



DEGREE PROJECT, IN COMPUTER SCIENCE , FIRST LEVEL  
*STOCKHOLM, SWEDEN 2015*

# Stock Forecasting Using Artificial Neural Networks

A QUANTITATIVE STUDY OF A  
FEEDFORWARD NEURAL NETWORK'S  
ACCURACY WITH RESPECT TO THE NUMBER  
OF NEURONS AND THE TRAINING DATASET  
DISTRIBUTION

DANIEL MILLEVIK AND MICHAEL WANG





**KTH Computer Science  
and Communication**

# **Stock Forecasting Using Artificial Neural Networks**

A Quantitative Study of a Feedforward Neural Network's  
Accuracy with Respect to the Number of Neurons  
and the Training Dataset Distribution

DANIEL MILLEVIK  
MICHAEL WANG

Degree Project in Computer Science, DD143X  
Supervisor: Michael Schliephake  
Examiner: Örjan Ekeberg

CSC, KTH, 8 May 2015



## Abstract

This paper studies the potential of artificial neural networks (ANNs) in stock forecasting. It also investigates how the number of neurons in the network, as well as the distribution of the training data into training, validation and testing sets, affect the accuracy of the network. By using MATLAB and its Neural Network Toolbox tests were carried out with a two-layer feedforward neural network (FFNN). These are carried out by collecting five years of historical data from the Dow Jones Industrial Average (DJIA) stock index, which is then used for training the network. Finally, retraining the network with different configurations, with respect to the number of neurons and the training data distribution, in order to perform tests on a separate year of the DJIA stock index. The best acquired accuracy for predicting the closing stock price one day ahead is around 99%. There are configurations that give worse accuracy. These are mainly the configurations using many neurons as well as the ones with low training data percentage. The conclusion is that there is potential for stock forecasting using ANNs but only predicting one day forward might not be practically useful. It is important to adapt the network to the given problem and its complexity and thus choosing the number of neurons accordingly. It will also be necessary to retrain the network several times in order to find one with good performance. Besides the training data distribution it is more important to gather enough data for the network's training set to allow it to adapt and generalize to the problem at hand.

## Sammanfattning

Denna rapport studerar ifall artificiella neuronnät (ANN) potentiellt kan tillämpas på den finansiella marknaden för att förutspå aktiepriser. Det undersöks även hur antalet neuroner i nätverket och hur fördelningen av träningsdatat i träning, validering och testning, påverkar nätverkets noggrannhet. Tester utfördes på en "two layer feedforward neural network" (FFNN) med hjälp av MATLAB och dess Neural Network Toolbox. Dessa utfördes genom att samla fem år av historisk data för "Dow Jones Industrial Average" (DJIA) aktieindex som används för att träna nätverket. Slutligen så tränas nätverket i omgångar med olika konfigurationer bestående av ändringar på antalet neuroner och fördelningen av träningsdatat. Detta för att utföra tester på ett separat år av DJIA aktieindex. Den bästa noggrannheten som erhöles vid förutsägning av stängningspriset i börser efter en dag är ca 99%. Det finns konfigurationer som ger sämre noggrannhet. Dessa är i synnerhet konfigurationer med ett stort antal neuroner samt de med låg andel träningsdata. Slutsatsen är att det finns potential vid användning av artificiella neuronnät men det är inte praktiskt användbart att bara förutspå aktiepriser en dag framåt. Det är viktigt att anpassa nätverket till det givna problemet och dess komplexitet. Därför ska antalet neuroner i nätverket väljas därefter. Det är också nödvändigt att träna om nätverket ett flertal gånger för att erhålla ett med bra prestanda. Utöver fördelningen av träningsdatat så är det viktigare att samla tillräckligt med data för träningen av nätverket för att försäkra sig om att den anpassar och generaliserar sig till det aktuella problemet.

# Contents

<b>Contents</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem Statement . . . . .	2
1.2 Purpose . . . . .	2
<b>2 Background</b>	<b>3</b>
2.1 Artificial Neural Networks (ANNs) . . . . .	3
2.1.1 Networks . . . . .	3
2.1.2 Supervised Learning . . . . .	7
2.2 Datasets and Time Series Prediction Using ANNs . . . . .	8
2.2.1 Data Pre-processing . . . . .	8
2.2.2 Dividing the Data . . . . .	9
<b>3 Method</b>	<b>11</b>
3.1 Collecting Data . . . . .	11
3.2 Importing and Pre-processing the Data . . . . .	11
3.3 Training the Network . . . . .	12
3.4 Testing and Producing Results . . . . .	13
<b>4 Results</b>	<b>15</b>
4.1 Standard Values . . . . .	15
4.1.1 Changing the Number of Neurons . . . . .	17
4.1.2 Changing the Training Dataset Distribution . . . . .	20
<b>5 Discussion</b>	<b>27</b>
5.1 Changing the Number of Neurons . . . . .	27
5.2 Changing the Training Dataset Distribution . . . . .	28
5.3 Method Discussion . . . . .	29
5.4 Improvements and Future Work . . . . .	30
<b>6 Conclusion</b>	<b>31</b>
<b>7 References</b>	<b>33</b>





# Chapter 1

## Introduction

The stock market is one of the most popular investment places because of its high return investment field. Investors can gain a profit by investing their resources in shares and derivatives of various companies. The global stock market capitalization for 60 major stock market exchanges was estimated to 62.4 trillion USD in 2014 [1]. The stock market is a complicated environment, meaning that behavioural traits of stock prices are volatile and hard to predict. It is influenced by many economical, political and psychological factors that make it difficult to predict its effects. For instance, there could be changes in the demand and supply that affect the psychology of the investors. Because of the various internal and external factors that affect the prices of the stock market, the prices fluctuate in a non-linear way.

Research is continuously being made on how to develop and use methods to predict nonlinear patterns. With the evolution of computer hardware and development of efficient programming technologies, it is now possible to make more accurate predictions compared to the classical statistical methods, such as linear regression. The classical methods were unable to deal with the nonlinearity of the datasets, thus it became necessary to develop and utilize more advanced forecast procedures. Financial time series prediction has thus gained a considerable amount of importance in the last two decades [2].

Artificial neural networks (ANNs) are proposed as one of the most promising methods for time series prediction [2]. The ANNs are inspired by the activity in human brain cells and can learn the data patterns and generalize their behaviour to predict future data. In other words, ANNs are able to find accurate solutions in a complex environment, or even deal efficiently with situations where a lot of information is unavailable. Pattern recognition methods based on machine learning, such as ANNs in this case, predict future data by training the system with previously available data, called the training set [3]. Consequently, with these adapted characteristics, ANNs are well applied to the problems where extracting the relationships among data is difficult. At the same time, a large enough dataset is required for the network to learn the patterns. Since ANNs are good at generalizing relationships in specific data it means that after training they can recognize new patterns even

if they have not been present in the previous data.

In this paper, the investigated stock forecasting model is known as the feedforward neural network (FFNN). The FFNN is a simple and fundamental ANN and was chosen because many other models are based on it. The proposed model will be used to forecast Dow Jones Industrial Average (DJIA) index values and examine how two factors of the FFNN affect the predicted values. By using a FFNN the objective is to investigate how the number of neurons in the network affect the accuracy of the forecasted values, as well as the distribution of the training or learning dataset.

## 1.1 Problem Statement

The development of methods applied to financial forecasting is rapidly improving. Integrating ANNs and machine learning, in general, to areas such as finance is useful and needed for innovation and development. ANNs certainly seem to have the potential to distinguish unknown and hidden patterns in data which can be very effective for stock market prediction.

The aim is to study how the accuracy of the forecasted prices are affected for a two layer FFNN depending on:

- the number of neurons
- the distribution of the training dataset

## 1.2 Purpose

The main purpose of this paper is to carry out a project to study the use of ANNs on financial forecasting. This raises the question if it is feasible to use ANNs or similar models to predict the future trend of the stock market and the fluctuation of price. Considering how many factors are believed to contribute to the development of stock market index prices, it is appropriate to investigate different ways to predict the price development in order to gain more insight and knowledge about the subject. It also implies that finding hidden patterns in financial data will contribute to this same purpose.

Another important question in this thesis is how two significant factors; the number of neurons and the training data distribution, can affect the network's accuracy. These two factors were chosen because of their presence in all types of ANNs making them fundamental and important. This question is not only applicable on the financial forecasting problem but is also relevant to other problems using ANNs.

## Chapter 2

# Background

The background section is divided mainly into two parts. The first is about artificial neural networks (ANNs) and also more specifically about the multilayer feedforward neural network (FFNN) used in this project. The second part goes into the data and how it is used in the FFNN.

### 2.1 Artificial Neural Networks (ANNs)

It is important to note that the definition of ANNs varies depending on the source since it is an evolving technology and different notions are used. Another reason is that there is a large amount of different ANNs where the model, structure and uses of the ANN varies. The differences among them might be the topology, functions, accepted values, learning algorithms, etc.

#### 2.1.1 Networks

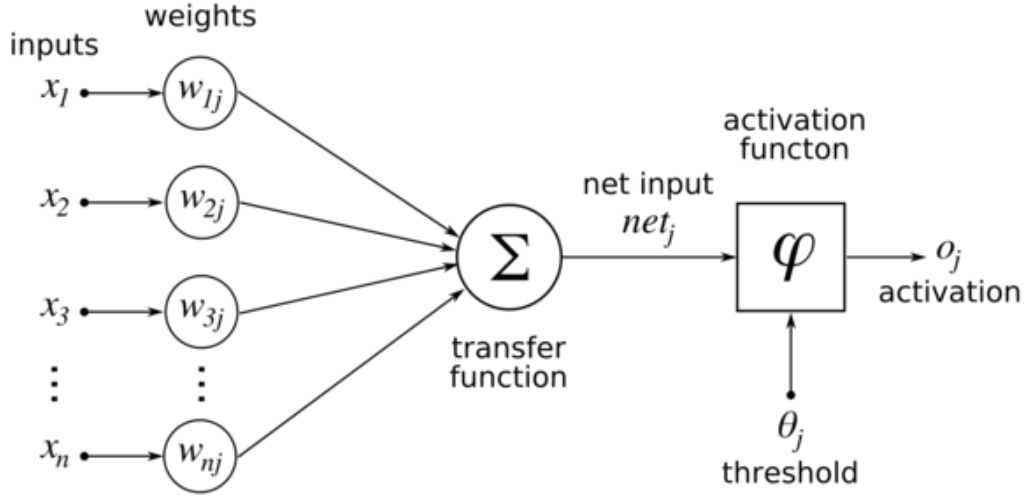
Networks are used in many different situations and can be designed for various purposes, for instance computer networks and communities. In general, a network consists of units called nodes and connections between these nodes. The nodes can be seen as computational units, designed to complete different tasks [4]. They take inputs and process it in order to produce an output. The connections are for the information flow, meaning the input and/or the output, and can be directed. If the connections are directed there is a constraint on them which only lets the information flow in one or the other direction. While a sole node can be different in complexity and function the idea of a network is to combine the power of several nodes to be able to complete more difficult tasks. This global behaviour is called *emergent* and means that the power of the network supersedes the ability of the nodes [4].

One type of network is an artificial neural network which consists of artificial neurons that can be connected in different ways in order to obtain desired complexity and functionality. This network is a computational model inspired by the

natural biological neurons. The complexity has been considerably abstracted from the biological neural network and is based on the idea that the natural neurons send and receive signals through synapses.

### Artificial Neurons

An artificial neuron is the basic building block of every ANN. Its design and characteristics are derived from observation of natural neurons in biological neural networks. An overview of the structure and functionality of an artificial neuron with its inputs, weights, transfer and activation functions, threshold and output can be seen in Figure 2.1 to get a general idea of an artificial neuron.



**Figure 2.1.** The basic functionality of a simple artificial neuron.

As seen in Figure 2.1, an artificial neuron can be fed with multiple numbers of inputs. Depending on the weights' value the total input's influence to the neuron will differ as well as the computation of the transfer function and output. Basically higher weight values increase the power and influence of the associated inputs [4]. This is because all the inputs are multiplied by its corresponding weight and the weights will thus affect the neurons' output. Finally the transfer function, that is the sum, sums the weighted inputs to produce the net input to the neuron, as in Equation 2.1.

$$net_j = \sum_{i=1}^n w_{ij}x_i + b \quad (2.1)$$

Where:

- $j$  denotes the actual neuron number
- $x_i$  is an input value where  $i$  goes from 1 to  $n$

- $w_{ij}$  is a weight value where  $i$  goes from 1 to  $n$
- $b = -threshold$

The value  $b$ , called the bias of the neuron, is equal to the negative threshold value of a neuron [5]. The threshold value decides what output the neuron gives depending on the activation functions value. For instance if the threshold is set to a real number, one could define:

$$output = \begin{cases} 0 & \text{if } \sum_{i=1}^n w_{ij}x_i \leq threshold \\ 1 & \text{if } \sum_{i=1}^n w_{ij}x_i > threshold \end{cases} \quad (2.2)$$

The threshold is moved to the other side of the comparison operator and then the transfer function becomes like Equation 2.1. Instead the transfer function's output is now either positive or negative depending on the threshold value.

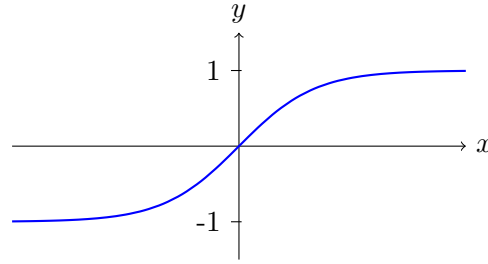
Further on, the final output from the neuron is computed as  $o_j = f(net_j)$  where  $f$  is an activation function. The activation function defines the properties of an artificial neuron and is chosen depending on the problem that the ANN needs to solve. The most commonly used activation functions are variations of the sigmoid function. The Tan-sigmoid function can output values between  $-1$  and  $1$ , other variations can however output values in a different range. The Tan-sigmoid function is defined as Equation 2.3.

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \quad (2.3)$$

As illustrated in Figure 2.2, large and positive input values gives output that is approximately 1 and with large and negative inputs the output will be closer to  $-1$ . Only when the input is of modest size will the function fluctuate between 1 and  $-1$  [5]. A useful characteristic of the Tan-sigmoid function is that it is a smooth and continuous function (as can be seen in Figure 2.2) which means that small changes to the weights and bias of a neuron will not affect the output of the neuron in a considerable way and make it hard for the network to slowly generalize the hidden patterns. The function's derivative is also easily found, which is useful when used with the backpropagation algorithm that is explained in Section 2.1.2.

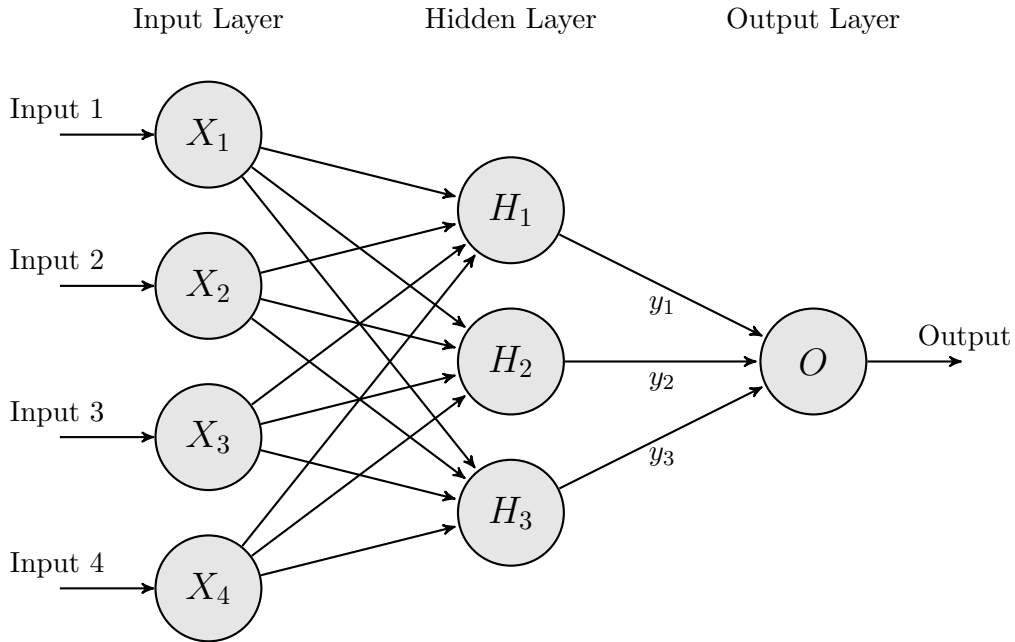
### Multilayer Feedforward Neural Networks (FFNNs)

An artificial neural network with feedforward topology is called a feedforward neural network (FFNN) and is the most popular and widely-used ANN topology. The only condition for such a network is that the information must flow in only one direction, from input to output, with no connections such as cycles or loops. A FFNN consists of layers of interconnected neurons where each neuron in the input layer is connected to all neurons in the next layer, and each neuron in this layer is connected to all neurons in the next layer and so on until finally they are connected to the final



**Figure 2.2.** Curve of the Tan-Sigmoid function.

output layer. There are no limitations on the number of layers, increasing the number of layers does however increase the complexity of the network. All layers between the input and output layers are called the hidden layers, simply meaning that they are neither input nor output neurons [5]. The figure below illustrates a multilayer FFNN.



**Figure 2.3.** Depicting the basic architecture of a two layer FFNN.

The input layer is made up of neurons  $X_1$  to  $X_4$ , the inputs to the network. At the neurons in the hidden layer, denoted  $H_1$  to  $H_3$  in Figure 2.3, each neuron processes its input using the principles of Equation 2.1 and 2.3 to calculate this layer's outputs, as seen in Equation 2.4.

$$y_j = f\left(\sum_{i=1}^4 w_{ij}x_i + b_j\right) \quad (2.4)$$

where  $f$  is a continuous activation function,  $b_j$  correspond to the bias at  $H_j$  and  $w_{ij}$  denotes the weight assigned to input  $x_i$  at neuron  $H_j$ . This layer's outputs,  $y_j$ , are then fed forward to the output neuron  $O$ , where the network output  $o$  is computed as in Equation 2.5.

$$o = g\left(\sum_{j=1}^3 w_j y_j + b_O\right) \quad (2.5)$$

Here  $w_j$  denotes the weight in  $O$  corresponding to the hidden neuron output  $y_j$ ,  $b_O$  denotes the bias at the output neuron, and  $g$  denotes a linear activation function.

The popularity of FFNNs derives from the fact that they have been successfully applied for the purpose of function approximation. Meaning that they are able to approximate any continuous function arbitrarily well given the right number of neurons in the hidden layer. For this reason FFNNs are applicable in situations with unknown nonlinear relationships, making it suitable for the stock market [6].

### 2.1.2 Supervised Learning

The learning or teaching process for ANNs is essential for its functionality. This is carried out by one out of two types of training algorithms based on different concepts. The type that is used for the described models is called supervised learning. Supervised learning means that given a vector of inputs to the network, it will produce the respective outputs and compare it to the correct answers at hand [7]. The answers are available to the control process and can therefore be compared to the computed outputs and this is utilised in the learning process. Furthermore supervised learning is divided into methods using two other types of learning, the one used in this paper is error correction [7]. The magnitude of the error together with the input determines the correction to the weights. In the end the goal is to minimize or fully eliminate this error. The backpropagation algorithm used with the FFNN uses supervised learning with error correction and is explained in further detail in the next section.

### Backpropagation Algorithm

In order to use an ANN it has to train and learn how to handle the input data. This learning process corresponds to setting the values of weights and biases so that it minimizes the error between the outputs and expected values. There are different training algorithms that can be used for the ANN depending on, for instance, the topology and activation function. The backpropagation algorithm is mainly used in FFNNs together with a sigmoid function because of its easily computed and continuous derivative [4]. Also, since data is only fed forward in an FFNN, errors

are propagated backwards while adjusting the weights and biases according to the error. Finally, for practical reasons, FFNNs implementing the backpropagation algorithm do not have many layers because of the time for training the network grows exponentially [4].

The backpropagation algorithm uses the concept of supervised learning with error correction to train the network. This means that given a sample of inputs and a target (desired output) to the network, the error is calculated between the target and actual output. The idea of the backpropagation algorithm is to obtain the target as output when the respective inputs are given. Since the error is the difference between the target and output the error depends on the weights and biases in the neurons. Therefore the weights and biases are adjusted in order to minimize the error. The change to the weights and biases is decided based on their impact on the error, how this is achieved is out of scope for this report. More specifically, the error is calculated using the mean square error function as seen in Equation 2.6.

$$MSE = (t_i - o_i)^2 \quad (2.6)$$

The mean square error between the network output  $o_i$  and the target  $t_i$  will always be positive. The error will also be greater if the difference is big, and lesser if the difference is small. The total error of the network will simply be the sum of the errors of all the neurons in the output layer as seen in Equation 2.7 [4].

$$MSE_N = \sum_{i=1} (t_i - o_i)^2 \quad (2.7)$$

## 2.2 Datasets and Time Series Prediction Using ANNs

The dataset used in this project is a time series. A time series is univariate or multivariate quantitative data collected over some period of time [8]. There are different techniques and approaches for time series forecasting. The quantitative technique refers to forecasting based on historically and statistically analysed data [8]. This applies to stock data and how the values of the stocks change over time in a non-linear way. The historical data can then be used to investigate the variable of interest and forecast its future values, which in this case are the stock prices.

The data used in this project is described in further detail in Section 3.1.

### 2.2.1 Data Pre-processing

A neural network is solely based on the sample data, therefore the network can only be as accurate as the data that is used to train the network [9]. Before data is used by a network, it can go through several transformations in order to adjust the inputs to a given model. The quality of the input data is directly connected to the success of the network, because different methods can handle different samples of data. Therefore, it is recommended to take advantage of certain data features in order to find out which pre-processing transformation works best [2, 10].



### Normalization

The chaotic nature of the stock prices is more or less dependent on the market volatility and a company's performance. This can increase the difficulty for the learning process in the neural network, as it gets disturbed by the large fluctuations in the price. Because of the activation function used by the neural network is bounded, this can cause inconsistencies in the training and prediction phases. In order to avoid the inconsistencies, all the input data is normalized between same bounds as the activation function's [2]. Since the chosen method to normalize is decided by the activation function's range, one method to normalize the prices between  $-1$  and  $1$  is the min-max normalization function in Equation 2.8.

$$y_i = \frac{(a_{max} - a_{min})(x_i - \min(\bar{x}))}{\max(\bar{x}) - \min(\bar{x})} + \min(\bar{x}) \quad (2.8)$$

Here  $x_i$  denotes the  $i$ :th input of the vector, the minimum and maximum values of the vector are denoted  $\min(\bar{x})$  and  $\max(\bar{x})$  respectively. The  $a_{max}$  and  $a_{min}$  correspond to the upper and lower limit for the activation function's output range,  $-1$  and  $1$ .

#### 2.2.2 Dividing the Data

For training a multilayer FFNN one has to divide the available dataset into three different subsets [11]. These three subsets are called training, validation and testing sets.

The training set is used for training the network, which means updating the weights and biases of the neurons depending on the deviation between the target value and actual output, as explained earlier in Section 2.1.2.

The validation set is monitored during the training process and the error on this set is decreased initially, just as the training set error. When the process starts to overfit the data (see the next section) this validation set error typically starts to increase. Because of this the weights and biases of the neurons are saved when the validation set error is at its minimum [11]. In other words, the validation set is used as a way to know when to stop the training process.

The testing set error is not used during training but can be used to compare different models. The error can also be compared to the validation set error and if the minimum error is achieved at a different point than for the validation set it can indicate poor data distribution [11].

### Overfitting the Data

One problem that can occur while training an ANN is overfitting. This means that while the network gets small errors during training and testing it does not manage to generalize to the dataset's patterns. Thus, when new inputs are given to the network, it can not predict them accurately and gets significant deviations from the expected values [11].

There are a few methods that can be used to prevent overfitting. One way is making sure that the network is just large enough for the given problem [11]. Since a bigger network will be able to generalize more powerful and complex problems and smaller networks will not have enough power to overfit the data, it is important to find an appropriate network size suited for the given problem.

Other methods are available but they are mostly relevant if the collected training set is very small and limited. If possible, the best approach is to gather a big enough training set. This will make sure there is little to no risk, or at least a very small risk, of overfitting the data [11].

## Chapter 3

# Method

The method section is presented by first describing the collection of data, needed to perform the tests. The way the data is prepared for the network is then explained. Finally the training of the network and different tests that were performed are described.

The study was carried out with MATLAB because of similar studies using the same tools. MATLAB's Neural Network Toolbox was used to create and train the two layer FFNN.

### 3.1 Collecting Data

Enough data is required to investigate the different distributions of the training dataset. Therefore, the collected data for the study consisted of six years (2009-2014 inclusive) of the *Dow Jones Industrial Average* (DJIA) index. The first five years of DJIA were used to train, validate and test the neural network (Figure 3.1) while the last year of DJIA was used to make one day predictions. The historical data was downloaded from *stock historical data* [12]. The data from the site is represented in one of most common ways, the *Metastock* data format. Meaning that the data consists of the date, volume and the stock index's open-, high-, low- and close-price of the day. The file is in *csv* format and represents all values separated by commas.

### 3.2 Importing and Pre-processing the Data

The collected historical data was imported as matrices where the rows and columns can be chosen from the data.

Four matrices were required to perform the tests, two input matrices and two target matrices. The first pair of inputs and targets were used for training the network and the other pair was used for testing and predicting. In this work the primary focus was on the prices, therefore the indirect variable *volume* was dismissed. Hence, the inputs consisted of the stock index's open-, high-, low- and close-price while the next day's closing price was assigned as the corresponding tar-



**Figure 3.1.** Historical time series used as training data (DJIA, 2009-2013). The variables are difficult to distinguish because they follow the same patterns.

get value. This means that the FFNN was trained to predict one day forward. In other words, the next day's closing index price is predicted given the open-, high-, low- and close-price from the day before. Thus the imported inputs consisted of the whole historical data time series (2009-2013), except for the last day's prices, while the targets consisted of all the closing prices except for the first day's price. This can be observed in the example seen in Figure 3.2. This makes the number of input samples match the target samples evenly. The same thing was done for the other two matrices that consist of the values from 2014.

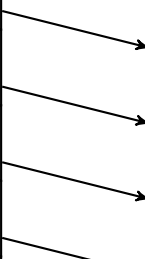
Before the data could be used for training the network, the matrices were normalized with the *mapminmax* function in MATLAB. The function corresponds to Equation 2.8 in Section 2.2.1. Hence the *mapminmax* function was used to match the bounds of the Tan-sigmoid activation function.

### 3.3 Training the Network

After preparing the necessary data the Neural Network Toolbox was used to construct and train a two layer FFNN, as described earlier in Section 2.1.

The prepared stock values for 2009 to 2013 was set as inputs and targets for the training. Since each sample had four inputs and one target, they also represent the respective number of neurons in the input- and output-layer.

The training data distribution was changed for each test when the network had to be reconfigured. This is for the purpose of examining one of the factors in the

Open Price	High Price	Low Price	Close Price		Close Price
10	15	8	14		14
14	19	13	16		16
16	16	5	6		6
6	8	1	3		3
3	9	2	7		7

**Figure 3.2.** Example of how the stock data corresponds to inputs and targets. The values filled with darker gray are left out of the imported data. Inputs to the left and targets to the right.

problem statement. The way the samples are divided into the different parts is set to random by default and is not changed in this work.

To configure the network size the number of neurons in the hidden layer are changed between the tests. This investigates the second factor in the problem statement.

The *Levenberg-Marquardt* backpropagation algorithm was used to train the network. The reason behind the chosen training algorithm was because of the fact that it suits most problems while the other alternatives in the toolbox are more specifically applied to smaller or larger problems.

Finally, some additional code was added to the network configuration and training script. Code for running the trained network with the input data from year 2014. The output for year 2014 (predicted prices) was plotted together with the targets. In addition, the errors, more specifically the absolute value of the difference between the output and target is also plotted. One example can be seen in Figure 4.3.

### 3.4 Testing and Producing Results

The network configurations varied by changing the training data distribution or the number of hidden layer neurons. For each configuration the network was retrained ten times. After each training the accuracies for the predicted values were calculated. The accuracy was calculated with the quotient between the output value and the target value (actual price). The lesser number of the two values was the numerator while the greater was the denominator in order to avoid values above one. For all the predicted values of 2014 the maximum, minimum and mean accuracy was recorded. This was done for all ten retrained networks and finally the mean values

were computed in order to get an average accuracy for the respective configuration. These average accuracies were used in order to find a new network, by retraining, deviating with less than 0.02 units, meaning less than 2%. All the diagrams for the respective network configuration were taken from this retrained network to get an estimated average prediction.

When investigating a specific factor in the network the other factors have to be constant through all the changes to guarantee they do not interfere with the results. Due to this the standard toolbox values are used for the number of neurons and the training data distribution when investigating the one or the other. While changing the number of hidden layer neurons the training data distribution is set to 70% for training and 15% for validation and testing. While changing the training data distribution the number of hidden layer neurons are set to 10. These were also tested together in order to confirm their suitability to the tests.

The first conducted tests were based on changing the number of neurons in the network, ranging from a low amount to a bigger amount and finally an even greater amount. Note that the toolbox only supports 1 to 10,000 neurons in the hidden layer. The training time also increases with the number of neurons, thus the highest number tested was 500 neurons.

The other tests were based on the training data distribution. The tested distributions were equal distribution, high training, validation and testing data and finally low training, validation and testing data respectively. The exact configuration values can be seen in Chapter 4 together with the results.

## Chapter 4

# Results

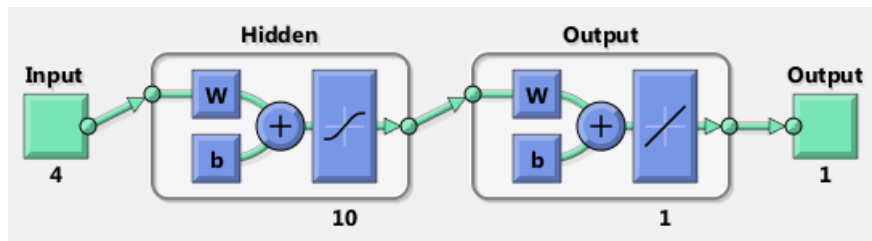
There are two types of tests executed in this project, as described in Chapter 3. The first type's purpose is to examine how the number of neurons in the FFNN impact the predicted prices. The second type's purpose is to study how the distribution of the training dataset impacts the same predictions.

The results are mainly represented with tables featuring the maximum, minimum and mean accuracy of the predicted next day closing price, during a year. Values for all ten retrained networks are presented for each configuration and table. These are denoted by *Session* and numbered from 1-10, as seen in Table 4.1. The average accuracy is rounded down to the same number of decimals as the other accuracies.

The diagrams are plotted with the predicted (output) and actual (target) values, as in Figure 4.2.

### 4.1 Standard Values

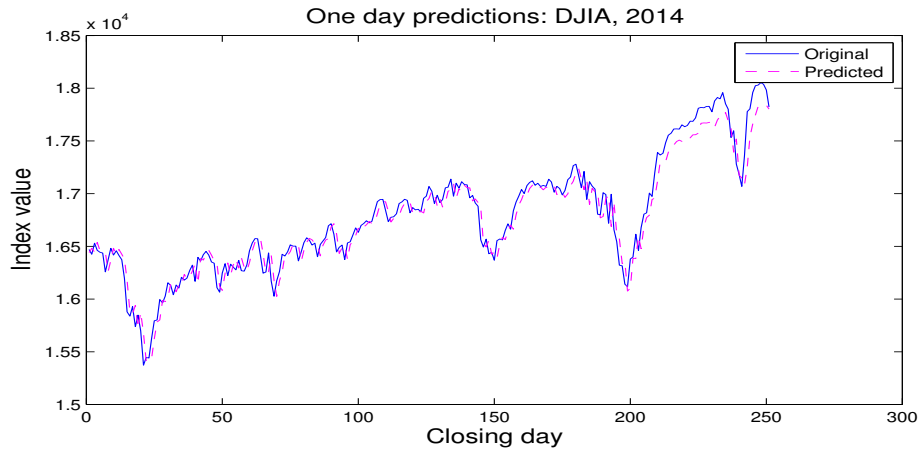
The default MATLAB values for the number of neurons and the training dataset distribution is used. These are ten neurons in the hidden layer (Figure 4.1) and the distribution is 70% for training, 15% for validation and 15% for testing. The following results use this standard configuration.



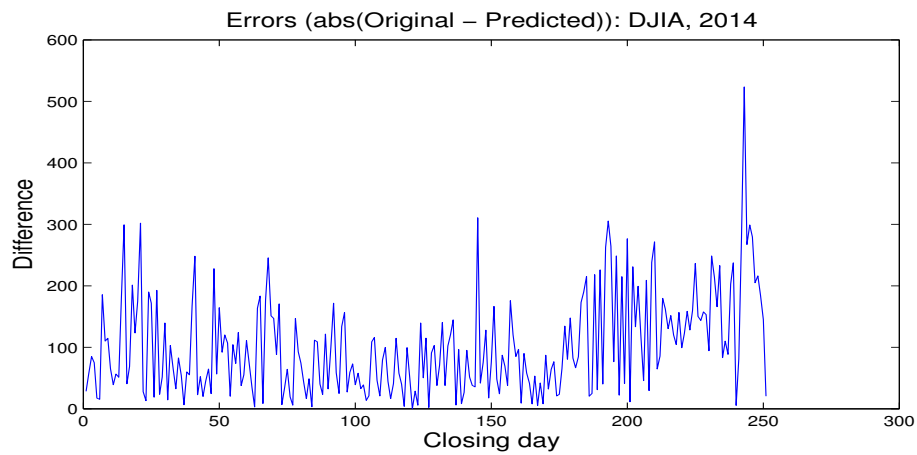
**Figure 4.1.** Two layer FFNN architecture with 10 hidden layer neurons.

Session	Minimum	Maximum	Mean
1	0.9720	1.0000	0.9945
2	0.9700	1.0000	0.9935
3	0.9753	1.0000	0.9950
4	0.9742	0.9999	0.9948
5	0.9729	1.0000	0.9946
6	0.9699	1.0000	0.9934
7	0.9695	1.0000	0.9927
8	0.9124	1.0000	0.9811
9	0.9760	1.0000	0.9946
10	0.9708	1.0000	0.9935
<b>Average</b>	0.9663	0.9999	0.9927

**Table 4.1.** Accuracy results based on 10 retrained networks.



**Figure 4.2.** One day DJIA stocks prediction and actual data from 2014.



**Figure 4.3.** Price difference between actual and predicted stock prices (USD).



### 4.1.1 Changing the Number of Neurons

The training dataset distribution for these results are set to the standard values (70% training, 15% validation and 15% testing), as in Section 4.1.

#### Accuracy Results Based on Ten Retrained Networks

Session	Minimum	Maximum	Mean
1	0.9529	0.9996	0.9873
2	0.9679	0.9999	0.9933
3	0.9746	1.0000	0.9950
4	0.9739	1.0000	0.9947
5	0.9747	1.0000	0.9950
6	0.9598	0.9998	0.9890
7	0.9712	1.0000	0.9945
8	0.9750	1.0000	0.9949
9	0.9735	1.0000	0.9949
10	0.9720	1.0000	0.9946
<b>Average</b>	0.9695	0.9999	0.9933

**Table 4.2.** Accuracy using a network with 2 hidden layer neurons.

Session	Minimum	Maximum	Mean
1	0.8298	0.9999	0.9670
2	0.9580	1.0000	0.9893
3	0.9667	1.0000	0.9926
4	0.9149	0.9999	0.9862
5	0.9637	0.9999	0.9919
6	0.9696	1.0000	0.9944
7	0.9487	1.0000	0.9913
8	0.9289	1.0000	0.9841
9	0.9586	0.9999	0.9891
10	0.9574	1.0000	0.9914
<b>Average</b>	0.9396	0.9999	0.9877

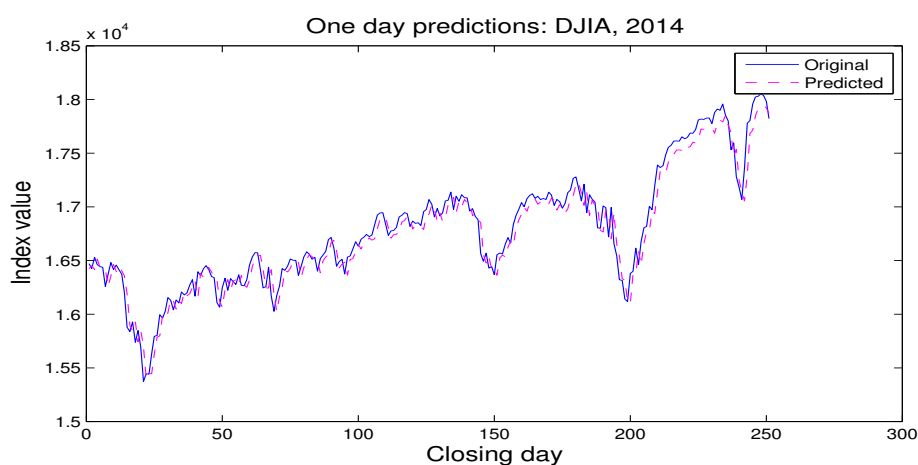
**Table 4.3.** Accuracy using a network with 50 hidden layer neurons.

Session	Minimum	Maximum	Mean
1	0.7798	0.9999	0.9845
2	0.5786	1.0000	0.9217
3	0.6542	1.0000	0.9486
4	-0.0942	0.9999	0.7821
5	-0.6160	0.9999	0.7807
6	0.1195	1.0000	0.8670
7	0.7099	0.9999	0.9614
8	0.4078	0.9998	0.9379
9	0.1512	0.9999	0.8407
10	0.8852	0.9999	0.9641
<b>Average</b>	0.3576	0.9999	0.8988

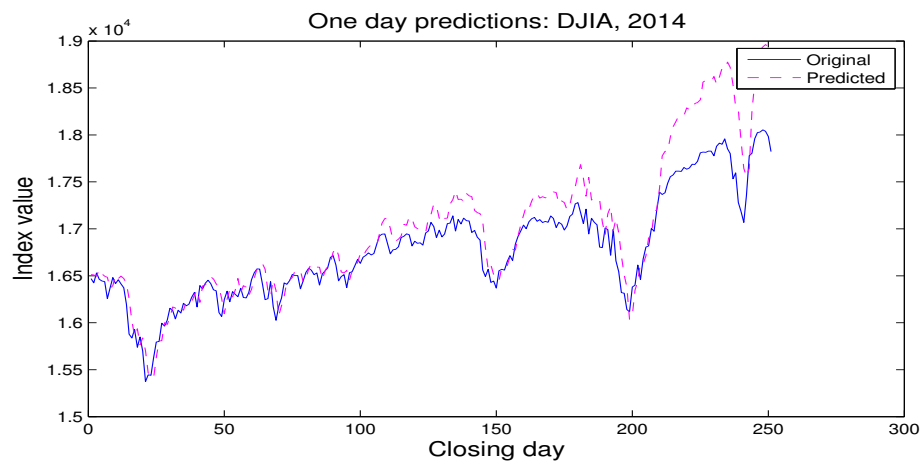
**Table 4.4.** Accuracy using a network with 500 hidden layer neurons.

### Prediction Diagrams

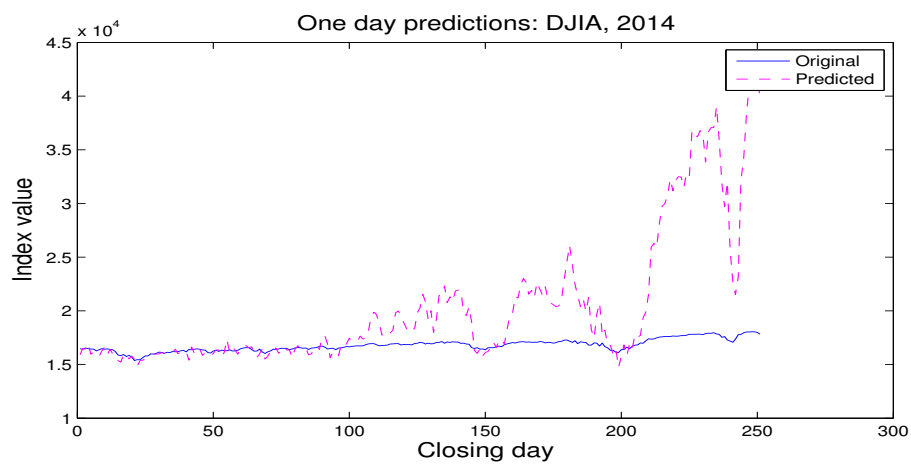
One day DJIA stocks prediction and actual data from 2014. The Diagrams are based on the average values from the previous tables (4.2 - 4.4), as described in Section 3.4. Meaning that they represent the average network predictions for each configuration.



**Figure 4.4.** One day DJIA stocks prediction using 2 neurons.



**Figure 4.5.** One day DJIA stocks prediction using 50 neurons.



**Figure 4.6.** One day DJIA stocks prediction using 500 neurons.

### 4.1.2 Changing the Training Dataset Distribution

The number of neurons for these results are set to the standard value (10 neurons), as in Section 4.1.

#### Accuracy Results Based on Ten Retrained Networks

Note the importance of taking the amount of data into account when analyzing the results, especially the low percentages. For the 5 years of collected training data, a 20% subset is equivalent to one year of historical stock market data and 5% is about one quarter of a year.

Session	Minimum	Maximum	Mean
1	0.9718	1.0000	0.9946
2	0.9751	0.9999	0.9934
3	0.9616	0.9998	0.9916
4	0.9644	1.0000	0.9930
5	0.9342	0.9997	0.9830
6	0.9741	1.0000	0.9934
7	0.9600	1.0000	0.9893
8	0.9562	0.9999	0.9877
9	0.9603	1.0000	0.9891
10	0.9727	1.0000	0.9943
<b>Average</b>	0.9630	0.9999	0.9909

**Table 4.5.** Equal distribution (34% training, 33% validation and 33% testing).

Session	Minimum	Maximum	Mean
1	0.9430	1.0000	0.9885
2	0.9738	1.0000	0.9950
3	0.9523	1.0000	0.9876
4	0.9723	1.0000	0.9927
5	0.9653	0.9999	0.9912
6	0.9647	0.9999	0.9913
7	0.9760	1.0000	0.9944
8	0.9699	1.0000	0.9937
9	0.9756	1.0000	0.9943
10	0.9714	1.0000	0.9939
<b>Average</b>	0.9664	0.9999	0.9922

**Table 4.6.** High training (90% training, 5% validation and 5% testing).

Session	Minimum	Maximum	Mean
1	0.9527	1.0000	0.9825
2	0.9187	1.0000	0.9712
3	0.8984	0.9984	0.9647
4	0.9609	1.0000	0.9908
5	0.9570	0.9999	0.9877
6	0.7702	0.9999	0.9420
7	0.9368	1.0000	0.9830
8	0.9021	0.9999	0.9869
9	0.9492	0.9970	0.9842
10	0.8540	0.9999	0.9416
<b>Average</b>	0.9100	0.9995	0.9734

**Table 4.7.** High validation (5% training, 90% validation and 5% testing).

Session	Minimum	Maximum	Mean
1	0.9625	1.0000	0.9941
2	0.8942	0.9970	0.9702
3	0.9588	0.9980	0.9890
4	0.9310	0.9998	0.9865
5	0.9621	0.9999	0.9915
6	0.9526	1.0000	0.9900
7	0.9219	0.9997	0.9708
8	0.9187	1.0000	0.9802
9	0.8751	0.9993	0.9546
10	0.9566	1.0000	0.9912
<b>Average</b>	0.9333	0.9993	0.9818

**Table 4.8.** High testing (5% training, 5% validation and 90% testing).

Session	Minimum	Maximum	Mean
1	0.9405	0.9999	0.9803
2	0.9472	0.9999	0.9826
3	0.8826	0.9991	0.9623
4	0.8563	0.9989	0.9365
5	0.9552	1.0000	0.9893
6	0.9575	1.0000	0.9923
7	0.9336	0.9999	0.9736
8	0.9407	0.9999	0.9847
9	0.9153	0.9998	0.9655
10	0.8941	0.9999	0.9571
<b>Average</b>	0.9223	0.9997	0.9724

**Table 4.9.** Low training (4% training, 48% validation and 48% testing).

Session	Minimum	Maximum	Mean
1	0.9743	0.9999	0.9935
2	0.9752	0.9999	0.9937
3	0.9639	1.0000	0.9922
4	0.9473	1.0000	0.9858
5	0.9687	1.0000	0.9942
6	0.9323	1.0000	0.9863
7	0.9545	1.0000	0.9862
8	0.9740	1.0000	0.9949
9	0.9616	1.0000	0.9903
10	0.9644	1.0000	0.9900
<b>Average</b>	0.9616	0.9999	0.9907

**Table 4.10.** Low validation (48% training, 4% validation and 48% testing).

Session	Minimum	Maximum	Mean
1	0.9662	0.9999	0.9921
2	0.9644	0.9998	0.9924
3	0.9727	0.9999	0.9942
4	0.9658	1.0000	0.9923
5	0.9327	0.9999	0.9844
6	0.9728	1.0000	0.9948
7	0.9767	1.0000	0.9948
8	0.9669	1.0000	0.9917
9	0.9773	1.0000	0.9944
10	0.9747	0.9999	0.9945
<b>Average</b>	0.9670	0.9999	0.9925

**Table 4.11.** Low testing (48% training, 48% validation and 4% testing).

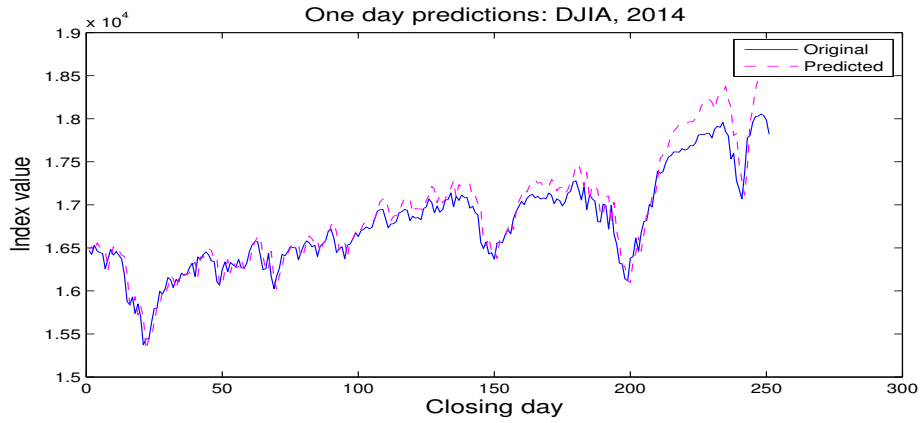
A summary of the average mean accuracies for each configuration (training dataset distribution) can be seen in Table 4.12.

Training %	Validation %	Testing %	Accuracy
70	15	15	0.9927
34	33	33	0.9909
90	5	5	0.9922
5	90	5	0.9734
5	5	90	0.9818
4	48	48	0.9724
48	4	48	0.9907
48	48	4	0.9925

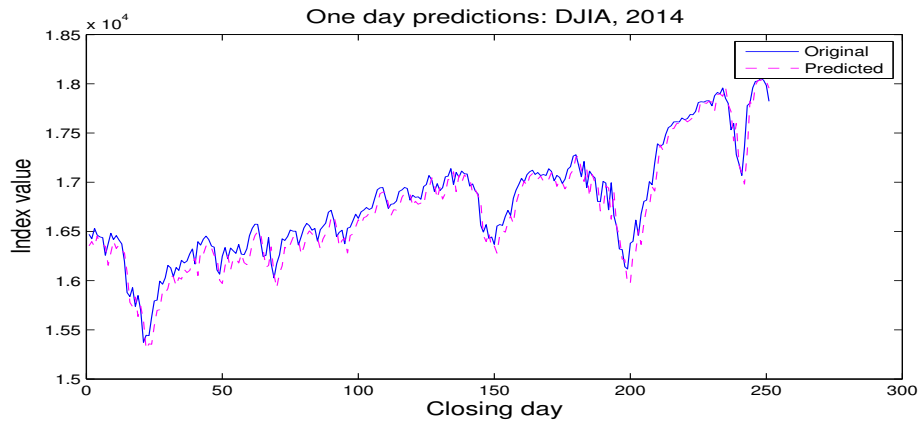
**Table 4.12.** Summary of the average accuracy results.

### Prediction Diagrams

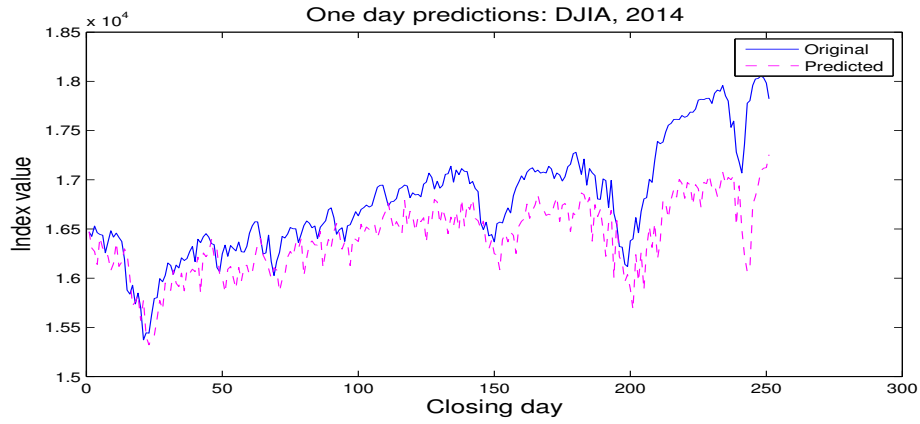
One day DJIA stocks prediction and actual data from 2014. The Diagrams are based on the average values from the previous tables (4.5 - 4.11), as described in Section 3.4. Meaning that they represent the average network predictions for each configuration.



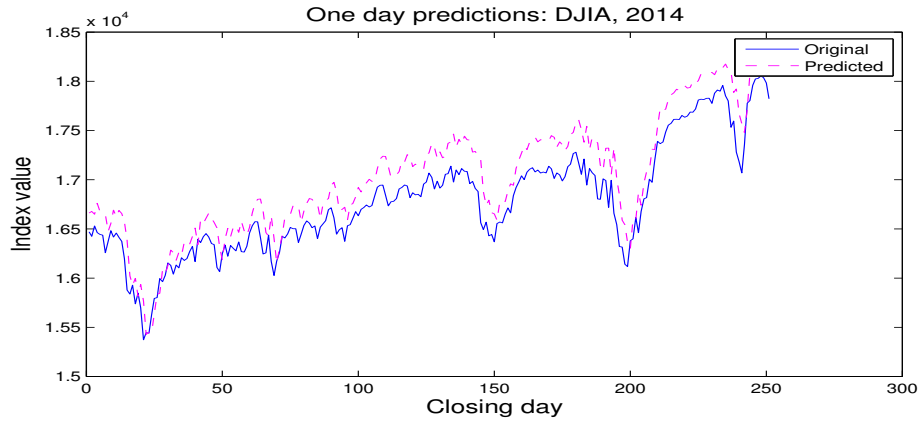
**Figure 4.7.** Equal distribution (34% training, 33% validation and 33% testing).



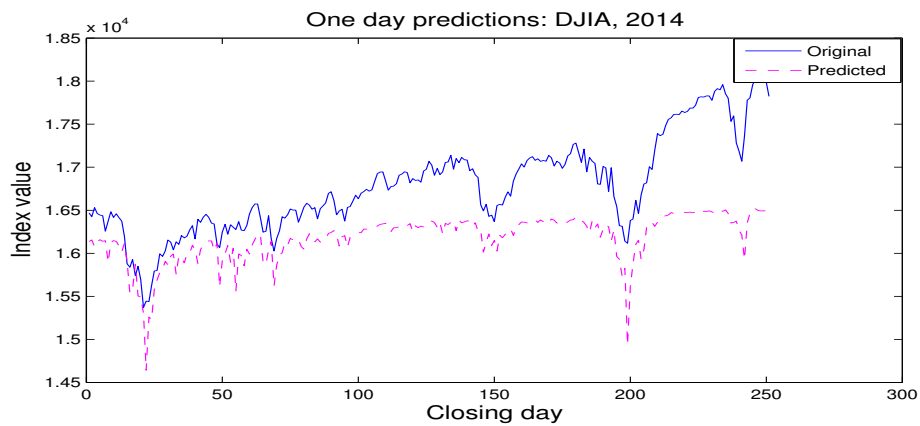
**Figure 4.8.** High training (90% training, 5% validation and 5% testing).



**Figure 4.9.** High validation (5% training, 90% validation and 5% testing).

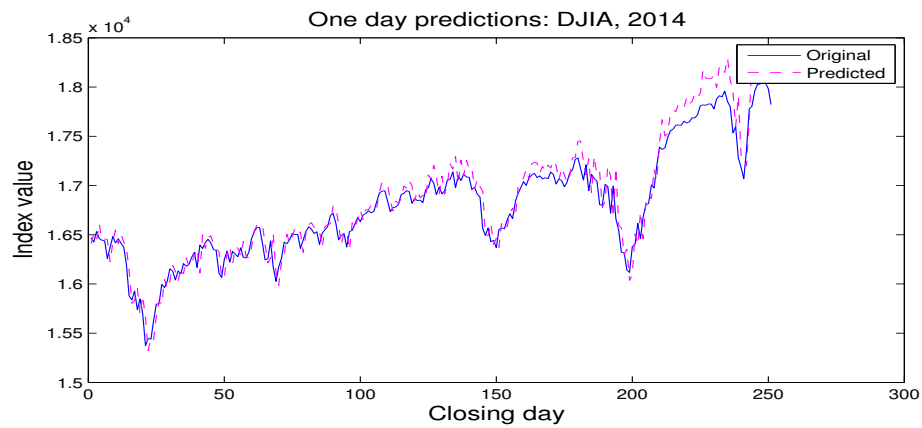


**Figure 4.10.** High testing (5% training, 5% validation and 90% testing).

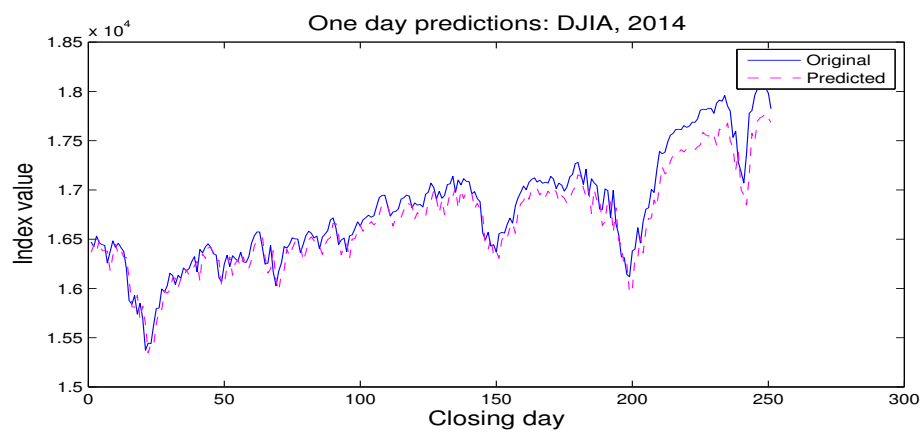


**Figure 4.11.** Low training (4% training, 48% validation and 48% testing).





**Figure 4.12.** Low validation (48% training, 4% validation and 48% testing).



**Figure 4.13.** Low testing (48% training, 48% validation and 4% testing).



## Chapter 5

# Discussion

The results in Section 4.1 uses the standard configuration and Figure 4.2 shows a fairly aligned and promising prediction curve. The predictions have a difference of at most 550 USD compared to the actual stock prices, as seen in Figure 4.3. However, this only applies to one peak and it is observed that most errors are below the 300 USD difference mark. Considering the stock index prices for the DJIA in 2014 are high values, mainly between 15,500 and 18,000 USD, this is not a huge difference, but of course it is still notable. Since the actual prices usually do not fluctuate more than 100 USD in a single day, except for some cases, this error is about three times the size of the actual fluctuation. It still manages to mimic the patterns fairly well and even though the predictions are a bit off one can see that the fluctuation of the original curve is similar to the predicted one, at least with respect to the prices rising and decreasing. For example, Table 4.1 shows a mean accuracy that is, for 9 out of 10 retrained networks, over 99%, meaning the predicted prices are in general off by less than 1%. The practical application of this on the stock market does still not seem feasible, especially considering the predictions being only one day forward. This will be discussed further in Section 5.3.

### 5.1 Changing the Number of Neurons

Comparing how the number of neurons affect the predictions brings us to Figure 4.4, 4.5 and 4.6. One can clearly see from these results how an increasing number of neurons give deteriorating predictions, especially during the second half of the 2014 data where the prices fluctuate more. This strengthens the fact that the complexity of a network is enhanced by the number of nodes. In this case, the problem applied to the network is not complex enough for it to require many nodes. As stated in Chapter 3, only four inputs and one target are taken into account, which does not apply many factors and relations for the network to generalize and adapt to. Since these tests are only applied to one problem it is not possible to fully confirm that the number of neurons in the network has to adapt to the problem's complexity, but they are definitely an indication of the fact.

The accuracy results for changing the number of neurons can be seen in Table 4.2, 4.3 and 4.4. The difference between the results from using ten neurons (Table 4.1) and two neurons (Table 4.2) is not significant. They are practically the same except for one retrained network (Session 8) with ten neurons that brought down the average mean accuracy. This just shows how important it is to retrain the network until one good enough is acquired. Except for this, the two network configurations both had above 99% average mean accuracy. Using 50 neurons decreases the average mean accuracy to below 99% and using 500 neurons is definitely a downgrade in this case. In Table 4.4, featuring the 500 neuron configuration, negative prices were occasionally predicted which is why the minimum accuracy is negative for Session 4 and 5. Considering the stock prices will never decrease to negative values this is an impractical prediction. This decreased the mean values for the 500 neuron configuration significantly but even the better values are below 97% except for the Session 1. However, the minimum accuracy for this retrained network (78%) is far below the minimum results using for example two neurons (95%). This makes it unreliable since the accuracy for every prediction varies within a 20% range compared to a 5% range. Using too many neurons, as seen in the results, is an example of overfitting the data and the network struggles to generalize the patterns into future predictions.

## 5.2 Changing the Training Dataset Distribution

Comparing how the distribution of the training data affect the predictions brings us to Section 4.1.2. Simply comparing all the diagrams (Figure 4.7 - 4.13) the less accurate predictions are the configurations with low training data, meaning 4-5% of the data, which is less than or equal to a quarter of a year. When comparing the accuracy results (Table 4.5 - 4.11) it can be seen that the average mean accuracy is above 99% in every other configuration when the training data is 34%, 48% or 90%. It is difficult to find a relation in how the actual distribution of training data affects the predictions, but it is clear that there has to be enough data for the training subset or the network will not be able to adapt and generalize to the problem as accurate as possible. This can be seen more clearly in the summary of the average mean accuracies in Table 4.12.

The testing data is not used for training the network and thus has no effect on the networks performance, which can be seen in the results as well. For instance in Table 4.5 and 4.11, which has equal distribution and low testing percentage respectively, both have average mean accuracy above 99% and average minimum accuracy above 96%. This is expected though since the testing data is only used after the network is trained to check its performance.

Since the testing data does not affect the training, it is easier to find relations between the training and validation data. Lesser accuracy be seen when the validation data is bigger than the training data (Table 4.7 and 4.9). The average mean accuracy is less than 98% and the average minimum accuracy is about 92% in both

cases. When it is the opposite (Table 4.6 and 4.10), better results are acquired with the average mean accuracy over 99% and the average minimum accuracy over 96%. This is not necessarily because of the distribution relationship between training and validation. Since the tests where they have equal distribution (Table 4.5 and 4.8), both lesser and higher accuracy results can be seen. This alleged relation between the training and validation data is more likely because of the lack of training data as stated earlier. Especially since the training data is only 5% or less during the two tests when the validation subset is bigger than the training subset, and it was already stated that this is too little to acquire a reliably trained network. No conclusion can thus be drawn from the relationship between the validation and training subsets in the distribution.

### 5.3 Method Discussion

Since there are many different models and structures of ANNs there are many other possibilities that can be used and tested for stock forecasting. Trying other models might give more insight into the problem at hand and manage to find more accurate and suitable systems for financial stock forecasting. The data used for training and testing could also benefit from being varied and could therefore generate other results because of more complex patterns or more volatile price development.

Even though the two layer FFNN generated highly accurate results it can be argued why only four out of five collected data variables were chosen as inputs for the network. The fifth variable referred to being the volume of the stock. Including more inputs or variables to the training data could contribute to the complexity of the generalization and impact the networks performance. The reason for choosing four inputs was because of previous similar work and in the end similar results were obtained in terms of accuracy [13]. The previous study managed to produce the best case accuracy at 96% compared to our best accuracy at 99%. This indicates better results but considering the study's method lacking details no conclusions can be drawn regarding the differences in the final results. The method to calculate the accuracy might be different but several tests were made for each configuration in this project in order to ensure empirical results.

Another point to take into consideration is that the average accuracy values for each network configuration is calculated with the mean. One problem is that if one retrained network's accuracy deviates significantly from the other results, it might affect the average value significantly (note that 1% (0.01) of  $1.65 * 10^4$  is 165 USD). This can affect the results considerably, especially when the mean value is based on solely ten retrained networks. One way to avoid this pitfall is to drastically increase the number of tests and retrained networks to generate more results and obtain a more accurate mean value. Alternatively, setting the average to the median would be a possible, if not better, approach to reduce the impact of the deviating results.

Furthermore, one could raise objections against the method to only predict one time sample ahead. This is not practically useful since both short-term and long-

term investments mostly span over periods longer than a day. This is not the case when all the information from the day before is needed to predict the price the next day. The method has nonetheless shown potential given the few input variables on the specific problem. The dimensionality of the network inputs, meaning the number of variables, required to predict stock prices is low and in reality there would be more factors that affect the stock market. Increasing the number of variables would also mean that the network and desired prediction model would have an increased complexity.

## 5.4 Improvements and Future Work

In possible future work one could investigate the increase of dimensionality of the training dataset. Meaning to simply take more variables into account and compare the results with lower dimensionality datasets. The goal is then focused to confirm the concept of the curse of dimensionality where higher dimensionality means that a longer time span of historical data is needed to achieve reliable results [14].

One potential model apart from the FFNN is the recurrent neural network that allows cycles and loops in the network and would thus allow for more complex learning.

Predicting only one day ahead might arguably not be very useful, as stated earlier. One could however develop this system by feeding the network with its own output repeatedly to predict the stock prices an arbitrary number of days ahead. This would of course stack the errors from each prediction and finally most likely be too inaccurate to be useful, but it would still be a possible study. One important note to keep in mind is that the number of inputs to the ANN have to match the number of outputs for this study to be carried out.

## Chapter 6

# Conclusion

ANNs certainly are good at generalizing and adapting to problems. There is potential for one day stock market prediction with accuracy above 99%. Even if this is not practically applicable it could instead be used as an indicator to if the stock index is rising or decreasing.

Adapting the network to fit the problem at hand is very important. First of all, the number of neurons chosen for the ANN should be decided by the complexity of the problem and in most cases it is required to retrain the network with different configurations and find the one that performs best. The same thing can be said about the training dataset distribution. It is important to have enough data for training the network and allowing it to generalize itself to it. Besides the distribution of the available training data, it would be more important to make sure that enough training data is collected for the network to adapt to during training. It can not be concluded whether it is possible to have too much training data but too little definitely lowers the accuracy of the network.





## Chapter 7

## References

- [1] P. Mark J. (2014, Apr. 15). “Global Stock Rally: World Market Cap Reached Record High In March” [Online]. Available:  
<http://seekingalpha.com/author/mark-j-perry> [May 8, 2015].
- [2] B. Lóránt, “Financial Time Series Forecasting Using Artificial Neural Network,” M.S. thesis, Dept. CS., Babes-Bolyai Univ., 2004. [Online]. Available:  
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.8658&rep=rep1&type=pdf> [Mar. 31, 2015].
- [3] P.N. Mahdi et al. “Stock Market Value Prediction Using Neural Networks,” International Conference on Computer Information Systems and Industrial Management Application, 2010, pp. 132-136. Available:  
[http://people.cs.pitt.edu/~hashemi/papers/CISIM2010\\_HBHashemi.pdf](http://people.cs.pitt.edu/~hashemi/papers/CISIM2010_HBHashemi.pdf) [Mar. 31, 2015].
- [4] G. Carlos. “Artificial Neural Networks for Beginners.” [Online]. Available:  
<http://arxiv.org/ftp/cs/papers/0308/0308031.pdf> [Mar. 31, 2015].
- [5] M. Nielsen. (2014, Dec. 21). “*Neural Networks and Deep Learning*” [Online]. Available:  
<http://neuralnetworksanddeeplearning.com/index.html> [Mar. 31, 2015].
- [6] H. Harry, “Performance Evaluation of Artificial Neural Networks in the Foreign Exchange Market,” M.S. thesis, Dept. Math., KTH Univ., Sthlm, Sweden, 2012. [Online]. Available:  
[www.math.kth.se/matstat/seminarier/reports/M-exjobb12/120607a.pdf](http://www.math.kth.se/matstat/seminarier/reports/M-exjobb12/120607a.pdf) [Mar. 31, 2015].
- [7] R. Rojas. (1996). *textitNeural Networks*. [Online]. Available:  
<http://page.mi.fu-berlin.de/rojas/neural/neuron.pdf> [Mar. 31, 2015].
- [8] T. Awokuse and T. Ilvento. “Using Statistical Data to Make Decisions: Time Series Forecasting.” [Online]. Available:  
<http://www.udel.edu/FREC/ilvento/BUAD820/MOD604.pdf> [Mar. 31, 2015].
- [9] Neural NetWork Toolbox User’s Guide, v 8.3., Mathworks, Inc. MA, USA, 2015. [Online]. Available:

[http://www.mathworks.com/help/pdf\\_doc/nnet/nnet\\_ug.pdf](http://www.mathworks.com/help/pdf_doc/nnet/nnet_ug.pdf) [Mar. 31, 2015].

[10] P. Ramani and P.D. Murarka. (2013, Apr). “Stock Market Prediction Using Artificial Neural Network.” *International Journal of Advanced Research in Computer Science and Software Engineering*. [Online]. 3(4), pp. 873-877. Available: [http://www.ijarcsse.com/docs/papers/Volume\\_3/4\\_April2013/V3I4-0350.pdf](http://www.ijarcsse.com/docs/papers/Volume_3/4_April2013/V3I4-0350.pdf) [Mar. 31, 2015].

[11] Mathworks Website. “Neural Network Toolbox Documentation.” [Online]. Available: <http://www.mathworks.com/help/nnet/index.html> [Mar. 31, 2015].

[12] Stock Historical Data Website. [Online]. Available: <http://www.stockhistoricaldata.com>, 2012 [Mar. 31, 2015].

[13] K. Abhishek. “Stock Prediction using Artificial Neural Networks” [Online]. Available: [http://www.cs.berkeley.edu/~akar/IITK\\_website/EE671/report\\_stock.pdf](http://www.cs.berkeley.edu/~akar/IITK_website/EE671/report_stock.pdf) [Apr. 19, 2015]

[14] M. Verleysen et al. “On the effects of dimensionality on data analysis with neural networks,” Springer-Verlag Berlin Heidelberg, 2003, pp. 105-1012. Available: <http://perso.uclouvain.be/michel.verleysen/papers/iwann03mv.pdf> [Apr. 19, 2015]

