Approximating European options using Artificial Neural Network and Finite Difference Method

Letladi William Rathaga (letladi@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Prof. Phillip Mashele
University of North West, South Africa

24 October 2019

Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

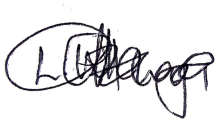
Abstract

In finance, pricing options has always been a topic of great interest for practitioners and researchers. Thereby, in 1973, Fisher Black and Myron Scholes formulated the famous Black-Scholes model (Black and Scholes, 1973) for pricing option. The derivation of the Black Scholes model leads to a second order Partial Differential Equation (PDE). In the literature, many numerical methods such as finite difference method (Duffy, 2013) have been used to solve the Black-Scholes PDE, however these numerical methods face the "curse of dimensionality" when it comes to solving the Black-Scholes PDE for multi-asset option. In this study, we introduce a machine learning method called Artificial Neural Network (ANN) for solving differential equtions. The aim of this project is to use the ANN method to solve the Black-Scholes PDE for European call option and compare it to the finite difference method.

Keywords: Black-Scholes, Artificial Neural Network, Finite Difference Method, European call option

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.



Letladi William Rathaga, 24 October 2019

Contents

1. Introduction

1.1 Background

Options are one of the most important derivatives in finance. Options are normally considered as the derivative instruments since the value of an option is actually obtained from a specific security. The securities that can be traded as an option are known as interest rates, bonds, indexes and commodities, to name few(Ianieri, 2009).

An option is simply a contract that grant the owner a right to trade an underlying security at a fixed agreement price and a specific agreed date, but the owner is not obligated to trade. There are variety of options in finance where the exercise time of the option is a main factor that makes them different from each other. The most common used options in theory and practical are known as European and American options. The difference between these two options is that European option have a fixed exercise time while American option gives owner the right to exercise at any time within the option life. American and European options are similar if an American option is exercised at expiry date.

There are several mathematical models that are used to price an option contract. One of those common used models is Black-Scholes model(Lütkebohmert, 2004). This model was discovered by Black and Scholes in 1973 (Black and Scholes, 1973) and the model was later modified by Merton on the same year of discovery. The analytical solution of the Black-Scholes partial differential equation only exists in European-style options as the expiry date is fixed. For American-style option, the solution of Black-Scholes partial differential equations are mostly approximated by numerical methods. The factors that need to be considered in Black-Scholes option pricing are the current underlying stock price, volatility, an agreement price ,dividends and risk-free rates.

1.2 Objective of the study

The objective of the study is to evaluate the value of European call option. The evaluation will be carried out by using two different methods namely Artificial Neural Network (ANN) method and Numerical method such as Finite Difference. Over the past years numerical methods have been used to solve Black-Scholes PDEs but they face the curse of dimensionality so we will employ method of Artificial Neural Network since (Jentzen et al., 2018) indicated that Deep Neural Networks (DNNs) are flexible to overcome the curse of dimensionality. We will also compare the analytical solution with approximated results from Artificial Neural Network and finite difference results to check the an accuracy.

1.3 Essay Structure

The structure of essay is carried out in this way, in chapter 2, we give a brief on definitions and notations that will be helpful in this study, also we will look at option theory. In chapter 3 we give an understanding of ANN techniques and Finite Difference Method. In chapter 4 we represent ANN and FDM methods results, the last chapter is conclusion from our findings.

1.4 Motivation Of The Study

Pricing an option has been an area of interest for researchers in the field of mathematical finance. There are several methods that have been employed to price an option by approximating the solution of the

Black-Scholes equation. The numerical methods are the most popular used techniques to approximate the solution of Black-Scholes equation. The motive of this study is to check wheather the approximated solution from Artificial Neural network method is close to the analytical solution so that the method can be employed to approximate the solution of Black-Scholes for American-style option.

2. Basic Notations and Option Pricing Theory

In this Chapter, we provide an understanding of the important notations and definitions that will useful in this study such as Probability Theory, Stochastic Process and lastly Option Theory.

2.1 Probability Theory

2.1.1 Definition. Sample space (Durrett, 2019) A Sample space is the set of all possible outcomes and it is denoted by Ω. The elements of a sample space are denoted by ω.

2.1.2 Definition. σ-Field (Durrett, 2019) Consider a sample space Ω, a collection $\mathcal{F}$ of subsets of Ω is called σ-field if it satisfies the following conditions:

(i). $\Omega \in \mathcal{F}$

(ii). $A \in \mathcal{F} \implies A^c \in \mathcal{F}$

(iii). $A_n \in \mathcal{F}$ for $n = 1, 2, 3, ... \implies \cup_{n=1}^{\infty} A_n \in \mathcal{F}$

The set $(\Omega, \mathcal{F})$ is called a measurable space.

2.1.3 Definition. Borel σ-field (Durrett, 2019) The smallest σ-field that contains all open intervals in $\mathbb{R}$ is called Borel σ-field and it is denoted by $\mathcal{B}(\mathbb{R})$.

2.1.4 Definition. Probability Measure (Durrett, 2019) Consider a σ-Field $\mathcal{F}$ and a sample space Ω, a real-valued function P is called Probability Measure on $\mathcal{F}$ if it satisfies the following conditions;

(i). $P(\Omega) = 1$

(ii). $0 \leq P(A) \leq 1$ for $A \in \mathcal{F}$

(iii). For $A_1, A_2, A_3, ..., \in \mathcal{F}$ with $A_i \cap A_j = \emptyset$ for $i \neq j$

$$P(\cup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i)$$

2.1.5 Definition. Probability Space (Durrett, 2019) A triple space $(\Omega, P, \mathcal{F})$ is a probability space formed with the following elements;

(i). Sample Space Ω which contains all possible outcomes

(ii). $\mathcal{F}$ contains all events of Ω

(iii). P is a Probability measure which range is between 0 and 1, i.e, $[0, 1]$

2.1.6 Definition. Random Variable (Focardi and Fabozzi, 2004) For a given probability space $(\Omega, P, \mathcal{F})$, a random variable X is defined as a measurable function $X(\omega)$ described over the sample space Ω that takes values in $\mathbb{R}$:

$$(\omega : X(\omega) \leq x) \in \mathcal{F}$$

2.2 Stochastic Process

2.2.1 Definition. Stochastic Process (Shreve, 2004) A stochastic process indexed by a set T, with values in a measurable space (E, ϵ) is a collection $X = (x_{t \in T})$ of measurable maps from a probability space to (E, ϵ).

2.2.2 Definition. Brownian Motion (Focardi and Fabozzi, 2004). Let $W = (W_t)_{t \geq 0}$ be a stochastic process, then W is said to be Brownian Motion or Weiner process if the following conditions hold:

1. $W_0 = 0$
2. W is a stochastic process with stationary and independent increments.
3. $W_t - W_s$ is distributed with $N(0, t-s)$ for $0 \leq s \leq t$

2.2.3 Definition. Geometric Brownian Motion(Dunbar, 2016). is the continuous time stochastic process $X(t) = z_0 e^{\mu t + \sigma W(t)}$ where $W(t)$ is standard Brownian Motion

2.2.4 Definition. Stochastic Differential Equation (Shreve, 2004) SDE is an equation of the form;

$$dX(u) = B(u, X(u))du + y(u, X(u))dW(u)$$

Where the functions $B(u, X(u))$ and $y(u, X(u))$ are known as drift and diffusion respectively and W is Brownian Motion

2.2.5 Lemma. Ito's lemma (Focardi and Fabozzi, 2004) Consider the process $X_{t \in [0,T]}$ with Stochastic Differential Equation(SDE) $dX_t = a(X_t)dt + b(X_t)dW_t$. For a function $f(t, x)$ with at least one derivative in t and at least two derivatives in x, we have,

$$df(t, X_t) = \left(\frac{\partial}{\partial t} + a(X_t)\frac{\partial}{\partial x} + \frac{b^2(X_t)}{2}\frac{\partial^2}{\partial x^2} \right) f(t, X_t)dt + b(X_t)\frac{\partial}{\partial x} f(t, X_t)dW_t \tag{2.2.1}$$

2.3 Finance Definitions

2.3.1 Definition. Volatility of a stock (Hull et al., 2009) is defined as a measure of unpredictability of returns of an underlying stock. It is denoted by σ also known as Standard deviation of the returns.

2.3.2 Definition. The drift rate is the average movement rate of an underlying assets and it is denoted by μ.

2.3.3 Definition. Delta of a Stock (Hull et al., 2009) is the ratio of the change in the price of the stock option to the change in the price of the underlying stock and it is denoted by $\triangle$.

2.3.4 Definition. Forward contract (Hull et al., 2009) An agreement to trade a certain underlying asset at specific time known as future time for a certain fixed price. When someone agrees to buy a certain underlying asset, the contact is called **long forward position** and when someone agrees to sell a certain underlying asset, the contract is called **short forward position**

2.4 Option Theory

An option is a contract which provides the owner with the right to buy or sell an underlying asset at a specified price at a specific date. The main feature of an option is that the owner of an option can choose to exercise the right or allow an option to expire since it is a right not an obligation. If the option owner is obligated to buy or sell, then the contract becomes a forward contract. Stock, property, and financial instruments are few examples of underlying assets.

In general, option contracts are considered as an agreement between two parties which are the holder(buyer) and the writer(seller). Long the position refers to someone who buys an option contract and short the position refers to someone who sells an option contract.

Options are divided into two categories namely call and put option. A call option is a contract that gives the buyer the right to buy a specific security at a specific price by the expiry date. A put option is a contract that allows the buyer to sell a specific security at a specific price over a period of time. The date at the end of the contract is called expiry date of an option or maturity date. The agreement price that gives the buyer or seller the right but not an obligation to buy or sell the underlying security, is called strike price or exercise price denoted by K. A case where the strike price, stock price and expiration date for call and put option are the same is called put-call parity. In an open market, option holder is allowed to trade the position at any time.

There are three ways to describe the behaviour of an option before the expiry date which are given as follows, In The Money(ITM), On The Money(OTM) and At The Money(ATM). These three cases of an option are called Time Value Option.

At The Money(ATM) option(Ianieri, 2009) is a case where the exercise price is equal to the current underlying stock price. This is a case where call payout is equal to the put payout.

In The Money(ITM) option (Ianieri, 2009) is a case where the current underlying stock price is greater than the strike price which is profitable to exercise at this time for a call option. For put option, it is a case where strike price is greater than stock price which is profitable to exercise at this time.

On The Money(OTM) option (Ianieri, 2009) is a case where the strike price is greater than the current underlying stock price for a call option which is not profitable to exercise at this time. For a put option, is a case where the stock price is greater than the strike price which is not profitable to exercise at this time. Table 2.1 gives the summary of behaviour of an option.

	Put	**Call**
In-the-money	Stock price is less than strike price	Stock price greater than strike price
On-the-money	Stock price greater than strike price	Stock price is less than strike price
At-the-money	Stock price is equal to strike price	Stock price is equal to strike price

Table 2.1: The behaviour of an option

In finance, the options are distinguished by the exercise time. An option that can be exercised at any time by a holder before expiry date is called American option. An option where a holder can only exercise at the expiry date is called European option. Asian option is the type of option where the pay-off of the option depends on the average of the underlying stock over period of time. An option that allows the holder to exercise more than once before expiry date is called Bermudan. The most popular used options are American and European options, where in this study we consider European option.

European Options are known as options that can only be exercised at the end of option life or expiry date, unlike in American-style options where the owner of an option have right to exercise at any time as long as it is profitable to at that time within the option life. A pay-off of the European option is equivalent to the American option pay-off if the American option is exercised at the expiry date . European options are considered as a simple option to analyse (Emmanuel et al., 2014). European options are often used in stock and foreign currency and they are considered as cheaper options compare to American option.

The current value of European option can be obtained by solving option pricing model such as Black-Scholes equation. The European option-style is considered as easy option to model using Black-Scholes as the analytical solution is known unlike in American Option-style where the solution of Black-Scholes model is approximated by numerical methods. There are factors that need to be considered to obtain the value of European option using classical Black-Scholes model such as current underlying stock price, risk free rate, volatility of stock, and time until expiry date.

The main idea of trading of an option is to make profit that come as pay-off minus the premium paid. The pay-off of European call option at the end of a contract is given as $max(S - K, 0)$ which simply indicate that the buyer will get a pay-off as the difference between current stock and strike price or 0 otherwise. For European put option, the seller will receive pay-off of $max(K - S, 0)$, which suggest that the greater the strike price the better the profit. The following figures 2.1 and 2.2 illustrate the behaviour of call and put option separately.

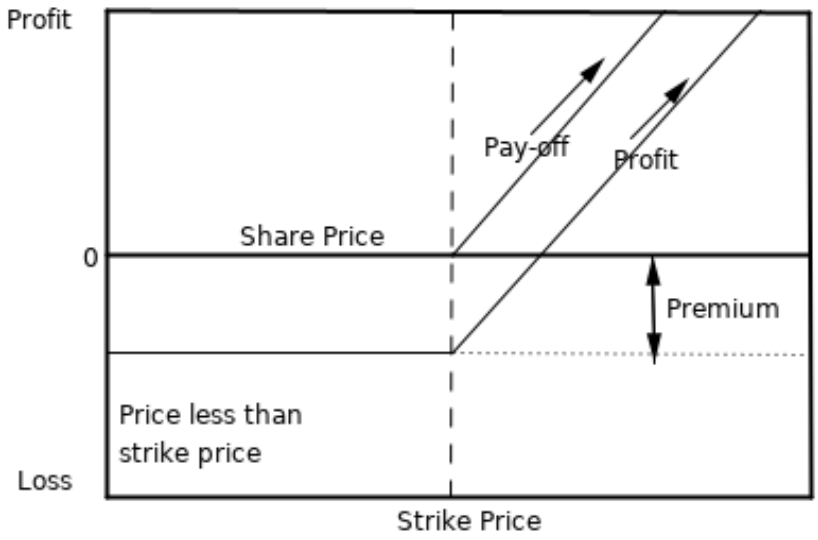


Figure 2.1: Call option

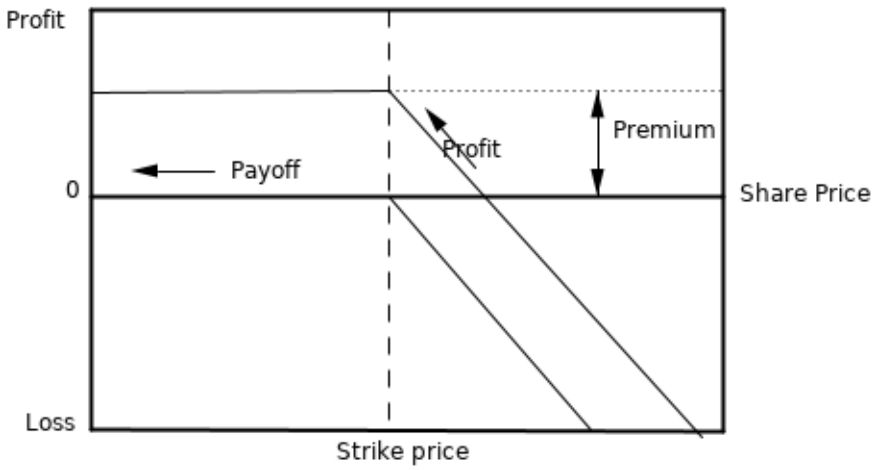


Figure 2.2: Put option

There are several studies that were conducted to model European option with different popular known methods. Cui and Yu (2012) has conducted study on simulating European call option's sensitivity which was based on Black-Scholes option pricing mode. Bohner et al. (2014) used Adomian Decomposition method to price the European call option in risk-neural world. Klar and Jacobson (2002) suggested Monte-Carlo simulation and numerical analysis method to price European call option by determining the numerical solution of the Black-Scholes option pricing model. Jentzen et al. (2018) used a different approach to the European option pricing model by solving Fractional Black-Scholes (FBS) and Generalized Fractional Black-Scholes (GBS) using homotopy pertubation and Adomian decomposition methods.

2.5 Derivation of Black-Scholes Pricing Model

Black-Scholes model is a mathematical model that is used for option pricing in financial market(Black and Scholes, 1973). The model is governed by the following assumptions(Hull et al., 2009):

(i). The stock price follows Geometric Brownian Motion

(ii). Asset Volatility and risk-free are known functions

(iii). Short selling is allowed.

(iv). the number of assets to be sold or bought is not limited

(v). Zero transaction costs

Before deriving black-Scholes PDE, let's first define the following important notations:

(i). $S(t)$ - Underlying stock price at time $t \in [0, T]$

(ii). σ- Volatility of the underlying stock returns

(iii). K - Strike price

(iv). r -The risk free interest rate

(v). T- The time to expiry

(vi). $\tau = T - t$ - The remaining time to expiry.

(vii). $V(S,t)$ - the value of an option.

To derive Black-Scholes PDE, we consider the fact that the underlying stock price follows a Geometric Brownian motion,

$$dS = \mu S dt + \sigma S dZ \tag{2.5.1}$$

where $dZ = \epsilon\sqrt{dt}$ and ϵ is normally distributed with mean 0 and variance 1. Next, we will need Ito's lemma, which is regarded as a change in value of option as function of underlying assets price in the time,

$$dV = \frac{\partial V}{\partial S}dS + \frac{\partial V}{\partial t}dt + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}dS^2 + \frac{1}{2}\frac{\partial^2 V}{\partial t^2} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}dtdS \tag{2.5.2}$$

Lastly, we consider the Delta-hedge portfolio which consist of two terms , a long position in call and a short position in stock.

$$\Pi = V - \triangle S \tag{2.5.3}$$

From 2.5.1, we have,

$$dS^2 = \mu S^2 dt^2 + \sigma^2 S^2 dZ^2 + 2\mu\sigma S dt dZ \tag{2.5.4}$$

dt is very small in a way that $(dt)^2$ approaches zero so, we have $(dt)^2 \approx 0$ which eliminate the term $\mu S^2 dt^2$ from 2.5.4 The term $dZ^2 = \epsilon dt \approx dt$ and the term $dtdZ = \epsilon(dt)^2 \approx 0$. Now equation 2.5.4 has been reduced to

$$dS^2 = \sigma^2 S^2 dt \tag{2.5.5}$$

From equation 2.5.1, we observe that,

$$dtdS = dt(\mu S dt + \sigma S dZ) \tag{2.5.6}$$
$$= \mu S(dt)^2 + \sigma S dt dZ \tag{2.5.7}$$
$$= 0 + 0 = 0 \tag{2.5.8}$$

Substituting 2.5.5 into 2.5.2, we have

$$dV = \frac{\partial V}{\partial S}dS + \frac{\partial V}{\partial t}dt + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}dt \tag{2.5.9}$$

From 2.5.3, the change in portfolio value at the given time interval is given by

$$d\Pi = dV - \triangle dS \tag{2.5.10}$$

Substituting 2.5.9 into 2.5.10, we get

$$d\Pi = \frac{\partial V}{\partial S}dS + \frac{\partial V}{\partial t}dt + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}dt - \triangle dS. \tag{2.5.11}$$

For any risk-less variable, the rate of return must always be equal to the change in portfolio value, $d\Pi = r\Pi dt$. $\triangle = \frac{\partial V}{\partial S}$, Then we can write equation 2.5.11 as

$$r\Pi dt = \left(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}\right)dt$$

$$r\left(V - S\frac{\partial V}{\partial S}\right) = \frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}$$

$$rV = \frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} \tag{2.5.12}$$

The equation 2.5.12 is called Black-Scholes Partial Differential Equation, where the solution of this PDE can be obtained numerically following important boundary conditions: For a call option,

$$V(S,T) = max\{S_T - K, 0\}$$

$$V(0,t) = 0$$

$$V(\infty,t) \sim S_t - Ke^{-r(T-t)}$$

For a put option,

$$V(S,T) = max\{K - S_T, 0\}$$

$$V(0,t) = Ke^{-r(T-t)}$$

$$V(\infty,t) \sim 0$$

Hull et al. (2009),the closed form solution of Black-Scholes equation for European option at maturity time T on a non-dividend paying stock is given as

$$C = S_0 N(d_1) - Ke^{-rT}N(d_2)$$

for a call option and for a put option

$$P = Ke^{-rT}N(-d_2) - SN(-d_1)$$

where

$$d_1 = \frac{ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

$N(.)$ is the cumulative probability distribution function for a standardized normal distribution.

3. Finite Difference and Artificial Neural Network method

3.1 Finite Difference Method

The finite difference method (FDM) is one of the numerical methods that are used to approximate the solution of differential equations. The methods solve an equation by discrete points or grid. So in our case, the domain of space and time are given as $[0, x_{max}]$ and $[0, T]$ respectively. The uniform step size of space is h and for time we have $\triangle t$. The discretizations of the domain of space and time are given as follows:

$$x = 0, h, 2h, ..., Nh, (N+1)h$$
$$\tau = 0, \triangle t, 2\triangle t, ..., M\triangle t, (M+1)\triangle t$$

The terminals $(N+1)\triangle t$ and $(M+1)\triangle t$ correspond to x_max and T. The value of an option on a discrete time and space is given as $V_{i,j}(i\triangle t, jh)$ where $i = 0, 1, 2, ..., M, M+1$ and $j = 0, 1, 2, ..., N, N+1$

The finite difference methods have various different approaches to solve a differential equation such as backwards, forward and central difference. The equations for these approaches are as follows:

(i) Backward difference:

$$\frac{\partial V}{\partial x} = \frac{V_{i,j} - V_{i,j+1}}{h} \tag{3.1.1}$$

(ii) Forward difference:

$$\frac{\partial V}{\partial x} = \frac{V_{i,j+1} - V_{i,j}}{h} \tag{3.1.2}$$

(iii) Central difference:

$$\frac{\partial V}{\partial x} = \frac{V_{i,j+1} - V_{i,j-1}}{2h} \tag{3.1.3}$$

(iv) second order derivation approximation.

$$\frac{\partial^2 V}{\partial x^2} = \frac{V_{i,j-1} - 2V_{i,j} + V_{i,j+1}}{h^2} \tag{3.1.4}$$

The above derivations were taken with respect to the underlying assets x only.

Now to apply the finite difference method, let's consider Black-Scholes PDE with $\tau = T - t$.

$$\frac{\partial V}{\partial \tau} = \frac{1}{2}x^2\sigma^2\frac{\partial^2 V}{\partial x^2} + xr\frac{\partial V}{\partial x} - rV \tag{3.1.5}$$

Discretizing the underlying stock price x first, the central difference approach method is used to approximate the underlying stock price x for first order derivative and second order derivation approximation,

$$\frac{\partial V}{\partial x} \approx \frac{V_{j+1} - V_{j-1}}{2h} \tag{3.1.6}$$

$$\frac{\partial^2 V}{\partial x^2} \approx \frac{V_{j+1} - 2V_j + V_{j-1}}{h^2} \tag{3.1.7}$$

Substituting into 3.1.5, We get,

$$\frac{\partial V_j}{\partial \tau} = \frac{1}{2}x_j^2\sigma^2\left(\frac{V_{j+1} - 2V_j + V_{j-1}}{h^2}\right) + x_j r\left(\frac{V_{j+1} - V_{j-1}}{2h}\right) - rV_j \tag{3.1.8}$$

$$\frac{\partial V_j}{\partial \tau} = \frac{x_j^2\sigma^2}{2h^2}V_{j+1} - \frac{x_j^2\sigma^2}{h^2}V_j + \frac{x_j^2\sigma^2}{2h^2}V_{j-1} + \frac{rx_j}{2h}V_{j+1} - \frac{rx_j}{2h}V_{j-1} - rV_j \tag{3.1.9}$$

Rearranging 3.1.9, we get

$$\frac{\partial V_j}{\partial \tau} = \left(\frac{x_j^2\sigma^2}{2h^2} + \frac{rx_j}{2h}\right)V_{j+1} + \left(\frac{-x_j^2\sigma^2}{h^2} - r\right)V_j + \left(\frac{x_j^2\sigma^2}{2h^2} - \frac{rx_j}{2h}\right)V_{j-1} \tag{3.1.10}$$

$$\frac{\partial V_j}{\partial \tau} = a_j V_{j+1} + b_j V_j + c_j V_{j-1} \tag{3.1.11}$$

where,

$$a_j = \left(\frac{x_j^2\sigma^2}{2h^2} + \frac{rx_j}{2h}\right) \tag{3.1.12}$$

$$b_j = \left(\frac{-x_j^2\sigma^2}{h^2} - r\right) \tag{3.1.13}$$

$$c_j = \left(\frac{x_j^2\sigma^2}{2h^2} - \frac{rx_j}{2h}\right) \tag{3.1.14}$$

Equation 3.1.11 can be written in expanded from as follows

$$\begin{cases} \frac{\partial V_1}{\partial \tau} & = a_1 V_2 + b_1 V_1 + c_1 V_0 \\ \frac{\partial V_2}{\partial \tau} & = a_2 V_3 + b_1 V_2 + c_2 V_1 \\ \frac{\partial V_3}{\partial \tau} & = a_3 V_4 + b_3 V_3 + c_3 V_2 \\ & \cdot \\ & \cdot \\ & \cdot \\ \frac{\partial V_{N-1}}{\partial \tau} & = a_{N-1} V_N + b_{N-1} V_{N-1} + c_{N-1} V_{N-2} \\ \frac{\partial V_j}{\partial \tau} & = a_N V_{N+1} + b_N V_N + c_N V_{N-1} \end{cases}$$

Matrix representation of the expanded form of 3.1.11,

$$\begin{bmatrix} \frac{\partial V_1}{\partial \tau} \\ \frac{\partial V_2}{\partial \tau} \\ \frac{\partial V_3}{\partial \tau} \\ \vdots \\ \frac{\partial V_{N-1}}{\partial \tau} \\ \frac{\partial V_N}{\partial \tau} \end{bmatrix} = \begin{bmatrix} b_1 & a_1 & 0 & \cdots & 0 \\ c_2 & b_2 & a_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & c_{N-1} & b_{N-1} & a_{N-1} \\ 0 & \cdots & 0 & c_N & b_N \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ \vdots \\ V_{N-1} \\ V_N \end{bmatrix} + \begin{bmatrix} c_1 V_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ a_N V_{N+1} \end{bmatrix}$$

The above system of equations can be written in a compact form as follows:

$$\frac{\partial V}{\partial \tau} = AV + B$$

For time discretization, we use θ-method which is another FDM that is used to approximate differential equation. We apply the method by first discretizing time by partitioning the interval $[0, T]$ into finite grids as follows $\tau_i = i\triangle t$, $i = 0, 1, 2, ..., M, M+1$ where $\triangle t = \frac{T}{M+1}$

Applying Euler-θ-method to approximate $\frac{\partial V}{\partial \tau}$ where $\theta \in [0, 1]$, we have

$$\begin{aligned}
\frac{V^{i+1} - V^i}{\triangle t} &= \theta\Big(AV^{i+1} + B^{i+1}\Big) + (1-\theta)\Big(AV^i + B^i\Big) \\
[I - \triangle t\theta A]V^{i+1} &= [I + \triangle t(1-\theta)A]V^i + \triangle t\theta B^{i+1} + \triangle t(1-\theta)B^i \\
V^{i+1} &= [I - \triangle t\theta A]^{-1}\Big([I + \triangle t(1-\theta)A]V^i + \triangle t\theta B^{i+1} + \triangle t(1-\theta)B^i\Big)
\end{aligned}$$

Boundary Condition The boundary condition in a discretized form are as follows: For a call option,

$$V(j, N+1) = max\{j \times (N+1) - K, 0\}$$

$$V(0, i) = 0$$

$$V(h \times (M+1), i) \sim (M+1) \times h - Ke^{-ri\triangle t}$$

For a put option,

$$V(j, N+1) = max\{K - jh, 0\}$$

$$V(0, i) = Ke^{-ri\triangle t}$$

$$V(h \times (M+1), i) \sim 0$$

3.2 Artificial Neural Network(ANN)

Artificial Neural Network is defined as mathematical model that resembles the characteristics of a biological neural networks. The ANN was found after the discovery of the simplified neuron in 1943 by Pitts and McCulloh (Yadav et al., 2015). The biological neurons are known for sending signal to nervous system. A single biological neuron has four main components namely soma, dendrites, axons and synapses. These components work together to ensure that the signal reaches the destination. The dendrites receives information from other neurons, soma process signal from dendrites and the axon transfers output of the neuron. Synapse connect the neurons . Figure 3.1 below shows the structure of a Biological neuron

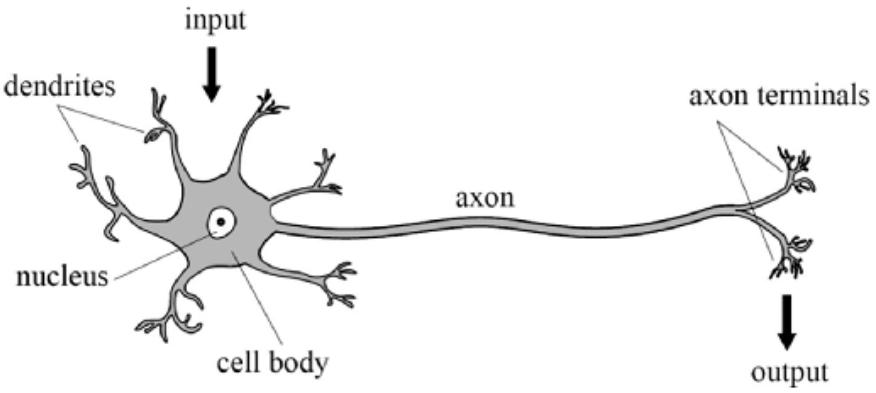


Figure 3.1: A simple structure of biological neuron
(Neves et al., 2017)

So artificial neurons are known to be the building block of ANN and they are used to process information from one layer to another. There are three sets of rules that artificial neuron uses to process information, those rules are multiplication, summation and activation. Each input is being multiplied by it's associated weight, the weighted inputs are summed up together with associated bias and then the information is transferred by passing the sum of weighted input and bias into the activation function which takes the information to the outside world.

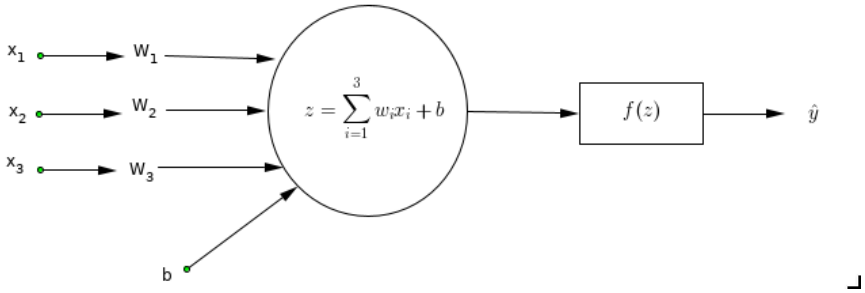


Figure 3.2: A simple structure of Artificial neuron

For all input signal x_i in the network, there is a weight w_i which multiply the associated input and sum them up together with their bias b. Then $\sum_{i=1}^{n} w_i x_i + b$ is passed through transfer function f, thus,

$$z = \sum_{i=1}^{n} w_i x_i + b \tag{3.2.1}$$

and

$$\hat{y} = f(z) \tag{3.2.2}$$

where z is the weighted input , b is bias and $f(.)$ is a transfer function which will be explained in the next section.

3.3 Activation functions

Activation functions are mathematical equations that are used to generate the output of neural network. The output is computed by taking the adjusted weight sum of input and bias, of which the purpose is to check wheather a neuron can be transferred or not. They are regarded as the gate between the current layer of the neural network and to the next layer. These functions are chosen by the complexity of the data. Activation functions are divided into three categories namely, Binary-step, linear and non-linear functions.

Binary step-function - is a function that depend on the threshold to activate the neuron. The neuron is activated by checking if the input value is above or below a threshold so that it can it can send exactly same signal to the next layer. The step-function allows only single value output.

Linear activation function The output is created by multiplying the inputs with the weights for each neuron, unlike binary step function linear function allows multiple outputs and it is not possible to perform back-propagation as the derivative of the function is a constant, which means it doesn't have any effect on the input.

Non-linear activation functions - are the type of activations that are commonly used to model the complex inputs and outputs of the neural network. They are known for their ability to learn and model complex data have sets high dimensionality such as video, image and data. Non-linear activation functions are better than binary and linear function as the derivative of the function exists and it is not a constant, which makes it to allow back-propagation.

(i). **sigmoid function** - Mathematical function that outputs the values between 0 and 1. The function is known as the logistic function and it has S-shaped curve. the mathematical representation of the function is as follows;

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.3.1}$$

The function is known to be differentiable function and also to have positive derivative. This function suffers from the problem of vanishing gradient.

(ii). **Hyperbolic tangent function** - Mathematical function that output values from -1 and 1, also the output is zero centred. The function suffers from vanishing gradient problem. The mathematical representation of the function is as follows:

$$f(x) = \left(\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}\right)$$

(iii). **Softmax activation function** - The function that outputs the values between 0 and 1, same as sigmoid function. the function is mostly used to compute the probability functions. Mathematical representation of the function is as follows;

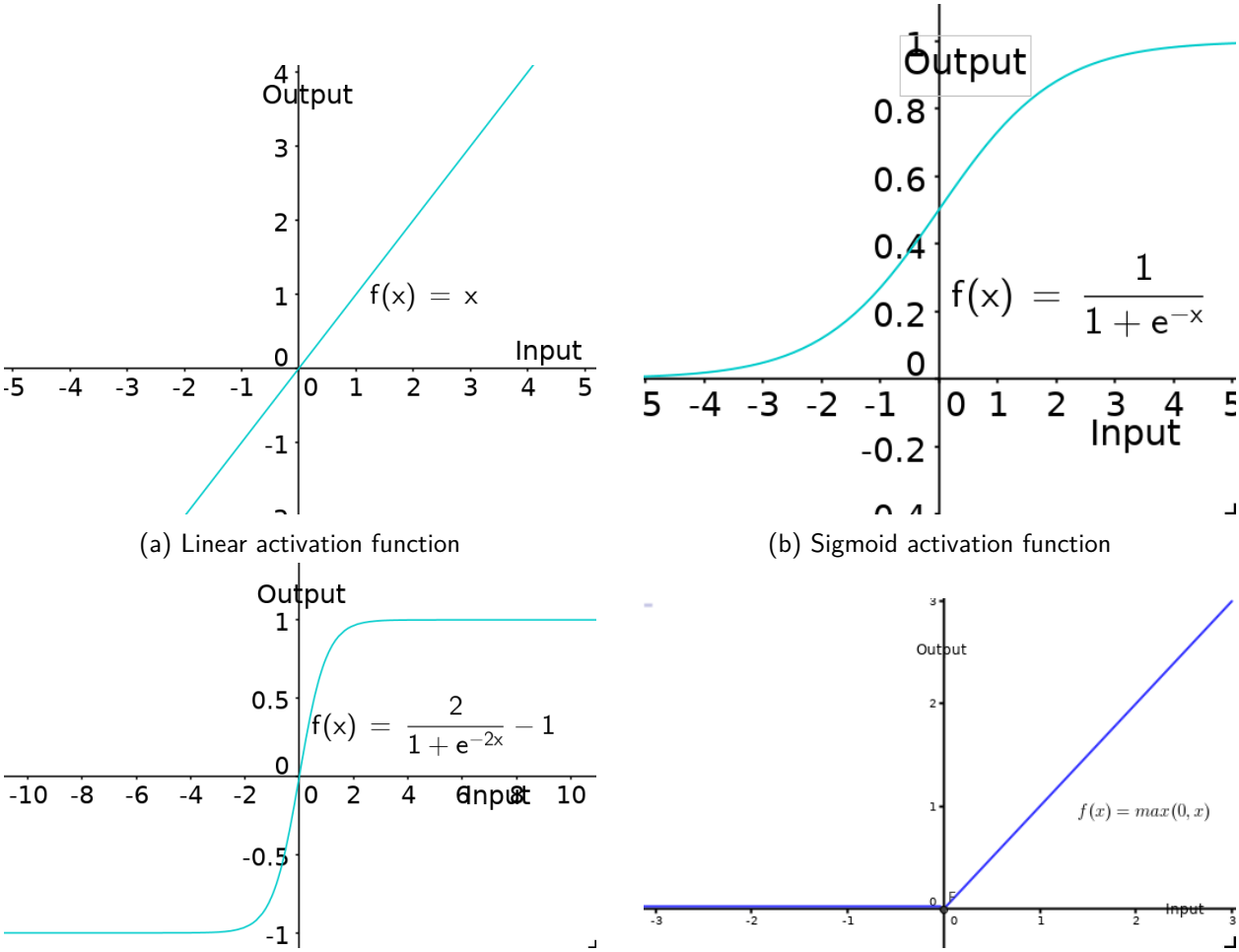
$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{3.3.2}$$

The function is suitably used for multivariation classification which makes it different from sigmoid function.

(iv). **Rectified linear unit(ReLu)**

$$f(x_i) = \begin{cases} x_i & x_i > 0 \\ 0 & x_i < 0 \end{cases} \tag{3.3.3}$$

ReLu is the preferable activation function than sigmoid and tanh as it allows the network to converge quickly and also have better performance. It resembles the linear activation functions properties but it has derivative which it is easy to perform back-propagation



(a) Linear activation function

(b) Sigmoid activation function

(c) Tanh activation function

(d) ReLu activation function

Figure 3.3: These are the four common used activation function

3.4 Multilayer Perceptron (MLP)

Multilayer perceptrons are known as feedforward ANN that contains or have more than two layers. In general MLP is the connection of single layered perceptrons. There exists an invisible layer that connects input and output layer, called hidden layer. Hidden layers process the signal received from the input layer to the next hidden layer or output layer. The number of hidden layers depend on the complexity of the problem and it is chosen randomly or just a guess.

An input layer is an entrance layer of MLP, where there is no computation, the node in input layer take

the information from outside and pass it to the next hidden node in the hidden layer. A hidden node is where the first computation and transformation of an information begins in MLP. The hidden node passes the computed signal to the output node in the hidden layer where the last computation of the MLP is done before the signal is released out to the outside world. Figure 3.4 shows a simple multilayer perceptron with three layers

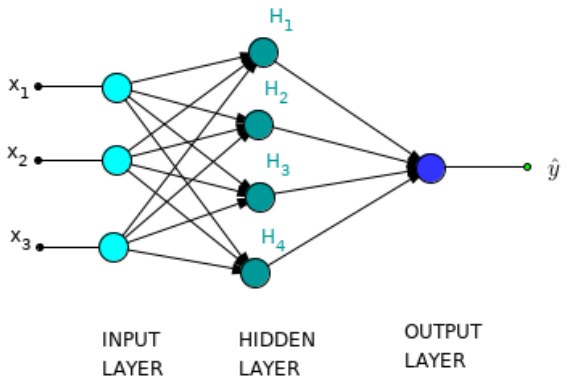


Figure 3.4: A Simple Multilayer Perceptron with three layers

We can summarise the general setup of MLP simply as folows, let L represent the number of layers in the neural network, we know that the 1^{st} layer correspond to input layer and the L^{th} layer correspond to output layer. The dimension of the 1^{st} layer is given by n_1 and the dimension of the last layer which is output layer is given by n_L where n_l are number of neurons in a single layer for $l = 1, 2, .., L$. Defining the weight matrix W^l which has the following entries w_{ij}^l. The weight w_{ij} is applied to the i^{th} neuron at the certain layer l^{th} to the particular output corresponding to the $(l-1)^{th}$ layer. Finally we define $b^l \in \mathbb{R}$ as vector which entries are threshold. To check the performance of the model, we employ the loss function which measures how close the predicted outcome it is to the actual.

3.5 Loss Function

In every prediction or approximation of a certain model there is always an error or mistake in a process. In ANN, the construction of neural network is affected by many factors, to name few, the wrong choice of number of hidden layers and the wrong parameters. An error of the neural network can be calculated using loss function which is defined as mathematical functions that take the difference between the actual output value and the predicted value from neural network. The most common used loss function is Mean Square Error (MSE), which will be employed in this study which is a function of weights and biases. The mathematical representation of the equation is given as follows;

$$ls(w, b) = \frac{1}{2}\sum_{i=1}^{N}(y(x_i) - \hat{y}(x_i)^L)^2 \tag{3.5.1}$$

where $y(x_i)$ is the true value and $\hat{y}^L$ is the approximated output value. Since the function depend on the parameters weights and biases, then the error can be reduced by updating the parameters using optimization methods.

3.6 Optimization

In ANN, the main goal is to design a model that fits the data well which means that the error has to be small. There are several methods that can be used to minimise the loss function. A gradient descent method is mostly used technique for optimization in ANN, which will be considered in this study. Gradient descent is defined as optimization algorithm to minimize a function.

3.6.1 Batch gradient descent. is an optimizing technique that perform gradient for all the given data set and update the parameter after later. The efficiency of a method also depends on the speed of computer as the training data set could be large.

3.6.2 Mini-Batch gradient descent. - is the gradient descent technique that divide the given data set into mini-samples then compute loss of every sample and uses the average loss of samples to update the parameters.

The gradient of the loss function can be obtained through backpropagation.

3.7 Backpropagation Algorithm

Let's consider a loss function for a single training point,

$$ls = \frac{1}{2}(y - \hat{y}^L)^2 \tag{3.7.1}$$

The output corresponding to the i^{th} layer is given as follows,

$$Z^l = W^l \hat{y}^{l-1} + b^l \in \mathbb{R}^n \text{ for } l = 2, 3, .., L \tag{3.7.2}$$

The z_j^l is regarded as the weighted input at layer l for neuron j. Then we can write the output at layer l as follows,

$$\hat{y}^l = f(z^l) \text{ for } l =, 2, 3, ..., L \tag{3.7.3}$$

We define $\delta^l \in \mathbb{R}^{n_l}$ as follows:

$$\delta_j^l = \frac{\partial ls}{\partial z_j^l} \text{ for } 1 \leq j \leq n_l \text{ and } 2 \leq l \leq L \tag{3.7.4}$$

Lemma,

$$\delta^L = f^{'}(z^L) \circ (\hat{y}^L - y) \tag{3.7.5}$$

$$\delta^l = f^{'}(z^l) \circ (W^{l+1})^T \delta^{l+1} \text{ for } 2 \leq l \leq L-1 \tag{3.7.6}$$

$$\frac{\partial ls}{\partial b_j^l} = \delta_j^l \text{ for } 2 \leq l \leq L-1 \tag{3.7.7}$$

$$\frac{\partial ls}{\partial w_j^l} = \delta_j^l a_k^{l-1} \text{ for } 2 \leq l \leq L-1 \tag{3.7.8}$$

Proof

To prove this *lemma*, we start by proving equation 3.7.5 using a fact that $\hat{y}^l$ depend on z^l then equation 3.7.3 can be written as follows;

$$\hat{y}_j^l = f(z_j^l) \tag{3.7.9}$$

The derivative of equation 3.7.9 with respect to the weighted input z_j^l is given by;

$$\frac{\partial \hat{y}_j^l}{\partial z_j^l} = f^{'}(z_j^l) \tag{3.7.10}$$

Further more, we have,

$$\delta_j^L = \frac{\partial ls}{\partial z_j^L} = \frac{\partial ls}{\partial \hat{y}_j^L} \times \frac{\partial \hat{y}_j^L}{\partial z_j^L} \tag{3.7.11}$$

We consider the fact that loss function is defined as sum of all errors over in the output layer for the corresponding nodes,

$$\frac{\partial ls}{\partial \hat{y}_j^L} = \frac{\partial}{\partial \hat{y}_j^L}\left(\frac{1}{2}\sum_{s=1}^{n_L}(y_s^L - \hat{y}_s^L)^2\right) \tag{3.7.12}$$

$$= -(y_j^L - \hat{y}_j^L) \tag{3.7.13}$$

Since equation 3.7.11 depend on equation 3.7.13 and 3.7.10, we have

$$\delta_j^L = (\hat{y}_j^L - y_j^L \hat{y}_j^L) f^{'}(z_j^l) \tag{3.7.14}$$

From properties of vectors, we know that for any two vectors $r, v \in \mathbb{R}^n$, $r \circ v$ can be defined by: $(r \circ v)_i = r_i v_i$ for $i = 1, 2, ..., n$. Then we apply this property on equation 3.7.14 and 3.7.5, we have,

$$\delta^L = f^{'}(z^L) \circ (\hat{y} - y) \tag{3.7.15}$$

Equation 3.7.6 can be proven by first considering the chain rule method to convert z_j^l to $\{z_s^{l+1}\}_{k=1}^{n_l+1}$, and also by using equation 3.7.4, we have,

$$\delta_j^l = \frac{\partial ls}{\partial z_j^l} = \sum_{s=1}^{n_{l+1}} \frac{\partial ls}{\partial z_s^{l+1}} \frac{\partial z_s^{l+1}}{\partial z_j^l} \tag{3.7.16}$$

Taking partial derivative of z_s^{l+1} with respect to z_j^l, we have,

$$\frac{\partial z_s^{l+1}}{\partial z_j^l} = \frac{\partial}{\partial z_j^l}\left(W_{sj}^{l+1} f^{'}(z_j^l)\right) \tag{3.7.17}$$

$$= W_{sj}^{l+1} f^{'}(z_j^l) \tag{3.7.18}$$

Now we can write δ_j^l as;

$$\delta_j^l = \sum_{s=1}^{n_{l+1}} \delta_j^{l+1} W_{sj}^{l+1} f^{'}(z_j^l) \tag{3.7.19}$$

Re-arranging equation 3.7.19, we get,

$$\delta_i^l = f^{'}(z_i^l)((W^{l+1})^T \delta^{l+1})_i \tag{3.7.20}$$

Representing equation 3.7.20 in a component-wise form, we have,

$$\delta^l = f^{'}(z^l) \circ (W^{l+1})^T \delta^{l+1} \tag{3.7.21}$$

To prove equation 3.7.7, We know that z_j^l is connected to b_j^l, since,

$$z_j^l = \left(W^l f'(z^{l-1})\right)_j + b_j^l \tag{3.7.22}$$

it is clear that z_j^{l-1} is independent of b_j^l, so,

$$\frac{\partial z_j^l}{\partial b_j^l} = 1 \tag{3.7.23}$$

Using equation 3.7.4 and application of chain rule, we have,

$$\frac{\partial ls}{\partial b_j^l} = \frac{\partial ls}{\partial z_j^l}\frac{z_j^l}{\partial b_j^l} = \frac{\partial ls}{\partial z_j^l} = \delta_j^l \tag{3.7.24}$$

To prove the last equation of this *lemma*, we first consider the fact that z_j^l is connected with W_{sj}^l, we have,

$$z_j^l = \sum_{s=1}^{n_{l-1}} W_{js}^l \hat{y}_s^{l-1} + b_j^l \tag{3.7.25}$$

The derivative of z_j^l with respect to W_{js}^l is given by,

$$\frac{\partial z_j^l}{\partial W_{js}^l} = \hat{y}_s^{l-1} \tag{3.7.26}$$

which is independent of j.

$$\frac{\partial z_r^l}{\partial W_{js}^l} = 0 \tag{3.7.27}$$

for $r \neq j$.

Equations 3.7.26 and 3.7.27 follow because the j^{th} neuron at layer l uses the weights from only the j^{th} row of weight matrix $W^{[l]}$ and applies the weights linearly. Then by considering 3.7.26 and 3.7.4 with the use of chain rule on these equations, we obtain,

$$\frac{\partial ls}{\partial W_{js}^l} = \sum_{r=1}^{n_l} \frac{\partial ls}{\partial z_r^l}\frac{\partial z_r^l}{\partial W_{js}} = \frac{\partial ls}{\partial z_j^l}\frac{\partial z_j^l}{\partial W_{js}} \tag{3.7.28}$$

$$= \delta_j^l \hat{y}_s^{l-1} \tag{3.7.29}$$

Thus, conclude the proof for our lemma.

The parameters for each layer can be updated using these equations,

$$W_{ij}^{l_{new}} = W_{ij}^{l_{old}} - \mathcal{E}\frac{\partial ls}{\partial W_{ij}^{l_{old}}} \tag{3.7.30}$$

and

$$W_{ij}^{l_{new}} = b_{ij}^{l_{old}} - \mathcal{E}\frac{\partial ls}{\partial b_j^{l_{old}}} \tag{3.7.31}$$

$\mathcal{E}$ is the learning rate.

3.8 Neural Network for solving differential equations

In this section, we present the method of approximating the solution of differential equations using feedforward neural network which is the method that will be used in this study. This method for solving differential equation was introduced by Likas and Fotiadis in 1997. The solution for differential equation obtained via ANN is said to be differentiable and it is of closed analytic form,(Lagaris et al., 1998). The method uses low memory as the solution is compact and requires few number of parameters.

3.8.1 Description of the method. The general differential equation definition will be used to illustrate the proposed method;

$$F(\vec{x}, g(\vec{x}), \nabla g(\vec{x}), \nabla^2 g(\vec{x})) = 0, \vec{x} \in D \tag{3.8.1}$$

Which it is governed by some certain boundary conditions such as Neumann or Dirichlet, where $\vec{x} = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$. $D \subset \mathbb{R}^n$ is regarded as the Domain and $g(\vec{x}))$ is the solution to be approximated.

The solution from ANN method is obtained by first discretizing the boundary S and the Domain D into some set of points $\hat{S}$ and $\hat{D}$. After discretizing the boundary and Domain, the equation 3.8.1 is transformed into a system of equations as follows;

$$F(\vec{x_i}, g(\vec{x_i}), \nabla g(\vec{x_i}), \nabla^2 g(\vec{x_i})) = 0, \forall \vec{x_i} \in \hat{D} \tag{3.8.2}$$

The proposed solution from the method is called trial solution which is denoted by $g_t(\vec{x}), \vec{p})$ with adjustable parameters $\vec{p}$ and the problem is transformed into the following equation,

$$min_{\vec{p}} \sum_{\vec{x} \in \hat{D}} F(\vec{x_i}, g(\vec{x_i}, \vec{p}), \nabla g(\vec{x_i}, \vec{p}), \nabla^2 g(\vec{x_i}, \vec{p}))^2 \tag{3.8.3}$$

The proposed trial solution takes on the feed forward neural network where the parameter $\vec{p}$ denotes biases and weights of the proposed neural architecture. The trial solution is chosen in a way that it satisfies the boundary conditions and it is also the combination of two terms as follows;

$$g_t(\vec{x}) = A(\vec{x}) + P(\vec{x}, N(\vec{x}, \vec{p})) \tag{3.8.4}$$

where $N(\vec{x}, \vec{p})$ is the feedforward neural network with parameter $\vec{p}$ and input vector $\vec{x}$. $A(\vec{x})$ is the term that is constructed in a way that it fulfils the boundary conditions and it is independent of the adjustable parameters. $P(\vec{x}, N(\vec{x}, \vec{p})$ is constructed in a way that it doesn't contribute to the boundary condition and it contains the neural that is used to deal with minimization of the problem by adjusting the weights and bias.

3.8.2 The Gradient computation. Some training of the neural network method use gradient of the ANN with respect to the parameters to minimise an error. The training of the ANN is in this method is taken as the minimization of equation 3.8.3. The error corresponding to each input vector x_i is the value of $F(\vec{x_i}))$ which has to be zero(Lagaris et al., 1998).

To compute the gradient of the neural network with respect to the inputs, we start by considering feedforward neural network with n inputs units, single hidden layer with c activation functions and some output unit(Yadav et al., 2015). We can define an output of neural network as;

$$N = \sum_{i=1}^{c} d_i \sigma(z_i) \tag{3.8.5}$$

for a particular vector of inputs $\vec{x} = (x_1, x_2, ..., x_n)$ and $z_i = \sum_{j=1}^{N} w_{ij}x_j + b_i$, where, w_{ij} represent the weight from input unit to the hidden unit, d_i represent the weight from the hidden unit to the output unit, b_i is the threshold of hidden unit, $\sigma(z)$ is the activation function now the derivative with respect to the inputs is given by,

$$\frac{\partial^k N}{\partial_j^k} = \sum_{i=1}^{c} d_i w_{ij}^k \sigma_i^{(k)} \tag{3.8.6}$$

The k^{th} derivative of the activation function is illustrated by $\sigma^{(k)}$. The derivative with any particular inputs of the neural network which is similar to a feedforward neural network N_g with single hidden layer. The difference between the output of neural network N and N_g is the weight d_i is replaced by $d_i E_i$ and the activation function of each hidden unit is replaced by Λ^{th} order of the activation function. Both networks have the same values of weights and biases. The neural network network N_g;

$$\frac{\partial^{\lambda_1}}{\partial_{x_1}^{\lambda_1}} \frac{\partial^{\lambda_2}}{\partial_{x_2}^{\lambda_2}} ... \frac{\partial^{\lambda_n}}{\partial_{x_n}^{\lambda_n}} N = \sum_{i=1}^{n} d_i E_i \sigma_i^{\Lambda} \tag{3.8.7}$$

where $E_i = \prod_{k=1}^{n} w_{ij}^{\lambda_k}$ and $\Lambda = \sum_{i=1}^{n} \lambda_i$.

Now, we consider the of N_g with respect to the parameters of neural network N, which can be done as follows;

$$\frac{\partial N_g}{\partial d_i} = E_i \sigma_i^{\Lambda} \tag{3.8.8}$$

$$\frac{\partial N_g}{\partial b_i} = d_i E_i \sigma_i^{\Lambda+1} \tag{3.8.9}$$

$$\frac{\partial N_g}{\partial w_{ij}} = x_j d_i E_i \sigma_i^{\Lambda+1} + d_i \lambda_j w_{ij}^{\lambda_{j-1}} \Big(\prod_{k=1, k\neq j} w_{ik}^{\lambda_k} \Big) \sigma_i^{(\Lambda)} \tag{3.8.10}$$

3.8.3 Updating parameters. The update of weights and biases are carried out in the following manner using backpropagation algorithm, using equation 3.8.8,3.8.9 and 3.8.10

$$d_i(q+1) = d_i(q) + t\frac{\partial N_k}{\partial d_i}$$

$$b_i(q+1) = b_i(q) + r\frac{\partial N_k}{\partial b_i}$$

$$w_i j(q+1) = w_{ij}(q) + t\frac{\partial N_k}{\partial d_i}$$

3.9 Application of ANN method on Black-Scholes model

Let's recall the discretization of the space x and time τ in section 3.1, $\tau_i = i\triangle t$ where $i = 0, ..., M+1$ and $\triangle t = \frac{T}{M+1}$.

$x_j = jh$ where $j = 0, ..., N+1$ and $h = \frac{x_{max}}{N+1}$.

The trial function of the Black-Scholes equation is given as follows;

$$V_t(x,\tau) = A(x,\tau) + B(x,\tau)N(x,\tau,\vec{p}) \tag{3.9.1}$$

where

$$A(x,\tau) = x\left(1 - \frac{Ke^{-r\tau}}{x_{max}}\right) + \tau\left(1 - \frac{\tau}{T}\right)\left(max(x-K,0) - x\left(1 - \frac{K}{x_{max}}\right)\right) \tag{3.9.2}$$

and

$$B(x,\tau) = \tau x(x_{max} - x) \tag{3.9.3}$$

The function $A(x,\tau)$ has to satisfy the following boundary conditions,$x_{max} > K$

$$V(0,\tau) = 0 \tag{3.9.4}$$
$$V(x_{max},\tau) = x_{max} - K.e^{-r\tau} \tag{3.9.5}$$
$$V(x,0) = max(x-K,0) \tag{3.9.6}$$

Loss function to minimize the error of artificial neural network can be defined by first letting $V_t(\tau_i, x_i) = V_{ij}$ then,

$$E(\vec{p}) = \sum_{i=1}^{M+1}\sum_{j=1}^{N+1}\left[rV_{ij} + \frac{\partial V_{ij}}{\partial \tau_i} - rx_j\frac{\partial V_{ij}}{\partial x_j} - \frac{1}{2}\sigma^2 x_j^2\frac{\partial^2 V_{ij}}{\partial x_j^2}\right]^2 \tag{3.9.7}$$

The accuracy of predicted solution of PDE is checked by using the following equation;

$$Error = \sqrt{\frac{1}{N+1}\frac{1}{M+1}\sum_{i=1}^{M+1}\sum_{j=1}^{N+1}\left[V_{ij}^a - V_{ij}^t\right]^2}$$

4. Results

The analytical solution and ANN methods were carried out using python. The FDM method was carried out using MATLAB.

4.1 Numerical results

The neural network method was carried out with three layers where the input layer contains two neurons which takes values from numerical values of τ and x. The hidden layer which is the second layer have ten neurons, the transfer function associated with each neuron of the hidden layer is sigmoid function. The last layer which is called output layer has only one neuron and the associated transfer function is linear function. The learning rate $\beta = 0.001$ is chosen for backpropagation algorithm to train the neural network.

The option price was approximated using the following inputs: $K = 1$, $x = [0, 3]$, $r = 0.01$, $T = 0.1$, $\sigma = 0.05$

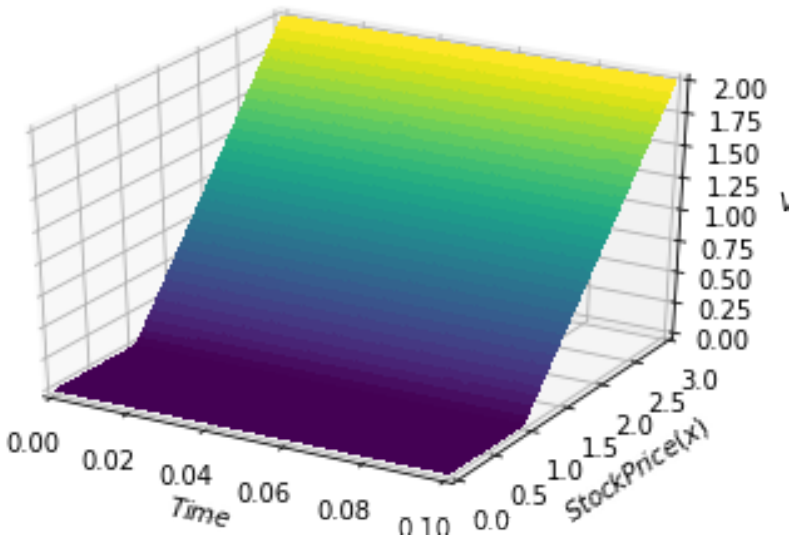


Figure 4.1: The graph shows analytical Solution of Black-Scholes PDE

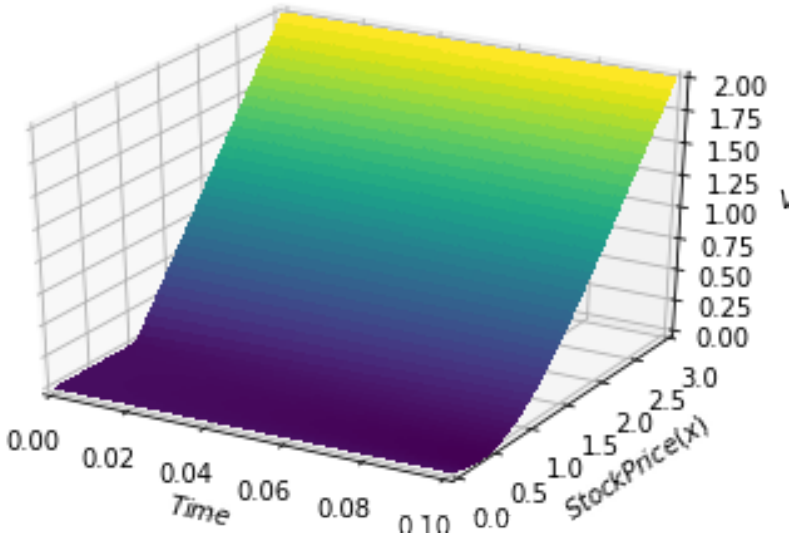


Figure 4.2: The graph shows ANN solution of Black-Scholes PDE

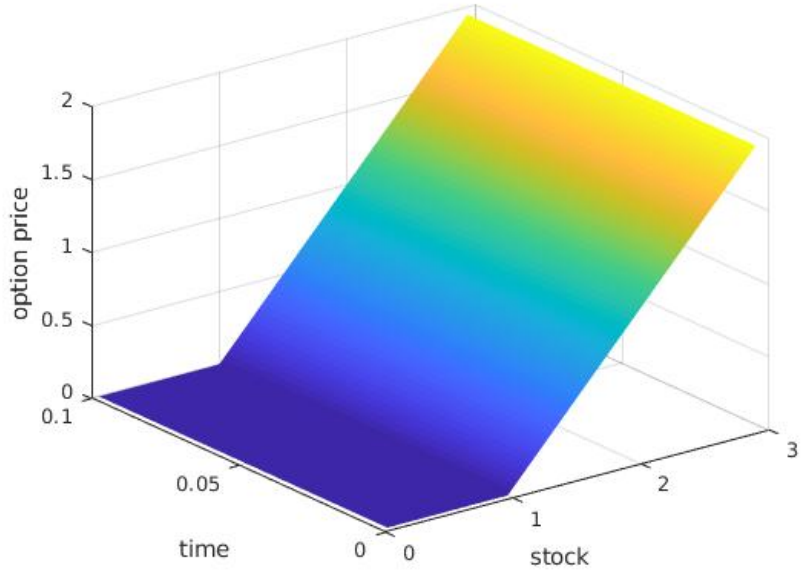


Figure 4.3: The graph shows Finite Difference solution of Black-Scholes

4.2 Approximated Error

We represent the mean square error with respect to analytical solution to compare the ANN method and finite difference method

Grid	Artificial Neural Network Method	Finite Difference Method
50× 50	0.005060021	0.00447
100× 100	0.0013413693	0.000753
150× 150	0.017101426	0.000314
200× 200	0.020428551	0.000714
250× 250	0.023040416	0.000132

Table 4.1: Table of errors

4.3 Discussion

The results from finite difference method which are represented by figure 4.3 are close to the analytical solution results as the option price before the strike price is surfaced close to zero. The results from ANN method which are represented by figure 4.2 shows a slight difference to the analytical results as the option price before strike price are a bit high than zero.

The results from table 4.1 shows that the FDM has less error compare to the Artificial Neural network method

5. Conclusion

Artificial Neural Network approach to option pricing model was introduced. A clear understanding of the Finite Difference Method and Artificial Neural Network technique were provided. An option pricing model Black-Scholes for European call option was considered where Finite Difference Method and Artificial Neural Network method were used to approximate the price of an option. Based on the evidence provided from Artificial Neural Network and Finite Difference Method, it shows that the Finite Difference Method approximate option price better than neural network method. We can conclude that finite difference method is more accurate compared than Artificial Neural Network method.

For future work, the Artificial Neural Network Method can be applied to a data in high-dimensional space to check how it behave compare to when it is applied to data in low-dimensional space.

Acknowledgements

Firstly, I would like to thank God for spiritual guidance and protection throughout my project.

Secondly, I would like to give special thanks to my supervisor Prof Phillip Mashele for his supervision on my project and also not forgetting Rock Koffi for providing me with financial material for my project. To my tutor Emilia Magnani, thanks for guiding throughout my project.

Lastly, Special thanks to our academic director Dr. Simukai Utete, Prof. Barry Green, Prof. Turok Neil and Jan Groenewald. Also AIMS football team.

References

Black, F. and Scholes, M. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654, 1973.

Bohner, M., Sánchez, F. H. M., and Rodriguez, S. European call option pricing using the adomian decomposition method. *Adv. Dyn. Syst. Appl*, 9:75–85, 2014.

Cui, Y. and Yu, B. The simulation of european call options' sensitivity based on black-scholes option formula. *Journal of Mathematical Finance*, 2(03):264, 2012.

Duffy, D. J. *Finite Difference methods in financial engineering: a Partial Differential Equation approach*. John Wiley & Sons, 2013.

Dunbar, S. R. Stochastic processes and advanced mathematical finance. *Department of Mathematics, University of Nebraska-Lincoln, USA*, 2016.

Durrett, R. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

Emmanuel, F. S., Adedoyin, A. O., and Hammed, O. O. Performance measure of binomial model for pricing american and european options. *Applied and Computational Mathematics*, 3(6-1):18–30, 2014.

Focardi, S. M. and Fabozzi, F. J. *The mathematics of financial modeling and investment management*, volume 138. John Wiley & Sons, 2004.

Hull, J. et al. Options, futures and other derivatives/john c. hull., 2009.

Ianieri, R. *Options theory and trading*. Wiley Online Library, 2009.

Jentzen, A., Salimova, D., and Welti, T. A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients. *arXiv preprint arXiv:1809.07321*, 2018.

Klar, L. and Jacobson, J. Pricing of european call options. *Uppsala: Uppsala University*, 2002.

Lagaris, I. E., Likas, A., and Fotiadis, D. I. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

Lütkebohmert, E. An asymptotic expansion for a black–scholes type model. *Bulletin des sciences mathematiques*, 128(8):661–685, 2004.

Neves, A. C., González, I., Leander, J., and Karoumi, R. A new approach to damage detection in bridges using machine learning. In *International Conference on Experimental Vibration Analysis for Civil Engineering Structures*, pages 73–84. Springer, 2017.

Shreve, S. E. *Stochastic calculus for finance II: Continuous-time models*, volume 11. Springer Science & Business Media, 2004.

Yadav, N., Yadav, A., Kumar, M., et al. *An introduction to neural network methods for differential equations*. Springer, 2015.

Yavuz, M. and Özdemir, N. A different approach to the european option pricing model with new fractional operator. *Mathematical Modelling of Natural Phenomena*, 13(1):12, 2018.