Stock Market Prediction using Artificial Neural Networks and Recurrent Neural Networks

Phumudzo Lloyd Seabe (Phumudzo@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by. Prof. Ronnie Becker
African Institute for Mathematical Sciences, South Africa

12 October 2018

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

Abstract

Forecasting the movement of stock prices is a difficult task because there are "various" factors that influence the stock prices. Highly accurate prediction of stock prices is of paramount interest to investors and to portfolio management. However due to the nonlinearity of the stock market, it is difficult for fundamental prediction techniques to capture the inside law of the market. In response to such limitations, artificial neural networks have been introduced to accomplish the task. This work attempted to train two models and compared their performances in predicting the direction of movement in the daily New York Stock Exchange index (S&P500). The two models are based on two classification techniques: Artificial Neural Networks (ANN) and Recurrent Neural Networks (RNN) classifiers. The opening prices of five stocks, some of which are part of the indeces, were chosen as features of the two models. The classification reports show that performance of Recurrent Neural Network (56%) was found better than Feed forward Neural Network (50%).

U bvumba kutshimbilele kwa mitengo ya mikovhe zwi a konga sa musi huna zwinzhi zwine zwa tutuwedza kutshimbilele kwa mitengo yeneyo. U nethedza vhutanzi ha vhukuma musi hu tshi khou bvumbiwa kutshimbilele kwa mitengo ya mikovhe ndi zwa ndeme kha vhabindudzi na vhalanguli vha mikovhe. Fhedzi nga nthani ha uri hu dzula hu na tshanduko maragani wa mikovhe, zwi a konda u tou bvumba zwavhudi zwine zwa do bvelela. Nga nthani ha yeneyi khaedu, ho bveledzwa ndila ine ya divhea sa Articial Neural Networks u itela u bvumba zwavhudi. Afha hu vha hu tshi khou lingedzwa u shumisa ndila mbili, ha vhambedzwa kushumele kwadzwo kha u bvumba uri mikovhe i do shuma hani duvha linwe na linwe kha maraga wa America, (New York Stock Exchange S&P500). Idzo ndila mbili dzo di tika nga khetekanyo dzo fhambanaho, khethekanyo ya Artificial Neural Network na Recurrent Neural Network. Mbalo-mbalo dza ndeme dza mikovhe dzo topolwa ho sedzwa idzo ndila mbili. Muvhigo wa nga ha khethekanyo kha idzo ndila mbili u sumbedza uri kushumele kwa Recurrent Neural Network (56%) ku khwine kha Feed Forward Neural Network (50%)

Keywords: Stock market Prediction; S&P500 index; Recurrent Neural Network; Artificial Neural Network; Confussion Matrix; Back propagation Method; Multi-layer perceptron

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.

Phumudzo LLoyd Seabe, 12 October 2018

Contents

1. Introduction

Forecasting the direction of stock market has been of interest to variety of people nowadays. It has become a common concept that massive amounts of capital are traded through the stock market all around the world. The prevailing notion in society is that wealth brings comfort and luxury, so it is not surprising that there has been so much work done on ways to forecast the direction of stock markets. Indeed, being able to make accurate predictions of the stock market has strong implications on trading strategies. Recently stock markets have become an investment tool, not only to investors but for common people as well. Various techniques have been proposed to predict stock market but non has been totally successful. Artificial Neural networks have been used to improve upon them. This forecasting technique uses historical time series data as inputs to make informed estimates that are predictive in determining the direction of stock market. This implies a reasonable comprehension of lead-lag relations and statistical significance among numerous variables and learning which variables are the more vital ones to use as signal for foreseeing the market direction. Accurate prediction of stock market can be regarded as an indicator to investors and act as a key component in decisions that involves financial investments, during increase of market volatility and internationalized capital flows. (Guresen et al., 2011).

This essay seeks to predict the direction of the daily prices of Standard and Poor 500 index (S& P 500), which is an American stock market. Stock Market prediction has been regarded as the most essential subject of human life in recent years for academic researchers and for portfolio management by commercial and investment banks. Many efforts have been made by researchers and investors to predict the direction of stock prices including the closing prices, the opening prices, the trading volume of stocks in the market, the highest and the lowest prices. According to the theory of Efficient Market Hypothesis which states that it is not possible to "beat the market" because stock market behave in a random manner and its unpredictable. Predicting stock market behaviour is a challenging task but it also leads to increased profits.

In practice dealing with financial time series data is a difficult task, which leads to stock market prediction being a classic problem, it's complicated in nature with always fluctuating, dynamic, chaotic and nonlinear behavior (Abu-Mostafa and Atiya, 1996). Stock markets are influenced by many macro economic factors such as political events in the world, firm policies, general economic conditions, investor expectations, institutional investor choices, correlation within stock markets, and psychology of investors etc (Tan et al., 2007). However individuals are always attempting to bypass the above mentioned issued regarding stock market prediction. Researchers have came up with so many techniques and various models that attempts to predict the financial market. This techniques are classified into two type "Technical" and "fundamental", the technical approach is the predicting technique that make use of historical data to forecast the market whilst the fundamental involves the "in-depth" analysis of the market.(Kim and Han, 2000)

In order to model the stock market value, a standout amongst other ways is the use of technical approach techniques called Artificial Neural Network and Recurrent Neural Network. The choice of this approach is motivated by the fact that the two models can easily adapt to the changes of the stock market. Stock market predictions are mainly classified into two types: linear and non linear models: Linear models are made of exponential smoothing models, autoregressive integrated moving average models as well as stochastic volatility model (Durham, 2007). Non-linear models include the above mentioned two models. They help to conquer the limitations of linear models as they are able to capture the non linear pattern between features (input and outputs) variables in the financial time series/stock market

data. They also improve predictive performance of stock market through their adaptability and ability to perform non linear modelling. They are driven by data, meaning they do not need prior knowledge or assumptions about the relations between inputs and output variables and lastly when they come across new data/inputs patterns after training they can still perform the same task on the new inputs or unseen data.

Artificial Neural Networks are useful tools in predicting the direction of stock market prices although the large amount of noise and non stationary behavior of financial time series data limits research in its capabilities (Kara et al., 2011). Researchers has made use of various types of artificial neural networks in predicting stock market fluctuations. These include multi-layer perceptron (MLP), back propagation neural network and radial basis function neural networks(Shen et al., 2011) etc.Asadi et al. (2012) reported that prior to down-streaming analysis of these models in the stock market, data pre-processing technique must be performed. This technique helps to improve the accuracy of the artificial neural network models by transformation and extracting the desired input variables from the data.

In this study we aim to predict the stock market fluctuations of the American stock index (S&P 500) one day ahead, based sorely on their historical prices. Our problem can be mathematically formalized as follows:

The linear relationship between input x and output y variables in the financial time series prediction is the main problem to identify. Let x_t represent a set of input vector $X = (x_1, x_2, \ldots, x_t)$ R, and Y_T, denote the closing price of stock market for one day interval at time T. T is the maximum lag time defined as $(t = 1, 2, \ldots, T)$ given by $Y = (Y_1, Y_2, \ldots, Y_T)$ i.e $Y = f(X)$. The function Y represent the stock market price at a point in future and it could be learned from input data so that when new data is presented a new prediction can be made. The prediction results can be learned by regression in which y R, and classification, Y $\{0, 1\}$. Since the stock data is a time series, we choose RNN and ANN models, which are widely applied to time series predictions. Thus we want to predict output y_{T+1} based on input data X, that is:

$$Y_{t+T} = f(x_m) \tag{1.0.1}$$

where $\left(x_m^1 \ldots x_m^t, y_{t-1} \ldots y_{t-L}\right)$

With this background, the results from the two techniques will be validated to compare how well the models are performing. The results of this study will be useful for portfolio management.

The remainder of this study is organized as follows. Chapter 2 provides an outline of Artificial Neural Network and Back Propagation (BP). Chapter 3 describes Recurrent Neural Network and the back propagation through time (BPTT) method. Chapter 4 presents the results and the improvement of the hyperparameters that control these models on the training data set using cross validation. Finally, chapter 5 provides a discussion of the experimental results and conclusion.

2. Machine Learning

In this chapter we explore Machine Learning (ML) at a high level of prediction and classification, and introduce two classification techniques that are used in this study, which are: Multi-layer Perceptron (MLP) and Recurrent Neural Network. The study uses the two techniques mentioned above, with the idea that can validate the classification performance on financial time series prediction problem.

2.1 Building intelligent machines to transform data into knowledge

In the age of modern technology, there is one resource that we have in abundance and that resources is large amount of data. In the beginning of the twentieth century, machine learning was regarded as a subfield of Artificial Intelligence (AI), a field that involves algorithms which are trained by humans to transform the structured and unstructured data into knowledge thus helping the world to make informed decisions and draw conclusions. (Raschka and Mirjalili, 2017). Artificial intelligence is defined as the art of creating machines that perform functions that require a state of intelligence when performed by people, instead of requiring humans to manually derive rules and build models from analyzing large amounts of data. Machine learning offers a more efficient alternative for capturing the knowledge in data to gradually improve the performance of predictive models and make data-driven decisions. Machine learning is an exciting field that makes sense of the data, where its main objective is to create a system that transform and learn from data. Intuitively, learning in machine learning means improving at performing on some task with experience by updating the parameters of the system/model during training. Machine learning algorithms are typically categorized into three types, namely supervised learning, where according to studies is defined as the type of system in which both input and out are given. The main goal of it is to develop a relationship that could estimate the correlation between input and output variables. Supervised learning is mostly used for classification of discrete class labels and regression for continuous outcomes. In this essay we are only going to be focused on classification task. The second type is unsupervised learning, in which, given unlabelled data as input and its main goal is to explore the hidden structure in the data without prior knowledge of the the group/data. The third type is The Reinforcement learning, for which the task is to create a system (agent) that improves its performance based on interactions with the environment. The model learns from a series of action by interacting with its environment in order to maximize a reward function. As discussed before,we only going to focus on classification techniques feed forward and Recurrent neural networks.

2.2 Artificial Neural Network

Artificial Neural Network have caused a surge of interest for researchers in the last recent years and are being explored in a vast and unprecedented range of problems arising from different domains like medicine, finance, physics, computer science, robotics and many more. Researchers have come up with numerous definitions of Artificial Neural Network. Historically artificial neural networks grew out of research in the field of artificial intelligence. Artficial neural networks were inspired biological by neuronal structure. According to McCulloch and Pitts (1943) the first concept of a simplified brain cell was called McCullock-Pitts (MCP) neuron. Figure 2.1 illustrates the idea of MCP concept, the study showed that, the nerve cells in the body act as a logic gate with binary outputs, information/signals arrives at the dendrites (Input) then integrated into the cell body and if the accumulated signal exceeds a

certain threshold, an output signal is generated that will be passed on by the axon. are together and act as a simple logic gate with binary outputs, multiple signals arrive at the dendrites, then integrated into the cell body, and, if the accumulated signal exceeds a certain threshold, an output signal is generated that will be passed on by the axon (Raschka and Mirjalili, 2017).

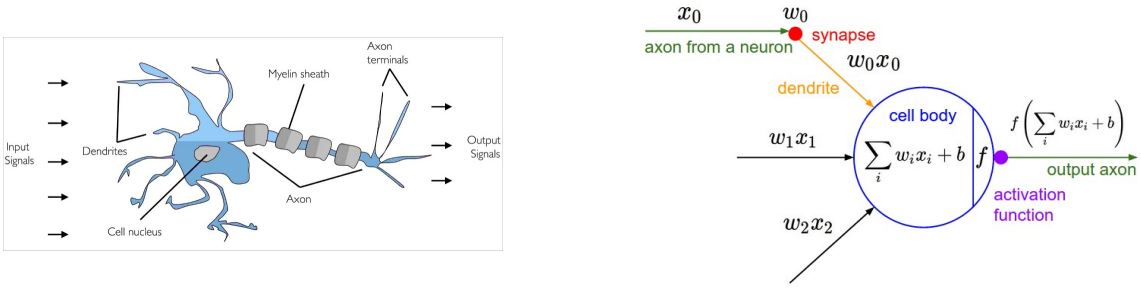


Figure 2.1: The figures shows biological neuron structure: Adopted from Artificial Neural Networks.

The same concept of creation of complex network of neurons by MCP concept inspired artificial neural networks. In AI, neurons are artificially created on a computer and they get connected together to form a complex network of neurons, which we call it artificial neural network. Artificial neural networks are able to perform well in areas where the problems are ill-defined and for time series predictions. Thus AI network are able to adapt to stock market fluctuations. In this case, neural network provides the output that correspond to a taught input pattern which is different from the input pattern and Multilayer perceptron (MLP) is used to overcome the limitation (handle nonlinear problems). Multilayer perceptron (MLP) is one of the types of artificial neural network, it is also called a feed-forward artificial neural network. The MLP architecture can be represented by the figure 2.2.

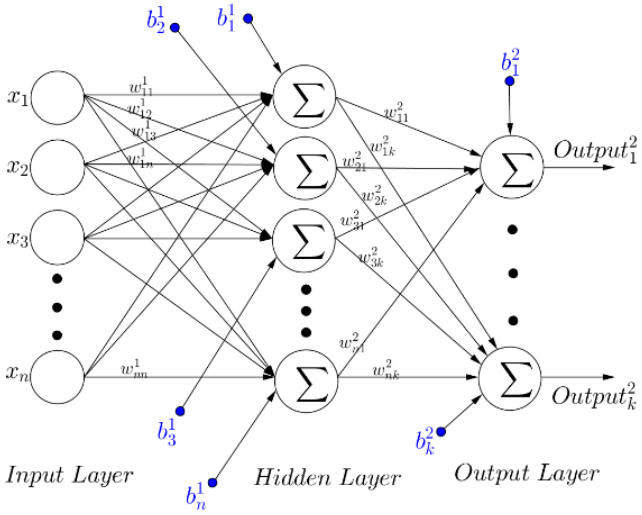


Figure 2.2: Multilayer neural network architecture: Adopted from Addai (2016).

The MLP can be defined as a feed-forward artificial neural network that generates a set of outputs from a set of inputs. The model shown in 2.2 is a graphical representation of MLP architecture. It can be used to illustrates how transmission of information/inputs in the MLP neural network through the edges

happens. The architecture of the MLP model above can be described as follows: It consists of at least three layers namely the Input, middle and the Output layer. The input layer receives and transmit the weighted information to the middle layer. In the middle layer, a lot of decisions are made and its more abstract than the input layer. Lastly the output layer produces the results and returns a neural network. The principle of MLP imitate the idea of human brain by learning the information from the data and transforms it into an intelligent machine as knowledge. There have been immense number of studies who have demonstrated their capabilities in financial modelling using Artificial Neural Networks. Artificial Neural Network algorithms have two possible: classification and regression. Classification is the process of predicting the class of given data points. It is used to identify a set of categories (sub-populations) of new observations. Regression is used to predict the continuous outcomes of target variables. It allows predictions from data by learning the relationship between features in the data and some observed, continuous-valued responses (Outputs). In this study we are going to classification models to predict stock market fluctuations using Recurrent or Interactive network that allows signals to have feedback loops (which will discuss later in chapter 3) and Feed forward Neural Networks (ANNs), that allow signals to travel one way only; from input to output.

2.3 Multilayer Perceptron (MLP)

The popular deep learning ANN is Multi-layer perceptron (MLP). The architecture of this Neural Network is typically represented by nodes. The Multi-layer perceptron in Figure 2.2 is made out of an input layer to receive the signal, an output layer that makes a decision or prediction about the input, and in between those two layers, an arbitrary number of hidden layers that are the true computational engine of the MLP. The difference between MLP and Single Layer Neural Network is the architecture. The MLP architecture can take more than one hidden layers depending on the underlying problem to be modelled. They are widely utilized for predictions, classification, approximation and pattern recognition (Devadoss and Ligori, 2013). According to studies it is proved that a Multi-layer perceptrons (MLP) can approximate any complex continuous function that enables in learning any complicated relationship between the input and the output of the system (Jabin, 2014). The MLP network is influenced by the weights which are connected amongst neurons and changing of the weights in specific manner results in learning of the network and the procedure is called learning/training process. We can use Figure 2.2 to illustrates MLP.

Let $a_j^l - 1$ refers to the output of the k^{th} neuron in the layer $l - 1$, a_j^l as the output of the j^{th} neuron in layer l, w_{jk}^l as the weight of the dendrite that connects the k^{th} neuron in layer $l - 1$ to the j^{th} neuron in layer l, and explicitly, use b_j^l as the bias of the j^{th} neuron in layer l. With these notations, the sum of all k dendrites in the $L - 1^{th}$ layer is the product sum between the output of each neuron in layer $l - 1$ and the weight of the edge that connects the neuron to the j^{th} neuron in layer l and add the bias vector b^l. Mathematically, we can write it as:

$$z_j^l = \sum_{j=1} w_{j,k}^l a_k^{l-1} + b_j^l \tag{2.3.1}$$

Equation (2.3.1) is called the Net input or the weighted input to the neurons in layer l which takes a linear combination as we saw above, it represent a linear approximation. The output always remains to be a linear combination regardless of the number of hidden layers we choose, but the choice of using number of hidden layers depends on the complexity of the problem at hand. Subsequently, in order to avoid linear relationship between input and output we introduce a non-linear activation function that

allows us to deal with non linear problems. The activation a_j^l of the j^{th} neuron in the l^{th} layer is related to the activations function φ in the $l - 1^{th}$ layer by equation

$$a_j^l = \varphi(z_j^l) = \varphi\left(\sum_{j=1} w_{j,k}^l a_k^{l-1} + b_j^l\right) \tag{2.3.2}$$

Equations (2.3.1) and (2.3.2) can be written in a matrix form. Lets define a weight matrix w^l whose entries are the weights of the connections that link the neurons in layer $l-1$ to neurons in layer l and the bias vector b^l for each layer l. Lastly an activation vector a^l whose components are the activations of a_j^l. With those notations defined above, we can rewrite two expressions in a vectorized form as follows:

$$z^l = w^{[l]^T} a^{l-1} + b^l \tag{2.3.3}$$

and

$$a^l = (z^l) = \varphi\Big(w^{[l]^T} a^{l-1} + b^l\Big), \tag{2.3.4}$$

Equation (2.3.4) gives us more understanding of how activations in the network works. Substantially how the activations in one layer is related to activations in the previous layer: This happen by applying the weight matrix to the activations, then by adding the bias vectors with the φ activation function. The choice of the activation that we can use influences the performance of the model and has an impact on the learning process. The activation function represented by φ is a sigmoid function. In the next section, we are going to introduce a formal definition of an activation function and then look at some commonly used activation functions.

2.4 Activation Function

The choice of the activation functions in neural networks has an important impact on the network performance. Many neural network use activation function in nonlinearity components to approximate any arbitrary complex function (Hornik et al., 1989). Lets define what activation function is:

Definition 2.4.1. An activation function is a mapping φ: R $\longrightarrow$ R that is differentiable.

The purpose of having activation function in a neural network is to ensure that the mapping between the input sapce and output space . It is mostly used when we dealing with non linear relationship in the network, especially when dealing with complex problems that cannot be solved using linear approximation. It is used to perform a threshold on the calculated similarity measure between the learned weights and the input. The have been so many proposed activation functions over the years. However, in many cases, best practice limits the utilization to just a restricted sort of activation functions that are used in ANN which are Sigmoid, Tanh, ReLU, Softmax.

2.4.1 Sigmoid Units. The Sigmoid function is used for hidden layer neuron output, is a special case of the logistic function having a characteristic "S"-shaped curve. It is defined by the formula:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.4.1}$$

Sigmoid activation function is used for binary classification, it takes real-valued numbers and "squashes" them into a range between 0 and 1, i.e. $\sigma(x) \quad (0, 1)$. The main properties of sigmoid function are non-linearity (as we said before), it is a continuous and differentiable at the same time allowing the back propagation of the error to occur. But it has two major drawbacks:

- Sigmoids saturate and kill gradients. The Sigmoid function has the property that when the neuron's activation saturates at either tail of 0 or 1, the gradient $\sigma(x) = \sigma(x).(1 - \sigma(x))$ vanishes when the weighted sum is close either to zero or one. During back propagation (which will discuss later), If the gradient is very small or close to zero, almost no signal flows through the neuron to its weights and recursively to its data, which affect the leaning process, so the weights of sigmoid neurons must be initialized to prevent saturation.
- Sigmoid outputs are not zero-centered. The outputs of sigmoid function are always positive and this has implications during gradient descent, because if the network recives information into a neuron and is always positive (e.g. $x > 0$ elementwise in $f(x) = w^T x + b$), then the gradient on the weights w will become either all be positive, or all negative during back propagation (depending on the gradient of the whole expression f).

2.4.2 Tanh function. The tanh function is an alternative to Sigmoid function. It is defined by the formula:

$$\tanh(x) = \frac{2}{1 + e^{(-2x)}} - 1 \tag{2.4.2}$$

It is also sigmoidal ("S"-shaped) and squashes real valued number to the range between -1 and 1, i.e., $\tanh(x) \quad [-1, 1]$. Like the sigmoid function, its output is zero-centred but it allows us to solve the non-zero centrality of sigmoid function. It also has the weakness o =f gradient vanishing when the weighed sum is either -1 or 1.

2.4.3 Rectified Linear Units (ReLU). The ReLU function is defined as

$$f(x) = max(0, x) \tag{2.4.3}$$

where x is the input to a neuron. This activation function is simply thresholded at zero, its ouput is zero when the input is less than or equal to zero, and for any postive input, RELU function is linear with a slope equal to one. The outputs of the function lie in the interval $[0, +\infty]$

The ReLU function is more effectively than the widely used logistic sigmoid; because it is capable to reduce the computation cost and its gradient does not vanishes like in sigmoid and Tanh functions. Unfortunately, ReLU also suffers several drawbacks.

2.4.4 Softmax activation function. The Softmax function which is usually used for multi-classification neural network output, is a generalization of the logistic function that "squashes" a K-dimensional vector z from arbitrary real values to a K-dimensional vector $\sigma(z)$ of real values in the range [0,1] that add up

to 1. The function is given by:

$$\sigma^l = \frac{e^{z_j^{[l]}}}{\sum_{k=1}^{k} e^{z^{[l]}{}_k}} \tag{2.4.4}$$

where K is the number of classes, $z_j^{[l]}$ and $z_k^{[l]}$ are the j^{th} and the k^{th} entry of the vector $z^{[l]}$ respectively. The output a $\sigma^{[l]}$ is a vector whose components are the probabilities associated to each class. This activation function has some interesting properties: the output is invariant when you add a scalar to the inputs and its gradient saturate when the probability of the true class is close to one. The key disadvantage of this activation function is that it is expensive computational.

The Figure (2.3) below shows a graphical presentation of activations functions we discussed above.

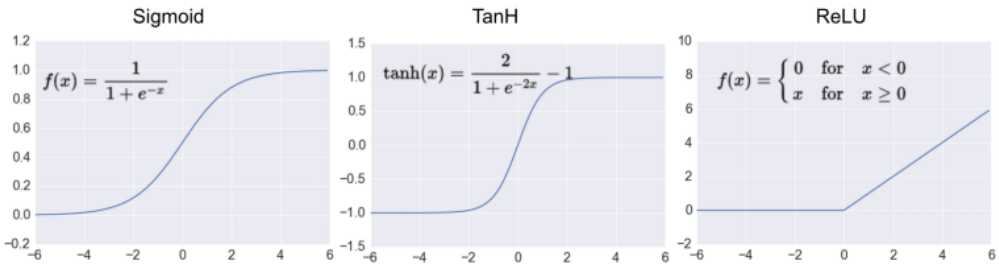


Figure 2.3: Graphical representation of Activation functions $Sigmoid$,$\tanh$ and $Relu$: Adopted from Activation functions.

2.5 Cross- Entropy Error Function

The process of learning in NN involves adjusting the weights and minimizing the error that is summed up over all the training samples. The network does this by calculating Mean square Error (MSE), which is average squared error between the target values and and the computed output. It used as the loss function for least squares regression and defined as:

$$E_i(w,b) = \frac{1}{2}\sum_{i=1}^{n}\Big(y_j(x_i) - a_j(x_i)\Big)^2 \tag{2.5.1}$$

Where $y_j(x_i$ and $a_j^L(x_i)$ are the output of the input x_i.

Error in most neural network is calculated using mean squared error, as the cost function, but during this process the network goes through stages in which the reduction of the error can becomes very slow which is referred as MSE stagnation period. This period affects the learning times of the network and in order to solve this problem, the MSE is going to be replaced by entropy error function which exhibit a better network performance with a shorter stagnation period (Hornik et al., 1989). The cross entropy for multilayer binary classification task is defined as:

$$E_m(w,b) = -\frac{1}{n}\sum_i\sum_j\Big[y_j(x_i)\ln a_j^L(x_i) + (1 - y_i(x_i))\ln(1 - a_j^L(x_i))\Big] \tag{2.5.2}$$

where $a_j(x_i) = \varphi(z_j) = \varphi(\sum_j w_j x_j + b_j)$, and the derivative $\varphi(.)$ is given by $\varphi\,(x_i) = a(x_i)(1 - a(xi))$

The error is minimized through updates on weights. To minimize the error E_m each weight w^l_{jk} is updated by an amount proportional to the partial derivatives of E_m with respect to weight. These updates for w and b are determined using the Backpropagation method. We can now introduce the Backpropagation method and see how the partial derivatives change with respect to w and b.

2.6 Back Propagation Algorithm

Backpropagation method was first introduced around 1970 and it got accepted through a paper published in 1986 by (Zipser and Andersen, 1988). The main goal of back propagation is to compute the partial derivatives $\partial E_m \big/ \partial w^L_{jk}$ and $\partial E_m \big/ \partial b^L_j$ of the the cost function E, with respect to weight w and bias b in the network.

To understand how changing of the weights and biases in a neural network changes the cost function during back propagation. Let us use some notation we defined above. Considering the weighted input to the activation function for j^{th} neuron in layer l for network of L layers, and the output of the k^{th} neuron given by the following Equations (2.3.1) and (2.3.2):

$$z^l_j = \sum_{j=1}^{M} w^l_{j,k} a^{l-1}_k + b^l_j \tag{2.6.1}$$

where z^l_j is a weighted input to the activation function for the neuron j in layer l.

$$a^l_j = \varphi(z^l_j) = \varphi\left(\sum_{j=1}^{M} w^l_{j,k} a^{l-1}_k + b^l_j \right) \tag{2.6.2}$$

Equation (2.6.1) gives a global picture of thinking about how activation in one layer relates to activation previous layer, weight matrix are applied to the activation, then add a bias vector. Thus, then result in applying the φ function to get equation (2.6.2). Back propagation is all about understanding the rate of change of weight and biases with respect to cost function. In order to do that we must compute $\partial E_m \big/ \partial w^L_{jk}$ and $\partial E_m \big/ \partial b^L_j$. The derivatives are:

$$\begin{aligned}
\frac{\partial E_m}{\partial w^L_{j,k}} &= \frac{\partial}{\partial w^L_{j,k}} \left(-\frac{1}{n} \sum_i \sum_j \Big[y_j(x_i) \ln a^L_j(x_i) + (1 - y_i(x_i)) \ln(1 - a^L_j(x_i)) \Big] \right) \\
&= -\frac{1}{n} \sum_i \sum_j \left(y_j(x_i) \frac{\varphi\,(z^L_j)}{\varphi(z^L_j) a^{L-1}_k} + \frac{(1 - y_i(x_i))(-\varphi^{\,(z^L_j) a^{L-1}_k})}{(1 - \varphi(z^L_j)} \right) \\
&= -\frac{1}{n} \sum_i \sum_j \left(\frac{y_j(x_i)}{\varphi(z^L_j)} - \frac{(1 - y_i(x_i))}{1 - \varphi(z^L_j))} \right) a^{L-1}_k \varphi\,(z^L_j)
\end{aligned} \tag{2.6.3}$$

now by applying Equation (2.6.2) and (2.6.1) to Equation (2.6.3) we get the following equation.

$$\frac{\partial E_m}{\partial w^L_{j,k}} = \frac{1}{n}\sum_i \sum_j \Big(a^L_j - y_i(x_i)\Big)a^{L-1}_k \tag{2.6.4}$$

And similarly for bias we the same operation with respect to b^L_j

$$\begin{aligned}\frac{\partial E_m}{\partial b^L_j} &= -\frac{\partial}{\partial w^L_{j,k}}\left(\frac{1}{n}\sum_i\sum_j \Big[y_j(x_i)\ln a^L_j(x_i) + (1-y_i(x_i))\ln(1-a^L_j(x_i))\Big]\right)\\ &= -\frac{1}{n}\sum_i\sum_j\left(y_j(x_i)\frac{\varphi\,(z^L_j)}{\varphi(z^L_j)} - \frac{(1-y_i(x_i))(\varphi^{\,(z^L_j)})}{(1-\varphi(z^L_j)}\right)\end{aligned} \tag{2.6.5}$$

similalrly applying Equation (2.6.2) and (2.6.5) to Equation (2.6.5) we get the following equation.

$$\frac{\partial E_m}{\partial b^L_j} = \frac{1}{n}\sum_i \sum_j \Big(a^L_j - y_i(x_i)\Big) \tag{2.6.6}$$

Equations (2.6.4) and (2.6.5) improves the way the network learns, the network turns to output $\varphi(z^l_j + \Delta z^l_j)$ instead of $\varphi(z^l_j)$. this results to variation in cross entropy by an amount $\left(\frac{\partial E_m(w,b}{\partial z^l_j})\Delta z^l_j\right)$. This turns out to be good news since the main objective in back propagation is to minimize the error, thus the error improves the way the network learns. such error is defined as:

$$\frac{\partial E_m}{\partial w^l_{jk}} = \frac{\partial E_m}{\partial z^l_j}\frac{\partial z^l_j}{\partial w^l_{jk}} \tag{2.6.7}$$

where the error δ^l_j is the error defined as:

$$\delta^l_j \equiv \frac{\partial E_m}{\partial z^L_j} \tag{2.6.8}$$

we now compute the error defined in Equation (2.6.8) as:

$$\delta^l_j = \frac{\partial E_m}{\partial z^L_j} \tag{2.6.9}$$

$$\begin{aligned}&= -\frac{1}{n}\sum_i\sum_j\left(y_j(x_i)\frac{\varphi\,(z^L_j)}{\varphi(z^L_j)a^{L-1}_k} + \frac{(1-y_i(x_i))(-\varphi^{\,(z^L_j)a^{L-1}_k})}{(1-\varphi(z^L_j)}\right)\\ &= \frac{\partial E_m}{\partial w^L_{j,k}} = \frac{1}{n}\sum_i\sum_j\Big(a^L_j - y_i(x_i)\Big)\end{aligned} \tag{2.6.10}$$

due to some similarities on our equations, we can write equation (2.6.6) in terms of the error δ_j^l as follows:

$$\frac{\partial E_m}{\partial w_{j,k}^{[l]}} = \delta_j^{[l]} a_j^{[L-1]}$$

and

$$\frac{\partial E_m}{\partial b_j^l} = \delta_j^l$$

Instead of starting equation δ_j^l in terms of the next layer δ_j^{l+1}, lets observe how would cross entropy function for the layer $l-1$ which is close to the output which can be generalized in any hidden layer would look like, such evaluation is as follows:

$$\frac{\partial E_m}{\partial w_{ij}^{L-1}} = \frac{1}{m}\sum_i\sum_k \Big(\varphi(z_k^L - y_k)\Big)\frac{\partial z_K^L}{\partial w_{ij}^{L-1}}\frac{1}{n}\sum_i\sum_k \Big(\varphi(z_k^L - y_k))\Big)\frac{\partial z_K^L}{\partial a_j^{L-1}}\frac{\partial a_j^{l-1}}{\partial w_j^{L-1}} \tag{2.6.11}$$

Applying (2.6.1) and $\varphi\,(x_j)$ in equation (2.6.11), we get

$$\begin{aligned}
\frac{\partial E_m}{\partial w_{ij}^{L-1}} &= \frac{1}{m}\sum_i\sum_k \Big(\varphi(z_k^L - y_k)\Big) w_{jk}^L \frac{\partial a_j^{L-1}}{\partial w_j^{L-1}} \\
&= \frac{\partial a_j^{L-1}}{w_j^{L-1}}\Big[\sum_k wjk^L\Big(\sum_i (\varphi(z_K^L - y_k))\Big)\Big] \\
&= \frac{\partial a_j^{L-1}}{\partial w_j^{L-1}}\Big(\sum_k w_{jk}^L \delta_k^L\Big) = \varphi\,(z_K^{L-1})\frac{\partial z_j^{L-1}}{\partial w_{ij}^{L-1}}\Big(\sum_k w_{jk}^L\delta_k^L\Big) \\
&= \varphi(z_j^{L-1})(1-\varphi(z_j^{L-1}))a_j^{L-2}\sum_k w_{jk}^L\delta_k^L
\end{aligned}$$

The final expression becomes:

$$\frac{\partial J}{\partial w_{i,j}^{[L-1]}} = \delta_j^{[L-1]} a_j^{[L-2]} \tag{2.6.12}$$

Where

$$\delta_j^{L-1} = \varphi(z_j^{L-1})(1-\varphi(z_j^{L-1}))a_j^{L-2}$$

In the equation above, we see that the error δ^{l-1} in the layer $L-1$ is expressed in terms of δ^l. We can also do the same and express the equation for the error δ^l in terms of the error in the next layer, δ^{l+1}, which is given by

$$\delta^l = a_j^{l+1}(1-a_j^{l+1})w_{ji}^{l+1}\delta^{l+1}$$

We can now on explain or discuss how weights and biases varies during back propagation method. However we noted that the bias units have no incoming weights, thus meaning we don't have to compute gradient per bias unit. Note that we only need gradient for out coming bias weights. This happens for any algorithms. We will discuss how gradient descent is used to minimize the cost function.

2.7 Update of Weight and Biases

During training the model, we need to have parameters that minimizes the error function. This is done by applying a technique called Gradient descent. We use this technique to find optimum parameters, We first by randomly initializing the weights and biases and then iteratively changing the parameters in the direction of negative gradients of the cost function until a global minimum is reached. If η is the current iteration, the new updates of the weights and biases are as follows:

$$w_{ij}^{lnew} = w_{ij}^{lold} - \eta \frac{\partial E_m}{\partial w_{jk}^{L}} = w_{ij}^{lold} - \eta \sum_j a_j^{l-1} \delta_j^l \quad and \quad b_j^{lnew} = b_j^{lold} - \eta \frac{\partial E_m}{\partial b_j^{L}} = b_j^{lold} - \eta \sum_j \delta_j^l$$

where the factor η is the learning rate. It is known as the step size between two successive positions of the parameters. To come up with an optimum learning rate is a difficult task during model training, if the step size is small, the algorithm slows down and takes long time to converge and if the step size is too large, the algorithm takes the opposite effect, it diverges which leads to overshooting. 0.35 has been taken as standard learning rate when training artificial neural network. We will discuss the learning rate that is used in this study later on. The following figure illustrates how how supervised learning process of neural network works.

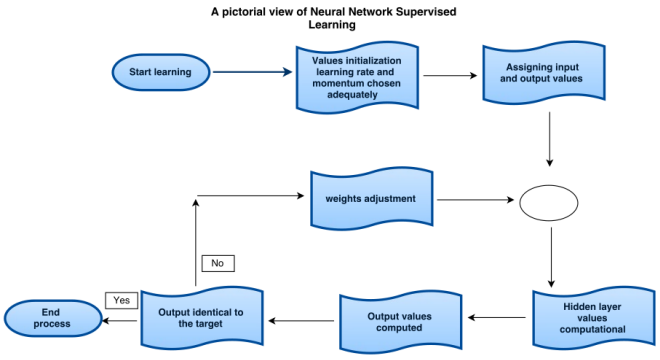


Figure 2.4: Supervised Learning Process: Adopted from Addai (2016).

3. Recurrent Neural Network

In recent years, Recurrent Neural Networks (RNNs) have been of interested to academia, as well. This is because they are able to learn features and long term dependencies from sequential time-series data. If one believes that the data has some kind of autoregressive structure, a recurrent neural network can be used (Hansson, 2017). Recurrent Neural Networks (RNNs) are defined as artificial neural networks whose neurons send feedback signals to each other. Recurrent neural Netowrk incorporates at least than one feed-back connection, so the activation flows in a cyclic fashion. RNN allow temporal processing and learning of sequences, e.g; perform sequence recognition, reproduction or temporal association and stock market prediction. Traditional neural networks assumes that the inputs and outputs are independent of each other, but that is not a good idea when dealing with other problems. For example if a person wants to predict the next word in a sentence using neural networks, it is better if he/she knows which words come before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being dependent on the previous computations. Another way to think about RNN is that they have a memory which stores information about what has been calculated so far. It is worth to saying that RNN are hypothetically able to treat arbitrarily long sequences of data, but in practice they are constrained to looking back only for few steps (Di Persio and Honchar, 2016).

In this chapter, we will discuss the architecture of a simple recurrent neural network and how the back propagation algorithm works in this particular case. We will also look at technique used to tune hyperparameters and finish by discussing some metrics that we use to evaluate our model.

3.1 RNN architecture

In recurrent neural networks we introduce a notion of time to the model. They can also be referred as feedforward neural networks because they are made by incoporation of edges that reach adjacent time steps (Lipton et al., 2015). Edges that link adjacent time steps are called recurrent edges. They form cycles in the network. This includes self connection cycles from a node to itself across time in the network. This cycles can be seen in the Figure 3.1 below. In order to understand the formulas that govern recurrent neural networks, Suppose that x_t is a sequence of input data. Nodes with recurrent edges receive input from the current input point and also from hidden state values h_{t-1} in the network's previous state. Each neuron in the hidden nodes performs a linear matrix operation with the input data x_t and result of this operation is fed into an activation function (e.g. tanh, sigmoid, softmax etc.) depending on the type of prediction in consideration. The output $\hat{y}_t$ at each time t is calculated using the hidden values h_t. Input x_{t-1} at time $t-1$ influences the output $\hat{y}_t$ at time t and later by the way of the recurrent connections. The results of linear matrix operation of these parameters gives two equations that are necessary for computation at each time step on the forward pass in a folded simple recurrent neuaral network shown in Figure 3.1. The two equations are:

$$h_t = \sigma(W^{xh}x_t + W^{hh}h_{t-1} + b_h \tag{3.1.1}$$

$$\hat{y}_t = \sigma(W^{hy}h_t + b_y) \tag{3.1.2}$$

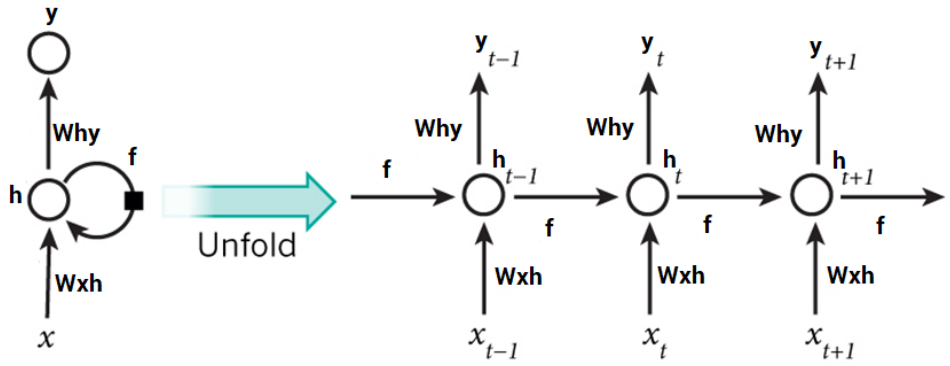


Figure 3.1: A folded simple Recurrent Neuaral Network: Adopted from Recurrent Neural Network.

Equation 3.1.1 is used to compute the hidden layer output at each time step and Equation 3.1.2 is used to compute $\hat{y}^{(t)}$ output. Here W^{hh} is the weight matrix connecting the hidden layers, W^{xh} is the weight matrix between the input layer and the hidden layer and W^{hy} is the weight matrix between the hidden layer and the output layer. The vectors b_h and b_y represent bias parameters. The weights matrices here in recurrent neural networks are shared parameters as like in feedfoward neural networks, they are just used in different time steps. If the dimensions of the input layer, output layer and hidden layer are D_x, D_y and D_h respectively. Then the dimensions of weights W^{hh} , W^{xh} and W^{hy} are $D^h \times D^h$, $D^h \times D^x$ and $D^y \times D^h$ respectively.

The Figure 3.1 below shows the unrolled Recurrent Neural. In this picture the network is no longer interpreted as cyclic but a complete sequence sharing weights across time steps. The unrolled full network can be trained across many time steps. Training RNN is similar to simple artificial neural network, it also uses the backpropagation method with slight difference. Since the weights and biases are shared by all time steps in this network, the gradient at each output depends on the calculation of the previous time steps and not only on the calculations of current time steps. For example if we are training RNN, in order to calculate the gradient at t=3, we would need to back propagate 2 steps and add the gradients. This method is called Backpropagation Through Time(BPTT) which almost all recurrent neural networks use it.

3.1.1 Initialization of Weights and biases. Prior to training the model, initialization of weights and biases in RNNs is critical. The previous hidden state is initialized to a zero vector and the weights are assigned to a small values or can be initialized randomly. Pascanu et al. (2013) pointed that a general rule is to assign small values to the weights and also drawing a Gaussian distribution with the standard deviation of 0.001 or 0.01 is a reasonable choice. We will discuss the drawbacks involved in RNNs in the subsequent sections.

3.2 Back-propagation through time

In previous section, we showed how we can use Backpropagation algorithm in a Multi-layer perceptron. Now in RNN, the same process can be applied but with some twist in the method. Back-propagation through time is the supplement of traditional (BP) used in the FeedForward Neural Network. This algorithm is used for updating of weights and biases in the network in order to minimize cost function.

It is also called a propagated gradient descent algorithm. During weight updates, different gradient descents are used for learning.

During weights updating, the change in weight is proportional to the negative gradient of the error function with respect to the weight. Mathematically is given by:

$$\Delta = -\eta \frac{\partial E_m}{\partial W} \tag{3.2.1}$$

Here η is referred to as the learning rate. It is a hyperparameter that gets chosen by the user during training. The main goal here is to calculate the gradient of the error function with respect to the parameters. To compute gradient of the error with respect to the weight, we sum up all the errors and also sum up the gradients at each time step t.

$$\frac{\partial E_m}{\partial W} = \sum_t \frac{\partial E_t}{\partial W} \tag{3.2.2}$$

.

To get these gradient of the error at time step t, $\frac{\partial E_t}{\partial W}$, we use chain rule:

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W} \tag{3.2.3}$$

$\frac{\partial h_t}{\partial h_k}$ is the partial derivative of h_t, with respect to all previous time steps $k = 1, 2, \ldots, t-1$

$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \tag{3.2.4}$$

The dimensions of input layer, output layer and hidden layer are D_x, D_y and D_h, respectively. Thus each $\frac{\partial h_i}{\partial h_{i-1}}$ is the jacobian matrix for the hidden layer. The elements of the matrix are all pointwise derivatives.

$$\frac{\partial h_i}{\partial h_{i-1}} = [\frac{\partial h_i}{\partial h_{i-1,1}} \cdots \frac{\partial h_i}{\partial h_{i-1,D_h}}] = \begin{bmatrix} \frac{\partial h_{i,1}}{\partial h_{i-1,1}} & \cdots & \frac{\partial h_{i,1}}{\partial hi-1,D_h} \\ \vdots & \vdots & \ddots \\ \frac{\partial h_{i,D_h}}{\partial h_{i-1,1}} & \cdots & \frac{\partial h_i}{\partial h_{i-1,D_h}} \end{bmatrix} \tag{3.2.5}$$

We can rewrite the above gradient by substituting (3.2.4) into (3.2.3) and then in (3.2.2), to get:

$$\frac{\partial E_m}{\partial W} = \sum_{t=1}^{T} \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \Big(\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \Big) \frac{\partial h_k}{\partial W} \tag{3.2.6}$$

3.3 Exploding and vanishing gradient problem

In the previous section we saw how training a neural network using gradient descent works by calculating the gradient of the output with respect to the weight matrix. The Backward pass of the gradient of the error function could cause an explosion or vanishing problem. Bengio et al. (1994) introduced the exploding and vanishing problems, he referred to the explosion problem as being large increases in the norm of the gradient during training, which are caused by the explosion of the long term components which grows exponentially over time. The vanishing gradients were referred as the opposite case of the exploding gradients, such that the long term components go exponentially fast to 0, causing the model not to be able to learn correlation between temporary distant events. The biggest challenge with training RNN models is that they have difficulties learning long range dependencies, which results in the two problems we introduced above. The choice of activation functions and network parameters, such as initializing of the weight matrix could results to the exploading and vanishing problems. have:

3.4 Evaluation metrics

There are different performance metrics used to evaluate the perfomance of classifiers. In this essay we will focus on the ones for the classification problem. There are also many performance metrics for classification such as Accuracy, Log-Loss, Area under curve(AUC)etc . In this study, we consider only confusion matrix as our performance measure.

3.4.1 Confusion Matrix. Confusion matrix score is used for finding the accuracy of the model. It's mainly used for classification problems where the outputs can be two or more classes. But in this essay, since the output is Up Or Down movement, it is used to show the number of correct and incorrect Up and Down predictions made by ANN and RNN classifiers compared to the actual/target outcomes in the data. This metric gives the summary of a classification when looking as the error made by the classifier. In confusion matrix columns we find the classifier's prediction labels and in the rows we find the actual labels. On the diagonal of this matrix, we find elements which the predicted class matches the true class and other entries contains elements misclassified.

	Class 1 Predicted	*Class 2 Predicted*
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

Figure 3.2: Confusion Matrix: Adopted from Confusion Matrix in Machine Learning.

3.4.2 Accuracy. Is defined as the ratio of the total number of correct predictions elements to the total number of elements classified. It is the most used metric and it is easy to interpret. Rechenthin

(2014) reported that the accuracy metric is not really descriptive for measuring performance of highly imbalanced data. The accuracy of classifier is given by:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total observations}} \tag{3.4.1}$$

There are several other approaches that are used to compare models with imbalanced data, Including:

3.4.3 Precision. This is defined as the ratio of correct positive predicted elements in a class to the total number of positive predicted elements in a class. It gives us the percentage that the model correctly predicted positive during training. Precision tells us about how precise the model is out of the predicted positive, we actually use precision to know how many of them are actually positive. it is also referred to as (Positive predictive value) and is given by:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True positives} + \text{False Positives}} \tag{3.4.2}$$

3.4.4 Recall. . This metric gives the percentage of positives correctly identified out of all existing positive elements. It tells us how many of the actual positives our model captures through labelling it as true positive. It is given by:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True positive} + \text{False Negative}} \tag{3.4.3}$$

3.4.5 F1 score. This metric is used to seek the balance between Recall and Precision. It is the harmonic mean of the two metrics. The measure ranges from 0 to 1 and if the score is 1, it possible corresponds to perfect classifier capturing of precision and recall and if the measure is 0, it is the worst classification. It is given by:

$$\text{F- measure} = \frac{2(Precision)(Recall)}{Precision + Recall} \tag{3.4.4}$$

4. Data and Methodology

4.1 Data Collection

This chapter presents the source of data and its usefulness in our study. It also explains the techniques used to transform the dataset into a form thhat suit the purpose of the study and how it has been used for both two models. It will discuss the feature selection of the models and the results of the two models models will be compared to determine the best classifier for predicting the S&P 500 index.

4.1.1 Data Collection. The data of this study has been collected from Yahoo Finance. It contain the daily closing price of S & P 500 as the response variable and 5 financial and economical variables as the input features for 3652 days partially taken on the period starting from 09 August 2008 to 05 August 2018. The availability of financial data at this time is considered as one of the essential factors. This data also contains historical information on the opening prices, low prices, high prices volume traded, closing prices and adjusted closing prices. Selection of financial and economical variables is one of the main consideration when building predictive models od stock market. Thus, the features were chosen based on whether they have significant influence on the S & P 500 closing price. Stocks/indices used in this study are regarded as input features and are believed to have significant effect on the closing price of S & P 500. Table 4.1 gives review of stocks/indices utilized in this study.

Table 4.1: Stocks/indeces

Stock/Index	Stock Market Traded
Exxol oil(XOM)	New York Stock Exchange (NYSE)
Microsoft (MSFT)	NASDAQ
Johnson and Johnson (JNJ)	New York Stock Exchange (NYSE)
Dow Jones Industrial Average (DJI)	New York Stock Exchange (NYSE)
NASDAQ Composite (IXIC)	NASDAQ

4.1.2 Data Preprocessing. Even though neural networks are able to predict the movement of stock market prices, they still have some limitations due to instability of financial time series data. Data preprocessing must be done prior to training the models. It is a technique of transforming and cleaning data into understandable form in order to improve the performance of the model and the quality of the data.

The main limitations that face the artificial neural network in predicting stock market is that financial time series data contain a tremendous amount of noise, non-stationary characteristics, and their complexity (Kara et al., 2011). Therefore, data preprocessing must be performed prior to downstream analysis. Asadi et al. (2012) showed that preprocessing procedures, such as data transformation and extraction of desired input variables improves the accuracy of the model. According to Addai (2016), the performance of predictive models depends on the stage of preprocessing, because training the models with the raw financial data, which is noisy in nature, results into over fitting and under fitting of the model in predicting the output.

4.1.3 Data splitting. Prior to training the Neural Networks, the dataset must be partitioned into two portions namely training set and test set. This is done for performance purpose. The two distinct sets are used for different reasons, Training set is used to develop and the model and the Test set is used for

evaluation. In practise, after preprocessing the data, the first 80% of the compiled data (for 2920 days) was used for training, the next 10% was used for validation and the last 10% was used for testing the performance of trained models.

4.1.4 Data Normalization. Another preprocessing stage called or scalling technique, involved in the field of computing and is called Normalization. It is a process of rescalling data to range of 0 and 1. This means that the smallest value in the data set is 0 and the largest value in the data is 1. A study done by Patro and Sahu (2015) showed that normalization of data can be helpful for the prediction or forecasting purpose. There are many methods for data normalization, this includes Min-Max normalization, Z-score normalization and Decimal scaling normalization. In this essay was normalized using Min-Max normalization. Mathematically, this techniques is applied as follows:

Let R be a data set, R_{min} and R_{max} be the minimum and maximum values of R respectively. Then the formula for normalizing (scaling) R is given by Normalization:

$$R = \frac{R - R_{min}}{R_{max} - R_{min}} \tag{4.1.1}$$

This technique was applied to our dataset prior to training the model.

4.1.5 Potential Influential features. The features used in this essay to predict the direction of S& P 500 are taken from various indices/stocks, based on their statistical significant with the output/closing price of S&P 500. Some of the features used in this study were used in previous studies. Table 4.2 gives description of the selected features used. Table 1 describes the state variables.

Table 4.2: Description of state variables

Feautures	Explanation
GSPC.Open(t)	Open Price of S&P 500 index in day t
MSFT.Open(t)	Open price of Microsoft stock in day t
JNJ.Open(t)	Open price of Amazon stock in day t
DJI.Open(t)	Open price of MIcrosoft stock in day t
IXI.Open(t)	Open price of Twitter stock in day t

The Opening prices of various indices/stocks above were considered as input variables to predict the closing price of S&P 500 the following day $t + 1$. mathematically this can be represented as:

$$\left[Open_t\right]_{\textbf{Input}} = \quad [close_{t+1}]_{\textbf{Output}}$$

4.2 Neural Networks Training

4.2.1 Artificial Neural Network. The Feed forward algorithm for a neural network was implemented in python with some useful help from built in functions including pandas, keras and Tensorflow. The ANN had been trained using proposed indices/stocks discussed above and after that the performance of the model was tested and then evaluated. The predictions done in this study for ANN was focused in a short term forecasting, which in our case it was a daily forecasting where the model everyday predicts

the closing price of the next day. During learning process in ANN, the previous predictions are not used in the next target prediction. The ANN model trained in this study was based on the following parameters values:

- Inputs and output (All variables discussed above were included)
- Training set (80%)
- Testing set (10%)
- Validation set (10%)
- Learning rate (0.03)
- Hidden layers($N_H = 2$)
- Number of epochs(100 and 150)
- batch size (300 and 200)

4.2.2 Recurrent Neural Netwok Implementation. Similarly, for the recurrent neural network model was implemented in python built in functions, Tensorflow and keras. The model used was done with moving ("rolling") window, at each step, a constant size features (inputs) and outputs were extracted during the learning process and each series were treated as a source of many input/output records. This method was chosen because time series forecasting use a lot of parameters and a single short time series does not provide enough data for successful training, which happens a lot in a non recurrent neural networks. The RNN model considered was there to learn across many time series, which we see failing to happen in non recurrent neutral networks because the series diverge a lot for smaller past values. This is as we will see later in the results section, why recurrent neural network method performed better than simple feed forward. The parameters used in model were:

- Inputs and output (All variables discussed above were included)
- Training set (80%)
- Testing set (10%)
- Validation set (10%)
- Learning rate (0.03)
- Number of epochs(100 and 150)

5. Results and Conclusion

5.1 Results and Discussion

This chapter present results of our essay for both Neural Networks models we implemented. The aim is to analyze the obtained findings and to validate and compare both classifiers.

5.1.1 Correlation and feature selections. Before we implemented the models, choosing the variables as input features, which are relevant to the underlying problem of predicting the direction of S & P 500 index, can results in obtaining a good forecasting accuracy. Niaki and Hoseinzade (2013) reported that the key factor to successful forecast the finacial time series is to consider models which have least complexity and only relevant features. But since we know that financial time series data are noisy by nature, selection of features has its own drawbacks. For example, if the variables chosen are believed to have some useful information, and neural networks fails to extract that information to use and to capture the relations between variable and the output, this is one of the drawbacks. In this essay in order to decide the variables used as features, we performed a Pearson correlation test between the potential variables mentioned above to establish the relationship between the variables. In person correlation we say that variables are correlated when a change in the value of one affects the other. Correlation between two variables is said to be significant when the coefficient r is greater than 0.5, the coefficient lies between $[-1,1]$

In the correlation matrix below, we see that the the diagonal elements take the value of 1, this is because, each variable is perfectly linearly correlated with itself. The idea of performing person's correlation was to be able to find the relationship between the response variable and inputs. We observe that Exxon company index was negatively correlated with the closing price of our response variable (see table 5.1) which caused us to reduce it as a predictor variable the model.

Table 5.1: Correlation Matrix

	MSFT_open	JNJ_open	DJI_open	XOM_open	IXI_open	GPSC_open	GPSC_close
MSFT_open	1.000000						
JNJ_open	0.883786	1.000000					
DJI_open	0.972886	0.920273	1.000000				
XOM_open	-0.539284	-0.380011	-0.475101	1.000000			
IXI_open	0.975688	0.915722	0.985579	-0.538865	1.000000		
GPSC_open	0.958869	0.944026	0.985102	-0.479829	0.993132	1.000000	
GPSC_close	0.958495	0.943947	0.984330	-0.478908	0.992623	0.999205	1.000000

Figure 5.3 below shows the plot of opening prices of selected features with the closing price of our response variable. We used it to validate the findings of correlation matrix. We can see that the opening price of stock index XOM was not following the trend flow of our response variable ($GPSC_{close}$). This is because according to the correlation findings, the XOM stock index was negatively correlated with the closing price of our output variable. This resulted in reducing the variable from inputs data list. According to Addai (2016), the reduction of a such feature is a necessity when training a neural network model, due to the fact that big models stands the risk of over-fitting and hence models should be built in such a way that they account for the trade-off between the data fitting and the model uncertainties.

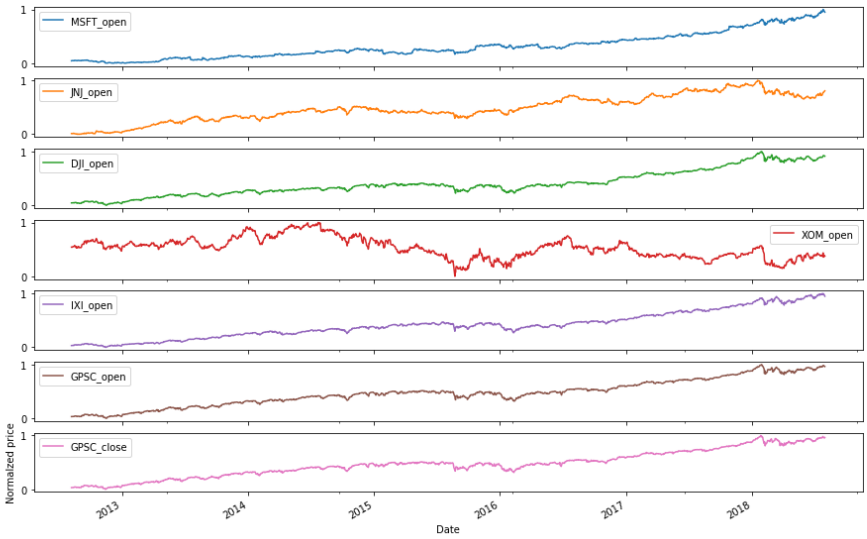
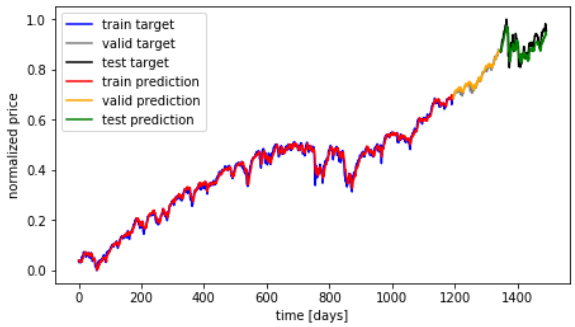


Figure 5.1: The figure represent the graphical representation of opening price flow of the features and closing price for response variable (S & P 500)

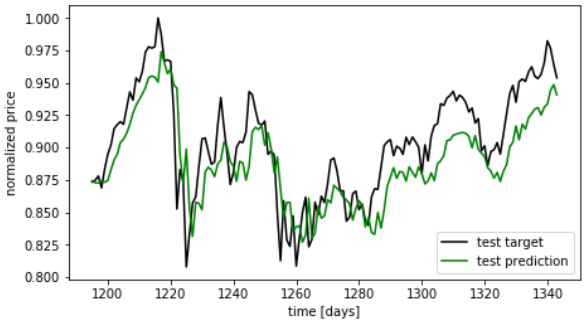
.

5.1.2 Recurrent Neural Network Results. Stock market Predictions is very crucial. If one could be able to come up with a forecasting tool, we could increase return on investment. Stock markets could be described as a multi-dimensional monstrosity full of complexities and inter-dependencies which is influenced by various of factors. Now what if this dilemma could be clarified by means of machine learning. so far neural networks has been seen to posses the capabilities to tackle the above mentioned dilemma. That's why we used Recurrent Neural Network to predict the direction of S & P 500 index. The RNN model used was trained with some useful help from built-in functions in python. Our results clearly shows that RNN are able to predict the direction of closing price of S & P 500 index as shown in figure, 5.2a and 5.2b. In simmulation of the model, we used python 3.0 to develop the RNN model to predict S & P index. Numpy, pandas, matplotlib, scikitlearn and tensorflow packages in Python were used in this work. Scikitlearn was used for Pre-processing the data. We have used the indices and stocks with the parameters discussed in Section 4.2. The two figures, 5.2a and 5.2b mentioned helped us to validate if the RNN model was able to predict the direction of our response variable. As we can see in Figure 5.2a. Comparing actual against predicted set, we see that the RNN model was able to flow the trend of S & P closing price based on the features we mentioned before. This study focuses mainly in classifying the up and down movement of the S & P 500 index. In section 5.1.3 will see how the RNN model performed in the classification task using the confusion matrix.



(a) Comparing Actual Against Predicted in (RNN)

.



(b) Test target and test prediction in (RNN)

.

Figure 5.2: The figures shows actual against predictions for RNN model.

5.1.3 Recurrent Neural Network Confusion Matrix. In predicting the movement of the S & P 500 index for the next day we used a regression. This was done because financial time series data are continuous. However, in this essay we are only interested in the binary output of the closing price of S & P 500, hence we classified the predicted outputs into two classes, "Up" and "Down", where we assigned 0 with "Down" and 1 with "Up". This enabled us to measure the performance power of the model by implementing the confusing matrix. In figure 5.3 we can see that the values in the predicted set of the recurrent neural network model predicted 83 trends correctly. Out of 64 downward trends only 15 were misclassified as upward trends and also out 85 upward trends, 34 were misclassified as downward trends which lead to 56% accuracy score of the model. The accuracy of the model defined as ratio of the total number of correct predictions elements to the total number of elements, helped us to the compare performance power of the two models present in this study. We used Equation 3.4.1 to calculate the accuracy of the model. The formula of accuracy is given by :

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total observations}} \tag{5.1.1}$$

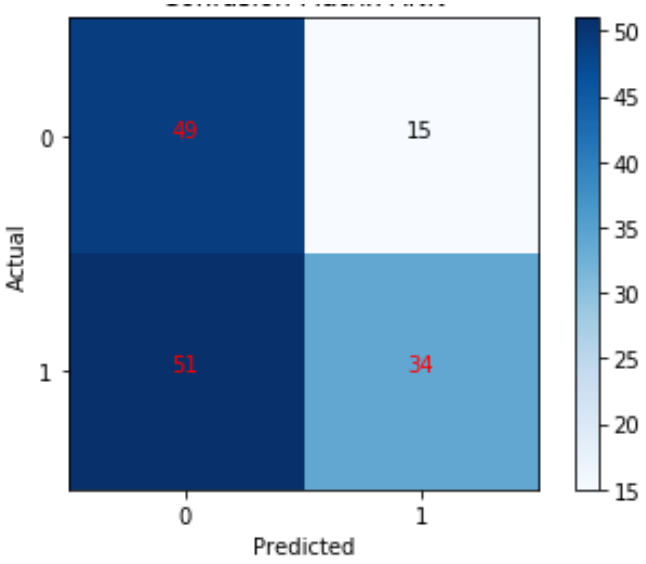
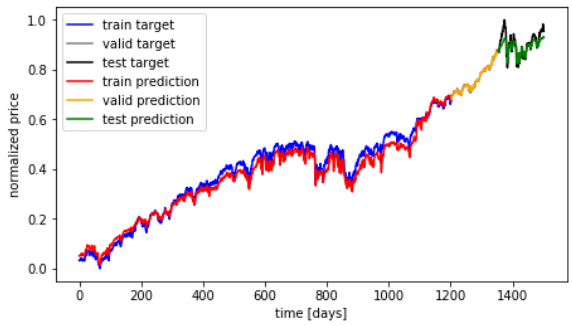
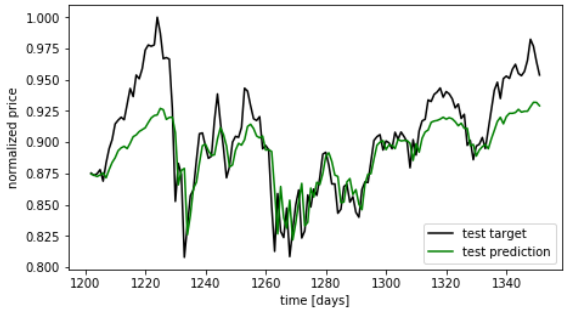


Figure 5.3: A confusion matrix showing a RNN Classification performance

5.1.4 Artificial Neural Network Results. In this section we present the obtained regression predictions results of ANN method. In Figure 5.4a and 5.4b we can see that the ANN model was able to predict the price movement of S & P 500 index. We use the two plots presented here, to validate the fact that the model can be used in predictions, we did not use it as a metric measure to compare the predictive power of the models.



(a) Comparing Actual Against Predicted (ANN)



(b) Test target and test prediction in (ANN

Figure 5.4: The figures shows actual against predictions for ANN model.

5.1.5 Artificial Neural Network Confusion Matrx. In Figure 5.5, we present a summary of the performance of the artificial neural network Model we used. We observe that, the model was correct in predicting that prices will go down whiles it actually went down on 59 occasions. Also, it correctly predicted the upward movements 85 times. On the negative side, the model misclassified the direction of the returns on 76 instances. This limits the overall accuracy of the model in the classification to approximately 50 percent.

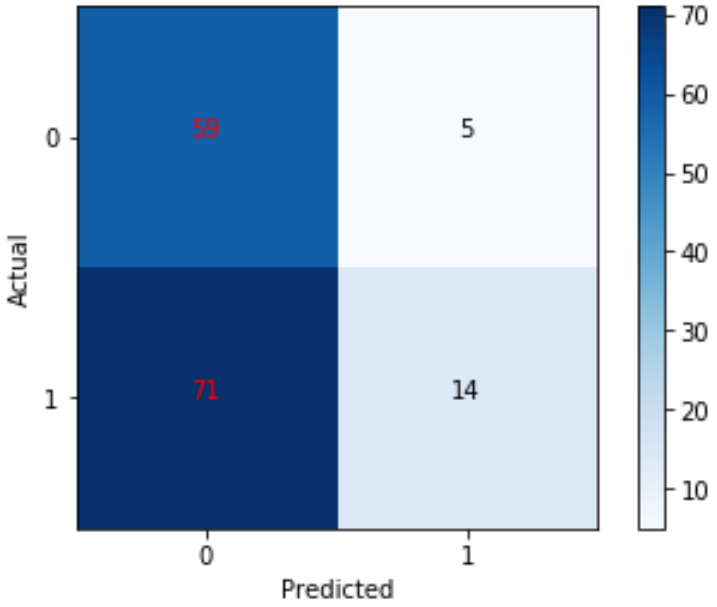


Figure 5.5: A confusion matrix showing a ANN Classification performance

Table 5.2 provides a summary of classification accuracies of the two algorithms implemented, using the opening prices of DJI, MSFT, JNJ, and IXIC stock indeces to predict the S & P 500 closing price one day ahead.

Table 5.2: Performance summary of the two algorithms

Algorithm	Number of correctly classified	Accuracy
ANN	73	50%
RNN	83	56%

The predicted results of the Recurrent Neural Netowk (RNN) and Artificial Neural Network (ANN) using the proposed performance criteria, clearly shows that RNN perform better than ANN in the terms of classification accuracy.

6. Conclusion

In this work, two artificial neural network models were presented, namely ANN and RNN to predict American stock market index (S & P 500) particularly with respect to it's oscillatory nature for trend movements. It has been observed that RNN perform better than ANN measured on classification accuracy. According to the results obtained in this work it is observed that using a larger dataset during the training phase is capable of improving accuracy without negatively impacting the performance of the model. This could have improved the classification accuracy of the RNN model used in this work since it consists of complex activation functions to capture the long term dependencies. In a case of short term dependency like what was used in this study, the performance of RNN and Feed forward neural network were not too different. In terms of classification measure both models performed well. In the instance of binary classification, both models were able to classify the direction of the S & P 500 index i.e if the price went "up" or "down" the next day the model was able to accurately predict the upward or downward movement. Although the scope of this work did not include extensive study of the improvements made by tuning parameters this is a viable option in attempting model accuracy improvement.

References

Abu-Mostafa, Y. S. and Atiya, A. F. Introduction to financial forecasting. *Applied Intelligence*, 6(3): 205–213, 1996.

Activation functions. Activation functions. KD nuggets, https://www.kdnuggets.com/2017/09/neural-network-foundations-explained-activation-function.html, Accessed September, 2018.

Addai, S. Financial forecasting using machine learning. *African Institute for Mathematical Science (AIMS)*, pages 1–32, 2016.

Arlot, S., Celisse, A., et al. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

Artificial Neural Networks. Artificial neural networks. Tum wiki, https://wiki.tum.de/display/lfdv/Artificial+Neural+Networks, Accessed September, 2018.

Asadi, S., Hadavandi, E., Mehmanpazir, F., and Nakhostin, M. M. Hybridization of evolutionary levenberg–marquardt neural networks and data pre-processing for stock market prediction. *Knowledge-Based Systems*, 35:245–258, 2012.

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Confusion Matrix in Machine Learning. Confusion matrix. Geeks for Geeks, https://www.geeksforgeeks.org/confusion-matrix-machine-learning/, Accessed September, 2018.

Devadoss, A. V. and Ligori, T. A. A. Forecasting of stock prices using multi layer perceptron. *International journal of computing algorithm*, 2:440–449, 2013.

Di Persio, L. and Honchar, O. Artificial neural networks architectures for stock price prediction: comparisons and applications. *International Journal of Circuits, Systems and Signal Processing*, 10:403–413, 2016.

Durham, G. B. Sv mixture models with application to s&p 500 index returns. *Journal of Financial Economics*, 85(3):822–856, 2007.

Guresen, E., Kayakutlu, G., and Daim, T. U. Using artificial neural network models in stock market index prediction. *Expert Systems with Applications*, 38(8):10389–10397, 2011.

Hansson, M. On stock return prediction with lstm networks. 2017.

Haykin, S. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Jabin, S. Stock market prediction using feed-forward artificial neural network. *growth*, 99(9), 2014.

Kara, Y., Boyacioglu, M. A., and Baykan, Ö. K. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert systems with Applications*, 38(5):5311–5319, 2011.

Kim, K.-j. and Han, I. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert systems with Applications*, 19(2):125–132, 2000.

Lipton, Z. C., Berkowitz, J., and Elkan, C. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

McCulloch, W. S. and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

McNelis, P. D. *Neural networks in finance: gaining predictive edge in the market*. Academic Press, 2005.

Nasr, G. E., Badr, E., and Joun, C. Cross entropy error function in neural networks: Forecasting gasoline demand. In *FLAIRS Conference*, pages 381–384, 2002.

Niaki, S. T. A. and Hoseinzade, S. Forecasting s&p 500 index using artificial neural networks and design of experiments. *Journal of Industrial Engineering International*, 9(1):1, 2013.

Nigrin, A. *Neural networks for pattern recognition*. MIT press, 1993.

Oksanen, R. New technology-based recruitment methods. Master's thesis, 2018.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

Patel, J., Shah, S., Thakkar, P., and Kotecha, K. Predicting stock market index using fusion of machine learning techniques. *Expert Systems with Applications*, 42(4):2162–2172, 2015.

Patro, S. and Sahu, K. K. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*, 2015.

Qiu, M. and Song, Y. Predicting the direction of stock market index movement using an optimized artificial neural network model. *PloS one*, 11(5):e0155133, 2016.

Raschka, S. and Mirjalili, V. *Python machine learning*. Packt Publishing Ltd, 2017.

Rather, A. M., Agarwal, A., and Sastry, V. Recurrent neural network and a hybrid model for prediction of stock returns. *Expert Systems with Applications*, 42(6):3234–3241, 2015.

Rechenthin, M. D. Machine-learning classification techniques for the analysis and prediction of high-frequency stock direction. 2014.

Recurrent Neural Network. Recurrent neural network. Analytics vidya, https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/, Accessed September, 2018.

Roh, T. H. Forecasting the volatility of stock price index. *Expert Systems with Applications*, 33(4): 916–922, 2007.

Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

Shen, W., Guo, X., Wu, C., and Wu, D. Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm. *Knowledge-Based Systems*, 24(3):378–385, 2011.

Stone, M. Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society. Series B (Methodological)*, pages 111–147, 1974.

Tan, C. N. *Artificial neural networks: applications in financial distress prediction & foreign exchange trading*. Wilberto, 2001.

Tan, T. Z., Quek, C., and Ng, G. S. Biological brain-inspired genetic complementary learning for stock market and bank failure prediction 1. *Computational Intelligence*, 23(2):236–261, 2007.

Wang, J., Wang, J., Fang, W., and Niu, H. Financial time series prediction using elman recurrent random neural networks. *Computational intelligence and neuroscience*, 2016, 2016.

Webos. Back propagation. Wikipedia, the Free Encyclopedia, http://en.wikipedia.org/wiki/Backpropagation., Accessed October 2018.

Widrow, B. Darpa neural network study. *Armed Forces Communication and Electronics Association, Fairfax, VA1988*, 1988.

Zipser, D. and Andersen, R. A. A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*, 331(6158):679, 1988.

Zurada, J. M. *Introduction to artificial neural systems*, volume 8. West publishing company St. Paul, 1992.