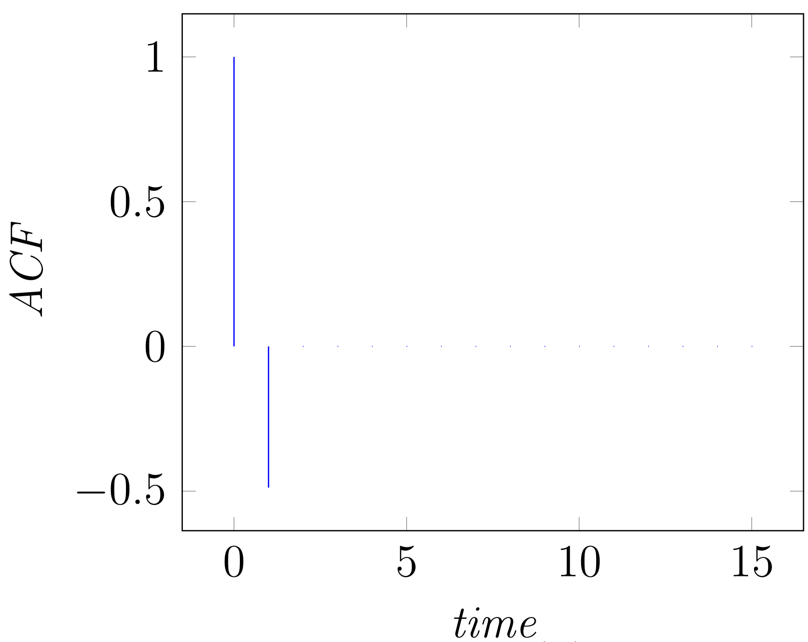

Figure 2.16: ACF of MA(1) model when $\theta = .8$

2.4 Autoregressive Moving average Model

It consists of both of AR(p) and MA(q) parameters, p, q positive integer numbers. It's given by [25]

$$y_t - \mu - \phi_1(y_{t-1} - \mu) + \phi_2(y_{t-2} - \mu) + ... + \phi_p(y_{t-p} - \mu) =$$
$$\epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - ... - \theta_q\epsilon_{t-q}. \tag{2.46}$$

where θ_i are regression parameter for MA process, Where ϕ_i are regression parameter for MA process, $\forall i \in \mathbb{N}$, ϵ_t is a white noise.

$$y_t = \alpha + \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + ... + \theta_q\epsilon_{t-q} + \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p}. \tag{2.47}$$

Where $\alpha = \mu(1 - \phi_1 - \phi_2 - ... - \phi_p)$.

Let B be a shift operater that's given by

$$y_t B = y_{t-1}. \tag{2.48}$$

And

$$(\epsilon_t)B = \epsilon_{t-1}. \tag{2.49}$$

Then

$$(1 - \theta_1 B - \theta_2 B^2 - ... - \theta_q B^q)\epsilon_t = (y_t - \mu)(1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^P). \tag{2.50}$$

Or

$$\phi(B)(y_t - \mu) = \theta(B)\epsilon_t. \tag{2.51}$$

Where

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - ... - \theta_q B^q. \tag{2.52}$$

is the nonseasonal MA operator of order q. Where

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p. \tag{2.53}$$

is the nonseasonal AR operator of order p.

ARMA(p,q) can be written as a p^{th} order AR process is given by

$$\phi(B)(y_t - \mu) = e_t, \tag{2.54}$$

where e_t follows the q^{th} order MA process

$$e_t = \theta(B)\epsilon_t. \tag{2.55}$$

Also ARMA(p,q) can be written as a q^{th} order MA process is given by

$$(y_t - \mu) = \theta(B)b_t.$$

Where b_t follows the p^{th} order AR process.

$$b_t\phi(B) = \epsilon_t.$$

By substitute e_t from (2.55) into equation(2.54) results

$$\phi(B)(y_t - \mu) = \theta(B)\epsilon_t. \tag{2.56}$$

The ARMA(p,q) process contains both the pure AR and MA processes as subsets,

So AR(p)=ARMA(p,0), Ma(q)=ARMA(0,q) and ARMA(0,0) is the white noise ϵ_t.

The ARMA(1, 1) process

Substitute p=q=1, $y_t - \mu = y_t$ in (2.47) results

$$y_t = \phi y_{t-1} + \epsilon_t - \theta\epsilon_{t-1} \tag{2.57}$$

is called ARMA(1, 1) process can be written as

$(1 - \phi B)y_t = (1 - \theta B)\epsilon_t$

Where B be a shift operator that's $\epsilon_t B = \epsilon_{t-1}$,

but $var(y_t) = \gamma_0$ and $var(\epsilon_t) = var(\epsilon_{t-1}) = {\sigma_\epsilon}^2$,

take variances of both sides of (2.57) results

$\gamma_0 = \phi\gamma_{-1} + {\sigma_\epsilon}^2 + \theta^2{\sigma_\epsilon}^2 - \phi\theta{\sigma_\epsilon}^2$

but $\gamma_{-1} = \gamma_1 = \gamma_0\phi - \theta{\sigma_\epsilon}^2$,

so $\gamma_0 = \phi^2\gamma_0 - 2\phi\theta{\sigma_\epsilon}^2 + {\sigma_\epsilon}^2 + \theta^2{\sigma_\epsilon}^2$, so

$$\gamma_0 = {\sigma_\epsilon}^2\left(\frac{1 - 2\phi\theta + \theta^2}{1 - \phi^2}\right) \tag{2.58}$$

but $\gamma_k = \phi\gamma_{k-1}$, and by dividing on γ_0, we get

$\rho_k = \phi\rho_{k-1}$, then $\rho_k = \phi^{k-1}\rho_1$

but $\rho_1 = \frac{\gamma_1}{\gamma_0} = \frac{\gamma_0\phi - \theta{\sigma_\epsilon}^2}{\gamma_0}$, so

$$\begin{aligned}\rho_1 &= \frac{{\sigma_\epsilon}^2\left(\frac{1-2\phi\theta+\theta^2}{1-\phi^2}\right)\phi - \theta{\sigma_\epsilon}^2}{{\sigma_\epsilon}^2\left(\frac{1-2\phi\theta+\theta^2}{1-\phi^2}\right)} \\ &= \frac{(1-\phi^2)(\left(\frac{1-2\phi\theta+\theta^2}{1-\phi^2}\right)\phi - \theta)}{1-2\phi\theta+\theta^2} \\ &= \frac{(1-2\phi\theta+\theta^2)\phi - \theta(1-\phi^2))}{1-2\phi\theta+\theta^2} \\ &= \frac{(1-\phi\theta)(\phi-\theta)}{1-2\phi\theta+\theta^2}\end{aligned}$$

so

$$\rho_k = \phi^{k-1}\rho_1 = \frac{(1-\phi\theta)(\phi-\theta)}{1-2\phi\theta+\theta^2}\phi^{k-1}[4] \tag{2.59}$$

remark 2.4.1. *ACF for AR(p) is the same as ACF for ARMA(p,q) for* $k > q - p$.[3]

Proof. Multiply both sides of (2.46) by y_{t-k} and take the expected value of the result to get

$$\gamma_k - \phi_1\gamma_{k-1} - \phi_2\gamma_{k-2} - ... - \phi_p\gamma_{k-p} =$$

$$\gamma_{2\epsilon(k)} - \theta_1\gamma_{2\epsilon(k-1)} - \theta_2\gamma_{2\epsilon(k-2)} - ... - \theta_q\gamma_{2\epsilon(k-q)} \tag{2.60}$$

$\gamma_{2\epsilon}(k) = E[(y_{t-k} - \mu)]$ is the cross autocovariance function between y_{t-k} and ϵ_t.

Since y_{t-k} is dependent only upon the shocks occurred up to time t-k so

$$\gamma_{2\epsilon}(k) = \begin{cases} 0 & k < t \\ nonzero & k \geq t \end{cases} \tag{2.61}$$

multiply both sides of(2.47) by ϵ_{t-k} and take the expected value of the result to get

$$\gamma_{2\epsilon}(-k)-\phi_1\gamma_{2\epsilon}(-k+1)-\phi_2\gamma_{2\epsilon}(-k+2)-...-\phi_p\gamma_{2\epsilon}(-k+p) = -[\theta_k]{\sigma_\epsilon}^2. \tag{2.62}$$

where

$$[\theta_k] = \begin{cases} \theta_k & k = 1, 2, .., q \\ -1 & k = 0 \\ 0 & otherwise \end{cases} \tag{2.63}$$

solving (2.60) and (2.62) for γ_k, we get

$$\gamma_k - \phi_1\gamma_{k-1} - \phi_2\gamma_{k-2} - ... - \phi_p\gamma_{k-p} = 0 \tag{2.64}$$

or

$$\phi(B)\gamma_k = 0, \tag{2.65}$$

dividing (2.65) by γ_0 we get

$$(1 - \phi_1 B - \phi_2 B^2 - ... - .\phi_p B^P)\rho_k = \phi(B)\gamma_k = 0 \qquad k > q. \tag{2.66}$$

(2.66) is identical to (2.25).

That's ACF for AR(p) is the same as ACF for ARMA(p,q) for $k >$

$q - p$.[3] $\square$

If $k \leq q$ then ρ_k is a function of both of the AR(p) and MA(q)parameters.

$$\phi(B)(y_t - \mu) = \theta(B)\epsilon_t. \tag{2.67}$$

so

$$\theta(B)^{-1}\phi(B)(y_t - \mu) = \epsilon_t, \tag{2.68}$$

where $\theta(B)^{-1}$ is infinite series in B.

2.5 ARIMA model(p,q,d)

Many time series are non stationary, so we use differencing to make them stationary.

The original undifferenced series is called integrated. That's ARIMA model [25] is the integrated time series that we make differences for it to become stationary time series.

Autoregressive Integrated Moving-Average model is a generalization of ARMA model, called ARIMA(p, q, d), where d is the order of differencing (the number of times needed to change the ARIMA(p, q, d) to stationary time series), it's a non-stationary linear time series model that's written by

$$(1-\phi_1 B-\phi_2 B^2-...-\phi_p B^p)(1-B)^d y_t=(1-\theta_1 B-\theta_2 B^2-...-\theta_q B^q)\epsilon_t. \tag{2.69}$$

So

$$\phi(B)(1-B)^d y_t=\theta(B)\epsilon_t. \tag{2.70}$$

Where

$$\theta(B)=1-\theta_1 B-\theta_2 B^2-...-\theta_q B^q$$

is the nonseasonal MA operator of order q. Where

$$\phi(B)=1-\phi_1 B-\phi_2 B^2-...-\phi_p B^p$$

is the nonseasonal AR operator of order p.

where B be a shift operater such that $y_t B=y_{t-1}$ and $\epsilon_t B=\epsilon_{t-1}$.

Where θ_i are regression parameter for MA(q) process, where ϕ_i are regression parameter fo AR(p) process, $\forall i \in \mathbb{N}$, ϵ_t is a white noise.

When $d>0$, the ACF of ARIMA process decays slowly since ARIMA is nonstationary.

ARIMA(p,q,0)=ARMA(p,q), ARIMA(0,1,0) is a random walk process. ARIMA(0,0,d) is the white noise ϵ_t, ARIMA(p,0,0)=AR(p), ARIMA(0,q,0)=MA(q).

Chapter 3

Parameter Estimation Approaches in Time Series Models

Frequently one wishes to fit a parametric model to time-series data and determine accurate values of the parameters and reliable estimates for the uncertainties in those parameters. It is important to gain a thorough understanding of the noise and develop appropriate methods for parameter estimation, where the most interesting effects are often on the edge of detectability. Underestimating the errors leads to unjustified confidence in new results, or confusion over apparent contradictions between different data sets. Overestimating the errors inhibits potentially important discoveries. In this chapter different approaches of parameter estimation for time series models will be presented.

3.1 Yule Walker equations

Substitute k=1, 2, ..., p into (2.22), parameters $\phi_i's$ can be estimated by the theoretical values of the ACF, results the linear equations that are called the Yule Walker equations [32] are given by

$$\begin{array}{ccccccccc} \rho_1 & = & \phi_1 & + & \phi_2\rho_1 & + & ... & + & \phi_p\rho_{p-1} \\ \rho_2 & = & \phi_1\rho_{k-1} & + & \phi_2 & + & ... & + & \phi_p\rho_{p-2} \\ . & & . & & . & & ... & & . \\ . & & . & & . & & ... & & . \\ \rho_p & = & \phi_1\rho_{p-1} & + & \phi_2\rho_{p-2} & + & ... & + & \phi_p \end{array} \tag{3.1}$$

Write the Yule Walker equations in the matrix form, we get

$$\phi = {P_p}^{-1}\rho \tag{3.2}$$

where

$$\phi = \begin{bmatrix} \phi_1 \\ \phi_2 \\ . \\ . \\ . \\ \phi_p \end{bmatrix}, \rho = \begin{bmatrix} \rho_1 \\ \rho_2 \\ . \\ . \\ . \\ \rho_p \end{bmatrix}, P_p = \begin{bmatrix} 1 & \rho_1 & \rho_2 & \cdots & \rho_{p-1} \\ \rho_1 & 1 & \rho_1 & \cdots & \rho_{p-2} \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ \rho_{p-1} & \rho_{p-2} & \rho_{p-3} & \cdots & 1 \end{bmatrix} \tag{3.3}$$

Where P_p is an invertible matrix .

To find Yule Walker estimates for the AR parameters, replace the $\rho_k's$ in (3.3) by their estimates $\rho_k, k = 1, 2, ..., p$ in (3.1).

For AR(1), substitute p=1 in (3.1), we get

$$\begin{aligned} \rho_1 &= \phi_1 \\ \rho_2 &= \phi_1\rho_1 = {\phi_1}^2 \\ \rho_3 &= \phi_1\rho_2 = {\phi_1}^3 \end{aligned}$$

In general

$$\rho_k = {\phi_1}^k \tag{3.4}$$

For AR(2), substitute p=2 in (3.1), we get

$$\begin{aligned} \rho_1 &= \phi_1 + \phi_2\rho_1 \\ \rho_2 &= \phi_1\rho_1 + \phi_2 \end{aligned} \tag{3.5}$$

Where $\rho_0 = 1, \rho_k = \rho_{-k}$, k=1, 2, ..., p.

In general

$$\rho_k = \phi_1\rho_{k-1} + \phi_2\rho_{k-2} \tag{3.6}$$

Solve the system in (3.5) for ρ_1, ρ_2, we get

$$\begin{aligned} \rho_1 &= \frac{\phi_1}{1-\phi_2} \\ \rho_2 &= \frac{{\phi_1}^2+\phi_2-{\phi_2}^2}{1-\phi_2} \end{aligned} \tag{3.7}$$

We can find higher lag autocorrelation by using the recursive relation (3.6).

For example

$$\rho_3 = \phi_1\rho_2 + \phi_2\rho_1 \tag{3.8}$$

Substitute ρ_1, ρ_2 from (3.7) into (3.8) results

$$\begin{aligned} \rho_3 &= \phi_1 \frac{{\phi_1}^2 + \phi_2 - {\phi_2}^2}{1-\phi_2} + \phi_2 \frac{\phi_1}{1-\phi_2} \\ &= \frac{{\phi_1}^3 + 2\phi_1\phi_2 - \phi_1{\phi_2}^2}{1-\phi_2} \end{aligned} \tag{3.9}$$

3.2 Method of moments

The method of moments (MOM)[5] is equating sample moments to the corresponding population moments expressed in the parameter of interest and solving the resulting system of equations for the model parameters.

The method of moments is the easiest but not the most efficient for parameter estimation.

Method of moments for the AR(p) models

AR(1) formula that's given by $y_t = \phi y_{t-1} + \epsilon_t$.

In AR(1), we want to estimate the parameter ϕ, the population lag one correlation ρ_1= the sample lag 1 autocorelation that's given by

$$r_1 = \frac{\sum_{t=1}^{n-k}(X_t - \bar{X})(X_{t+k} - \bar{X})}{\sum_{t=1}^{n-k}(X_t - \bar{X})^2}.$$

Let k=1 in (2.17), then we have the MOM estimator of ϕ is $\hat{\phi} = r_1$.

AR(2) formula that's given by $y_t = \phi_1 y_{t-1} + \phi_2 \epsilon_{t-2} + \epsilon_t$, we want to estimate the parameters ϕ_1, and ϕ_2, recall the Yule Walker equations in (3.5), let $\rho_1 = r_1$, and $\rho_2 = r_2$, so we get

$$\begin{aligned} r_1 &= \phi_1 + \phi_2\rho_1 \\ r_2 &= \phi_1\rho_1 + \phi_2 \end{aligned} \tag{3.10}$$

Solving the system (3.10) for ϕ_1 and ϕ_2, we get the MOM estimators

$$\begin{aligned} \hat{\phi}_1 &= \tfrac{r_1(1-r_2)}{1-{r_2}^2} \\ \hat{\phi}_2 &= \tfrac{r_2-{r_1}^2}{1-{r_2}^2} \end{aligned} \tag{3.11}$$

AR(p) formula that's given by $y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + ... + \phi_p y_{t-p} + \epsilon_t$ we recall the Yule Walker equations in (3.1), let $\rho_1 = r_1, \rho_2 = r_2, ..., \rho_p =$

r_p, so we get

$$\begin{array}{ccccccccc} r_1 & = & \phi_1 & + & \phi_2\rho_1 & + & ... & + & \phi_p\rho_{p-1} \\ r_2 & = & \phi_1\rho_{k-1} & + & \phi_2 & + & ... & + & \phi_p\rho_{p-2} \\ . & & . & & . & & ... & & . \\ . & & . & & . & & ... & & . \\ r_p & = & \phi_1\rho_{p-1} & + & \phi_2\rho_{p-2} & + & ... & + & \phi_p \end{array} \tag{3.12}$$

Solving the system (3.12) using software, gives us the MOM estimators for $\phi_1, \phi_2, ..., \phi_p$.

Method of moments for the MA(q) models

MA(1) formula that's given by: $y_t = \epsilon_t - \theta_1\epsilon_{t-1}$.

In MA(1), we want to estimate the parameter θ.

But in (2.32), $\rho_1 = \frac{-\theta_1}{1+{\theta_1}^2}$.

Let $\rho_1 = r_1$, we get $r_1 = \frac{-\theta_1}{1+{\theta_1}^2}$, so

$$r_1\theta^2 + \theta + r_1 = 0 \tag{3.13}$$

Solve (3.13) for θ, we get

$\theta = \frac{-1\pm\sqrt{1-4{r_1}^2}}{2r_1}$.

Real solutions for θ exist when $1 - 4{r_1}^2 \geq 0$ that's $\mid r_1 \mid \leq 0.5$.

- If $\mid r_1 \mid > 0.5$, there's no real solution for θ.

- If $\mid r_1 \mid = 0.5$, then $\theta = \pm 1$, that's $\mid \theta \mid = 1$, so MA(1) model invertible.

- If $\mid r_1 \mid< 0.5$, the real solution for which MA(1) is invertible, so the MOM estimator for θ is $\hat{\theta} = \frac{-1\pm\sqrt{1-4{r_1}^2}}{2r_1}$.

MA(q) formula that's given by: $y_t - \mu = \epsilon_t - \theta_1\epsilon_{t-1} - \theta_2\epsilon_{t-2} - ... - \theta_q\epsilon_{t-q}$. But in (2.44),

$$\rho_k = \begin{cases} 1 & k = 0 \\ \frac{-\theta_k+\theta_1\theta_{k+1}+\theta_2\theta_{k+2}+...+\theta_{q-k}\theta_q}{1+{\theta_1}^2+{\theta_2}^2+...+{\theta_k}^2} & k = 1, 2, ..., q-1 \\ \frac{-\theta_q}{1+{\theta_1}^2+{\theta_2}^2+...+{\theta_k}^2} & k = q \\ 0 & k > q \end{cases}$$

Let $\rho_k = r_k$, for k=1, 2, ..., q, we get

$$r_k = \begin{cases} 1 & k = 0 \\ \frac{-\theta_k+\theta_1\theta_{k+1}+\theta_2\theta_{k+2}+...+\theta_{q-k}\theta_q}{1+{\theta_1}^2+{\theta_2}^2+...+{\theta_k}^2} & k = 1, 2, ..., q-1 \\ \frac{-\theta_q}{1+{\theta_1}^2+{\theta_2}^2+...+{\theta_k}^2} & k = q \\ 0 & k > q \end{cases} \tag{3.14}$$

Then the MOM estimators for θ are the solutions of (3.14) that can be solved by software.

Method of moments for the ARMA(1, 1) models

ARIMA(1, 1) formula is given by $y_t = \phi y_{t-1} + \epsilon_t - \theta\epsilon_{t-1}$, we want to estimate the parameters ϕ, and θ, but in (2.59),

$$\rho_k = \frac{(1-\phi\theta)(\phi-\theta)}{1-2\phi\theta+\theta^2}\phi^{k-1}$$

It follows that $\frac{\rho_2}{\rho_1} = \phi$.

Let $\rho_1 = r_1$, $\rho_2 = r_2$, then the MOM estimator of ϕ is $\hat{\phi} = \frac{r_2}{r_1}$.

$r_1 = \frac{(1-\hat{\phi}\theta)(\hat{\phi}-\theta)}{1-2\hat{\phi}\theta+\theta^2}$ have two solutions, the solution when the ARIMA is invertible that is the MOM estimator of θ is $\hat{\theta} = 1-\hat{\theta}x$ has root x such that $\mid x \mid > 1$.

Method of moments for white noise variance(MOM)

For any stationary ARMA model, the process variance $\gamma_0 = var(y_t)$ can be estimated by the sample variance that's given by

$$S^2 = \frac{1}{n-1}\sum_{t=1}^{n}(Y_t-\bar{Y})^2 \tag{3.15}$$

For AR(q): Recall γ_0 for AR(p) in (2.20) that's given by

$\gamma_0 = \frac{{\sigma_\epsilon}^2}{(1-\phi_1\rho_1-\phi_2\rho_2-...-\phi_p\rho_p)}$. Then

$${\sigma_\epsilon}^2 = \gamma_0(1-\phi_1\rho_1-\phi_2\rho_2-...-\phi_p\rho_p). \tag{3.16}$$

Then the MOM estimator of the ${\sigma_\epsilon}^2$ is obtained by substituting in $\hat{\phi}$ for ϕ, r_k for ρ_k and S^2 for γ_0. So the MOM estimator of the ${\sigma_\epsilon}^2$ is given by

$$\hat{\sigma_\epsilon}^2 = S^2(1-\hat{\phi}_1 r_1-\hat{\phi}_2 r_2-...-\hat{\phi}_p r_p). \tag{3.17}$$

For MA(q): Recall γ_0 for MA(q) in (2.43) that's $\gamma_0 = (1+{\theta_1}^2+{\theta_2}^2+...+{\theta_q}^2){\sigma_\epsilon}^2$. Then

$$ {\sigma_\epsilon}^2 = \frac{\gamma_0}{(1+{\theta_1}^2+{\theta_2}^2+...+{\theta_q}^2)}. \tag{3.18} $$

Then the MOM estimator of the ${\sigma_\epsilon}^2$ is obtained by substituting in $\hat{\theta}$ for θ and S^2 for γ_0. So the MOM estimator of the ${\sigma_\epsilon}^2$ is given by

$$ \hat{\sigma_\epsilon}^2 = \frac{S^2}{(1+\hat{\theta}_1^2+\hat{\theta}_2^2+...+\hat{\theta}_q^2)}. \tag{3.19} $$

For ARMA(1, 1): Recall γ_0 for ARMA(1, 1) in (2.58) that's $\gamma_0 = {\sigma_\epsilon}^2\left(\frac{1-2\phi\theta+\theta^2}{1-\phi^2}\right)$. Then

$$ {\sigma_\epsilon}^2 = \gamma_0 \frac{1-\phi^2}{1-2\phi\theta+\theta^2} \tag{3.20} $$

Then the MOM estimator of the ${\sigma_\epsilon}^2$ is obtained by substituting in $\hat{\theta}$ for θ , $\hat{\phi}$ for ϕ and S^2 for γ_0. So the MOM estimator of the ${\sigma_\epsilon}^2$ is given by

$$ \hat{\sigma_\epsilon}^2 = S^2 \frac{1-\hat{\phi}^2}{1-2\hat{\phi}\hat{\theta}+\hat{\theta}^2} \tag{3.21} $$

3.3 The Least square method(LSE)

The Least square method [5] based on minimizing the sum of the squared residuals(errors).

Autoregressive models

AR(1) formula that's given by $y_t - \mu = \phi(y_{t-1} - \mu) + \epsilon_t$ this is a regressive model with predictable variable y_{t-1} and response variable y_t, we estimate ϕ and μ by the values that minimize the sum of the square of the differences

$S_C(\phi,\mu) = \sum_{t=2}^{n}[(y_t - \mu) - \phi(y_{t-1} - \mu)]^2$, called the conditional sum of squares function, given the observed values $y_1, y_2, ..., y_n$, and minimizing $S_C(\phi,\mu)$ with respect to μ results

$\frac{\partial S_C}{\partial \mu} = \sum_{t=2}^{n} 2[(y_t - \mu) - \phi(y_{t-1} - \mu)](-1+\phi)$=0. So

$(\phi - 1)[\sum_{t=2}^{n} y_t - (n-1)\mu - \phi \sum_{t=2}^{n} y_{t-1} + \phi\mu(n-1)] = 0$

So $\mu((n-1)(1-\phi)(1-\phi) = (1-\phi)[\sum_{t=2}^{n} y_t - \phi \sum_{t=2}^{n} y_{t-1}]$, then

$$\mu = \frac{1}{(n-1)(1-\phi)}\left[\sum_{t=2}^{n} y_t - \phi \sum_{t=2}^{n} y_{t-1}\right] \tag{3.22}$$

For large n, $\frac{1}{n-1}\sum_{t=2}^{n} y_t \approx \frac{1}{n}\sum_{t=2}^{n} y_{t-1} \approx \bar{y}$.

So regardless of ϕ, (3.22) reduces to $\hat{\mu} \approx \frac{1}{1-\phi}(\bar{y} - \phi\bar{y}) = \bar{y}$

minimizing $S_C(\phi,\mu)$ with respect to ϕ results

$\frac{\partial S_c(\phi,\bar{y})}{\partial \phi} = \sum_{t=2}^{n} 2[(y_t - \bar{y}) - \phi(y_{t-1} - \bar{y})](y_{t-1} - \bar{y}) = 0.$

So $\sum_{t=2}^{n}(y_t - \bar{y})(y_{t-1} - \bar{y}) - \sum_{t=2}^{n} \phi(y_{t-1} - \bar{y})(y_{t-1} - \bar{y}) = 0$

So $\hat{\phi} = \dfrac{\sum_{t=2}^{n}(y_t - \bar{y})(y_{t-1} - \bar{y})}{\sum_{t=2}^{n}(y_{t-1} - \bar{y})^2}.$

For AR(p), by using the same methods that can be extended to get $\hat{\mu} = \bar{y}$, to generalize the estimation of ϕ, we consider AR(2) and substitute $\mu = \bar{y}$ in the conditional sum of squares function,

So $S_c(\phi_1, \phi_2, \bar{y}) = \sum_{t=3}^{n}[(y_t - \bar{y}) - \phi_1(y_{t-1} - \bar{y}) - \phi_2(y_{t-2} - \bar{y})]^2$

$\frac{\partial S_c}{\partial \phi_1} - 2\sum_{t=3}^{n}[(y_t - \bar{y}) - \phi_1(y_{t-1} - \bar{y}) - \phi_2(y_{t-2} - \bar{y})](y_{t-1} - \bar{y}) = 0$, then

$$\sum_{t=3}^{n}[(y_t - \bar{y})(y_{t-1} - \bar{y}) = \phi_1 \sum_{t=3}^{n}[(y_{t-1} - \bar{y})]^2 + \phi_2 \sum_{t=3}^{n}(y_{t-1} - \bar{y})(y_{t-2} - \bar{y}). \tag{3.23}$$

Dividing the both sides of (3.23) over $\sum_{t=3}^{n}[(y_t - \bar{y})]^2$ results

$$\begin{aligned} r_1 &= \phi_1 + \phi_2 r_1 \\ r_2 &= \phi_1 r_1 + \phi_2 \end{aligned} \tag{3.24}$$

(3.24) are the Yule Walker equations and solved previously in (3.11) and it is written as

$$\begin{aligned} \hat{\phi}_1 &= \tfrac{r_1(1-r_2)}{1-{r_2}^2} \\ \hat{\phi}_2 &= \tfrac{r_2-{r_1}^2}{1-{r_2}^2} \end{aligned} \tag{3.25}$$

In general for AR(p): The conditional least square estimates of $\phi's$ are the solutions of the sample Yule Walker equations in (3.15).

Moving Average models

Consider MA(1) formula that's given by $y_t = \epsilon_t - \theta\epsilon_{t-1}$.

To use least square method, we convert it to AR model, but $MA(1) = AR(\infty)$

That's MA(1) is given by $y_t = \epsilon_t - \theta y_{t-1} - \theta^2 y_{t-2} - \theta^3 y_{t-3} + \ldots$,

and $S_c(\theta) = \sum {\epsilon_t}^2 = \sum y_t + \theta y_{t-1} + \theta^2 y_{t-2} + \theta^3 y_{t-3} + \ldots$.

We can't use the least square method by calculating $\frac{\partial S_c}{\partial\theta} = 0$, it isn't practical method here.

So we'll use techniques of numerical optimization by calculating S_c for a given value of θ.

Rewrite MA(1) as $\epsilon_t = y_t + \theta\epsilon_{t-1}$.

Let $\epsilon_0 = 0$, then conditional on $\epsilon_0 = 0$, given the observed values

$y_1, y_2, ..., y_n$. We get

$$\begin{aligned}
\epsilon_1 &= y_1 \\
\epsilon_2 &= y_2 + \theta\epsilon_1 \\
\epsilon_3 &= y_3 + \theta\epsilon_2 \\
. &= . \\
. &= . \\
. &= . \\
\epsilon_n &= y_n + \theta\epsilon_{n-1}
\end{aligned} \tag{3.26}$$

and thus find $S_c(\theta) = \sum_{t=1}^{n} {\epsilon_t}^2$ conditional on $\epsilon_0 = 0$ for a single given value of θ.

We should do a grid search over the range (-1, 1) for θ to find the minimum sum of squares when MA(1) is invertible. For more general MA(q), a numerical optimization algorithm are used.

For higher order moving average models, we compute $\epsilon_t = \epsilon_t(\theta_1, \theta_2, ..., \theta_q)$ recursively from

$\epsilon_t = y_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + ... + \theta_q\epsilon_{t-q}$, with $\epsilon_0 = \epsilon_{-1} = ... = \epsilon_{-q} = 0$.

and $S_c(\theta_1, \theta_2, ..., \theta_q) = \sum_{t=1}^{n} {\epsilon_t}^2 = \sum_{t=1}^{n} y_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + ... + \theta_q\epsilon_{t-q}$

The sum of squares is minimized jointly in $\theta_1, \theta_2, ..., \theta_q$ by using a multivariate numerical method, searching over all possible values of $\theta_1, \theta_2, ..., \theta_q$ that give an solution for which MA(1) is invertible.

Autoregressive Moving Average models

Consider the ARMA(1, 1) that's given by $y_t = \phi y_{t-1} + \epsilon_t - \theta\epsilon_{t-1}$,

rewrite it $\epsilon_t = -\phi y_{t-1} + y_t + \theta\epsilon_{t-1}$

$S_c(\phi, \theta) = \sum_{t=1}^{n} {\epsilon_t}^2$,

set $\epsilon_1 = 0$ and minimize $S_c(\phi, \theta) = \sum_{t=2}^{n} {\epsilon_t}^2$ with respect to ϕ and θ

For general ARMA(p, q), we compute $\epsilon_t = \epsilon_t(\phi_1, \phi_2, ..., \phi_p, \theta_1, \theta_2, ..., \theta_q)$ recursively from

$\epsilon_t = y_t - \phi_1 y_{t-1} - \phi_2 y_{t-2} - ... - \phi_p y_{t-p} + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + ... + \theta_q\epsilon_{t-q}$,

with $\epsilon_p = \epsilon_{p-1} = ... = \epsilon_{p+1-q} = 0$.

Then minimizing $S_c(\phi_1, \phi_2, ..., \phi_p, \theta_1, \theta_2, ..., \theta_q) = \sum_{t=2}^{n} {\epsilon_t}^2$ numerically to get the conditional least square estimates of all parameters.

The least square estimation is nearly identical to the method moments for large samples. The least square estimation is consistent that's for large samples, the parameter estimate is close to the parameter being estimated.

3.4 The Maximum Likelihood estimation

The maximum likelihood estimation (MLE) [5] is a method of estimating unknown parameters in time series models. The MLE selects the sets of the values of the model parameters which maximizes the likelihood function.

The likelihood function is the function that describes the joint distribution of $X_1, X_2, ..., X_n$, it's a function of the model parameters with the observed data being fixed.

The Likelihood Function is defined to be :

$L_N : \Theta \times \mathbb{R}^N \longrightarrow \mathbb{R}^+$

$(\theta; x_1, ..., x_n) \longmapsto L_N(\theta; x_1, ..., x_n) = \prod_{i=1}^{N} f_X(x_i; \theta)$

Autoregressive models

AR(1) formula that's given by $Y_t - \mu = \phi(Y_{t-1} - \mu) + \epsilon_t$.

In AR(1), we want to estimate the parameters ϕ, μ and ${\sigma_\epsilon}^2$,

The probability density function of $\epsilon_t \sim \mathcal{N}(0, {\sigma_\epsilon}^2)$ is given by

$f(\epsilon_t) = \frac{1}{\sqrt{2\pi\sigma_\epsilon}} \exp\left(\frac{-{\epsilon_t}^2}{2{\sigma_\epsilon}^2}\right)$, for all $-\infty < \epsilon_t < \infty$.

But $\epsilon_1, \epsilon_2, ..., \epsilon_n$ are independent, so the joint pdf of $\epsilon_1, \epsilon_2, ..., \epsilon_n$ is given by

$$\begin{aligned} f(\epsilon_1, \epsilon_2, ..., \epsilon_n) &= \prod_{t=2}^{n} f(\epsilon_t). \\ &= \prod_{t=2}^{n} \frac{1}{\sqrt{2\pi\sigma_\epsilon}} \exp\left(\frac{-{\epsilon_t}^2}{2{\sigma_\epsilon}^2}\right). \\ &= (2\pi{\sigma_\epsilon}^2)^{\frac{1-n}{2}} \exp\left(\frac{-1}{2{\sigma_\epsilon}^2} \sum_{t=2}^{n} {\epsilon_t}^2\right). \end{aligned} \tag{3.27}$$

We perform the multivariate transforming

$$
\begin{aligned}
Y_2 &= \mu + \phi(Y_1 - \mu) + \epsilon_2. \\
Y_3 &= \mu + \phi(Y_2 - \mu) + \epsilon_3. \\
&\quad . \quad\quad . \quad\quad\quad . \quad\quad\quad . \\
&\quad . \quad\quad . \quad\quad\quad . \quad\quad\quad . \\
Y_n &= \mu + \phi(Y_{n-1} - \mu) + \epsilon_n.
\end{aligned} \tag{3.28}
$$

Let Y is the joint pdf of $Y_1, Y_2, ..., Y_n$, so the conditional joint distribution of $Y_1, Y_2, ..., Y_n$ given $Y_1 = y_1$.

The likelihood function (i.e. the joint pdf of Y) is given by

$L = L(\phi, \mu, {\sigma_\epsilon}^2; y \backslash y_1) = \prod_{i=2}^{n} f_{Y \backslash Y_1}(y_i \backslash y_1; \phi, \mu, \sigma_\epsilon) = f(y_2, y_3, ..., y_n \backslash y_1) f(y_1)$.

But $f(y_2, y_3, ..., y_n \backslash y_1) = f(\epsilon_1, \epsilon_2, ..., \epsilon_n) = (2\pi{\sigma_\epsilon}^2)^{\frac{1-n}{2}} \exp\left(\frac{-1}{2{\sigma_\epsilon}^2} \sum_{t=2}^{n} {\epsilon_t}^2\right)$.

But $\epsilon_t = Y_t - \mu + \phi(Y_{t-1} - \mu)$.

So $f(y_2, y_3, ..., y_n \backslash y_1) = (2\pi{\sigma_\epsilon}^2)^{\frac{1-n}{2}} \exp\left(\frac{-1}{2{\sigma_\epsilon}^2} \sum_{t=2}^{n} [y_t - \mu + \phi(y_{t-1} - \mu)]^2\right)$.

But in (2.3.4), AR(1)=MA(∞) , so AR(1) can be written as

$y_t = \epsilon_t + \epsilon_{t-1}\phi_1 + {\phi_1}^2\epsilon_{t-2} + {\phi_1}^3\epsilon_{t-3} + ...$ is a normal distribution.

Then $var(Y_1) = \sum_{k=0}^{\infty} \phi^{2k}{\sigma_\epsilon}^2 = \frac{{\sigma_\epsilon}^2}{1-\phi^2}$.

Then $Y_1 \sim \mathcal{N}(\mu, \frac{{\sigma_\epsilon}^2}{1-\phi^2})$.

So $f(y_1) = \left(\frac{1-\phi^2}{2\pi{\sigma_\epsilon}^2}\right)^{0.5} \exp\left(\frac{-(y_1-\mu)^2(1-\phi^2)}{2{\sigma_\epsilon}^2}\right)$, then

$L = f(y_2, y_3, ..., y_n \backslash y_1) f(y_1)$.

$L = (2\pi{\sigma_\epsilon}^2)^{\frac{1-n}{2}} \exp\left(\frac{-1}{2{\sigma_\epsilon}^2} \sum_{t=2}^{n} [y_t - \mu + \phi(y_{t-1} - \mu)]^2\right) \left(\frac{1-\phi^2}{2\pi{\sigma_\epsilon}^2}\right)^{0.5} \exp\left(\frac{-(y_1-\mu)^2(1-\phi^2)}{2{\sigma_\epsilon}^2}\right)$.

$L = (2\pi{\sigma_\epsilon}^2)^{\frac{-n}{2}}(1-\phi^2)^{0.5}\exp\left[-\frac{S(\phi,\mu)}{2{\sigma_\epsilon}^2}\right].$

Where $S(\phi,\mu) = (y_1-\mu)^2(1-\phi^2) + \sum_{t=2}^{n}[y_t - \mu + \phi(y_{t-1}-\mu)]^2$.

The maximum likelihood estimators of ϕ, μ and ${\sigma_\epsilon}^2$ are the values that maximize $L(\phi,\mu,{\sigma_\epsilon}^2\backslash y)$.

The function $S(\phi,\mu)$ is called the unconditional sum of squares function.

The unconditional least squares function (ULS) estimates of ϕ and μ can be found by minimizing $S(\phi,\mu)$.

When $S(\phi,\mu)$ is random, then $S(\phi,\mu) = (Y_1-\mu)^2(1-\phi^2) + S_C(\phi,\mu)$, where $S_C(\phi,\mu) = \sum_{t=2}^{n}[y_t - \mu + \phi(y_{t-1}-\mu)]^2$ is called the conditional sum of squares function .

The difference between $S(\phi,\mu)$ and $S_C(\phi,\mu)$ is only $(y_1-\mu)^2(1-\phi^2)$.

Since $S_C(\phi,\mu)$ is a sum of $n-1$ components, we have $S(\phi,\mu) \approx S_C(\phi,\mu)$ for large sample n.

The MLE's for any stationary ARMA(p, q) can be found in the same way we did for AR(1), but the likelihood function L becomes more complex in larger models, the Yule Walker estimators of the coefficients $\phi_1, \phi_2, ..., \phi_p$ of an AR(p) process have approximately the same distribution for large samples as the corresponding MLE's, and the Yule Walker estimators of the coefficients $\sigma_1, \sigma_2, ..., \sigma_p$ of an AR(p) process for large samples and the estimators are close to the true σ.

For stationary autoregressve models, the MOM, LSE, and the MLE have the same estimators for large samples.

The advantages of the MLE that it's used all the data rather than just the first and second moments as in the least squares , and that many large samples results are known under general conditions. By the weak law of large numbers, for large t, the sample average converges to its population mean.

But the disadvantage is that we must work for the first time with the joint probability density function of the process.

3.5 Box Jenkins method for ARIMA(p, q, d) models

The Box Jenkins method consists of four steps [10]:

- Order selection: First if the data isn't stationary, then we make differencing for the data until it becomes stationary, then choose the parameters p, q and d by plot the autocorrelation function and the partial correlation function and estimate p, q and d.

  If the partial autocorrelation function cuts off after a few lags, then the last lag with a large value would be the estimated value of p. If the partial autocorrelaton does't cut off, we have MA(q) (p=0) or ARIMA model with positive p and q.

  If the autocorrelation function cuts off after a few lags, then the last lag with a large value would be the estimated value of q. If the autocorrelaton doesn't cut off, we have AR(p) (q=0) or ARIMA model with positive p and q.

When neither the autocorrelatons nor the partial autocorrelations cuts off, then it's an ARIMA model, that's a mixture of exponential decay and damped sine waves after the first q-p , the partial correlation function has the same pattern after p-q lags, and we use error and trail approach until the residuals have small correlations then we estimate values for p and q.

- Estimation of the coefficients: The coefficients of the AR(p) are $\phi_1, \phi_2, ..., \phi_p$. The coefficients of the MA(q) are $\theta_1, \theta_2, ..., \theta_q$. These coefficients are estimated by estimators such as MLE.

- Diagnostic check: The fit of the ARIMA(p, q, d) with the estimated coefficients is checked. Check if the empirical autocorrelation function is close to 0. So if all the correlations and partial correlations of the residuals are small, then the model is adequate and we find the forecasts. But if there's a large correlation for the residuals, we repeat the previous steps again.

- The prediction of the future values of the original process: The forecasts are done.

3.6 Monte Carlo methods

Monte Carlo methods are from a class of computational algorithms can be applied to wide ranges of stochastic system problems.

The Monte Carlo method simulates the behavior of a system by taking repeated sets of random numbers (huge amount of random variables) from the probability distribution of the process under investigation, so the observations are independent in this method.

To perform the Monte Carlo method [6], we follow four steps:

- Define a distribution of possible inputs for each input random variable: Requires recognition of the probability distribution of the process.

- Generate inputs randomly from those distributions: Requires the selection of an appropriate random number generator to model the observed probability distribution.

- Perform a deterministic computation using that set of inputs: Computing the desired output variable or variables from the generated random numbers.

- Aggregate the results of the individual computations into the final result: The aggregation process is dependent on the specific simulation that can be as computing the average of the simulated results.

example 3.6.1. ***Numerical calculation of* π *[26]***

The area of the a unit circle is π*. So we can calculate* π *by numerical integration by the following algorithm:*

- *Draw a unit circle arc in the first quadrant, that's an arc of radius one circumscribed by a square.*

- *Choose N points randomly in the first quadrant, for instance N independent pairs x, y* $\in$ *[0, 1].*

- *Calculate* $r^2 = x^2 + y^2$.

- *Count the number of points within the unit circle and the number of points in the quarter circle that's the number of points where* $r^2 \leq 1$*. With a large number of points, these values will approximate the area of the circle and the area of the square.*
  $\frac{The\ number\ of\ points\ inside\ circle}{The\ number\ of\ points\ inside\ square} = \frac{0.25\pi r^2}{r^2} = \frac{\pi}{4}$.
  Then multiply the last value by 4 to get the result is the value of π.

This example applied the steps of Monte Carlo method mentioned above. A random number generator selects the coordinates for each dot. The coordinates were selected from uniform distribution that provided the probability density function. A sampling rule used the random numbers to select values from the uniform distribution, the scoring method *by the formula in step 4. Finally error estimation is performed by comparing the computed value of π to the theoretical value for π .*

3.7 Bootstrapping in ARIMA models[9]

Let $(X_1, ..., X_n)$ be a finite sample of n identical independent observations obtained from unknown probability distribution F(.) and let $T_n(X)$ be some static of interest. And $\hat{F}(.)$ is the empirical distribution that assigns probability mass n^{-1} to each sample element.

The bootstrap [18] approximate the sampling distribution of $T_n(X)$ under F(.) by the bootstrap distribution of a $T_n(X^*)$ under $\hat{F}(.)$, where $X^* = ({X_1}^*, ..., {X_n}^*)$ is a bootstrap sample (called pseudo data) of size n obtained by randomly sampling with replacement from sample X.

The bootstrap algorithm starts by generating a large number B of independent bootstrap samples denoted by ${X_i}^*$, i=1, 2, ..., B, each of size n. These samples are drawn from the empirical distribution $\hat{F}(.)$. Corresponding to each bootstrap sample ${X_i}^*$ is a bootstrap replication of $T_n({X_i}^*)$, the value of the statistic evaluated for ${X_i}^*$.

The set of bootstrap estimates $\{T_n({X_i}^*), i = 1, 2, ..., B\}$ are an approximation to the true sampling distribution of the statistic $T_n(X)$.

The bootstrap estimate of standard error

The bootstrap is a method for estimating standard errors by repeatedly resampling with replacement from the original finite that's a sample of identical independent observations from unknown probability distribution.

Let $\hat{F}$ be the empirical distribution which assigns probability mass n^{-1} to each sample element $x_i, i = 1, ..., n$. A bootstrap sample is a random sample of size n drawn from $\hat{F}$, that's $x^* = (x_1{}^*, ..., x_n{}^*)$ is a bootstrap sample of size n obtained by randomly sampling with replacement from sample x. $\hat{F} \rightarrow (x_1{}^*, ..., x_n{}^*)$

Corresponding to a bootstrap data set x^* is a bootstrap replication of $\hat{\theta}$, and $\hat{\theta}^* = s(x^*)$.

Where $s(x^*)$ is the result of applying the same function s(.) to x^* as was applied to x.

If $s = \sigma$ then $\hat{\sigma} = \sigma(\hat{F}) = [var_F(\hat{\theta})]^{0.5}$[8]

The bootstrap estimate of $\sigma_F(\hat{\theta})$ is defined by $\sigma_{\hat{F}}(\hat{\theta}^*)$, that's the standard error of $\hat{\theta}$ for data sets of size n randomly sampled from $\hat{F}$ in place of a unknown function F.

Chapter 4

Recurrent Neural Networks

4.1 Artificial Neural Networks structure

An artificial neural network is a simulation of the human nervous system, in the brain, we have neurons connected by synapses. The human brain is in high complexity and does nonlinear and parallel computation, also the artificial neural networks have functions with the same features to simulate the human brain real activity.

The input\output mapping in human brain that we give input to network and expect the output, so it's learned to do specific tasks and developing this feature in supervised (feed inputs and desired network output) or unsupervised way (feeding only inputs and and let network do associative procedure), that neural network adjusts its free parameter to get the desired output.

The neural network contains of processing elements neurons acting as like nodes connected by the interconnections, the neurons of the same layer aren't connected but the neurons of the adjacent layers are connected.

The one layer network has one or more neurons, and multiple layered network containing more than one layer.

In the multiple layered network, the first layer is the input layer, the last layer is the output layer, and the layers between the input layers and the output layers are called the hidden layers, each neuron has output and the inputs of the neuron (after the input layer) is the outputs of the previous neurons connected to it.

Definition 7. *The activation function is a function that limits the amplitude of the output of the neuron in the neural network, its input is the sum of the weighted sum of output of the and the bias of the same neuron.[9]*

The input p is transmitted through a connection line multiplies its strength by a weight w, where each link between two neurons is associated with a weight, the weight of the link from the i^{th} neuron to the j^{th} neuron are called w_{ij}.

In the layer with s neurons and r inputs, each input p_i, i=1, 2, ..., r, is connected to the input of each neuron n_j with weight w_{ij}, j=1, 2, ..., s, each neuron n_j is connected to bias b_j, j=1, 2, ..., s, the input to the transfer function for n_j is $\sum_{i=1}^{r} w_{ij} p_i$, j=1, 2, ..., s. So we get s outputs. The bias activated the network when the signal is low, and we adjust w and b to get the desired output with less error.

4.2 The activation functions

The neurons of the same layer uses the same transfer function most the time.

Examples of the transfer functions [21] are:

- The hard limit function (the threshold function or called the Heaviside step function): Limits the output to 1 when the input to it is positive or 0, or limits it to 0 when the input to it is negative, this function can be used in perceptron to take classification decisions, its formula is given as

$$f(y) = \begin{cases} 0 & y < 0 \\ 1 & y \geq 0 \end{cases} \tag{4.1}$$

  It's differentiable at all points except at 0.

- The linear transfer function: Gives linear output (the output =the input + the bias), it's differentiable function used in neurons as linear approximators .

- The hyperbolic tangent function (f(t)= tanh(t)) [11]. Its range is

[-1, 1], used for modeling and control, and in the hidden layers of RNNs and LSTMs to approximate the functions that take negative real numbers, it's differentiable where $f'(t) = \frac{\partial}{\partial t}(tanh(t)) = \frac{\partial}{\partial t}(\frac{sinh(t)}{cosh(t)}) = \frac{(sinh(t))\prime cosh(t) - sinh(t)(cosh(t))\prime)}{(cosh(t))^2}$.
so $f'(t) = \frac{cosh(t))^2 - (sinh(t))^2}{(cosh(t))^2} = \frac{1}{(cosh(t))^2} = (sech(t))^2$.

- The logistic functions: Its range is (0, 1), it's used for binary classifications, its formula is given by
  $f(t) = \frac{A}{1+e^{-k(t-t_0)}}$, where t is the the sum of the outputs of the previous neurons added the bias to it . And it's differentiable that's
  $$f'(t) = \frac{-kAe^{-k(t-t_0)}}{(1+e^{-k(t-t_0)})^2}. \tag{4.2}$$
  Where A is the maximum of its curve, t_0 is the midpoint of its curve(sigmoid curve) and k is the logistic growth rate or steepness of its curve.
  The logistic function is a scaled hyperbolic tangent function that's given by $f(t) = 0.5 + 0.5tanh(0.5x)$
  When k=A=1 and $t_0 = 0$, then it's called the sigmoid function (or standard logistic function), it's differentiable and it's used in propagation networks, its formula is given by $f(t) = \frac{1}{1+e^{-t}}$.
  So according to (4.2) $f'(t) = \frac{-e^{-(t-t_0)}}{(1+e^{-(t-t_0)})^2}$.
  It's used in the models that predicts the probability as an output,

and for classification problems and as a function approximater.

- RELU (Rectified Linear Unit) activation function [14]: Its formula is given by $f(x) = \max(0, x)$ that's

$$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \tag{4.3}$$

$$f'(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases} \tag{4.4}$$

It's not differentiable nor bounded.

RELU activates the neoron when the input is above a certain value and speeds the training time and make the training better when the neurons are either off or working in a linear system, and we can use it as a classifier.

- Leaky RELU (parametric rectified linear unit)[22] is given by

$$f(y) = \begin{cases} \alpha y & y < 0 \\ y & y \geq 0 \end{cases} \tag{4.5}$$

$$f\prime(y) = \begin{cases} \alpha & y < 0 \\ 1 & y \geq 0 \end{cases} \tag{4.6}$$

Where α is a parameter of order 0.01, it's chosen small and tested by the model to find the better one needed.

- The softmax function [22]: Transforms the k dimensional vector into another k dimensional vector of real values, each has range in (0, 1) and sum up to 1, it's used in the output layer classifier in multiclass classifications. Let $y = (y_1, ..., y_k)$ be a vector where $y_i \in \mathbb{R}, i = 1, ..., k$, then the softmax function $S(y) = (S(y_1), ..., S(y_k))$ is given by

  $S(y_i) = \frac{e^{y_i}}{\sum\limits_{r=1}^{k} e^{y_r}}$.

  As seen $S(y_i)$ since clearly the denominator is less the nominator,

  also $\sum\limits_{j=1}^{k} S(y_j) = \sum\limits_{j=1}^{k} \frac{e^{y_j}}{\sum\limits_{r=1}^{k} e^{y_r}} = \frac{\sum\limits_{j=1}^{k} e^{y_j}}{\sum\limits_{r=1}^{k} e^{y_r}} = 1$

  To use the softmax function as a classifier, we should use a layer with 10 neurons, each of it has output equals z_i, and after it let the last layer(output layer) has 1 neuron with softmax function as an activation function so its input is 10 outputs that is $z_j, j = 1, ..., 10$

Definition 8. *Squashing function is a nonlinear activation function with bounded range such as standard logistic function and tanh*

4.3 The perceptron

The perceptron is a single layered network has many neurons, it is a linear model binary classifier that uses the heaviside step function as an activation function and the output of the heaviside function is the output of the perceptron. The neural network modeling have i neurons for any time t is given by [1]

$\tau \frac{dx_i}{dt} + x_i = f(b_i + \sum_j w_{ij} x_j)$.

Where i=1, ..., N, the argument to the activation function f is the input to the i^{th} neuron, w_{ij} is the synaptic weight from the j^{th} neuron to the i^{th} neuron, x_j is the input to the neuron j, $w_{ij}x_j$ is the synaptic input to the j^{th} neuron, b_i is the bias is an input from outside the network or provided to the neuron to make it active, τ is a time constant show the rapid of the response of the variable x_i to the changes in input.

The feed-forward network contains several simple perceptrons have w_{ij}=0, $\forall i \leq j$ since the signal move only from input to output that's from a neuron with a small index to a neuron with bigger index. The feed-forward network converges to a unique steady state that's $\frac{dx_i}{dt}$=0, so it's given by

$x_i = f(b_i + \sum_j w_{ij} x_j)$, so the perceptron is a piece-wise function follows

$$f(x) = \begin{cases} 1 & b + wx > 0 \\ 0 & otherwise \end{cases} \tag{4.7}$$

Where w is the synaptic weight vector and x is the input vector and b is the bias vector.

The perceptron can be written in vector notation as f(wx+b), and the perceptron still changes weights until all inputs are classified properly.

A network with one hidden layer [28] is given by

$y_i = \sum_{r=1}^{n} w_{ir} f(\sum_{j=1}^{m} v_{rj} x_j + b_r)$, i=1, ..., l.

Where $x_j \in \mathbb{R}^m$ is the input and $y_i \in \mathbb{R}^l$ the output of the network and the activation function f(.), the weight matrices $W \in \mathbb{R}^{l \times n}$ for the output layer and $v_{rj} \in \mathbb{R}^{n \times m}$ for the hidden layer, w_{ir} is the weight from neuron i to the neuron r and $b_r \in \mathbb{R}^n$ is the bias vector with n the number of the hidden neurons

This process of propagation from the input of the network to the output are called forward propagation.

The multiple layer perceptron

The multiple layer perceptron are perceptrons with one or more hidden layers between input and output layers, using the sigmoid activation functions, they're universal approximators, when adjust weights, perform linear transformations and the neurons activation use local nonlinear transformations.

4.4 The learning procedure

The important elements in the design of any application containing neural networks is the number of layers, the number of neurons in each layer, and the transfer function of each layer, the power of the networks is in having many neurons in the hidden layers.

In learning process [20]; we modify the connections weights to get less error in the output of the neural network. Some networks are called fixed since their weights are prior fixed, others are adaptive neural networks which have changeable weights.

Definition 9. *The epoch is the time from entering input until all the patterns in the training set have been presented once in the neural network.*

The learning methods are applied for adaptive neural networks, the categories of learning methods are:

- The supervised learning: The training set consists of pairs of input and their corresponding desired outputs as a training pattern and learning acts as external teacher, and we still adjust the weights and biases until minimizing the error between the desired output and the produced output. For example object recognition.

  The reinforcement learning: Is a special case of supervised learning by adjusting the neural parameters depending on any qualitative or quantitative information obtained through the interaction of the system, then using the trail and error that's if the produced output is satisfactory, we increase the weights and biases to reinforce this state, it has a scalar performance index called the enforcement signal to know if the network system's output is correct or not, for example chess game, we have two types of reinforcement:

- The positive reinforcement: An event happens because of a behavior that increases strength and the frequency of the behaviour that maximizes the performance and keeps the change for a long period of time.
- The negative reinforcement: Strengthening of a behaviour since a negative condition is stopped or avoided, it gives challenge to minimum standard of performance, but it also gives enough to minimum of bad behaviour.

- The unsupervised learning: The training data is the input training patterns only without an external teacher and without any knowledge about the values of the desired output, so neural networks adapt weights, learn and respond relying on the inputs, may be the prior is the maximum or the minimum of the output before.
- The on line learning: Is changing weights and biases after each training sample when new input pattern added to the network. It's used when the behavior of the system is changing quickly
- The off line learning algorithm (or called batch learning): Is changing weights and biases after making all the training set, each adjustment depending on the number of errors that's occurred.

The adaptive learning rate

The learning rate [19] is about how much the weights and biases changes per time through optimization to reduce the neural network's error, if it's very high then the produced output fails to converge to the desired output,but if it's very small, the produced output reached the desired output very slowly. The learning rate $\eta(t)$ are assumed to be constant, but really the training starts with big $\eta(t)$ then it decreases by time. At t=0, many weights changed, so the number of epochs decreased, so the learning rate decreases. The equation that describes the learning rate [19] as a function of time is

$\eta(t) = \eta(0)e^{-\alpha t}$

Where α is the slope of the negative exponential.

The loss function

The loss function [24] computed the error of the network by using the least squared method after updating the weights and the biases, the sum of squared errors is given by

$E = \frac{1}{N}\sum_{i=1}^{N}(Y_{di} - Y_i)^2$

Where N is the number of samples (sets of inputs and corresponding outputs), Y_{di} is the desired output corresponding to the i^{th} input and Y_i is the real output collected from the neural network.

Backpropagation

The backpropagation [31] is a supervised learning in feed-forward nonlinear multiple layers neural networks used for function approximation, pattern association and pattern classification, the backpropagation is one of the the most popular steepest gradient descent methods.

Let M be a feed forward neural network with k layers called $L_1, ..., L_k$, where L_1 is the input layer, $L_2, ..., L_{k-1}$ are the hidden layers and L_k is the output layer and m_k is the number of neurons in the output layer. Let P training patterns (x_p, d_p) where x_p is the input value, d_p is the desired output value, and $1 \leq p \leq P$, let Q validation patterns (v_q, d_q) where v_q is the input value in the validation process , d_q is the desired output value in the validation process, and $1 \leq q \leq Q$,where y_p is the resulted output, the error of each neuron j in the output layer is $e_p = y_p - d_p$, then the squared error for pattern p is given by

$$E_p = \frac{1}{m_k}\sum_{p=1}^{P}(e_p)^2 = \frac{1}{m_k}\sum_{j=1}^{m_k}(y_j - d_j)^2 \tag{4.8}$$

Let E_{avg} be the average error for all input patterns that's given by

$E_{avg} = \frac{1}{P}\sum_{p=1}^{P} E_p$

Let (i, j) is an interconnected pairs of neurons, where i is a neuron in layer l, j is a neuron in layer l+1 and w_{ij} are the weights on their connections, where $f(t_j)$ is a differentiable activation function.

To adjust w_{ij} of the neuron j in the output layer k, we should find

$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial y_j}\frac{\partial y_j}{\partial w_{ij}}$

by differentiating (4.8) with respect to $y_{i,p}$, we get

$$\frac{\partial E_p}{\partial y_j} = y_j - d_j, \tag{4.9}$$

also $\frac{\partial y_j}{\partial w_{ij}} = \frac{\partial y_j}{\partial t_j}\frac{\partial t_j}{\partial w_{ij}} = f'(t_j)\frac{\partial}{\partial w_{ij}}\sum_k w_{jk}y_k$, so

$$\frac{\partial y_j}{\partial w_{ij}} = f'(t_j)y_i \tag{4.10}$$

So by (4.10) and (4.9), we get

$\frac{\partial E_p}{\partial w_{ij}} = (y_j - d_j)f'(t_j)y_i.$

But the local gradient is given by

$$\delta_j = (y_j - d_j)f'(t_j) \tag{4.11}$$

then

$$\frac{\partial E_p}{\partial w_{ij}} = \delta_j y_i \tag{4.12}$$

is the gradient of the error function for each pattern p.

But to adjust w_{ij} proportionally to the gradient but in the opposite direction, we use the learning rate η where $\eta \in (0,1)$, so

$\Delta w_{ij} = -\frac{\partial E_{avg}}{\partial w_{ij}} = -\frac{1}{m_{l+1}}\frac{\partial y_i}{\partial w_{ij}}\sum_{i=1}^{m_{l+1}}(y_i - d_i) = -\eta\frac{\partial E_p}{\partial w_{ij}} = -\eta\delta_j y_i.$

The learning rate is multiplied by the error gradient to choose how much of the gradient to be used.

In (4.11), δ using the output desired but we can't use (4.11) in hidden layers since there's no output desired in the hidden layers, but we know that

$$\delta = \frac{\partial E_p}{\partial t_j} = \frac{\partial E_p}{\partial y_j}\frac{\partial y_j}{\partial t_j} = \frac{\partial E_p}{\partial y_j} f'(t_j) \tag{4.13}$$

For hidden layer l, let i, j ,k neurons each in a different layer, so $\frac{\partial E_p}{\partial y_j} = \sum_{k=1}^{m_{l+1}} \frac{\partial E_p}{\partial t_k}\frac{\partial t_k}{\partial y_j} = \sum_{k=1}^{m_{l+1}} \frac{\partial E_p}{\partial t_k} w_{j,k} = \sum_{k=1}^{m_{l+1}} \delta_k w_{j,k}$, so

$$\frac{\partial E_p}{\partial y_j} = \sum_{k=1}^{m_{l+1}} \delta_k w_{j,k} \tag{4.14}$$

So by (4.14) and (4.13), we get the local gradient for neuron j in layer l

$$\delta = \sum_{k=1}^{m_{l+1}} \delta_k w_{j,k} f'(t_j) \tag{4.15}$$

The algorithm of the back-propagation[34]

We want to minimize the the loss function that reduces the error by taking the derivative of the loss function and calculating the gradient.

- let $E_{avg} = \infty$
- begin a new epoch
- $\forall p \in \{1, ..., P\}$, we have the pattern (x_p, d_p), first set $y_0 = x_p$ and m_k is the number of neurons in the k layer.
- Forward pass: It starts from the input towards the output units, the output of the neurons is calculated and stored as follows

$\forall j \in L_l$, where l=1, ..., k, find $t_j = \sum_{i=1} w_i x_i + b$ and $y_i = f(t)$.

- Backward pass: It begins from the output and goes towards the input. Propagate the error that's calculated at the output layer, and calculate the δ_j variables for each layer, then adapt the weights w_{ij}, as follows

  - $\forall i \in L_{k-1}, j \in L_k$, calculate $\delta_j = (y_j - d_j) f'(t_j)$ and $\Delta w_{ij} = -\eta \delta_j y_i$
  - For l=k-1, ..., 1 find $\forall i \in L_{l-1}, j \in L_l$, calculate $\delta_j = \sum_{k=1}^{m_{l+1}} \delta_k w_{j,k} f'(t_j)$ and $\Delta w_{ij} = -\eta \delta_j y_i$
  - For l=1, 2, ..., k $\forall (i,j) \in L_{L-1} \times L_l$, adjust Δw_{ij} by $\Delta w_{ij} = -\eta \delta_j y_i$
  - Find the error of the pattern p : $E_p = \frac{1}{m_k} \sum_{j=1}^{m_k} (y_j - d_j)^2$

- Let $E_{prev} = E_{avg}$, and find a new $E_{avg} = \frac{1}{P} \sum_{p=1}^{P} E_p$ for the validation data set.

- if $E_{prev} > E_{avg}$ then repeat the steps beginning from the second step.

When minimizing the cost function by backpropagation algorithm, we can use backpropagation as on line or off line learning algorithm.

4.5 Training

Training is using methods to find the weights that optimize the network performance.

To check our model with minimum error, we divide our dataset [7] into:

- Training set: Selected to train the model on it by using inputs and parameters in an optimizer method such as gradient descent, and updating weights and biases.

- Validation set: Its error is computed across the training process, the validation error and the training error decrease when the training starts, the validation error increase across overfitting the data, the weights and the biases saved when the validation error reaches its minimum that gives indication to end the training

- The test set: Compare different models by using the trained model and then check how it's working.

4.6 Recurrent Neural Networks

Recurrent means that it creates cycles in the network and models the time that's the output of the network became input to the network and learn from the sequences, so the output at a specific time relies on the current input and all inputs at previous time steps. The feedback neural networks let signals move in two ways, from input to output, and back to input again. Recurrent neural networks are feedback neural networks use both parallel and sequential computation with a large feedback network and do the same type of operation to all terms in the sequence, and predict the next terms using its internal memory for the previous terms, changes the weights until it reaches the desired output.

Definition 10. *Let f be a smooth bounded nonlinear function such as sigmoid or hyperbolic tangent, $x \in \mathbb{R}^M$ is an Upstream layer as vector of size M, W_{xh} is a weight matrix of size $N \times M$ for link from upstream layer to hidden layer, $b_h \in \mathbb{R}^N$ is bias of size N for the hidden layer then the hidden layer $h \in \mathbb{R}^N$ is calculated by*

$h = f(W_{xh}x + b_h)$,$t \in \mathbb{N}$. is the current time step, and $W_{hh} \in \mathbb{R}^{N\times N}$ is a weight matrix of size $N \times N$ for recurrent link from hidden layer of previous time step to hidden layer of the next time step, the RNN hidden layer $h_t \in \mathbb{R}^N$ is calculated by

$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$, $t \in \mathbb{N}$.[16]

The vanishing gradient problem

The vanishing gradient [17] occurs when the gradients are very small and becomes hard to model long range dependencies (10 time steps or more) in the input dataset. That happens when the output error fails to reach the farther neurons through training, that the back-propagation process propagates the output error backward to the hidden layers, the error comes to the first hidden layer hardly, and the weight can't adjusted, so the hidden layers after the first one aren't trained correctly, so they don't added and vanished.

The gradients of a hidden layer with respect to a another layer can be is the product of the gradient of the current hidden layer h_i [23] against the previous one h_{i-1}, then $\frac{\partial h_t}{\partial h_k} = \prod_{t \geq i > k} \frac{\partial h_i}{\partial h_{i-1}}$

If the gradient is smaller than 1, then after many time steps; the product of the gradients become more smaller until it vanishes, if it's larger than 1, then after many time steps; the product of the gradient become more larger until it explodes. This may be solved by true initialization of weights. But LSTM is often used to solve this problem.

The Long Short-Term Memory (LSTM)

LSTM [13] is a recurrent neural network that has four layers (3 gates and 1 hidden layer), the components of LSTM unit are:

- Three gates, the gates of the LSTM are:
  - The sigmoid input gate controls the degree of the data entering the input of the network.
  - The forget gate decides if and how the data stayed through time states, it's connected the memory carousel and controls

the memory transfer from time step to the next one, it uses the sigmoid function, if the forget gate is 1, the cell content stayed, if it's 0, the cell content is deleted.

  – The sigmoid output gate decided the size of data that exists the network

- block input

- Memory cell (the constant error carousel) has fixed weight of 1, the contents of the memory cell are feeded by the input gates and the forget gates, it has a linear activation function, and passes through squashing function as standard logistic function so that the backpropagaton be effective.

- Output activation function

- Peephole connections: [16] Point-wise weighted connections from the memory cell to the gates, to let the memory cell decide if the data should stayed or overwritten or passed to the next time step, two peepholes are recurrent to the forget gates and input gates

LSTM has many parameters that slow the training and need more data and longer training.
The output of the LSTM block is recurrent connected back to the block input and all of the gates for LSTM block. The input, forget and output gates have sigmoid activation function for [0, 1]. The LSTM block

input after forget gate and output activation after output gate uses a tanh activation function.

The back-propagation through time and the gradient descent optimization after time causing vanishing in gradients but LSTM solved that [12] by separating the memory cells and the output by using the input gates and the forget gates that are closed then the contents will stay unmodified between one time step and the next for a long time, so the information passes through many time steps, then gradients pass across many time steps if there's no new input or error signal, so that the learning in the recurrent network continues over many time steps.

The Gated Recurrent Units(GRU)

The gated recurrent units [13] are simplified LSTMs that combine the forget gate and the input gate into one update gate that determines how much previous memory to keep (control the data flows into the memory), and replaces the output gate by a reset gate controls the recurrent links to the block input(control the data flow outside the memory), and merges the memory cell layer and the block output layer into exposed memory layer, it decided the quantity from the hidden state being carried out from the previous time step.

GRUs don't have the controlled encapsulation of the memory content, control the data using separate forget and output gates, have no cell state; exposes the memory content at each time step, and transits between the previous memory content and the new memory content using leaky integration controlled by update gate, and neither have the independence between the inclusions of the present and past input, but it's simplified in computation and faster to train.

Overfitting

Overfitting [19] happens when the number of free parameters (that's the number of weight connections) is very big compared to the size of the training data, so there's a big gap between the train and the test data performance when the model is complex and the data set is small, that may be the number of training is fewer than the number of the parameters; so it may be infinite number of solutions with zero error, it will be poor performance because of the inference of the parameters, so it's better that the number of training data points be 2 or 3 times the number of parameters in the neural network. Also increasing the number of neurons may cause overfitting [26].

Regularization

Regularization is any modification to a learning algorithm that is used to reduce its test error but not the training error to get better result in the test set. One of the efficient methods of the regularization is the dropout that is used to solve the overfitting problem

Dropout:

The dropout uses node sampling instead of edge sampling, in each iteration removes nodes and all incoming and outgoing connections from these nodes that effect the training [27], then in the LSTMs, the stronger activations might make the units more independently so the LSTMs weights become higher, and the error gradient can be propagated to learn long-term dependences.

Dropping nodes is done by sampling it by very small probability (between 0.2 and 0.5) that determine the retaining of the activation of the layer and increase the training time [27], multiply that probability with the weights of the nodes, that's called the weight scaling inference rule, so the input of the unit is the same as the expected input in a sampled network. Dropout in all hidden layers is more effectively than in only one hidden layer. It added noise to the hidden layer so minimize the loss function.

Parameter Estimation

We use many methods to estimate error in recurrent neural networks, some of these methods is bootstrap methods and Monte Carlo methods.

The bootstrap is done by making many bootstrap samples of the training set and restimating the parameter on each bootstrap sample bootstrap pairs sampling algorithm [30]:

- Generate B samples, each one of size n chosen with replacement from the n training observations $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$. The

b^{th} sample is denoted by $(x^*{}_1, y^*{}_1), (x^*{}_2, y^*{}_2), ..., (x^*{}_n, y^*{}_n)$.

- For each bootstrap sample b=1, ..., B, minimize $\sum_{i=1}^{n}[y^*{}_i - y(x^*{}_i, \phi)]^2$ getting the parameter estimation for the parameter ϕ is ϕ^*.

- Estimate the standard error of the predicted value by $\left\{\sum_{b=1}^{B} \frac{[y(x_i, \phi^*) - y(x_i, .)]^2}{B-1}\right\}^{0.5}$, where $y(x_i, .) = \frac{\sum_{b=1}^{B}[y(x_i, \phi^*)]^2}{B}$.

Bootstrap residual sampling algorithm [30]:

- Estimate the parameter $\hat{w}$ from the training sample and let $r_i = y_i - y(x_i, \hat{w}), i = 1, 2, ..., n$.

- Generate B samples, each one of size n chosen with replacement from $r_1, r_2, ..., r_n$. The b^{th} sample is denoted by $r^*{}_1, r^*{}_2, ..., r^*{}_n$ and let $y^*{}_i = r^*{}_i + y(x_i, \hat{w})$

- For each bootstrap sample b=1, ..., B, minimize $\sum_{i=1}^{n}[y^*{}_i - y(x^*{}_i, w_i)]^2$ getting the parameter estimation for the parameter $\hat{w}$ is $\hat{w}^*$.

- Estimate the standard error of the i^{th} predicted value by $\left\{\sum_{b=1}^{B} \frac{[y(x_i, \hat{w}^*) - y(x_i, .)]^2}{B-1}\right\}^{0.5}$, where $y(x_i, .) = \frac{\sum_{b=1}^{B}[y(x_i, \hat{w}^*)]^2}{B}$.

The bootstrap pairs sampling algorithm is more strong than the bootstrap residual sampling algorithm since the errors $y - \hat{y}_i$ in the bootstrap residual sampling algorithm represent the true model errors, and the model may be misspecified or overfitted. But the bootstrap pairs in each bootstrap; result in a different set of predictor values that may be chosen by design and that may be less common in the applications of the neural networks.

Monte Carlo estimation

The Monte Carlo method is described in section 3.6, but how can we find the mean and the variance of the estimation.

Chapter 5

Recurrent Neural Networks Applications

In this chapter we give examples of recurrent neural networks of one layer that is used as a classifier, and use methods for estimating the error of that output of the layer in the recurrent neural networks, graphing the results and comparing it.

We test with a single layer neural network with 3 inputs that's 3 nuerons in its input layer, 3 neurons in its hidden layer and one neuron in its output layer, let f(x)=tanh(x) be the activation function for the hidden layer, where x is the input of the network, and let the linear function g(t)=2t+1 is the output layer function , where t is the input to the hidden layer. We want to estimate the error of the hidden layer in the recurrent neural network, the output of the neuron j in the hidden layer network is given by

$$Y_j = f(\sum_{i=1}^{3} w_{ij}x_i + b_j) = tanh(\sum_{i=1}^{3} w_{ij}x_i + b_j),$$

Where x_i is the input of the i^{th} neuron in the hidden layer, w_{ij} is the weight on the link from the i^{th} neuron in the input layer to the j^{th} neuron in the hidden layer, b_j is the bias of the thej^{th} neuron in the hidden layer.

The observed output of the network is y=1.46.

So the objective is minimizing the error in Y_j by bootstrapping and Monte Carlo methods then graphing the results and comparing it.

We choose a set of training patterns

$\{(x_i, y(x_i, \hat{w_{11}}))\} = \{(35, -1), (-7, -1), (2, 1.)\},$

Where x_i is the input of the network, y_i is the predicted output of the i^{th} neuron in the hidden layer and the bias on the i^{th} neuron in the hidden layer is $b_i = 1, \forall i = 1, 2, 3$

The resulted output for the i^{th} neuron in the hidden layer is

$y_i = \{-1.3, -1.4, 1.8\}$

To find the minimum of the error of the output of the i^{th} neuron in the hidden layer; we use the least square error method.

5.1 Using the Bootstrapping method in minimizing the error

The bootstrap residual sampling algorithm:

We want to update w_{11} to get the minimum error of the output of the i^{th} neuron in the hidden layer, so we will do these steps: Estimate the parameter $\hat{w_{11}}$ from the training sample and let the error in the output for the i^{th} neuron in the hidden layer is $r_i = y_i - y(x_i, \hat{w_{11}}), i = 1, 2, 3,$ to estimate $\hat{w_{11}}$ to get the minimum of the output, we find the derivative of the output with respect to the weight w_{11} that's

$\frac{\partial}{\partial w_{11}} Y_i(x) = \frac{\partial}{\partial w_{11}} tanh(\sum_{i=1}^{3} w_{ij} x_i + b_i). = 0.$

So $x_1 sech^2(w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + b_i) = 0$

By graphs, we have the minimum of the output of the of the i^{th} neuron in the hidden layer is $Y_1 = tanh(35w_{11} + .1)$ when $w_{11} = -0.1$ that's $\hat{w_{11}} = -0.1$.

Then set the weights from the i^{th} neuron in the input layer to the j^{th} neuron in the hidden layer are w_{ij} and the weights w_j on the link connected to the j^{th} neuron the output layer which are in the table below

Table 5.1: Table of weights

Inputs	neuron1	neuron2	neuron3	w_i
x_1	?	-0.6	0.5	0.1
x_2	0.1	0.3	-0.1	-0.5
x_3	-0.1	0.8	0.2	0.2

The predictive outputs for j^{th} neuron in the hidden layer outputs where j=1,2,3 are written as

$y_1(x_1, w_{11}) = tanh(w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + 1) = -1.$

$y_2(x_2, w_{11}) = tanh(w_{12}x_1 + w_{22}x_2 + w_{32}x_3 + 1) = -1.$

$y_3(x_3, w_{11}) = tanh(w_{13}x_1 + w_{23}x_2 + w_{33}x_3 + 1) = 1.$

But the error in the hidden layer output is $r_i = y_i - y_i(x_i, w_{11})$.

So the error in the hidden layer output is $\{r_i, i = 1, 2, 3\} = \{-.3, -.4, .8\}$,

Then we generate B samples, each one of size 10 are chosen with replacement from r_1, r_2, r_3. The b^{th} sample is denoted by $r^*{}_1, r^*{}_2, r^*{}_3$ and let $y^*{}_i = r^*{}_i + y(x_i, \hat{w_{11}})$, the error samples are $\{r_2, r_3, r_1\}, \{r_3, r_1, r_2\}$, $\{r_2, r_2, r_1\}, \{r_2, r_1, r_1\}, \{r_3, r_3, r_1\}, , \{r_1, r_3, r_3\}, \{r_1, r_1, r_3\}, \{r_3, r_1, r_3\}$, $\{r_2, r_3, r_2\}, \{r_1, r_2, r_1\}$

So for b=1:

$y^*{}_1 = r^*{}_1 + y_1(x_1, \hat{w_{11}}) = -.4 - 1 = -1.4$

$y^*{}_2 = r^*{}_2 + y_2(x_2, \hat{w_{11}}) = .8 + -1 = -.2$

$y^*{}_3 = r^*{}_3 + y_3(x_3, \hat{w_{11}}) = -.3 + 1 = .7$

Similarly the all bootstrap samples in the table below

Table 5.2: Table of bootstrap samples

b	$y^*{}_1$	$y^*{}_2$	$y^*{}_3$
1	-1.4 ± 0.4	-0.2 ± 0.8	0.7 ± 0.3
2	-0.2 ± 0.8	-1.3 ± 0.3	0.6 ± 0.4
3	-1.4 ± 0.4	-0.5 ± 0.4	0.7 ± 0.3
4	-1.4 ± 0.4	-1.3 ± 0.3	0.7 ± 0.3
5	-0.2 ± 0.8	-0.2 ± 0.8	0.7 ± 0.3
6	-1.3 ± 0.3	-0.2 ± 0.8	1.8 ± 0.8
7	-1.3 ± 0.3	-1.3 ± 0.3	1.8 ± 0.8
8	-0.2 ± 0.8	-1.3 ± 0.3	1.8 ± 0.8
9	-1.4 ± 0.4	-0.2 ± 0.8	0.6 ± 0.4
10	-1.3 ± 0.3	-1.4 ± 0.4	0.7 ± 0.3

For each bootstrap sample b=1, ..., 10, minimize $\sum_{i=1}^{3}[y^*{}_i - y(x_i, w_{11})]^2$ getting the parameter estimation for the parameter $\hat{w_{11}}$ is $\hat{w_{11}}^*$, to get the minimum error of the output, we find the derivative of the error with respect to the weight w_{11} that's

$\frac{\partial}{\partial w_{11}} \sum_{i=1}^{3}[y^*{}_i - y(x_i, w_{11})]^2 = 0,$

so $\frac{\partial}{\partial w_{11}} \sum_{i=1}^{3}[y^*_i - tanh(\sum_{i=1}^{3} w_{ij}x_i + b_i).]^2 = 0,$

so $\frac{\partial}{\partial w_{11}} \sum_{i=1}^{3}[(y^*_i)^2 - 2(y^*_i tanh(\sum_{i=1}^{3} w_i x_i + b_i) + (tanh(\sum_{i=1}^{3} w_i x_i + b_i))^2] = 0,$

so $-2y^*_i x_1 sech^2(w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + b_i) + 2x_1 tanh(w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + 1) sech^2(w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + 1) = 0$, so

$2sech^2(w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + 1)x_1[-y^*_i + tanh(w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + 1) = 0.$

By using graphs, we have the minimum of the least square error of the output when w=-.01 when $y^*{}_1 = -.2$, so the parameter estimation for the parameter $\hat{w_{11}}$ is $\hat{w_{11}}^* = -.01$ and $y(x_i, \hat{w_{11}}) = -.245$.

But when $y^*{}_1 = -1.4$ or -1.3, then the minimum of the least square error of the output when w=-.1, so the parameter estimation for the parameter $\hat{w_{11}}$ is $\hat{w_{11}}^* = -.1$ and $y(x_i, \hat{w_{11}}) = -1$.

So for b=1, the minimum of the least square error is

$\sum_{i=1}^{3}[y^*{}_i - y(x_i, \hat{w_{11}}^*)]^2 = (-1.4 + 1)^2 + (-.2 + 1)^2 + (.7 - 1)^2 = .89.$

Similarly for all of the booststrap samples in the table below

Table 5.3: Table of the minimum of least square error

b	The minimum of least square error
1	0.89
2	0.25
3	0.5
4	0.34
5	0.73
6	1.37
7	0.82
8	0.73
9	0.96
10	0.34

Then estimate the standard error of the i^{th} predicted value by

$$\left\{\sum_{b=1}^{10}\frac{[y(x_i,\hat{w_{11}}^*)-y(x_i,.)]^2}{9}\right\}^{0.5}, \text{ where } y(x_i,.)=\frac{\sum_{b=1}^{10}[y(x_i,\hat{w_{11}}^*)]}{10}$$

$$y(x_1,.)=\frac{\sum_{b=1}^{10}[y(x_1,\hat{w_{11}}^*)]}{10}=\frac{7(-1)+3(-.245)}{10}=-1.547$$

That's the estimated standard error of the i^{th} predicted value is

$$\left\{\sum_{b=1}^{10}\frac{[y(x_i,\hat{w_{11}}^*)-\frac{\sum_{b=1}^{10}[y(x_i,\hat{w_{11}}^*)]}{10}]^2}{9}\right\}^{0.5}$$

$$\left\{\sum_{b=1}^{10}\frac{[y(x_1,\hat{w_{11}}^*)-\frac{\sum_{b=1}^{10}[y(x_1,\hat{w_{11}}^*)]}{10}]^2}{9}\right\}^{0.5}=$$

$$\left\{\frac{7((-1+1.547)^2)+3((-.245+1.547)^2)}{9}\right\}^{0.5}=0.345984$$

Similarly for all of the i^{th} predictive values are shown in the table below

Table 5.4: Table of the average and the estimated standard error of the i^{th} predictive values

The i^{th} predictive values	the average	the estimated standard error
$y(x_1, \hat{w_{11}}^*)$	-1.547	0.345984
$y(x_2, \hat{w_{11}}^*)$	-1	0
$y(x_3, \hat{w_{11}}^*)$	1	0

Then we find the skewness and the kurtosis for the 1^{st} predicted value by using excel.

Skewness for the 1^{st} predicted value is

$$\frac{\sqrt{n(n-1)}}{n-2} \frac{\sum_{b=1}^{10} \frac{[y(x_i, \hat{w_{11}}^*) - y(x_i, .)]^3}{10}}{\left[\sum_{b=1}^{10} \frac{[y(x_i, \hat{w_{11}}^*) - y(x_i, .)]^2}{10}\right]^{1.5}} = 1.0351.$$

Skewness is positive so the tail of the graph of the distribution on the right as shown below in figure 4.5 and the graph not symmeteric, mean (the mean=-.20198) is to the right of the median (the median=-.99835) since skewness is positve

Kurtosis for the 1^{st} predicted value is

$$\frac{\sum_{b=1}^{10} \frac{[y(x_i, \hat{w_{11}}^*) - y(x_1, .)]^4}{10}}{\left[\sum_{b=1}^{10} \frac{[y(x_1, \hat{w_{11}}^*) - y(x_1, .)]^2}{10}\right]^{2}} - 3 = -1.2245,$$

Kurtosis is less than 3 so the graph is platykurtic

We draw the $y(x_1, \hat{w_{11}}^*)$'s, with the 1^{st} predicted value to compare them.

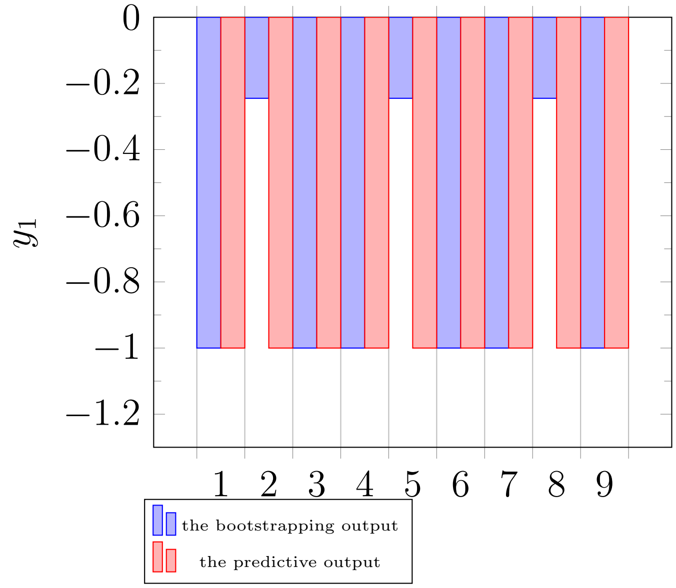

Figure 5.1: Estimation of y_1 by the bootstrapping method

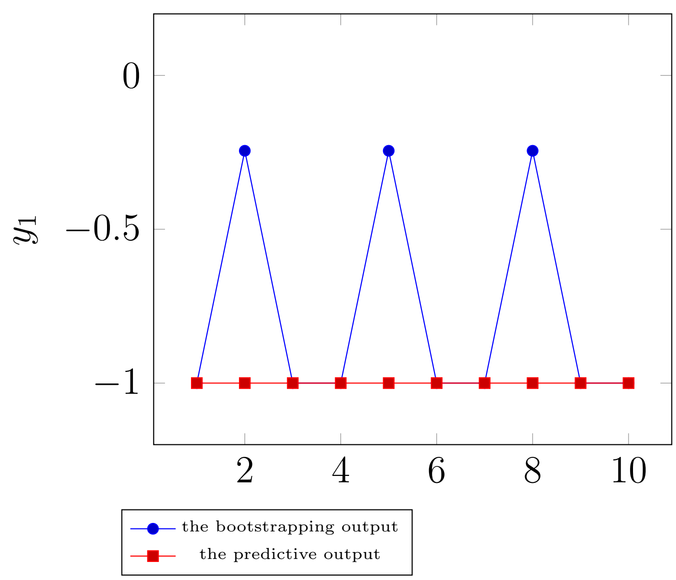

Figure 5.2: Estimation of y_1 by the bootstrapping method

5.2 Using the back-propagation method to update the weights

We say about the algorithm of the back-propagation method in section (3.4.2), we want to update the weights which we have in section (4.1), to update the weights that are on the links to the output layer, first we should calculate $\delta_{kj} = (y_j - d_j)f'(t_j)$ before calculate $\Delta w_{ij} = -\eta\delta_{kj}y_i$. Where f is the activation function in the output layer that's $2t_j + 1$, $f'(t_j) = 2, j = 1, 2, 3$ and set $\eta = 0.1$.

We have one neuron in the output layer connected to 3 neurons in the hidden layer so we have

So $t_1 = w_1 tanh(w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + 1) = -0.1$.

$t_2 = w_2 tanh(w_{12}x_1 + w_{22}x_2 + w_{32}x_3 + 1) = 0.5$.

$t_3 = w_3 tanh(w_{13}x_1 + w_{23}x_2 + w_{33}x_3 + 1) = 0.2$. $y_3 = y_2 = y_1 = f(t_3) = f(t_1) = f(t_2) = 2(-.1 + .5 + .2) + 1 = 2.2 = Y$

So $\Delta w_1 = \Delta w_2 = \Delta w_3 = -\eta\delta_{k1}y_1 = -\eta(Y - y)f\prime(t_1)y_1 = -.1 \times -.74 \times 2 \times 2.2 = 0.3256$.

We want to update the weights that are on the links between the input layer and the hidden layer, first we should calculate $\delta_j = \sum_{k=1}^{m_{l+1}} \delta_{kj}w_j f'(t_j)$ and $\Delta w_{ij} = -\eta\delta_j y_i$, where m_{l+1} is the number of the neurons in the l+1 layer which is here the output layer so $m_{l+1} = 1$, and i is the neuron in the l-1 layer and j is the neuron in the l layer.

Where f is the activation function in the hidden layer that's $tanh(t_j)$, $f\prime(t_j) = sech^2(t_j)$ and set $\eta = 0.1$.

But $t_1 = w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + 1 = -3.4$, so $y_1 = f(t_1) = tanh(-3.4) = -.9978$ and $f'(t_1) = sech^2(-3.4) = .004452$.

We have 3 neurons in the hidden layer connected to 3 neurons in the input layer so we have

So $\Delta w_{11} = -\eta\delta_1 y_1 = -\eta\delta_{k1} w_1 f'(t_1) y_1 = -0.1 \times -1.48 \times .1 \times .004452 \times -.9978 = -.00006574$.

Similarly we calculated the weights update on the links between the inputs x_i's and the neurons n_j's in the hidden layer L and the results are shown in the table below

Table 5.5: Table of the weights update w_{ij}

Inputs	neuron1	neuron2	neuron3
x_1	-0.00006574	0.00032872	-0.00013149
x_2	-3.35664×10^{-12}	1.67832×10^{-11}	6.17328×10^{-12}
x_3	9.0872×10^{-11}	-4.5436×10^{-10}	1.81744×10^{-10}

5.3 Using Monte Carlo method in estimating the error

We want to estimate the error of the recurrent neural networks.

The predictive outputs for the j^{th} neuron in the hidden layer are

$y_1(x_1, w_{11}) = tanh(w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + 1) = -1.$

$y_2(x_2, w_{11}) = tanh(w_{12}x_1 + w_{22}x_2 + w_{32}x_3 + 1) = -1.$

$y_3(x_3, w_{11}) = tanh(w_{13}x_1 + w_{23}x_2 + w_{33}x_3 + 1) = 1.$

First we choose the inputs x_i's, i =1, ..., 3 randomly by a random generator of the calculator; the set of inputs is $\{x_i, i = 1, ..., 3\} = \{39.7, 59.4, 76.6\}$, then we find the set of outputs randomly $y_i, i = 1, ..., 3,$

$y_1 = tanh(w_{11}x_1 + w_{21}x_2 + w_{31}x_3 + 1) = -.9999.$

$y_2 = tanh(w_{12}x_1 + w_{22}x_2 + w_{32}x_3 + 1) = -1.$

$y_3 = tanh(w_{13}x_1 + w_{23}x_2 + w_{33}x_3 + 1) = 1.$

But the error in the j^{th} neuron in hidden layer output is $r_i = y_i - y_i(x_i, w_{11})$.

So $r_1 = -.9999 + 1 = 0.0001$, $r_2 = -1 + 1 = 0$, $r_3 = 1 - 1 = 0$.

We generate another 9 sets of inputs randomly and all the results and its errors in the tables below (where n is the number of generating random numbers).

Table 5.6: Table of the Monte Carlo inputs generated randomly

Inputs	1	2	3	4	5	6	7	8	9	10
x_1	39.7	31	17.5	29.8	64.3	96.5	3.5	16	52.6	5.6
x_2	59.4	13.1	28.3	87.1	10.4	20.5	37.3	74.8	77.8	31.3
x_3	76.6	2.3	3	4	77.5	72.3	14.1	12.4	64.6	53

Table 5.7: Table of the Monte Carlo estimates

n	y_1	y_2	y_3
1	-0.9999 ± 0.0001	-1	1
2	-0.965 ± 0.035	-1	1
3	0.652 ± 0.348	-0.9967 ± 0.0033	0.9999 ± 0.0001
4	0.9999 ± 0.0001	0.9999 ± 1.999	0.9999 ± 0.0001
5	-1	-1	1
6	-1	-1	1
7	0.96 ± 1.96	$.997 \pm 1.997$	$.686 \pm .314$
8	0.998 ± 1.998	0.99 ± 1.99	0.995 ± 0.005
9	-0.9992 ± 0.0008	0.99 ± 1.99	0.995 ± 0.005
10	-0.992 ± 0.008	-1	0.9999 ± 0.0001

Estimate the standard error of the i^{th} predicted value by

$$\left\{\sum_{t=1}^{10} \frac{[y_i - y_i(x_i,.)]}{9}\right\}^{0.5}, \text{ where } y_i(x_i,.) = \frac{\sum_{t=1}^{10}[y_i]}{10}, \text{ so}$$

$$y_1(x_1,.) = \frac{\sum_{t=1}^{10}[y_1]}{10} = -0.235.$$

$$\left\{\sum_{t=1}^{10} \frac{[y_1 - \frac{\sum_{t=1}^{10}[y_1]}{10}]}{9}\right\}^{0.5} == 0.933034,$$

Similarly we calculated the average and the estimated standard error for all of the y_i's and shown in the table below

Table 5.8: Table of the average and the estimated standard error of the estimated outputs

The estimated output	the average	the estimated standard error
y_1	-0.235	0.933034
y_2	-0.20198	0.976702
y_3	0.96757	0.093877

Then we find the skewness and the kurtosis for the i^{th} resulted value by using excel

Skewness for y_1 is $\frac{\sqrt{n(n-1)}}{n-2} \frac{\sum_{b=1}^{10} \frac{[y_1 - y(x_1,.)]^3}{10}}{\left[\sum_{b=1}^{10} \frac{[y_1 - y(x_1,.)]^2}{10}\right]^{1.5}} = 0.51684.$

Kurtosis for y_1 is $\frac{\sum_{b=1}^{10} \frac{[y_1 - y(x_1,.)]^4}{10}}{\left[\sum_{b=1}^{10} \frac{[y_1 - y(x_1,.)]^2}{10}\right]^{2}} - 3 = -2.17679.$

Similarly we calculated the skewness and the kurtosis for all of the y_i's and shown in the table below

Table 5.9: Table of the skewness and the kurtosis of the estimated outputs

The estimated output	the skewness	the kurtosis
y_1	0.51684	-2.17679
y_2	0.484148	-2.27669
y_3	-3.15972	9.98804

We draw the $y^*{}_i$'s, with the i^{th} predicted value to compare

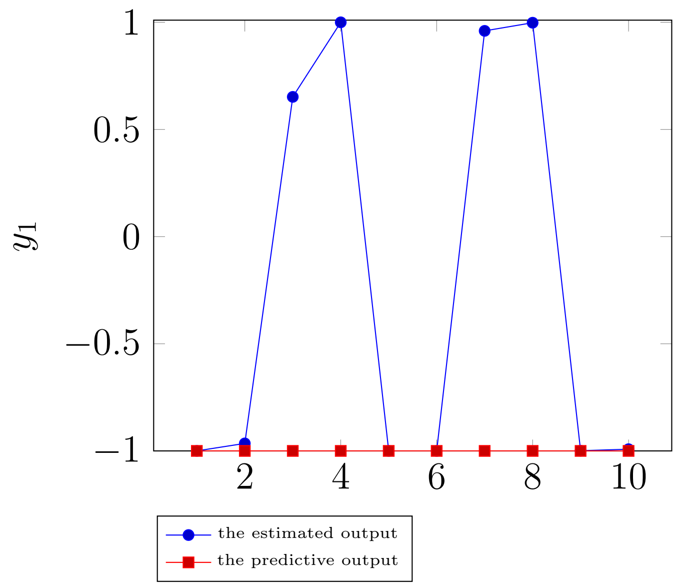

Figure 5.3: Estimation of y_1 by the Monte Carlo method

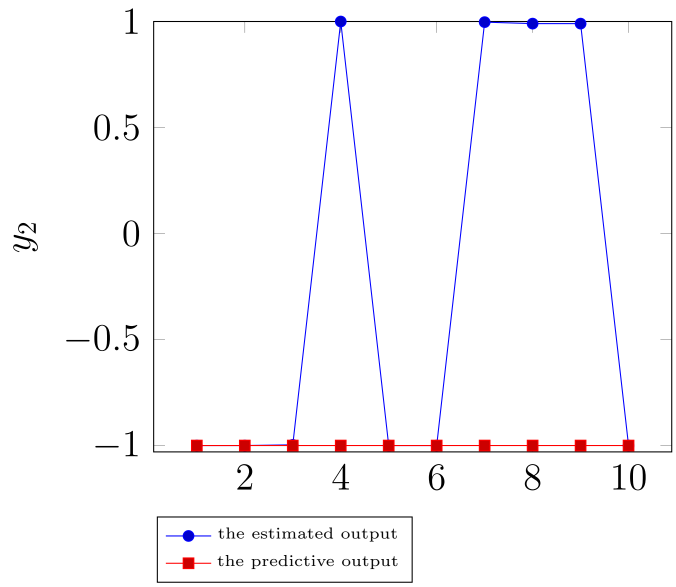

Figure 5.4: Estimation of y_2 by the Monte Carlo method

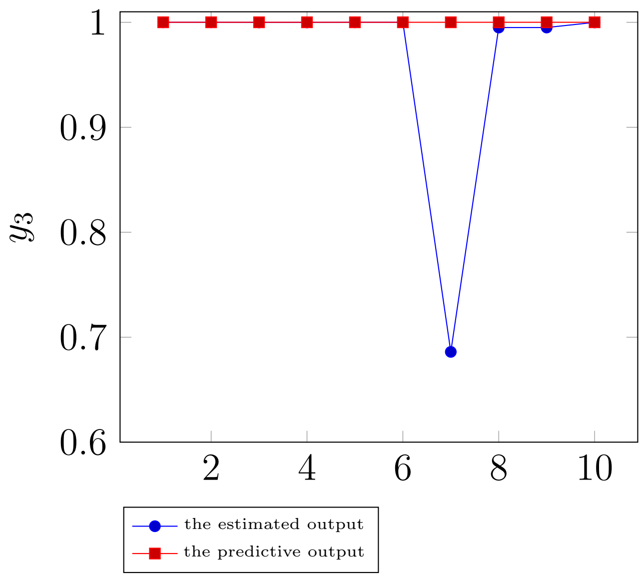

Figure 5.5: Estimation of y_3 by the Monte Carlo method

Table 5.10: Table of comparison between the Monte Carlo method and the bootstrapping

	the Monte Carlo method			*The bootstrapping*		
The statistical property	y_1	y_2	y_3	$y(x_1, \hat{w_{11}}^*)$	$y(x_2, \hat{w_{11}}^*)$	$y(x_31, \hat{w_{11}}^*)$
The standard error	0.933034	0.976702	.093877	0.345984	0	0
The skewness	0.51684	0.484148	-3.15972	1.0351	-	-
The kurtosis	-2.17679	-2.27669	9.98804	-1.2245	-	-

5.4 Discussion and Conclusion

In this thesis we explained the recurrent neural networks and the methods used in parameter estimation, then give an example of one hidden layer neural recurrent network that has three inputs, three neurons in hidden layer with tanh function as an activation function in it, and one neuron in the layer output with linear activation function and compared between the Monte Carlo method and between the bootstrapping method in estimating the output of the hidden layer and the estimated standard error of the estimated output, also compute the skewness and the kurtosis of the estimated outputs.

Skewness for y_1 is positive so the tail of the graph of the distribution on the right and the graph not symmeteric, mean (the mean=-.23462) is to the right of the median (the median=-.9785) since skewness is positive.

Kurtosis for y_1 is less than 3 so the graph is platykurtic.

Skewness for y_2 is positive so the tail of the graph of the distribution on the right as shown below in figure 4.5 and the graph not symmetric, mean (the mean=-.20198) is to the right of the median (the median=-.99835) since skewness is positive.

Kurtosis for y_2 is less than 3 so the graph is platykurtic.

Skewness for y_3 is negative so the tail of the graph of the distribution is longer on the left as shown below in figure 4.6 and the graph not symmetric, mean (the mean=.96757) is left to the median (the median=.9999) since skewness is negative.

Kurtosis for y_3 is higher than 3 so the graph is leptokurtic.

We would say that Bootstrapping is a type of Monte Carlo simulation for a very specific purpose: Estimate some characteristics of the sampling distribution where you are estimating the distribution of a sample statistic. Also Monte Carlo and bootstrapping both are based on repetitive sampling and direct examination of the results.

From the strategy of bootstrapping and Monte Carlo method, we see that bootstrapping uses the original initial sample as the population from which to resample and is used for choosing the best group of inputs (by replacement) from the input sample, but Monte Carlo method is used to choose the best input from random number so it is used in wider range of applications and complex ones.

The bootstrapping method is simple and straightforward way to estimate the standard error to estimate parameters and standard errors when there isn't enough statistical theory and it doesn't provide more information about the original data, the bootstrapping method doesn't need large sample size. Randomization is where you choose values from a population of data without replacement. If all the values are chosen, then the statistical characteristics are the same as those for the original observations so the output may depend on the representative sample and the bootstrapping can be time-consuming.

The bootstrapping is non-parametric computer intensive statistical method that uses a unique finite sample to describe the variability of a statistic without making any distributional assumptions about the data.

Monte Carlo Simulation is straightforward way generates large number of random numbers for inputs so need more time and it's slow to get precision in the output, and that precision doesn't depend on the number of inputs, but it's flexible (we can change inputs to get the best solution). It is also easy to see which input or the combination of inputs have the biggest effects on results.

In the Monte Carlo method, we should give all the data about the inputs, constrains and conditions for testing it in order to try to reduce the range of the random variables so we can't study the behavior of the output when the initial parameters are changed, we may get unrealistic results that can't be explained.

For large number of generating number for inputs, Monte Carlo is more accurate than the bootstrapping, the Monte Carlo can use any statistical distribution so it's applicable for large and complex systems without simplifications and also Monte Carlo can do many simultaneous simulations on many computers processors, each simulation is dependent of the other, it has time compression property, the Monte Carlo method has computational costs depend on the complexity of the problem and expensive for small applications that can be longer time to develop

References

[1] Anagnostopoulos, S. (2010): Artificial neural networks for data classification. International Hellenic university, Greece.

[2] Baron, M. (2014): Probability and statistics for computer scientist, 2nd edition. CRC press, USA.

[3] Barreto H. and Howland F. (2006): Introductory economics, using Monte Carlo simulation with Microsoft Excel, 2nd edition. Cambridge university press, UK.

[3] Box, G., Jenkins, M., Reinsel, G. and Ljung, M. (2016): Time series analysis. Forecasting and control, 5th edition. Wiley, USA.

[4] Brockwell, P. and Davis, R. (2002): Introduction to time series, 2nd edition. Springer, USA.

[5] Cryer, J. and Chan K. (2008): Time series analysis with applications in R, 2nd edition. Springer, USA.

[6] Davis, R., Coole, T. and Osipyw, D. (2014): "The application of time series modeling and Monte Carlo simulation: Forecasting volatile inventory requirements", applied mathematics, pp 1152-1168.

[7] Edara, P. (2003): Mode choice modeling using artificial neural networks, Virginia polytechnic institute and state university, USA.

[8] Efron, B. and Tibshirani, R. (1986): Bootsrap methods for standard errors, confidence intervals and other measures of statistical accuracy, statistical science, 1, pp 54-75.

[9] Estep, M. (2006): Self-Organizing Natural Intelligence, Issues of Knowing, Meaning, and Complexity. 1st edition. Springer. USA.

[10] Falk, M. (2011): A first course on time series analysis with SAS examples, 1st edition.

[11] Fausett, L. (N.D): Fundamentals of neural networks, architectures, algorithms and applications.

[12] Graves, A. (2008): Supervised sequence labeling with recurrent neural networks, Technische universitat munchen, Germany.

[13] Guli, A. and Pal, S. (2017): Deep learning with Keras, 1st edition. Pacht, UK.

[14] Heaton, J. (2015): Artificial intelligence for humans, Neural networks and Deep learning, 1st edition. Vol.3. Heaton research. UK.

[15] Hipel, K. and Mcleod, A. (1994): Time series modeling of water resources and environmental systems. Elsevier, Netherlands.

[16] Kaumanns, F. (2016): Assessment and analysis of the applicability of recurrent neural networks to natural language understanding with a focus on the problem of coreference resolution. Ludwig Maximilian university of Munich, Germany.

[17] Kim, P. (2017): Matlap deep learning with machine learning, neural networks and artificial intelligence, 1st edition. Apress, USA.

[18] Koster, F. (1999): The bootstrap approach to autoregressive time series analysis. University of Pretoria, South Africa.

[19] Kubat, M. (2017): An introduction to machine learning, 2nd edition. Springer, Switzerland.

[20] Lewis, N. (2016): Deep time series forecasting with Python, 1st edition.

[21] Livshin, I. (2019): Artificial neural networks with Java, 1st edition. Apress, USA.

[22] Michelucci, U. (2018): Applied deep learning, 1st edition. Apress, USA.

[23] Pascanu, R. (2014): On recurrent and deep neural networks, University of Montreal, Canada.

[24] Pattern, J. and Gibson, A. (2017). Deep learning, 1st edition. O'Reilly, Canada.

[25] Shumway, R. and Stoffer, D. (2015):Time series analysis and its applications with R examples, 3rd eition. Springer, USA.

[26] Sokolowski, J. and Banks, C. (2010): Modeling and simulation fundamentals, 1st edition. Willey, USA.

[27] Srivastava, N., Hinton, G., Sutskever, I. and Salakhutdinov, R. (2014): "Dropout: A simple way to prevent neural networks from overfitting", Journal of machine learning research,15, pp. 1929-1958.

[28] Suykens, J., Vandewalle, J. and DE Moor, B. (1996): Artificial neural networks for modelling and control of non-linear systems, 1st edition. Springer science and business media Dordrecht, Netherlands.

[29] Tebbs, J. (2013): Forecasting and time series. University of South Carolina, USA.

[30] Tibshirani, R. (1995): A comparison of some error estimates for neural networks models. University of Toronto, Canada.

[31] Vojt, J. (2016): Deep neural networks and their implementation. Charles university, Czech.

[32] Woodward, W., Gray, H. and Elliott, A. (2017): Applied time series with R, 2nd edition. CRC press, USA.

تعميم السلاسل الزمنية ذات المتغيرات المتعددة

مع تطبيقات الشبكات العصبية المتكررة

اعداد : صفاء نادر مصطفى شناعة.

اشراف: الدكتور خالد صلاح.

الملخص

بيانات السلاسل الزمنية ذات المتغيرات المتعددة تدخل في تطبيقات عملية مثل الرعاية الصحية وعلوم الارض والهندسة والاحياء وغيرها . هذه الرسالة تعرض تطور الاحصاء من تحليل السلاسل الزمنية الى الشبكات العصبية المتكررة ، مجال تحليلي ضروري لفهم وتنبؤ سلوك المتغيرات في مختلف المجالات .

بداية تم البحث والمناقشة في خصائص واساسيات بيانات لاسلاسل الزمنية بما تتضمنه من نماذج مختلفة من السلاسل الزمنية مثل نموذج الانحدار الذاتي AR ونموذج المتوسط المتحرك MA ونموذج الانحدار الذاتي المتوسط المتحرك ARMA ونموذج الانحدار الذاتي المتكامل المتوسط المتحرك ARIMA. غالبا نتمنى ان نجد نموذج بارامتري يناسب بيانات السلاسل الزمنية ويحدد قيم دقيقة للمتغيرات وتقديرات صحيحة لأوجه عدم اليقين في هذه المتغيرات . مهم جدا ان نأخذ فهما عميقا عن الضجيج ونطور الطرق المناسبة لتقريب المتغيرات لذلك تحدثنا في هذه الرسالة عن طرق كثيرة لتقريب المتغيرات مثل معادلات يول ووكر وطريقة المربعات الصغرى وطريقة العزوم وطريقة التقدير حسب الاحتمال الاعظم.

ثانيا: تم دراسة مختلف تقنيات النمذجة للسلاسل الزمنية لتتناول مواضيع مختلفة تهم الباحثين في الشبكات العصبية الاصطناعية تتضمن وصف نمط التغير في المتغيرات ونمذجة الاثار الموسمية وتقييم الاثر الفوري وطويل الاجل لحدث بارز وتوقع قيم مستقبلية . تركيب الشبكات العصبية الاصطناعية وخاصة الشبكات العصبية المتكررة تم مناقشتها بالتفصيل في هذه الرسالة وما تتضمنه من وحدات متكررة بوابية GRUs وذاكرات طويلة وقصيرة الامد LSTMs وخصائصهم وبعض المشاكل التي تواجه الشبكات العصبية المتكررة مثل تدرجات التلاشي والافراط في الملاءمة.

لتوضيح هذه الطرق فقد عملنا في هذا البحث تطبيق توضيحي يستند الى طريقة مونتي كارلو وطريقة البوتستراب بتصميم شبكة عصبية متكررة مكونة من طبقة مخفية واحدة وتطبيق عليها الانتشار الخلفي وطريقة مونتي كارلو وطريقة البوتستراب وتقدير التباين للخطأ في كل من الطريقتين ومقارنتهما.