CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Stock Market prediction using Artificial Neural Networks

Master's thesis in Computer Systems and Networks

RAFAEL KONSTANTINOU

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2017

Master's thesis 2017

Stock Market prediction using Artificial Neural Networks

RAFAEL KONSTANTINOU

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Division of Networks and Systems
Chalmers University of Technology
Gothenburg, Sweden 2017

Stock Market prediction using Artificial Neural Networks
RAFAEL KONSTANTINOU

Supervisor: Philippas Tsigas, Department of Computer Science and Engineering
Examiner: Marina Papatriantafilou, Department of Computer Science and Engineering

Master's Thesis 2017
Department of Computer Science and Engineering
Division of Networks and Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2017

Stock Market prediction using Artificical Neural Networks
RAFAEL KONSTANTINOU
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

Stock market is one of the most competitive financial markets and traders need to compute the financial workloads with low latency and high throughput. In the past, people were using the traditional store and process method to calculate the heavy financial workloads efficiently. However to achieve low latency and high throughput, data-centers were forced to be physically located close to the data sources, instead of other more economically beneficial locations. This is the main reason, the data-streaming model was developed and it can process large amount of data more efficiently. It was shown in studies that using data streaming we can solve the options pricing and risk assessment problems using traditional methods, for example Japanese candlesticks, Monte-Carlo models, Binomial models, with low latency and high throughput.
However instead of using those traditional methods, we approached the problems using machine learning techniques. We tried to revolutionize the way people address data processing problems in stock market by predicting the behaviour of the stocks. In fact, if we can predict how the stock will behave in the short-term future we can queue up our transactions earlier and be faster than everyone else. In theory, this allows us to maximize our profit without having the need to be physically located close to the data sources.
We examined three main models. Firstly we used a complete random model using a monkey trader. The name monkey trader comes from B.G. Malkiel's claim, that a blindfolded monkey throwing darts at a newspaper's financial pages could select a portfolio that would do just as well as one carefully selected by experts. It works by choosing random actions at random moments. Secondly we used a feed-forward artificial neural network (ANN) model and finally a model that uses Reinforcement Learning(RL). Each of those models was applied on real stock market data and checked whether it could return profit.

Keywords: Stock market, Artificial Neural Networks, Machine Learning.

Acknowledgements

I would like to thank my supervisor Philippas for all the support, advice and the time that he spent to help me construct this work.

I also want to thank my examiner Marina for the cooperation and all the useful comments and advice she gave me.

It would be a mistake not to thank all the professors and members of the department that helped me with their teachings through my master program here at Chalmers.

In addition, I want to thank all my friends that were always by my side encouraging me and supporting me.

Finally, I want to say a big "thank you" to my family that gave me the opportunity to move in Sweden and study. Without them I wouldn't be able to be the person I am today.

Rafael Konstantinou, Gothenburg, June 2017

Contents

List of Figures

List of Tables

1

Introduction and Background Information

In this section we introduce the reader to this thesis and we explain the background information and previous knowledge the reader should have to understand the rest of this thesis. It is separated in 3 parts. In the first part we state the Motivation Goals and Limitations of the tesis. In the second part there is all the information related to the Economical field and the stock market. In the final part is the information related to Algorithmic and Mathematical Methods.

1.1 Motivation

Predicting the movement of stocks in the competitive financial markets is a challenge even for the most experienced day trader. Even in a fraction of a second the price of a stock can change so drastically that the first one who is able to see it and act can win huge amount of money while the rest have to face a financial disaster. Through the years many experts used a variety of methods in order to try and predict the unpredictable stock market and earn money.

1.2 Goals and Limitations

In this section we discuss the Goals and the Limitations of this thesis. We explain in detail what we want to achieve through the thesis and what difficulties we had to overcome to make it happen.

1.2.1 Goals

For the concept of this thesis we tried to predict the price of the stock in the short term future and decide whether is better to buy, sell or hold our stocks. There is no strict definition of short term future. It can be any interval from nanoseconds until a few days. We decided that we will use 5-min intervals as our prediction time. As the stock price depends on the time, time interval is a parameter that had to be decided. We think that five minutes can be a good representation of short term future. Also using a constant time interval simplifies the problem significantly. The main objective is to maximize the profit by trying to increase the capital. Through the years the economists investigated many different methods to try and find an

optimal way to predict the movement of a stock. Some of them are Japanese candlesticks [9], Monte Carlo Models[4], Binomial Models[8, 6, 7], Black-Scholes formula [7] and more. However instead of using those traditional methods we approached the problem of predicting stock prices using machine learning techniques and specifically Artificial Neural Networks [23]. Then we tried to come up with an optimal trading strategy to maximize the potential profit. The main idea is to model a stock trading day into five-minute intervals and using historical information of the stock we tried to predict the stock price after five minutes. Also we tried to train and test our model on historical stock data collected during the period of November 2016 to June 2017 from The OMXS30 index of the Stockholm's stock market.

1.2.2 Challenges and Limitations

1.2.2.1 The Nature of stock market prediction

Stock market is so complicated and many things can affect the change in a price. Not only financial factors can influence the price of a stock. Things like news or the general mood can affect the price in many ways positive or negative. If it was possible to model the stock market with a function it would be a complex function that lives in high-dimensional, maybe infinite dimensional, space. Imagine what would happen if someone knew a away to calculate that function. That someone would be able to profit by taking advantage of it. However the nature of the space is so complicated that finding that function is an impossible thing to do. The real challenge is to try and approximate that function using neural-networks in a way that we can profit by applying it in the stock market. The focus of this thesis is to try to approximate the stock market as good as possible and try to maximize our profit.

1.2.2.2 Data collection

First difficulty we had to face was lack of free and accessible data. Although through the internet someone can find numerous of historical data those are limited to day prices. In order to implement our thesis we needed the history of all the transactions each day. This type of data are available to public until midnight of each day. To overcome this, we decided to collect the transactions ourselves. We created a script that was collecting all the transactions at the end of each day.

1.2.2.3 How did we check the winnings?

An other difficulty we had to face was the way to determine winnings. We had two different options.

1. Winnings is the difference between portfolio value plus the capital we have on our possession and the initial capital.
2. Winnings is the sum of all the differences in price between sequential transactions. For example if we bought a stock at price x and sold it at price y, then the winnings are y-x.

1.3 Road-map

The Thesis is divided into four main parts.

Goal of the first part, chapter 2, is to introduce the reader to all the necessary background information that he needs to understand in order to be able to follow the context of this Thesis.

The second part, chapters 3-5, states, explains and discusses the main research ideas produced by the authors.

In the third part, chapter 6, we list all the experimental results, their explanation and discussion.

The final part of the thesis, chapter 7, is the conclusion where we discuss and analyze the whole project but also mention what we can work in the future.

1.4 Financial Information

In this section we introduce the reader to the Financial knowledge needed to understand completely the context of this thesis.

1.4.1 Information about the Stock Market

1.4.1.1 Stocks

The stock of a company or an organization is described by the equity stock of its owners. A single share of the stock represents partial ownership of the corporation in ratio to the total number of shares. The stock of a corporation is divided into shares and the total number of them is stated at the time of the company's creation. The existing shareholders can authorize the company to issue additional shares. In some cases, each share of stock has a certain declared par value, which is a legal accounting value and it represents the equity on the balance sheet of the organization. There are also cases, where shares of a stocks are issued without having a par value.

1.4.1.2 Shares of a Stock

Shares represent a percentage of ownership in a business. A business may declare different types of shares, each having different ownership rules, privileges, or share values. The owner of the shares has a documented stock certificate that insures that ownership of the stock. A stock certificate is a legal document that documents the amount of shares owned by the shareholder, and other specifics of them, such as the par value, if any, or their class.

1.4.1.3 Stock Market

The Stock Market is the aggregation of buyers and sellers of stocks, which represent ownership claims on businesses. Most of the time, include securities listed on a public stock exchange as but can also be traded privately. An example of private trades, include shares of private companies which are sold to investors through equity crowd-funding platforms. Not only common equity shares are listed in stock exchanges but also shares of other security type e.g. corporate bonds and convertible bonds.

1.4.1.4 Stock market prediction

Stock market prediction is the act of trying to determine the future value of a company stock or other financial instrument traded on an exchange.

1.4.1.5 Shareholders

A shareholder is an individual or a company that legally owns a number of shares in a joint stock company. Shareholders have special privileges that depend on the class of their stock. Some of them are, right to vote on matters such as elections to the board of directors, the right to share in distributions of the company's income, the right to buy new shares issued by the company and the right to access the company's assets during a liquidation of the company. However, shareholder's rights to a company's assets are bound by the rights of the company's creditors.

1.4.1.6 Options

According to J. C. Cox in his paper "Option pricing: A simplified approach." [8], an Option is a security that grands the owner of the stock ability to trade in a fixed number of shares of a specific common stock at a fixed price at any time on or before a given date. The act of completing the transaction is known as exercising the option. The fixed price is called the strike price and the date is known as the expiration or maturity date. A call option gives the ability to buy the shares and a put option gives the ability to sell the shares.

1.4.2 Financial Indicators

In this part we present the different financial indicators and oscillators used in this thesis.

1.4.2.1 Relative Strength Index (RSI)

Relative strength index (RSI) is a momentum oscillator developed by J. Welles Wilder in his book "New Concepts in Technical Trading Systems", 1978 [24].

1.4.2.1.1 The Momentum Oscillator Concept The momentum oscillator measures the velocity of directional price movement. For example when the price of a stock moves up exceptionally fast, at some point it is considered to be overbought.

In the same concept when the price goes down fast it is considered oversold. The momentum oscillators are able to capture that and they are often characterized by a line on a chart drown down in two dimensions. The vertical axis represents the magnitude of the indicator movement. The horizontal axis represents time. Momentum oscillators generally move very fast at market turning points and tent to slow down as the market continues the directional move.

1.4.2.1.2 Relative Strength Index Equation To calculate Relative Strength RS of the first day we need the closing prices of the 14 previous days.

1. Obtain the sum of UP closes for the previous 14 days and divide by 14
2. Obtain the sum for the DOWN closes for the previous 14 days and divide by 14
3. Divide the average UP close with the average DOWN close.

$$RS = \frac{\text{Average of 14 day's closes UP}}{\text{Average of 14 day's closes DOWN}} \tag{1.1}$$

The initial RSI is calculated in the following steps

1. Divide 100 to RS+1
2. Subtract the result obtain above from 100

$$RSI = 100 - [\frac{100}{1 + RS}] \tag{1.2}$$

From this point on we can use the previous calculated average UP and DOWN in the calculation of the next RSI.

1. Multiply the previous average UP by 13 and add the today's UP close and then divide the total by 14
2. Multiply the previous average DOWN by 13 and add the today's DOWN close and then divide the total by 14

The steps to calculate the new RS and RSI are the same as the Initial's one.

1.4.2.1.3 Modified version of RSI In this thesis we used a modified version of RSI. Instead of calculating UP and DOWN averages every day we calculated them every 5 minutes. The reason behind this is that we want to study the momentum changes of a stock in the short term but the original RSI represents the daily momentum changes.

1.4.2.2 Moving averages

According to Gerald Appel in his book "Technical Analysis: Power Tools for Active Investors"[2], moving averages are used to deal with the noise of shorter-term price variations in order to identify with higher precision the significant trends. We have two types of moving averages.

1.4.2.2.1 Simple moving average (SMA)
This type of moving average treats all data in an equal way. Let $x = (x_1, ...x_n)$ be

a subset of our data. Then the simple moving average of that subset is calculated as,

$$SMA = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1.3}$$

where, n is the window size of the average calculation.

1.4.2.2.2 Weighted moving averages(WMA)

This type of moving averages assign different weights to each data point. For example, some weighted averages are assigning higher weights to more recent data and penalizing older data points by assigning them lower weights. Let again $x = (x_1, ...x_n)$ be a subset of our data and $w = (w_1, ..., w_n)$ be the weights. Then the weighted moving average of that subset is calculated as,

$$WMA = \frac{1}{n}\sum_{i=1}^{n} x_i w_i \tag{1.4}$$

where, n is the window size of the average calculation. There are many different types of weighted averages and the main difference is on the way they calculate their weights.

1.4.2.2.3 Time-based separation of Moving average

Moving averages can also be separated in three categories with respect to the market trend we want to observe.

1.4.2.2.3.1 Long-Term Moving average

This type of moving average reflects the longer-term market trends. Those types of averages are usually used to observe the behavior of the stock in the long term and moderate the odds of an imminent market reversal increase. An example is the 200-day moving average.

1.4.2.2.3.2 Intermediate-Term Moving average

This type of moving average reflects the intermediate-term market trends. They usually provide with entry points with in favorable, strongly rising stock market cycles. When intermediate-term moving averages are in decline, selling opportunities can develop. An example can be a 50-day moving average.

1.4.2.2.3.3 Short-Term Moving average

This type of moving average reflects the short-term market trends. They are usually used with a longer-term average. When short-term moving average crosses with a longer-term moving average, signals a shifting in momentum which means is time for a strong action. An example of short-term moving average can be 10-day moving average. They can also be used in day trading on, hourly, 30-minute, 15-minute and even 5-minute bases to model intra-day trading.

1.4.2.2.4 Moving average convergence divergence (MACD) Moving average convergence divergence (MACD) is an indicator used in technical analysis of stock prices and was developed by Gerald Appel [2]

1.4.2.2.4.1 Basic Concepts

MACD represents the difference of the short-term exponential moving average minus the long-term exponential average. When market trends are improving, short-term averages will rise more quickly than long term averages thus MACD lines will turn up. Also when market trends are losing strength, short-term averages will tend to flatten, falling below longer-term averages if declines continue.Then MACD lines will fall below 0. Finally weakening trends are reflected in changes of direction of MACD readings. During the course of price movements, short-term moving averages will diverge and converge with longer-term moving averages and that's why the indicator is named moving average convergence divergence.

1.4.2.2.4.2 Exponential moving average(EMA)

Exponential Moving average is a weighted moving average. The Weight Multiplier for n data-points is calculated as:

$$w = \frac{2}{n+1} \tag{1.5}$$

EMA is calculated as following:

1. Calculate the simple moving average (SMA) using equation 1.3 for the chosen number of time periods.
2. Calculate the weighted multiplier using equation 1.5
3. Calculate initial EMA as $\{EMA\}_0 = (C - SMA) \cdot w + SMA$
4. Calculate new EMA as $\{EMA\}_i = (C - \{EMA\}_{i-1}) \cdot w + \{EMA\}_{i-1}$

where, C is the closing price of the time interval.

1.4.2.2.4.3 MACD formula

MACD line is the subtraction of a 26-day EMA from a 12-day EMA. We calculate EMA using the steps in 1.4.2.2.4.2 and MACD is given by the following formula

$$MACD = EMA_{12} - EMA_{26} \tag{1.6}$$

1.4.2.2.4.4 Modified MACD

We introduced a modified version of MACD in this thesis. Instead of using daily data we are using 5-min intervals as data-points to calculate short-term moving average. The calculation is the same of the traditional MACD but with short-term reference as we are interested in day trading and this is more suitable for the general concept of this thesis.

1.4.2.3 Channels

There are two types of Channels used in stock trading. We discuss them in the following parts.

1.4.2.3.1 Price channels Price channels are commonly used in stock trading in order to discover trends to the stock price. They are separated in a high channel and a low channel.

1.4.2.3.1.1 High channel High channels are calculated connecting the two highest points of the chart in the stock price. It is commonly use in a time based window, for example connect the two highest prices the last 10 days.

1.4.2.3.1.2 Low channel Low channels are calculated by connecting the two lowest points of the chart in the stock price. In the same concept as the high channels, low channels are commonly use in a time based window.

1.4.2.3.1.3 Price channel usage The price channel usage indicate the momentum change. If the actual price is between the channels and the high and low channels are parallel it means that the stock price will follow a specific trend. When the channels are going to converge (connect together) it signals a breaking point. That means that the trader should either buy or sell. The same happens if the lines are diverging (moving away) signaling an other breaking point. In addition when the actual stock price moves outside of the channels it also signals a breaking point.

1.4.2.3.1.4 Channel Calculation Let x_{t1} and x_{t2} be the two highest (or lowest) prices in out time window. Without loss of generality we assume $t_1 < t_2$. the line $x(t)$ that describes the channel is given by,

$$x - x_{t1} = \frac{x_{t1} - x_{t2}}{t_1 - t_2} \cdot (t - t_1) \tag{1.7}$$

where, x_t is the stock price at any given time t.

1.4.2.3.2 Moving Average trading channels The moving average trading channels can be applied for both short time and long term treading. They are useful and help us determine the following:

1. Whether the markets are showing increasing strength or are losing upside momentum.
2. Whether forthcoming support and resistance levels are likely to develop.
3. Whether initial attempts to rally appear likely to develop a good follow-through.
4. When it is safe to buy market weakness.
5. Where and when market retracements are likely to occur and whether the odds favor subsequent recovery.

1.4.2.3.2.1 Channel Creation
Channels are created by drawing lines based on offsets by a predefined percentage above and below the level of the moving average's lines used. Those new lines act like the boundaries of the moving average's channels and the actual line of the moving average becomes the center of the channel.

1.4.2.4 Stochastic oscillator

Stochastic oscillator is a momentum indicator that was developed by Gorge Lane and uses resistance and support levels. The formula is given by:

$$K = 100 \cdot \frac{(C - L_{14})}{(H_{14} - L_{14})} \tag{1.8}$$

where,

- C is the closing price
- L_{14} is the low of the 14 previous intervals
- H_{14} is the high of the 14 previous intervals

The stochastic oscillator is given as the three period interval moving average of K

$$D = \frac{1}{3} \sum_{i=1}^{3} K_i \tag{1.9}$$

1.5 Mathematical and Algorithmic Methods

1.5.1 Chain Rule

According to [10] the definition of the chain rule is the following.
Let $w = f(x, y, w, ..., v)$ be a differentiable function of a finite set of variables $\{x, y, ..., v\}$ and $x, y, ..., v$ are differentiable functions of an other finite set of variables $\{p, q, ..., t\}$. Then w is a differentiable function of the variables $p, q, ..., t$ and the partial derivatives of w with respect to those variables is given by equations of the form

$$\frac{\partial w}{\partial p} = \frac{\partial w}{\partial x}\frac{\partial x}{\partial p} + \frac{\partial w}{\partial y}\frac{\partial y}{\partial p} + ... + \frac{\partial w}{\partial v}\frac{\partial v}{\partial p} \tag{1.10}$$

1.5.2 Artificial Neural Networks

The main motivation behind artificial neural networks is the human brain and the fact that the way it computes is completely different from a digital computer. The brain is a complex nonlinear and parallel computer and has the capability to organize its structural constituents to perform certain computations many time faster than the fastest digital computer in existence today. The definition of a neural network is given in [13].
A neural network is a massively parallel distributed processor made up of simple processing units which has a natural propensity for storing experimental knowledge and making available for use. It resembles the brain in two respects: Firstly knowledge is acquired by the network from its environment through a learning process and secondly interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge. A learning algorithm, the procedure used to perform the learning process, modifies the synaptic weights of the network in an orderly fashion to attain a desired design objective.
The neuron is constructed by three basic elements.

1. A set of synapses or connecting links, each of which is characterized by a weight or strength of its own. An input signal x_j at the input of synapse j connected to the neuron k is multiplied by the weight w_kj.
2. An adder for summing the input signals, weighted by the respective synapses of the neuron, also known as linear combiner.
3. An activation function for limiting the amplitude of the output of the neuron. Example activation functions can be tanh, relu, sigmoid etc.

A neuron can also be described using mathematical terms.

$$u_k = \sum_{j=1}^{m} w_k j x_j \tag{1.11}$$

and

$$y_k = \phi(u_k + b_k) \tag{1.12}$$

where $x_1, x_2, ..., x_m$ are the imput signals; $w_k1, w_k2, ..., w_km$ are the synaptic weights of neuron k; u_k is the linear combiner output due to the input signal; b_k is the bias;

$\phi(.)$ is the activation function and y_k is the output signal of the neuron. The bias b_k is used to apply an affine transformation to the output u_k of the linear combiner.

1.5.2.1 Examples of activation functions

- Linear Activation Function

$$\phi(u_k) = u_k \tag{1.13}$$

- Rectified Linear Activation Function

$$\phi(u_k) = \begin{cases} u_k, & u_k \geq 0 \\ 0, & u_k < 0 \end{cases} \tag{1.14}$$

- Sigmoid Activation Function

$$\phi(u_k) = \frac{1}{1 + e^{-u_k}} \tag{1.15}$$

- Hyperbolic Tangent Activation function

$$\phi(u_k) = \frac{e^{u_k} - e^{-u_k}}{e^{u_k} + e^{-u_k}} \tag{1.16}$$

where u_k is the linear combiner output due to the input signal; $\phi(.)$ is the activation function

1.5.2.2 Types of Artificial Neural Networks

1.5.2.2.1 Feed-forward artificial neural network : In this type of network the information moves in one direction, forward, from the input units to the hidden units (if there are any) and then to the output nodes. The connections cannot form any circles or travel any other direction than forward.

1.5.2.2.2 Recurrent Neural network : In this type of network the connections are allowed to form directed cycles. Information can move in any direction and RNNs can use their internal memory to process arbitrary sequences of inputs. This can make RNN suitable to solve more advanced and difficult tasks but also to have higher computational complexity than feed forward neural networks.

1.5.2.3 Types of Learning Algorithms

1.5.2.3.1 Supervised Learning : In this concept we provide to the network a desired response to the training vector. The desired response represents the optimum response of the neural network. Then the networks parameters are adjusted under the influence of the training vector and the error signal. The error signal is a function of the actual network response and the optimum desired output. The network parameters are adjusted step by step and the procedure repeats until the actual output of the network is close enough to the desired output. Supervised learning is usually used for two tasks classification and regression.

1.5.2.3.1.1 Classification: is the task where we want classify our datapoints into a specific number of classes. A classification example is when we cant to separate data pointer whether they are above or below a curve:

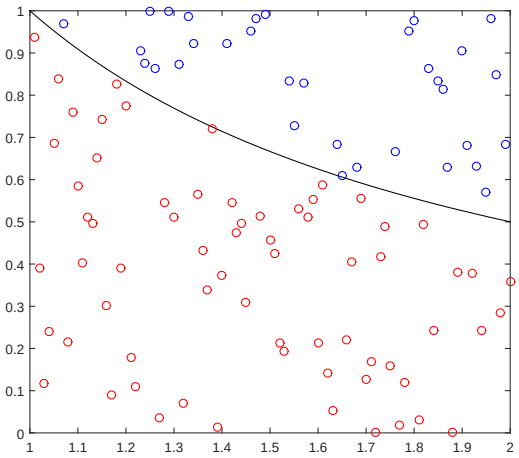


Figure 1.1: Simple classification example

In the above figure the separator is the function $\frac{1}{x}$. We have two classes, the blue and the red. The points that lie above the separator belong in the blue class and points that lie below the separator belong in the red class. Main task of this problem is to approximate the unknown separator function just by processing a finite number of observations and try to simulate the same behavior in the general case.

1.5.2.3.1.2 Regression: is the task where we want to estimate relationships between our datapoints. An example is function approximation, where we want to find a function that can describe our datapoints.

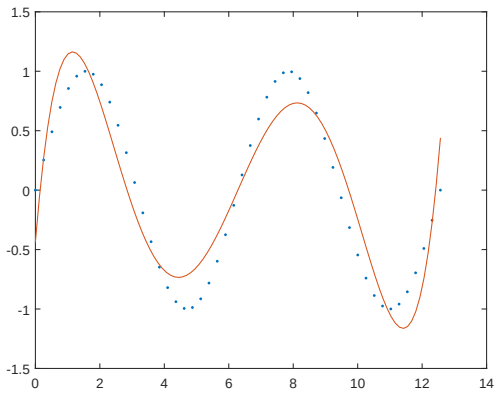


Figure 1.2: Simple regression example

In the above figure we have the red line where is a polynomial function that tries to approximate points from the function the $sin(x)$.

1.5.2.3.2 Unsupervised Learning : In this case we cannot provide the network with the desired response to the training vector. Instead a task independent measure of the representation of what the network has to learn and the network parameters are optimized with respect to that measure. When the network is optimized it can develop the ability to encode feature of the input and thus create new classes automatically. Unsupervised learning is often used to create auto-encoders and auto-decoders where we want our neural network to capture the important features from our data, discard the not so important features and create a good data representation in lower dimension.

1.5.3 Reinforcement Learning

According to R.S. Sutton in the book, "Reinforcement Learning: An introduction, Volume 1" [22], Reinforcement Learning (RL) is a field of machine learning that was originally inspired by behaviorist psychology. The object of Reinforcement learning is learning what to do, how to map situations (states) into actions in order to maximize a numerical reward signal. The learner(agent) is not told which action to take but instead must explore and discover which actions yield the most reward by trying them. However in the most challenging cases an action may affect not only the immediate reward but also the states following and all subsequent rewards.

1.5.3.1 Elements of Reinforcement Learning

The main concepts of the reinforcement learning is the agent and the environment.

1.5.3.1.1 Environment The Environment E is a black box that the agent doesn't know how it works but can only observe its current state. The agent experiments with it by taking some actions and observing the change in the numerical reward. We also assume that the agent does not have the power to modify the environment thus the change in the environment by doing an action is negligible.

1.5.3.1.2 Agent An agent A is a software agent that can react with the environment by observing the states and using action to change the agents environmental state. The main reason and quest is to maximize a numerical reward.

In addition to those we can identify four other sub-elements to a reinforcement learning system, a policy, a reward function, a value function and a model of the environment.

1.5.3.1.3 Policy A policy defines the agent's way behaving at any given time. In other words a policy is a mapping from perceived states of the environment to actions to be taken when the agent identifies those states. Usually a policy is a

simple function or a look-up table but there are cases where it can be an expensive computation for example a search process.

1.5.3.1.4 Reward Function A reward function defines the goal in a reinforcement learning problem. It actually maps perceived states and actions to a single numerical value, the reward, indicating the inartistic desirability of the state. The main objective of the agent is to maximize the total reward it receives in the long run. The reward function is fixed and can be used as a basis for changing the policy. Rewards determine the immediate, intrinsic desirability of environmental states.

1.5.3.1.5 Value Function In opposition to the reward function that indicates what is good in an immediate sense, the value function specifies what is good in the long run. It calculates the value of the state as the total amount of reward an agent can expect to accumulate over the future starting from that state. Values indicate the long-term desirability of states after taking into account the states that are likely to follow, and the rewards available in those states.

1.5.3.1.6 Model of the Environment Model of the environment is non mandatory for Reinforcement learning models. It mimics the behaviour of the environment and they are usually used for planning. With the term planning we mean a way of deciding on a course of action by considering possible future situations before they are actually experienced.

1.5.4 Stochastic Gradient Decent

In [5] the concept of Gradient Decent and Stochastic Gradient Decent is given. Lets consider a superviced learning setup. An example z is a pair (x, y) of an input x and a scalar y. We have a loss function $\ell(\hat{y}, y)$ that measures the cost of predicting $\hat{y}$ when the actual value is y and we choose a family Φ of functions $f_w(x)$ parametrised by a vector w. We seek the function $f \in \Phi$ that minimizes the loss $Q(z, w) = \ell(f_w(x), y)$. We would like to average over then unknown distribution $dP(z)$. Let $E(f) = \int \ell(f(x), y) dP(z)$ be the expected risk that measure the generalization performance and let $E_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i)$ be the empirical risk that measure the training set performance.
Now to minimize the empirical risk $E_n(f_w)$ using **Gradient Decent** we need to update the weights w after each iteration in the basis of the gradient of $E_n(f_w)$

$$w_{t+1} = w_t - \gamma \frac{1}{n} \sum_{i=1}^{n} \bigtriangledown_w Q(z_i, w_t) \tag{1.17}$$

where γ is an adequately chosen gain. When the initial estimate w_0 is close enough to the optimum and γ is sufficiently small this algorithm achieves linear convergence.

The **Stochastic Gradient Decent** is a simplification of the above. Instead of computing the gradient of $E_(f_w)$, each iteration calculates on the basis of a single randomly picked example z_t

$$w_{t+1} = w_t - \gamma_t \bigtriangledown_w Q(z_t, w_t) \tag{1.18}$$

To guarantee convergence we need $\sum_t \gamma_t^2 < +\infty$

1.5.5 Backpropagation

In [19] the authors present Backprobagation method to adjust the weights of connections in the network in order to minimize a measure of the difference between the actual output vector of the network and the desired output vector. The input x_j to unit j is a linear function of the outputs, y_j, of the units connected to j and the weights $w_{i,j}$

$$x_j = \sum_i y_i w_{ij} \tag{1.19}$$

A unit has a real-valued output y_j, which is a non-linear function of its total input

$$y_j = \frac{1}{1 + e^{-x_j}} \tag{1.20}$$

The total error E is defined as

$$E = \frac{1}{2} \sum_c \sum_j (y_{j,c} - d_{j,c})^2 \tag{1.21}$$

where c is an index over cases (input-output pairs), j is an index over output units, y is the actual state of an output unit and d is the desired state. To minimize E by gradient decent it is necessary to compute the partial derivative of E with respect to every weight of the network.
The method starts by computing $\frac{\partial E}{\partial y}$ for each of the output units. For a specific c and suppressing the index c gives

$$\frac{\partial E}{\partial y_j} = y_j - d_j \tag{1.22}$$

Now we can apply the chain rule to compute $\frac{\partial E}{\partial x_j}$

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \frac{\partial y_i}{\partial x_j} \tag{1.23}$$

Differentiating equation (1.20) to get the value of $\frac{\partial y_i}{\partial x_j}$ and substituting in (1.23) we get

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} y_j (1 - y_j) \tag{1.24}$$

This means that we know how a change in the total input x to an output unit will change the error. As the total input is just a linear function of the weights on the connections it is easy to compute how the error will be affected by changing these states and weights. The derivative $\frac{\partial E}{\partial w_{ij}}$ for a weight w for neuron i to j is given by

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_j} \frac{\partial x_j}{\partial w_{ij}} = \frac{\partial E}{\partial x_j} y_i \tag{1.25}$$

and for the output of the i^{th} unit the contribution to $\frac{\partial E}{\partial y_i}$ is simply,

$$\frac{\partial E}{\partial y_i} = \frac{\partial E}{\partial x_i}\frac{\partial x_j}{\partial y_i} = \frac{\partial E}{\partial x_j}w_{ij} \tag{1.26}$$

and thus if we take into account all the connections from a unit i we get

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j}w_{ij} \tag{1.27}$$

1.5.6 Overfitting

Overfitting is the situation where the neural network learns the data-set so well that it fails to distinguish noise and fail to generalize. We can identify overfitting using a validation set while training. The moment when the validation error starts to increase while the training error decreases, are the first signs of overfitting. A good counter to that is early stopping. In the figure below you can see an example of overfitting.

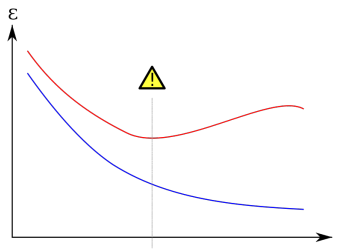


Figure 1.3: Example of over-fitting. Blue line is the training error and red line is the validation error. Figure taken from Wikipedia.

Overfitting is an important factor when training neural networks. We need our models to generalise and thus we should try to avoid overfitting. More information about how we dealt with it is given in section 5.3.1.1

1.5.7 Mean Absolute Percentage Error (MAPE)

We define Mean Absolute Percentage Error (MAPE) as it is presented in [12]. It is defined for forecasts made for periods 1 to N of single series. The forecast error at any given time t is defined by $e_t = A_t - F_t$, where A_t is the actual observation at time t and F_t the forecast made for period t. MAPE is given by

$$(MAPE) = \frac{100}{N}\sum_{t=1}^{N}|\frac{A_t - F_t}{A_t}| \tag{1.28}$$

1.5.8 Maximum Percentage Error

We define Maximum Percentage Error (MXPE) for forecasts made for periods 1 to N of single series. The forecast error at any given time t is defined by $e_t = A_t - F_t$,

where A_t is the actual observation at time t and F_t the forecast made for period t. MXPE is calculated using the equation below

$$(MXPE) = 100 \cdot \max_t |\frac{A_t - F_t}{A_t}| \tag{1.29}$$

1.5.9 Cross Entropy Error Function

G.E. Nasr in the paper "Cross Entropy Error function in Neural Networks",2002 [18] defines the cross entropy error function the following way

$$E_m = \frac{1}{m}\sum_{k=1}^{m}[t_k \cdot ln(y_k) + (1 - t_k) \cdot ln(1 - y_k)] \tag{1.30}$$

where, t_k represents the target value of the network (label) and y_k is the actual network value.

1.5.10 Tensorflow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well. [1]

1.6 Problem Definition

In this thesis we solved the trader's dilemma on when to buy, sell or hold our stocks in order to maximize our profit. Profit is defined as

$$P = C + S_{price} \cdot n - C_0 \tag{1.31}$$

where P is the profit, C the current capital, S_{price} is the price of our stock and C_0 the initial capital.
The lazy approach to the problem is to buy when the stock price is relatively low and sell when the price is relatively high. However this way limit the times we can buy and sell stocks to a few per year. Ideally we wanted to buy and sell stock as often as possible taking advantage of the short term price variations, something that allows us to increase our winnings even more. We can achieve this by predicting the behaviour of the stock in the short-term future.
We can separate our problem into two smaller problems:

1. Predict the stock price in the short term future
2. Use a trading strategy to ensure maximum profit

We explain them in detail in chapters 3 and 4.

2
Datasets

2.1 Data Collection

To collect the data we created a script that was crawling the web to collect all the daily transactions of the wanted stocks. We focus on collection data for the OMSX30 index of the Stockholm's stock market, that holds the 30 most important companies of Sweden. Each transaction had information about the time, name, traded volume and price of the selected stock. We collected data from November 10, 2016 until May 31st , 2017. In April we started collecting data from other companies as well to check if our model can generalize to stocks outside OMSX30 index. All the transactions from each trading day was stored in a plain ASCII text file sorted by stock name.

2.2 Data Processing

We separated each trading day into five minute intervals and modeled it into a vector $x_t = (o_t, c_t, h_t, l_t)$, where o is the opening price, c is the closing price, h is the highest price and l is the lowest price at any given 5-min interval t. Then we connected the trading days creating timeseries of sequential 5-min intervals for the whole dataset. Then we proceeded to calculate the values of the financial indicators and oscillators described above. Now each data entry is represented by a vector $x_t = (o_t, c_t, h_t, l_t, rsi_t, macd_t, stoc_t, hc_t, lc_t, avg_{1ht}, avg_{3ht})$ where,

- o is the opening price
- c is the closing price
- h is the highest price during the time period
- l is the lowest price during the time period
- rsi is the strength relative index
- $macd$ is the mean average convergence divergence
- $stoc$ is the stochastic oscillator
- hc is the channel of the highest prices
- lc is the channel of the lowest prices
- avg_{1h} is the last hour average
- avg_{3h} is the last three hour average
- t is any given 5-min interval

The data are stored in to a plain ASCII text file. We have one file for each stock. In the same time the labels are created. The label at any given time t,is $y_t = c_{t+1}$, where c_{t+1} is the closing price of the stock in the 5-min interval $t + 1$.

In addition we had to normalize the data before we could be able to use them in any prediction model. The answer to the question, "why normalization and how we normalize our data?", is stated in the next section.

2.3 Data Normalization

According to Guoqiang Zhang, B. Eddy Patuwo, Michael Y. Hu, in the paper "Forecasting with artificial neural networks: The state of the art", 1998 [25] data normalization is performed before the training process begins due to the fact that non-linear activation functions tend to squash the output of a node either in $(0, 1)$ or $(-1, 1)$. As a result we need a way to transfer the output of the neural network back to the original data range. Even in the case we use a linear output function it is also stated that normalizing the inputs as well as the outputs has many advantages. Some of them are, to avoid computational problems, A. Lapedes and R. Farber, 1987 [15], to meet algorithm requirements, R. Sharda and B. Patil, 1992 [20], or to facilitate network learning, D. Srinivasan et al, 1994 [21].

2.3.1 Methods for input data normalization

We have four methods to normalize out input data and they are explained by E. M. Azoff in his book, "Neural Network time series forecasting of financial markets" [3]

2.3.1.1 Along Channel Normalization

A channel is defined as a set of elements in the same position over all input vectors in the training or test set. This normalization method performs column by column normalization if the input vectors are put into a matrix. Each column is treated as independent input variable, and it is normalized individually.

2.3.1.2 Across Channel Normalization

In this case the normalization is performed for each input vector independently. In the case the input vectors are put into a matrix it can be seen as row by row normalization.

2.3.1.3 Mixed Channel Normalization

In this case a mixture of both the above methods is used to achieve along and across normalization.

2.3.1.4 External Normalization

In this case all the training data are normalized into a specific range.

2.3.2 Normalization Paradox

We choose to use Along channel normalization as the input vector is constructed in a way that each data point can be seen as an independent variable but when we applied Along channel normalization the following thing happened.
Let $x_j = (..., h_j, l_j, ...)$ be an input vector and c_h, c_l are the normalization constants for the columns of the high and low price respectively and thus $c_l \leq c_h$ and $l_j \leq h_j$. In the case that $0 < c_l < c_h$ the normalized vector is $||x_j|| = (..., \frac{h_j}{c_h}, \frac{l_j}{c_l}, ...)$. However if the stock was stable during the 5-min interval, which is really common, we have the paradox, $\frac{h_j}{c_h} < \frac{l_j}{c_l}$, that means the normalized highest price is less than the normalized low price. The two vectors are correlated and as a result the normalization affected the performance of the network.
The same thing can happen with the pair of the opening and closing price and thus a stock that experience loses, (opening price is greater than the closing price), can be seen as a winning stock after the normalization, (closing price is greater than the opening), if the difference in the normalization constants is high enough. As a result of this observation all the input data that are correlated are normalized with the same normalization constants.

2.3.3 Choosing a normalization constant

The normalization constant is going to be used in order to transfer the output of the neural network from the interval (0,1) to a subset of real positive numbers. We don't want it to bound the output of the network in a way that will prevent it to predict the stock price in the case the stock increases. For example lets say that a stock in day A has a stock price of 150 SEK. We want to predict the price in day B lets assume that the opening price is 160 SEK. If the normalization constant is 161 then the network wont be able to predict any price over 161. In the likely case that the stock price keeps increasing we face the danger of wrong prediction due to bad normalization. On the other hand if we use a huge number as a normalization constant for example 10^7 while the stock price is 150 SEK, we squish the input vectors to zero making it harder for our network to converge. As a result we have to choose the normalization constant in a way that we don't penalize out network and make it harder to converge.

3
Prediction Models

We approached the stock market prediction with two different models. First model tries to solve a classification problem and the second model tries to solve a regression problem.

3.1 Classification Feed Forward Artificial Neural Network

Main concept of this model is to try and classify our input data into three categories characterizing the stock movement in the next 5 minutes. Those classes are:

- First class consists of the stocks that their price will rise the following 5 minutes.
- Second class consists of the stocks that their price will drop with in the following 5 minutes.
- Third class consists of the stocks that their price will remain the same within a threshold ϵ.

3.1.1 Architecture

We created a 5 layer feed-forward artificial neural network, with three hidden layers, an input layer and an output layer.

3.1.1.1 Input Layer

Input layer constitutes of eleven neurons, one for each data point. The input is a vector in the form $x_t = (o_t, c_t, h_t, l_t, rsi_t, macd_t, stoc_t, hc_t, lc_t, avg_{1ht}, avg_{3ht})$ where,

- o is the opening price
- c is the closing price
- h is the highest price during the time period
- l is the lowest price during the time period
- rsi is the strength relative index
- $macd$ is the mean average convergence divergence
- $stoc$ is the stochastic oscillator
- hc is the channel of the highest prices
- lc is the channel of the lowest prices
- avg_{1h} is the last hour average

- avg_{3h} is the last three hour average
- t is any given 5-min interval

The input is normalized in the interval (-1,1). The only neuron that can be negative is the MACD data-point.

3.1.1.2 Hidden Layers

We have three hidden layers. Each hidden layer has a number of neurons and an activation function. The number of neurons in each layer is a hyper-parameter that should be optimized. We used exhaustive search using different number of neurons in each layer and came up with a setup that gives good results. It is important to note that with probability one there is an other setup that can provide better results.

3.1.1.2.1 First Hidden layer The first hidden layer consist of 8 neurons and has a hyperbolic tangent ($tanh$) activation function. The weights that connect the input layer with the first hidden layer are initialized using a random normal distribution. Let x be the input vector and w_1 , b_1 are the weight matrix and the bias vector respectively. Then the values of the first hidden layer h_1 is given by

$$h_1 = tanh(w_1' \cdot x + b_1) \tag{3.1}$$

where, $tanh$ is defined in section 1.5.2.1

3.1.1.2.2 Second Hidden layer The second hidden layer consist of 15 neurons and has a hyperbolic tangent ($tanh$) activation function. The weights that connect the input layer with the first hidden layer are initialized using a random normal distribution. Let h_1 be the first hidden layer vector and w_2 , b_2 are the weight matrix and the bias vector respectively. Then the values of the first hidden layer h_2 is given by

$$h_2 = tanh(w_2' \cdot h_1 + b_2) \tag{3.2}$$

where, $tanh$ is defined in section 1.5.2.1

3.1.1.2.3 Third Hidden layer The third hidden layer consist of 5 neurons and has a hyperbolic tangent ($tanh$) activation function. The weights that connect the input layer with the first hidden layer are initialized using a random normal distribution. Let h_2 be the second hidden layer vector and w_3 , b_3 are the weight matrix and the bias vector respectively. Then the values of the third hidden layer h_3 is given by

$$h_3 = tanh(w_3' \cdot h_2 + b_3) \tag{3.3}$$

where, $tanh$ is defined in section 1.5.2.1

3.1.1.3 Output Layer

Output layer has three neurons and each one represents a class. The result is a three dimensional vector and the dominating dimension is selected to be the class of the data input. Let h_3 be the third layer vector and w_4, b_4 the weight matrix and bias vector respectively. Then the output neuron value is given by

$$y = sigmoid(w_4' \cdot h_3 + b_4) \tag{3.4}$$

where *sigmoid*, is defined in section 1.5.2.1

3.1.2 Loss Function

3.1.2.1 Cross entropy error

Cross Entropy error between two probability distributions is defined in the book "Deep Learning" [11] as follows:
If the distributions p and q are discrete,

$$E(p, q) = -\sum p(x) \log q(x) \tag{3.5}$$

3.1.3 Train Batches

We input data into the neural network in train batches in order to speed up the training process. The size of the train batch is a hyperparameter we had to optimize. Feeding each data point separately and applying corrections on the neuron connectors every single time is really a time consuming. We can achieve almost the same result many times faster if we apply corrections based on the average error when we pass trough whole batch of data.

3.1.4 Accuracy

To calculate the accuracy, A of our neural network we used the following method. If the class of the datapoint was the same as the dominating dimension of the out vector then the prediction was correct and the accuracy was set to 1. If the class of the datapoint was different from the dominating dimension the prediction was wrong and the accuracy was set to 0. The training accuracy is defined as the sum of all the partial accuracy multiply it by 100 to get the percentage and divide it by the total number of datapoints.

$$A = \frac{100}{n} \sum_{i=1}^{n} A_i \tag{3.6}$$

3.2 Regression Feed Forward Artificial Neural Network

We decided to go with a feed-forward artificial neural network solving a regression problem. Basic hypothesis is that the stock price prediction is a high dimensional non linear function and we tried to approximate it using a simple feed forward artificial neural network. The details are explained below.

3.2.1 Architecture

We created a 5 layer feed-forward artificial neural network, with three hidden layers, an input layer and an output layer.

3.2.1.1 Input Layer

Input layer constitutes of eleven neurons, one for each data point. The input is a vector in the form $x_t = (o_t, c_t, h_t, l_t, rsi_t, macd_t, stoc_t, hc_t, lc_t, avg_{1ht}, avg_{3ht})$ where,

- o is the opening price
- c is the closing price
- h is the highest price during the time period
- l is the lowest price during the time period
- rsi is the strength relative index
- $macd$ is the mean average convergence divergence
- $stoc$ is the stochastic oscillator
- hc is the channel of the highest prices
- lc is the channel of the lowest prices
- avg_{1h} is the last hour average
- avg_{3h} is the last three hour average
- t is any given 5-min interval

The input is normalized in the interval (-1,1). The only neuron that can be negative is the MACD data-point.

3.2.1.2 Hidden Layers

We have three hidden layers. Each hidden layer has a number of neurons and an activation function. The number of neurons in each layer is a hyper-parameter that should be optimized. We used exhaustive search using different number of neurons in each layer and came up with a setup that gives good results. It is important to note that with probability one there is an other setup that can provide better results.

3.2.1.2.1 First Hidden layer The first hidden layer consist of 15 neurons and has a hyperbolic tangent ($tanh$) activation function. The weights that connect the input layer with the first hidden layer are initialized using a random normal

distribution. Let x be the input vector and w_1 , b_1 are the weight matrix and the bias vector respectively. Then the values of the first hidden layer h_1 is given by

$$h_1 = tanh(w_1' \cdot x + b_1) \tag{3.7}$$

where, *tanh* is defined in section 1.5.2.1

3.2.1.2.2 Second Hidden layer The second hidden layer consist of 25 neurons and has a hyperbolic tangent (*tanh*) activation function. The weights that connect the input layer with the first hidden layer are initialized using a random normal distribution. Let h_1 be the first hidden layer vector and w_2 , b_2 are the weight matrix and the bias vector respectively. Then the values of the first hidden layer h_2 is given by

$$h_2 = tanh(w_2' \cdot h_1 + b_2) \tag{3.8}$$

where, *tanh* is defined in section 1.5.2.1

3.2.1.2.3 Third Hidden layer The third hidden layer consist of 15 neurons and has a hyperbolic tangent (*tanh*) activation function. The weights that connect the input layer with the first hidden layer are initialized using a random normal distribution. Let h_2 be the second hidden layer vector and w_3 , b_3 are the weight matrix and the bias vector respectively. Then the values of the third hidden layer h_3 is given by

$$h_3 = tanh(w_3' \cdot h_2 + b_3) \tag{3.9}$$

where, *tanh* is defined in section 1.5.2.1

3.2.1.3 Output Layer

Output layer has only one neuron and represents the prediction value of the stock. It has a sigmoid activation function thus the output is the interval (0,1).Then we multiply the output with the normalization constant and as a result we get the predicted stock price. Let h_3 be the third layer vector and w_4, b_4 the weight matrix and bias vector respectively. Then the output neuron value is given by

$$y = sigmoid(w_4' \cdot h_3 + b_4) \tag{3.10}$$

where *sigmoid*, is defined in section 1.5.2.1

3.2.2 Loss Function

3.2.2.1 Mean Squared Error

In order to train our Neural network we need a Loss function. We used Mean Squared Error(MSE) as our loss function and it is defined as:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \tag{3.11}$$

We used Mean Square Error(MSE) instead of Mean Error (ME) as it is computationally cheaper. Also from basic calculus when a function f has a sup(inf) at a point x then the function f^2 has a sup(inf) at the same point x. Note that with f^2 we don't mean function composition (fof) but we just square the output of function f. As a result we can optimize our neural network by minimizing MSE.

3.2.3 Adam Optimizer

We used Adam optimizer to minimize the error of our neural network. According to D. P. Kingma in the paper "Adam: A method of stochastic optimization" [14], Adam optimizer is a computational efficient algorithm gradient-based optimization of stochastic objective functions. The method uses and combines the advantages of two other popular optimization methods. Those are the ability of AdaGrad to deal with sparse gradients and the ability of RMSProp to deal with non-stationary objectives. An other advantage of Adam, is that it requires less memory.

3.2.4 Train Batches

We input data into the neural network in train batches in order to speed up the training process. The size of the train batch is a hyperparameter we had to optimize. Feeding each data point separately and applying corrections on the neuron connectors every single time is really a time consuming. We can achieve almost the same result many times faster if we apply corrections based on the average error when we pass trough whole batch of data.

3.2.5 Training accuracy

To calculate the accuracy of our neural network while testing we used Mean Absolute Percentage Error (MAPE). We defined MAPE in section 1.5.7. We have a validation set that we consistently use to check how our neural network is behaving in order to use early stopping and avoid overfitting.

3.2.6 Training Windows

During training and testing our neural network a few important questions raised about the training. Questions that we need to investigate in depth. We list those questions below:

1. Do we have to train each stock separately or we can train them all together?
2. How much historical data we need to train our Neural network with?
3. Does the training and the testing has to be continues?
4. Do we have to retrain our network when we have new data or is it enough to train once and use it forever?
5. If we have to retrain our network how often does it have to happen? Daily, weekly, monthly?

We investigated those questions separately and we discuss them in depth in section 5.

4

Trading Strategy

We created a Neural Network that is able to successfully predict the stock movement in the next five minutes but we also need to come up with a trading strategy that can help us maximize profit when we buy and sell stocks. In this chapter we present and discuss the different trading strategies.

4.1 The monkey trader

According to B. G. Malkiel in his book "A random walk down Wall Street" [16], a blindfolded monkey throwing darts at a newspaper's financial pages could select a portfolio that would do just as well as one carefully selected by experts. We wanted to test this statement by using our own monkey trader. However instead of using an actual monkey throwing darts we modified the experiment a bit.

4.1.1 Our own blindfolded monkey trader

Our own "blindfolded monkey trader" is just a script written in python that decides what he will do by randomly choosing his action from a set of legal actions. The probability of selecting an action follows a uniform distribution. The set of allowed actions is $S = \{Buy, Sell, Hold\}$. However each time the trader makes a decision he considers a subset S' of S that holds all the possible actions. For example the trader cannot sell if he holds 0 stocks and then $S' = \{Buy, Hold\}$.
The probability of the trader selecting an action is given by

$$P(x = \theta) = \begin{cases} \frac{1}{|S'|}, & \theta \in S' \\ 0, & otherwise \end{cases} \tag{4.1}$$

The simulation starts with the trader having an initial capital and every five minutes he randomly chooses an action from the set of legal actions S'. The simulation ends after a set number of steps. Then we check whether the trader's capital is increased or decreased.

4.2 Simple Buy and Sell Strategy

We used the neural network described above in section 3.2. The output of the network is the predicted price of the stock after 5 minutes. In this strategy we start with a number of stocks and each 5-min interval we have two options:

- Sell stocks now and buy them after 5 minutes if we predict that the stock price will drop.
- Buy stocks now and sell them after 5 minutes if we predict that the stock price will rise.

4.2.1 Profit

We calculate the profit as the following way:

- If we sell now and buy later, the transaction profit(tp) is the difference between the current price and the future price, $cp - fp$.
- if we buy now and sell later, the transaction profit(tp) is the difference between the future price and the current price, $fp - cp$.

The total profit of the simulation (p) is the sum of all the transaction profits

$$p = \sum_i tp_i \tag{4.2}$$

4.2.2 Advantages and disadvantages

The main advantage of this is that we can identify the profit created by transaction to transaction. Also we maximize the number of transactions as every interval as we are buying and selling. However, the main disadvantage by maximizing the total number of transactions is that most of the time we are doing unnecessary transactions. When the stock price follows a monotonic trend then there is no reason buying and selling in the intermediate steps. In addition to this if the trader is required to pay a fee during transactions this strategy may not be that good and the winnings experience significant cut due to the transaction fees.

4.3 Decision based trading strategy

In order to improve our profit we need to find a more profitable strategy than those we described above. Thus we introduce the reader to our decision based trading strategy and its basic elements.

4.3.1 Buy and Sell zones

Buy and sell zones are custom indicators that can help us decide whether to buy or sell our stocks. To calculate them we use moving averages and basic theory from linear algebra.

4.3.1.1 Zone calculation

We calculate buy and sell zones with the following steps.

1. Calculate the moving average of the high price H.
2. Calculate the moving average of the low price L.
3. Now we have the interval $(L, H) \subset \mathbb{R}$.
4. Use the inversible linear transformation f to map $[L, H]$ to the interval [0,1] $f(L) = 0$ and $f(H) = 1$.
5. We have the interval $(a, b) \subset [0, 1]$ as the normalized buy and sell zone. For example (a, b) is $(0.3, 0.7)$.
6. Use the linear transformation f^{-1} on the interval (a, b) and expand it to $(B, S) \subset [L, H]$.
7. B is the buy zone and S is the sell zone.

4.3.1.2 Zone usage

The way we use the zones is simple. If the prediction of the stock price is greater than the sell zone then we sell our stocks. If the prediction of the stock is lower than the buy zone then we buy stocks.

4.3.2 Trading History

In order to ensure profit we introduce trading history. For a short time window we store our transactions in the sell and buy history accordingly. When we sell a stock we check if the price is higher that the buy history. When we buy a stock we check if the price is lower that the sell history. After the time window passes we reset the trading history. The reason is that we don't want the trader to do a transaction that is going to be unprofitable. However we want to prevent cases like monotonic behavior of the stock curve where not acting will have disastrous effect. An example of that is when the stock price is dropping and we hold the stock because we bought it higher. A better way of action would be to sell it now and buy it again when it starts to stabilize again at a lower price.

4.3.3 Legal Actions

When an action is decided we check if we can actually do that action. For example our strategy decides to sell but unfortunately we don't own any stocks. We can easily conclude that the action "Sell" is not a legal action and as a result we have to change the action to "Hold". The main use for the set of legal actions is to make sure we prevent any unorthodox scenario from happening.

4.3.4 Step Lock

In the case aggressive stock price slopes we want to make sure we take the correct decisions in order to minimize loses and maximize potential winnings. This is used as a fail-safe mechanism in the cases of extreme value changes due to external factors.

Some of those are the announcement company's quarter results, a terrorist attack or an accident to company's facilities etc. Step lock works by not allowing the trader to actually trade even if all the other conditions of an action is met.

4.4 Reinforcement Learning trading strategy

The stock market and the trader's dilemma can be perfectly modeled as a Reinforcement Learning problem. We have the stock market as our Environment, and our trading agent that has a list of legal actions over the environment $\{Buy, Sell, Hold\}$. The trading agent does not have enough power to affect the stock market by performing an action. The state is the stock price and history and the agents portfolio. The main objective of the trader is to maximize the reward which in this case is the value of the portfolio plus the owned capital.

4.4.1 Inspiration

The inspiration for this trading strategy was Deep-mind's paper "Playing Atari games with Reinforcement Learning "[17]. In their paper they created an agent that was able to outperform human players while playing Atari games Using Q-learning models. We can easily use the same concept and try to create our own agent to be able to observe and learn how to invest in the stock market using the same fundamentals.

4.4.2 Reinforcement learning in the stock market

As we mentioned above the stock market is the Environment, our trading software is the agent that can act over the environment. The agent experiments with the environment and learn what actions will maximize his reward. Why is the stock market a perfect match for a Reinforcement Learning agent? The answer is rather simple. An action, for example buying a stock, can affect the agent's reward in the long term as the price can grow or fall after some time. The ability of the agent to be able to learn through buying and selling can in theory produce some amazing results.

4.4.3 Model architecture

This model is consisted by a simple simulation of the environment, and two different neural-networks that act as the agent. Complementary we have a replay memory that stores all the past simulations with the states and the actions.

4.4.3.1 Simulation of the environment

This is a script written in python that returns to the agent the state of the stock market at any given time. It behaves like the stock market and feeds information to the agent. The information that the Environment can feed to agent is in the format we stated in section 2.2.

4.4.3.2 The trading agent

The agent is consisted by two neural networks. The agent asks the environment for its state and receives a vector with the information as stated in section 2.2. Then the agent passes the information through the first neural network that will predict the price of the stock after 5 minutes. This neural network is the same as the one described in section 3.2.
Now the agent has the predicted price and wants to decide on what to do. In general the agent needs to calculate the potential reward of each action in the long term. However this is a spanning tree that grows exponentially as time passes.

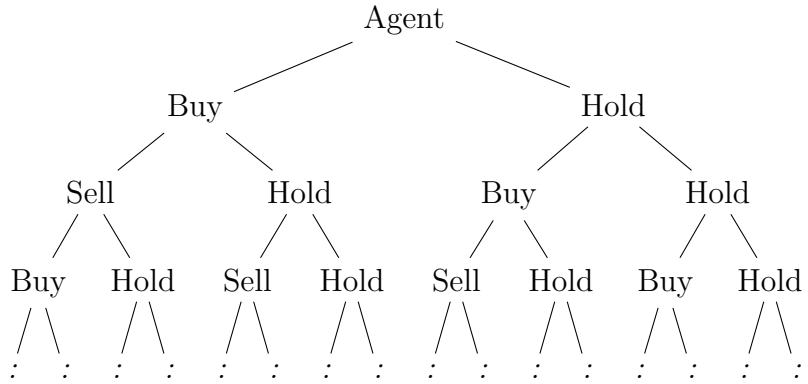


Figure 4.1: Decision tree example for the Agent

To overcome this exponential complexity we try to approximate the long term reward using a neural network that takes as input the current state and approximates the long term reward for each possible action. After that the agent picks the most rewarding action.

4.4.3.3 Training

Those two neural networks can be trained separately or together. We could use the already trained Neural network for the stock prediction and focus completely on training the reward approximation network or we could train both of them at the same time. To make our life easier in training and help our network to converge we used replay memory.

4.4.3.4 Replay Memory

Replay Memory is a buffer that stores the data points we already used for train together with the Environmental state and selected action. Occasionally during the training the agent samples over the replay memory and uses the sample to train the network.

5
Results

In this chapter we present all the major results we got from our experiments.

5.1 Monkey Trader

In this section we list and discuss the experiments we did with the monkey trader. We explained how the monkey trader works in 4.1.1. We only run the experiment once in order to see if the claim of B.G. Malkiel can be confirmed.

5.1.1 Monkey Trader Simulation

We used data from the ABB stock starting from 11th of November 2016 to 13th of December 2016. The trader randomly decides whether to buy, sell or hold the stocks. When the simulation ends we say that the simulation is profitable if the total value of the holding stocks plus the current capital is higher than the initial capital. We run the same simulation 1000 times. The surprising result was that the monkey trader was able to profit 734 out of 1000 simulations. This result actually completely confirms B. G. Malkiel's claim that a blindfolded monkey throwing darts at a newspaper's financial pages could select a portfolio that would do just as well as one carefully selected by experts [16]. The average winnings of the monkey trader was 0.36% of the initial capital, maximum winnings was 1.21% of the initial capital and maximum loses was -1.25% of the initial capital.

5.1.2 Monkey trader Discussion

The above results clearly state that the claim of B. G. Makiel is confirmed. We were able to simulate a simple trading strategy without the need of having any financial knowledge and for 73.4% of the simulations the trader was able to increase its initial capital. However the total amount of winnings vary from -1.25% to 1.21%. This can be interpreted as it is most likely for someone to win by randomly selecting actions but the losses can be higher. This result can be used as a baseline to compare how our proposed models and strategies can compare to a completely random one.

5.2 Classification Neural Network

In our first attempt to train a neural network we used the first model we defined in section 3.1. We trained and tested the classification neural network model and we

present the results below.

5.2.1 Training of the network

We trained our neural network with data that we collected. The procedures we followed to collect and process our data are explained in section 2.2. We separated our training data into two sets, a train set and validation set. We used the training set to train our network and used the validation set during training in order to forward pass the network and detect overfitting. In the cases that we observed overfitting we proceed with stopping the training process.

5.2.2 Testing of the network

To test the accuracy of our network we used data that we collected that chronologically come after the data collected from the training set. Then we passed the testing set through the network and calculated the accuracy using the procedure we defined in section 3.1.4

5.2.3 Presentation of Results

Unfortunately we were unable to train this neural network and have convincing results. In all the cases the neural network was diverging and the accuracy of correct classification was slightly over 50%. Maximum accuracy achiever was 62.5% and minimum accuracy was 51.2% The main results are presented in the table below.

It is important to note that the gap that you can see between the training and the testing in not important as the stock market was closed most of the days due to Christmas holiday. Also the three working days during the two weeks holiday period can be considered as special days and do not follow the pattern of a normal trading day. For that reason we decided not to include those days in the training and testing.

Table 5.1: Results of the classification Neural Network

Stock Symbol	Training Period	Testing period	Accuracy
ABB	10/11/2016-23/12/2016	10/1/2017-13/2/2017	51.465%
ALFA	10/11/2016-23/12/2016	10/1/2017-13/2/2017	54.691%
BOL	10/11/2016-23/12/2016	10/1/2017-13/2/2017	59.973%
GETI-B	10/11/2016-23/12/2016	10/1/2017-13/2/2017	57.824%
KINV	10/11/2016-23/12/2016	10/1/2017-13/2/2017	61.031%
NOKI	10/11/2016-23/12/2016	10/1/2017-13/2/2017	53.002%
SAAB-B	10/11/2016-23/12/2016	10/1/2017-13/2/2017	55.151%
SCA-B	10/11/2016-23/12/2016	10/1/2017-13/2/2017	60.823%
SECU-B	10/11/2016-23/12/2016	10/1/2017-13/2/2017	59.062%
SHB-A	10/11/2016-23/12/2016	10/1/2017-13/2/2017	61.048%
SWMA	10/11/2016-23/12/2016	10/1/2017-13/2/2017	51.289%
TEL2	10/11/2016-23/12/2016	10/1/2017-13/2/2017	62.554%
VOLV-B	10/11/2016-23/12/2016	10/1/2017-13/2/2017	53.096%

5.2.4 Discussion of Results

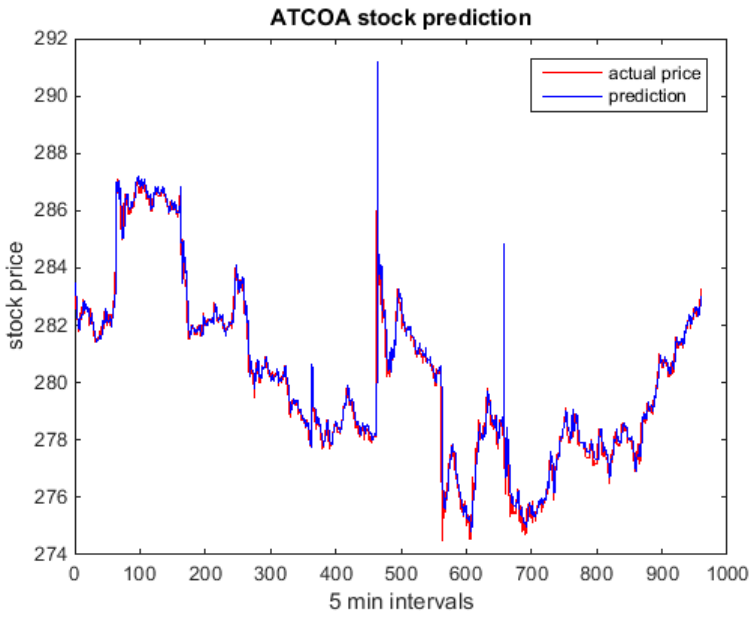
We can easily observe that the accuracy of the network is below expectations. Someone can argue that the results are random. Our approach and hypothesis for this model is that we can classify the input into three classes characterizing the stock movement. We would consider this model good if the accuracy was over 90%. However it was not possible to achieve this and we decided to look for different models. Due to the random nature of the results we decided not to proceed with the simulation of the market in later state. We have to note that this was our first attempt to predict the stock movement and the methods described in sections 4.2 and 4.3 were considered in a later state. Also the simulation of the stock market was implemented yet. By the time we implemented those models and the way to calculate our winnings we have already discarded this model and we thought there was no reason to return back to this method and test whether the initial capital would increase and by what percentage.
However there is the possibility that with a bit different architecture and activation functions someone can achieve better results and reach higher accuracy.

5.3 Regression Neural Network with decision based trading strategy

In this Section we present and discuss the various results we get from out experiments using the regression Neural Network and the decision based trading strategy.

5.3.1 Stock Prediction Preliminary Results

After the bad results we got from the classification we shifted our focus on the regression model. Our first approach was to try and predict the stock price of



ATCOA. The training process used data from 10th of November 2016 until 23rd of December 2016. To test our network we used data from 10th of January 2017 to 20th of January 2017. The input data were processed and normalized in the way we described in sections 2.2 and 2.3 respectively. To calculate the performance of our network we used Mean Absolute percentage Error(MAPE) defined in section 1.5.7.

5.3.1.1 Network Convergence, Overfitting and Early-stopping

We considered the network to be converged or close to convergence when MAPE for the validation set was below 0.07%. What this actually means is that when we predict a stock price the expected error of the prediction will be 0.07%. If the stock price is 100 SEK then the prediction will be in the range of 99.93 SEK and 100.07 SEK. This is acceptable as the stock price minimum change is 0.1 SEK.
Over-fitting is explained in section 1.5.6 and it is usually countered by early-stopping the training process. In the following figures we present our results.
Although MAPE error is below 0.1% we can easily identify over-fitting by using Maximum Percentage Error (MXPE). We defined MXPE in section 1.5.8. If MXPE is way higher than MAPE then it means that our neural network is either overfitting or the stock prediction is not converging to the stock curve because we need more data in order to do it. Overfitting can easily identified on the curves above at the points where the blue line explodes.

5.3.1.2 Discussion of results

We can easily observe that the regression Feed Forward Neural Network Model can successfully predict the future stock price of ATCOA stock price. However we have to be careful about over-fitting by occasionally forward pass the validation set through the network to see if we should stop the training processes. After those amazing results we are should come test our various trading strategies. We can

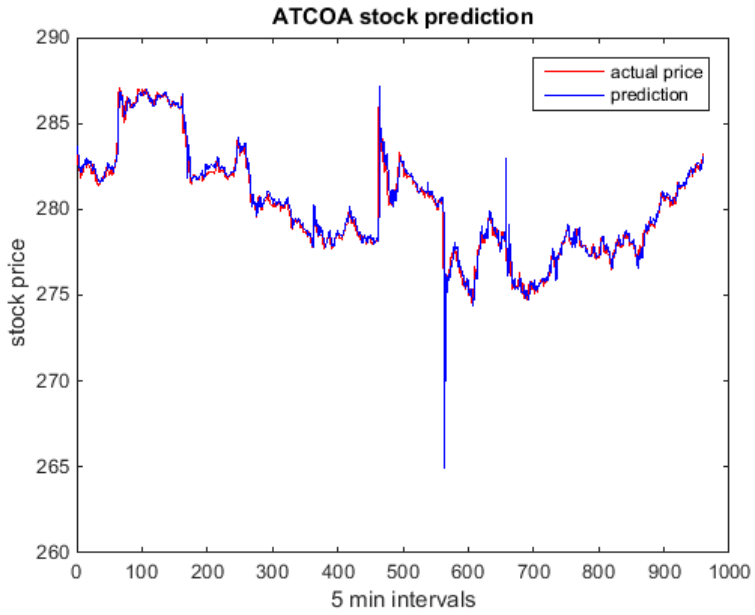
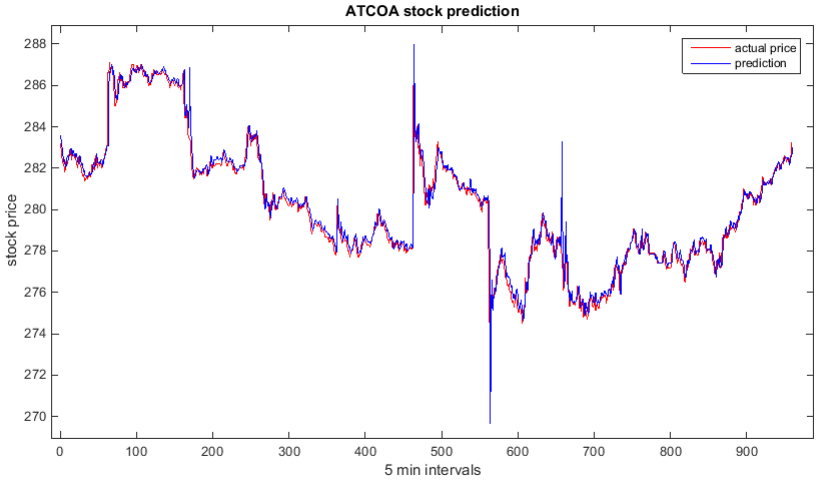


Figure 5.1: Stock Price prediction of ATCOA overfitting part A



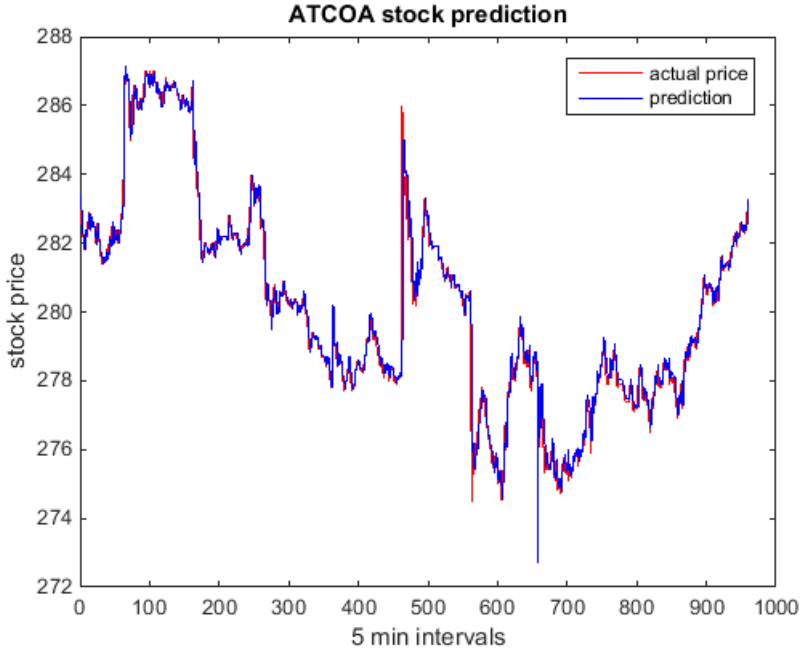
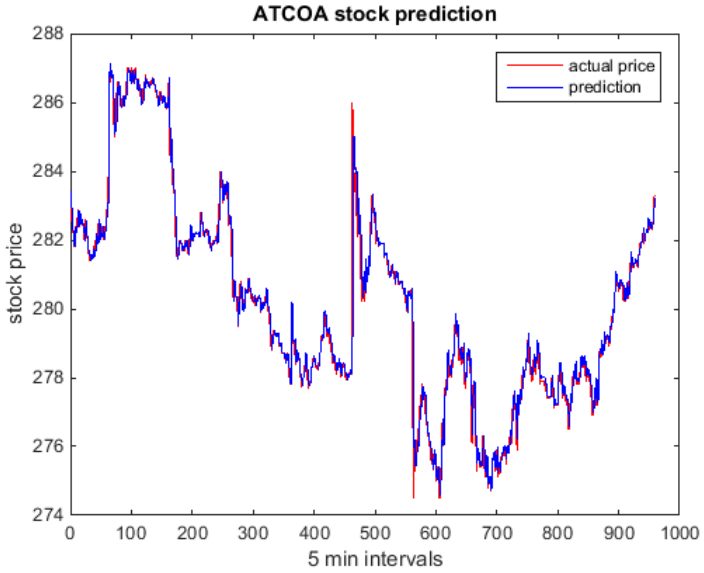


Figure 5.2: Stock Price prediction of ATCOA overfitting part B



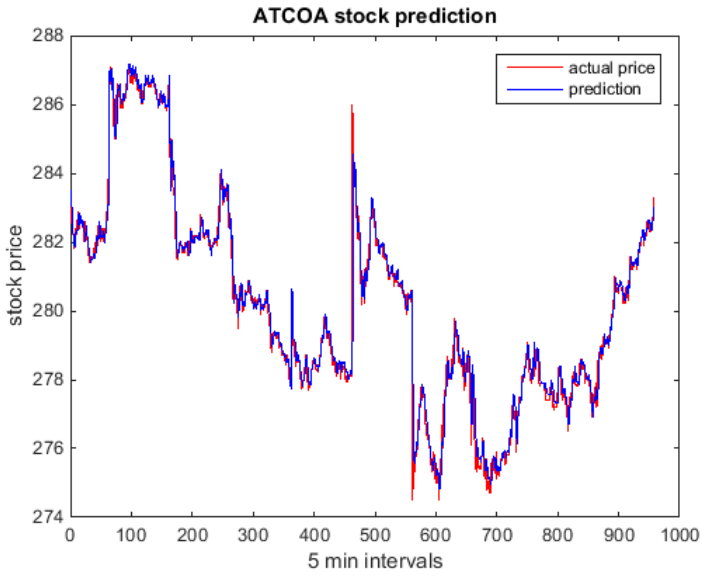


Figure 5.3: Stock Price prediction of ATCOA with early stopping

actually get a good enough fit of our prediction in the stock price curve.

5.3.2 Buying and selling constant number of stocks

The first experimental approach after implementing the decision based trading strategy was to buy and sell constant number of stocks each transaction. However the question raised though is what is the optimal number of stocks we should buy and sell? To solve that we decided to run the trading strategy from 1 stock to the maximum amount we can buy and sell. For all the experiments the initial capital is set to 30000 SEK.

5.3.2.1 Training of the network

To train the network we used data from 10th of November until 23rd of December 2016. Data were processed and normalized with the methods explained in section 2.2 and 2.3 respectively. Train data ware randomly separated into a training and a validation set. The reason for the validation set was to regularly testing the accuracy of the network and prevent overfitting.

5.3.2.2 Testing of the network

To test the network we used data from 9th of January until 10th of February 2017. The testing process was consisted by two steps.

1. Forward pass of the network with the testing data and calculate MAPE and MXPE.
2. Simulate the transactions and the trading at the stock market by buying and selling a constant amount of stocks. Calculate the winnings after the simulation.

Note we repeat step 2 for all the possible amount of stocks, as we want to find the optimal amount of stocks to trade. Below we present the total winnings in comparison to the total amount of stock we bought or sold each time.

We present below the graphs for some of the stocks.

5.3.2.2.1 Graphs of Stock A In the figures below you can see the winnings in comparison to the number of stocks in each transaction for the stock of ABB. First figure shows the numerical winnings in SEK and the second figure shows the percentage increase of the initial capital.

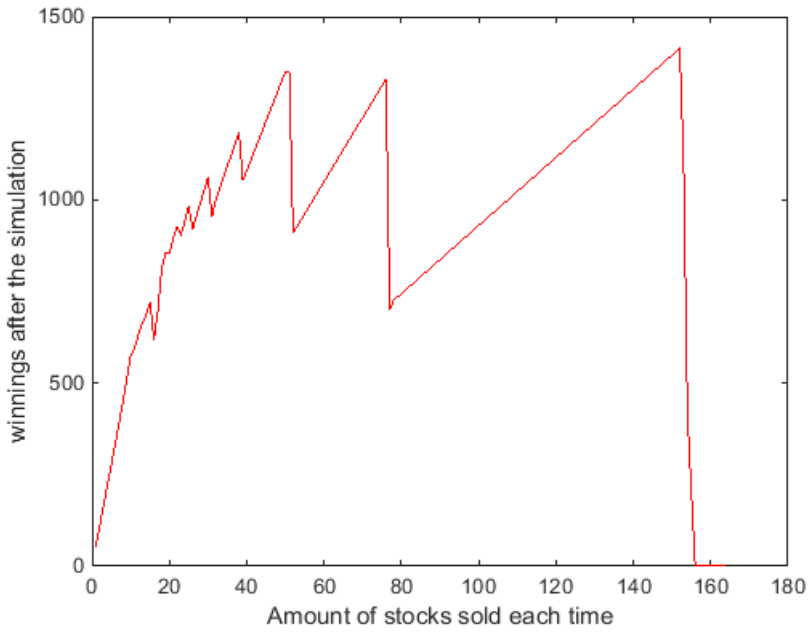
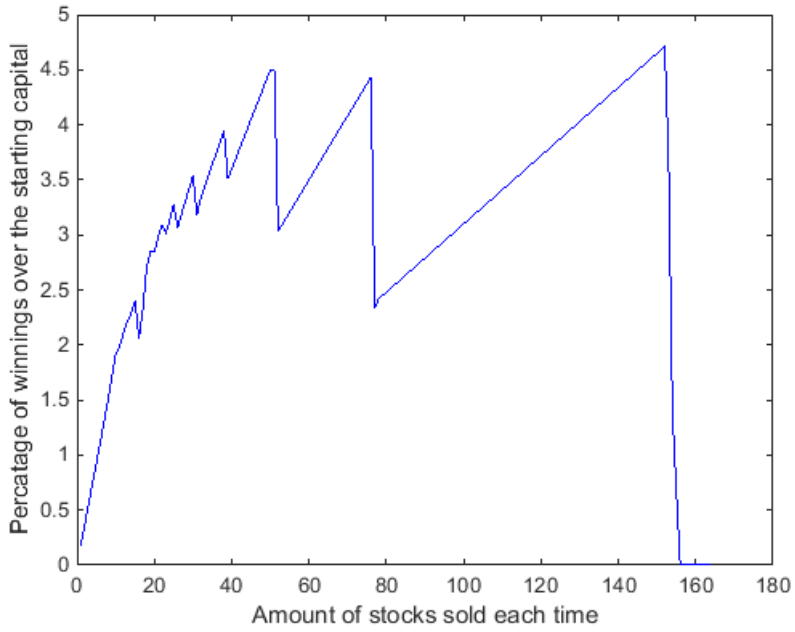
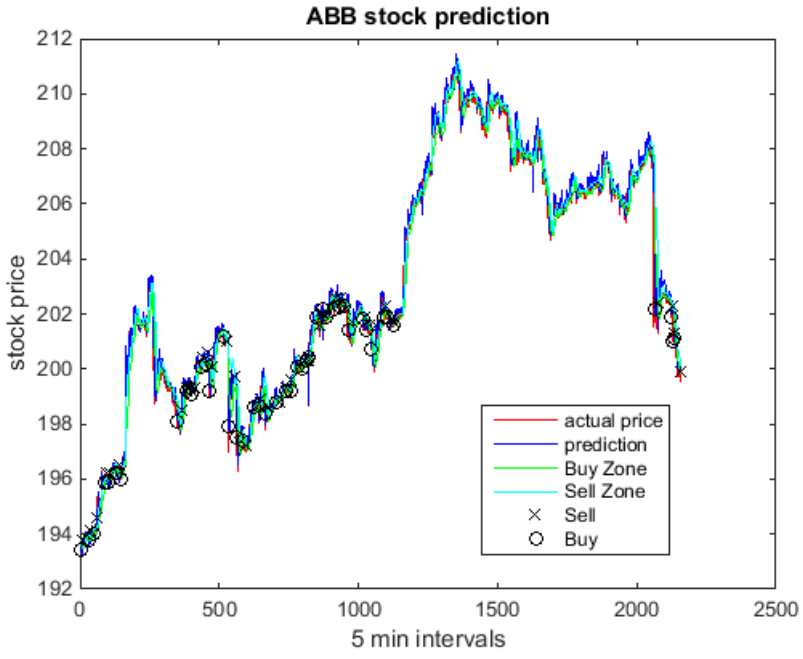


Figure 5.4: Winnings while doing transactions with different amount of stocks



The figures below show the stock prediction and a snapshot of the winnings for a specific number of stocks. We also show where each transaction happens.



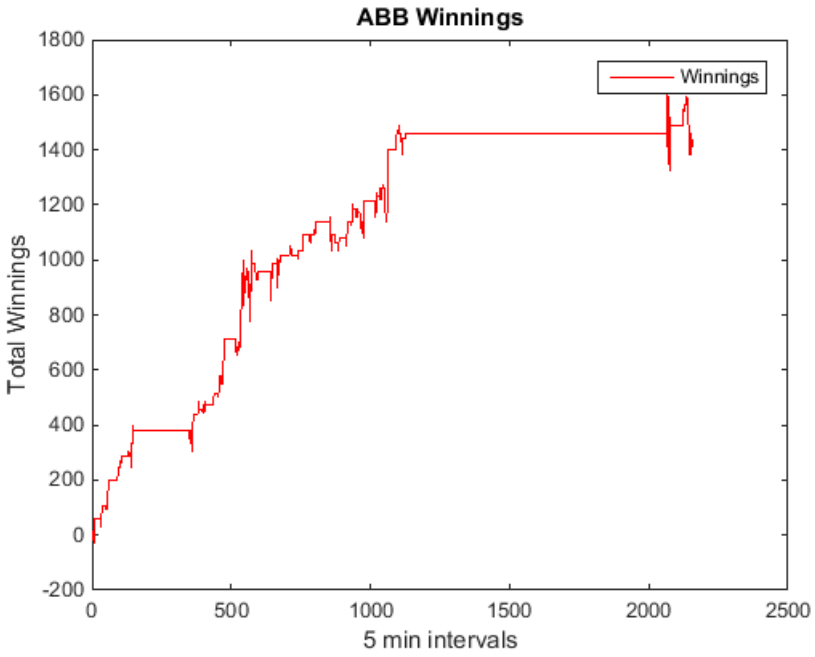


Figure 5.5: Stock Price prediction of ABB and the winnings

5.3.2.2.2 Graphs of Stock B In the figures below you can see the winnings in comparison to the number of stocks in each transaction for the stock of SAND. First figure shows the numerical winnings in SEK and the second figure shows the percentage increase of the initial capital.

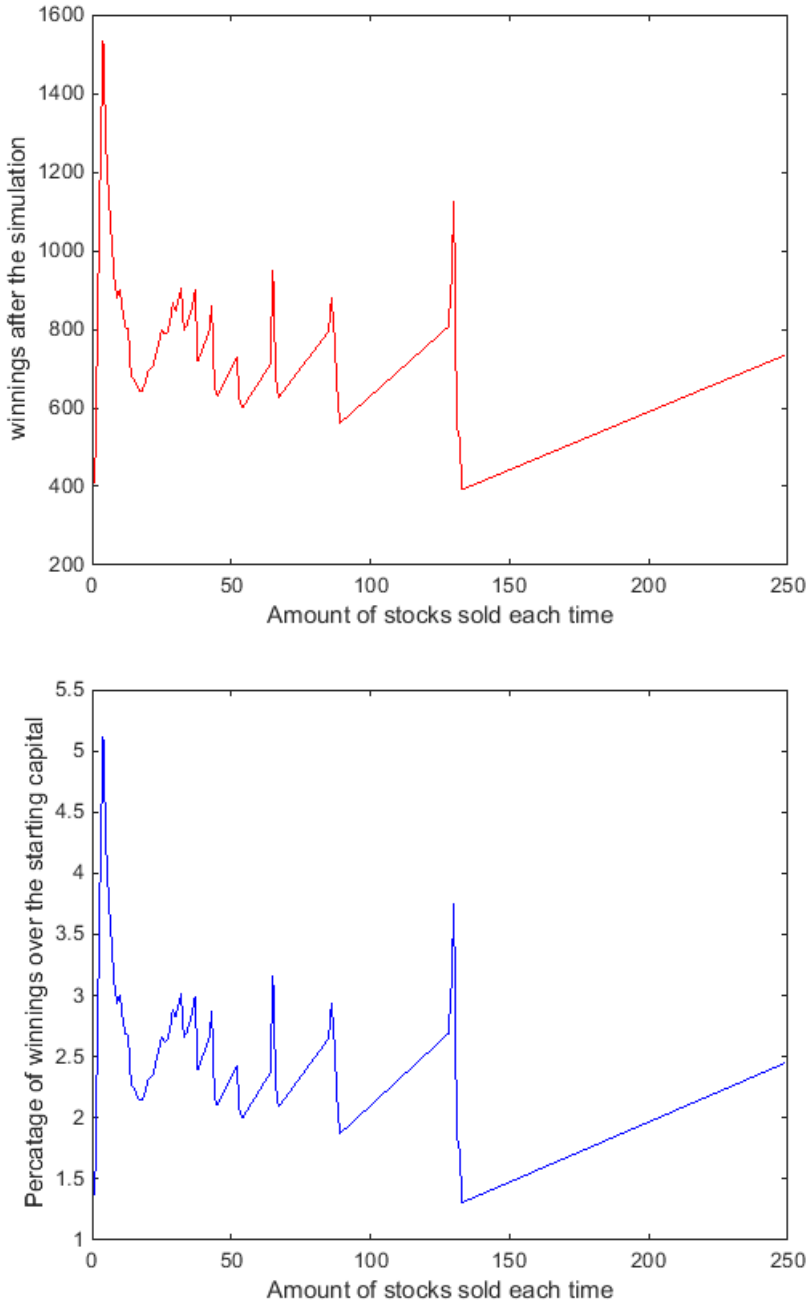
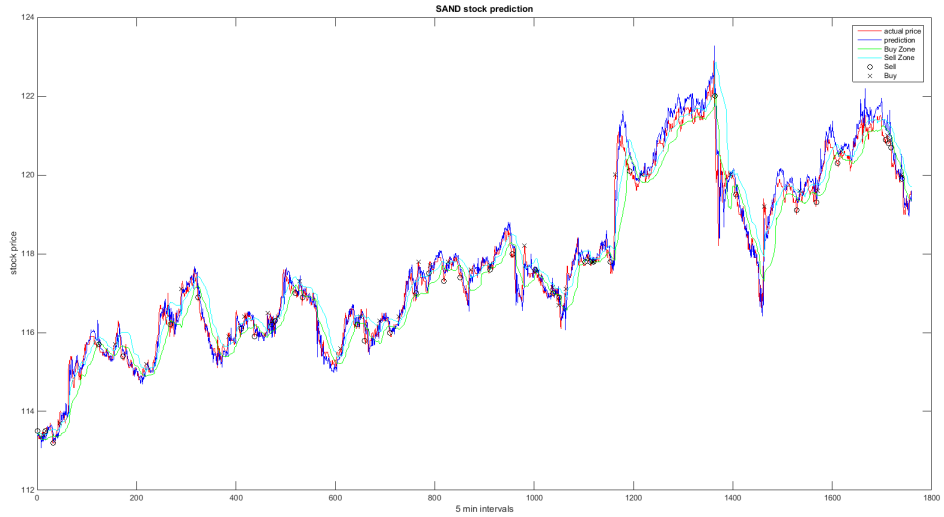


Figure 5.6: Winnings while doing transactions with different amount of stocks

The figures below show the stock prediction and a snapshot of the winnings for a specific number of stocks. We also show where each transaction happens.



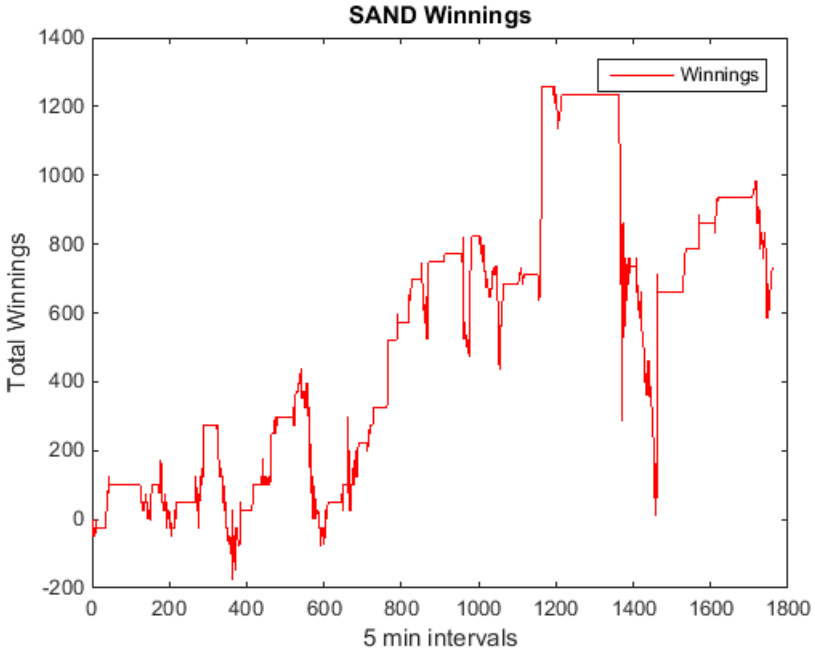


Figure 5.7: Stock Price prediction of SAND and the winnings

5.3.2.2.3 Graphs of Stock C In the figures below you can see the winnings in comparison to the number of stocks in each transaction for the stock of SKFB. First figure shows the numerical winnings in SEK and the second figure shows the percentage increase of the initial capital.

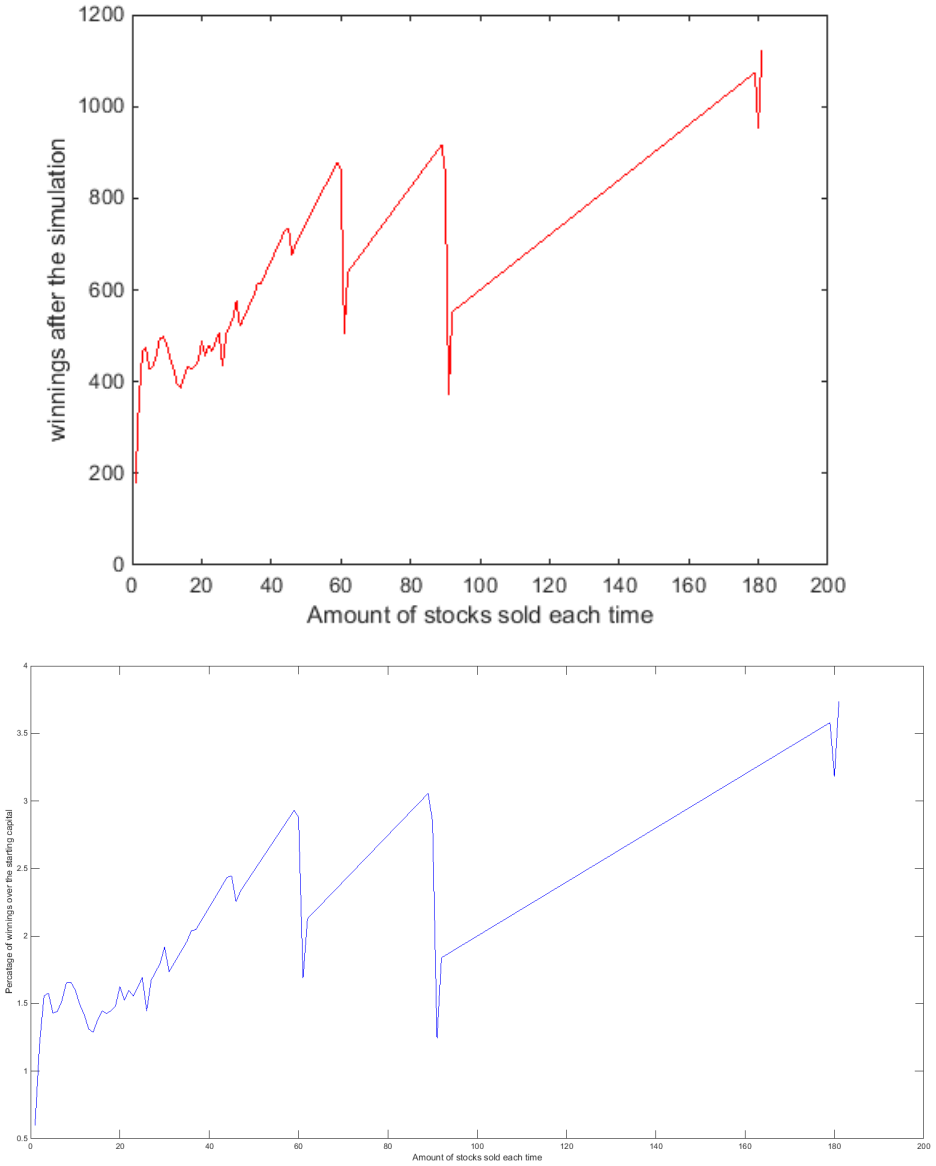


Figure 5.8: Winnings while doing transactions with different amount of stocks

The figures below show the stock prediction and a snapshot of the winnings for a specific number of stocks. We also show where each transaction happens.

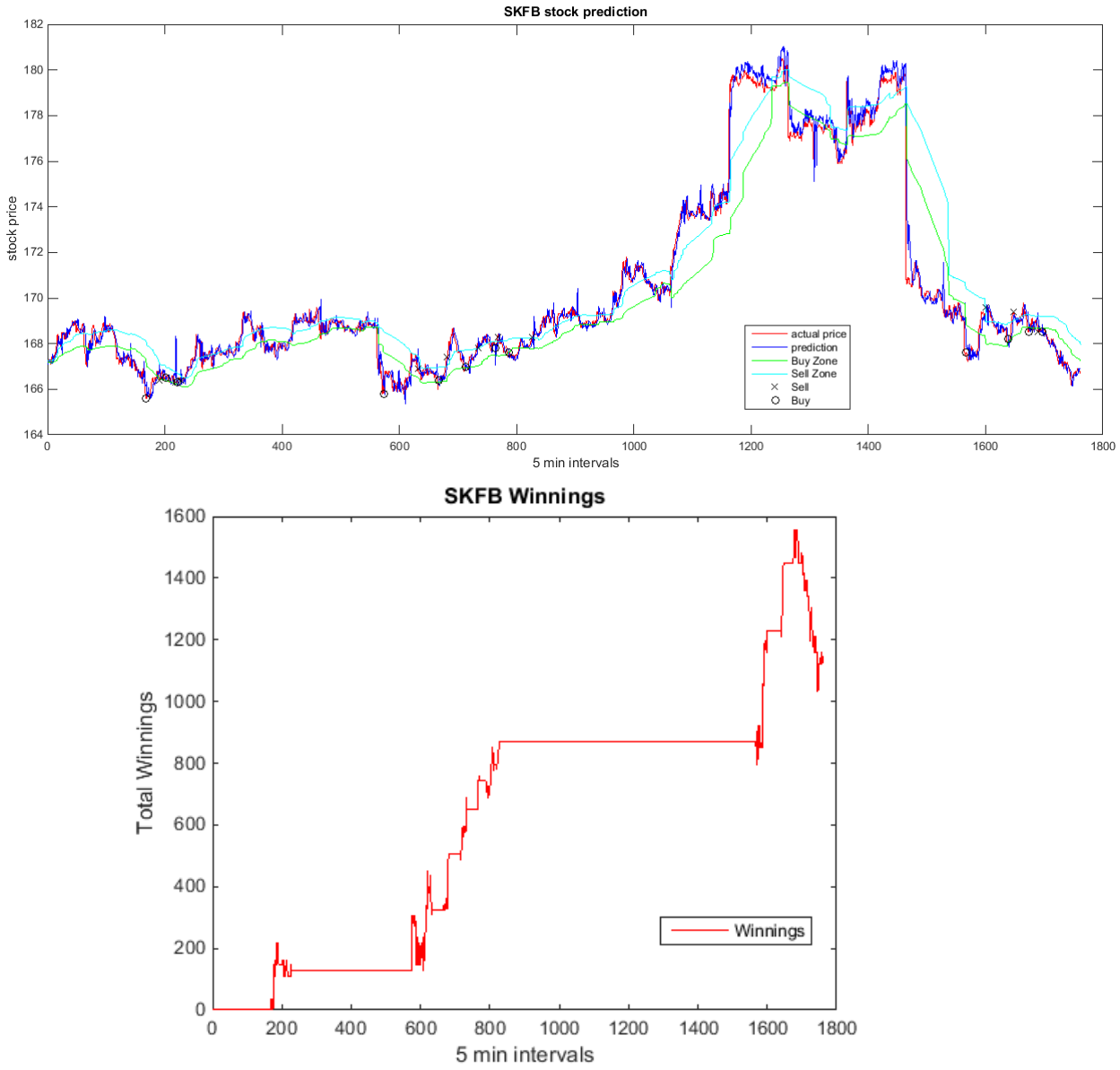


Figure 5.9: Stock Price prediction of SKFB and the winnings

5.3.2.2.4 Graphs of stock D In the figures below you can see the winnings in comparison to the number of stocks in each transaction for the stock of VOLVO. First figure shows the numerical winnings in SEK and the second figure shows the percentage increase of the initial capital.

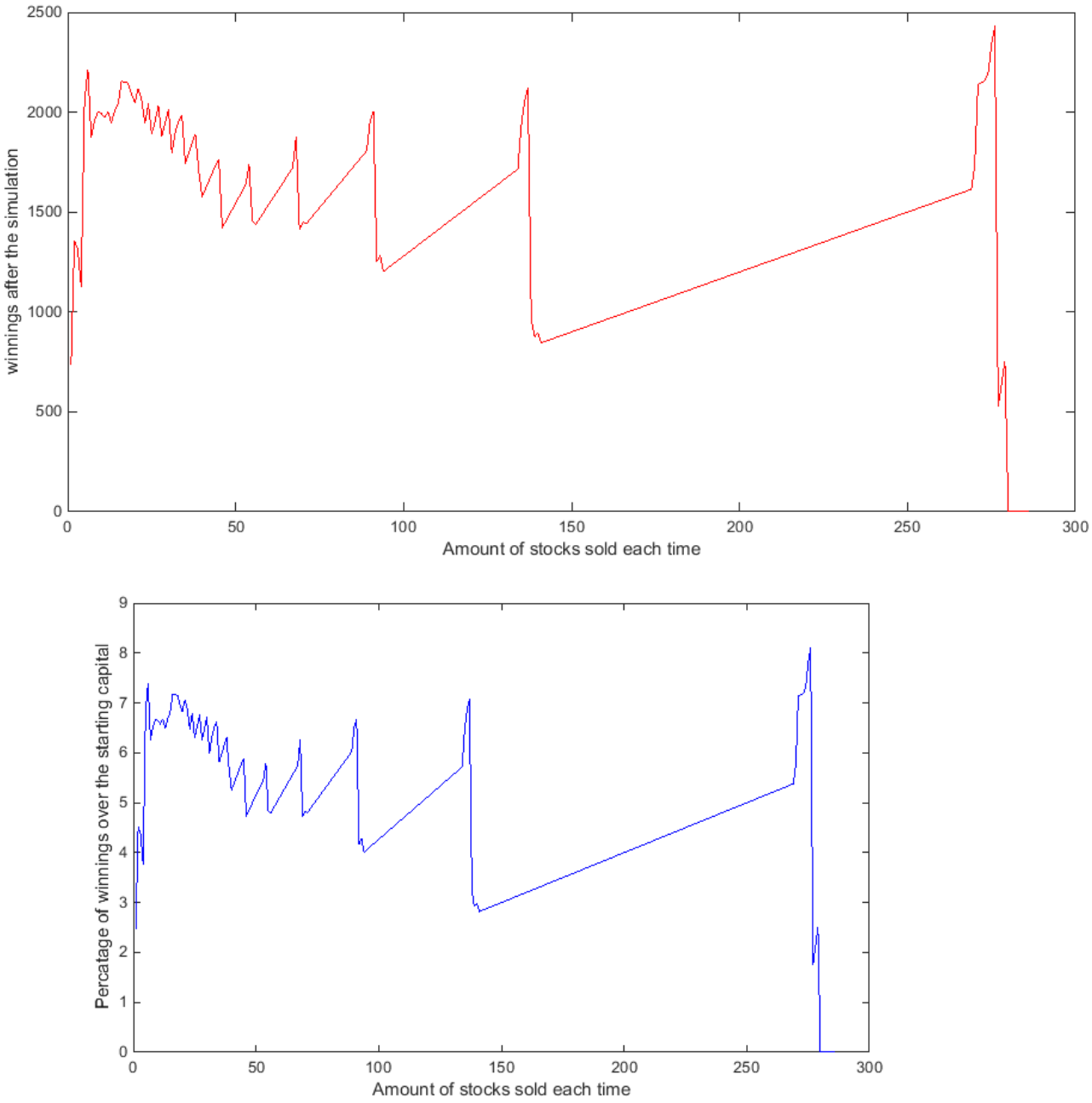


Figure 5.10: Winnings while doing transactions with different amount of stocks

The figures below show the stock prediction and a snapshot of the winnings for a specific number of stocks. We also show where each transaction happens.

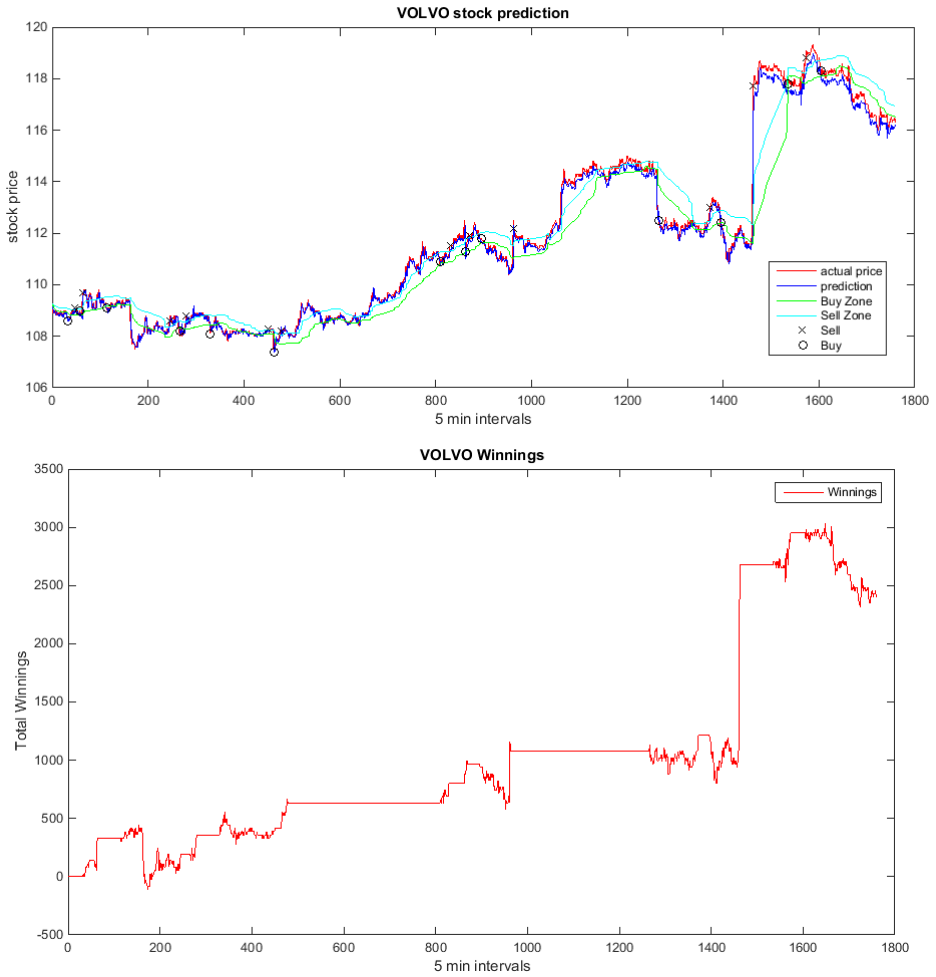


Figure 5.11: Stock Price prediction of VOLVO and the winnings

5.3.2.3 Discussion of Results

From the figures above we are able to conclude that the profit we make depends on the leftover unused capital after each transaction. For example lets assume we can buy 100 stocks with the initial capital we have. If our buy step is 60 stocks then we will have 40% of our initial capital unused thus we will have reduced winnings. It is important to understand that sometimes buying 50% of the max amount of stocks that we can buy with the initial capital, can behave better than buying 100% as it gives us the chance to react on mistakes. For example if the price of the stock increases we might not be able to buy stocks at all at that point. A good example can be seen in figure 5.5 where the trader cannot buy stocks when the price is over one specific point.

Our main conclusion is that instead of buying and selling a constant amount of stocks we should try to minimize the unused capital and to do that we should buy the max possible amount by recalculating the stock number depending on our current capital and the stock price. For more information refer to section 5.3.4.

5.3.3 Capital, profit and number of stocks

The next experiment we conducted we wanted to find out if there is a relationship between the initial capital, the profit and the number of stocks we buy and sell. For that reason we conducted simulations on LUPE stock with normal capital (30000 SEK) and with ten times more initial capital (300000 SEK). You can see the results in figures below.

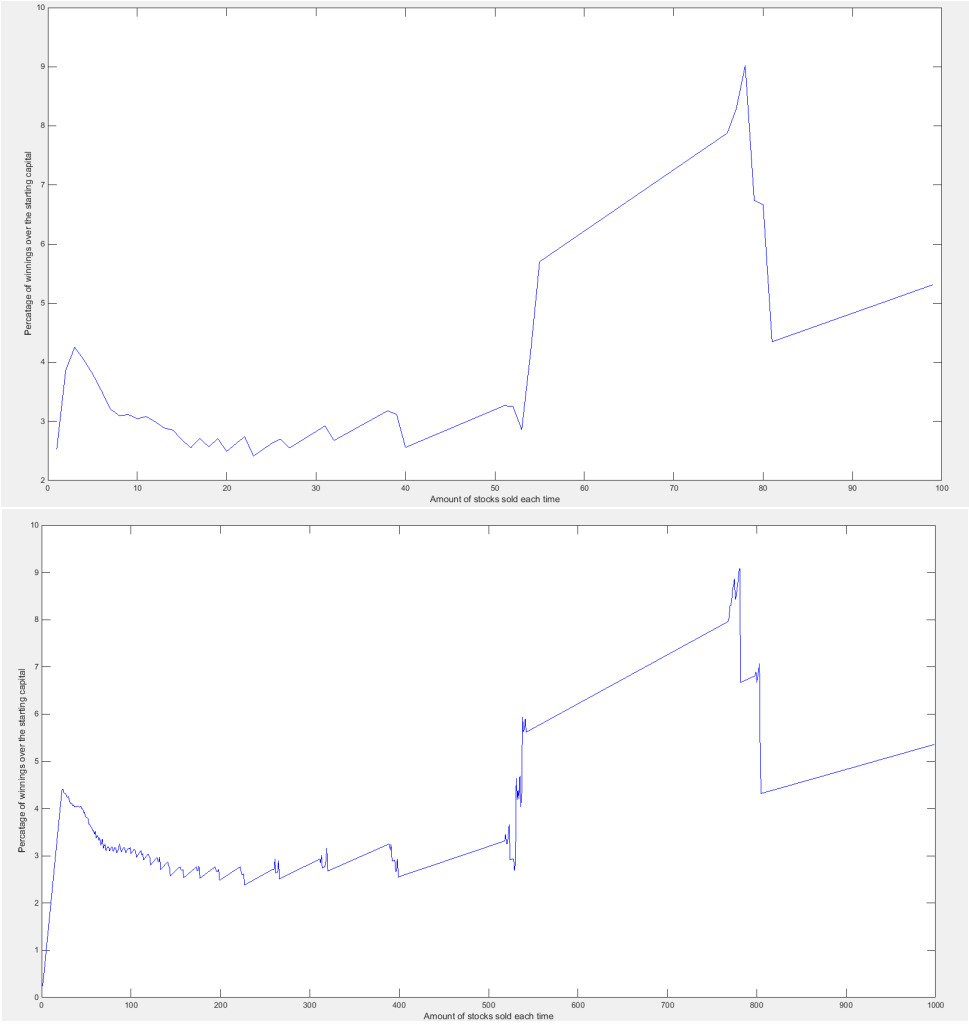


Figure 5.12: Percentage increase on capital for LUPE stock with normal and ten times more capital for a constant amount of stocks per transaction

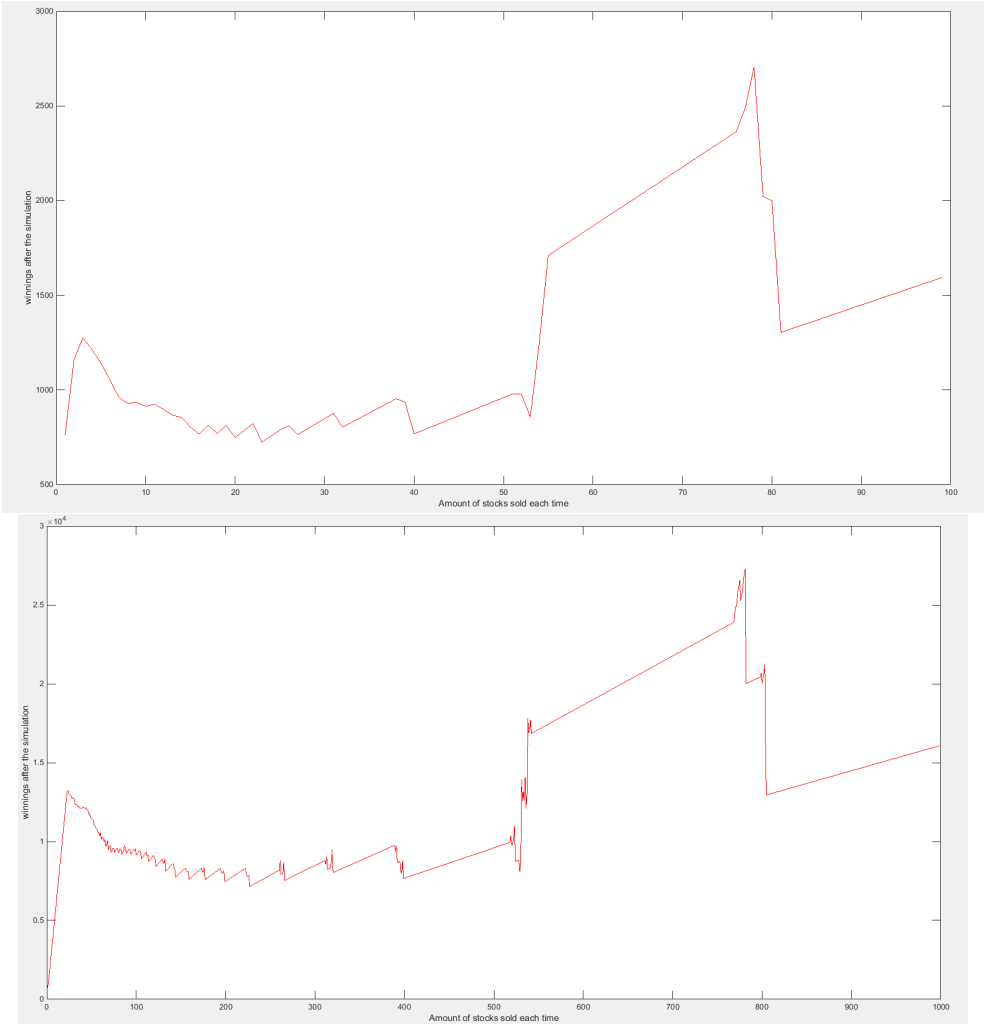


Figure 5.13: Total winnings for LUPE stock with normal and ten times more capital for a constant amount of stock per transaction

5.3.3.1 Discussion of results

We observe that there is a linear dependence between the winnings, step and initial capital. With ten times more capital and step the winnings is ten times more than the initial capital, see figure 5.13. The percentage of the winnings over the initial capital remains the same. Ten times more capital and 10 times the step, results to the same percentage of winnings. This is visible in figure 5.12

This observation is very important because it shows that it is enough to simulate only for specific initial capital as we can map the dependence of winnings between different initial capitals and different step.

5.3.4 Buying and selling the max possible number of stocks

In order to maximize the profit of our network we realized that we have to buy the max amount of stocks that we can minimize the unused capital. The amount of

stocks that we are going to buy each time is the integer fraction of the division of current capital with current stock price and the unused capital is the remainder of the division. When we are selling, we sell all the stocks we have in our possession unless we have a reduced selling occasion. We will present our results in the table below.

Stock name	Percentage Difference on winnings
ABB	0.465%
ALFA	0.528%
INVB	0.159%
SAND	0.386%
SKFB	0.764%
SAAB	0.972%
TELIA	0.822%
VOLVO	0.382%

Table 5.2: Difference on percentage of winnings using constant and dynamic step while Buying stocks

5.3.4.1 Discussion of Results

In the table 5.2 we can clearly see that buying the maximum possible number of stocks was able to increase the winnings for all the stocks. The average increase was 0.560%, smallest increase was 0.159% and the highest was 0.972%. We expected that the winnings would increase and the reason this happens is that we minimize the amount of unused capital when we buy stocks and as result our winnings increase.

5.3.5 How much historical data do we need?

We tried training our network by using different intervals to see how does the volume of data affect the training process. For each of the graphs below the training happened on a different time interval but the testing took place from 13th of February 2017 until 10th of March 2017.

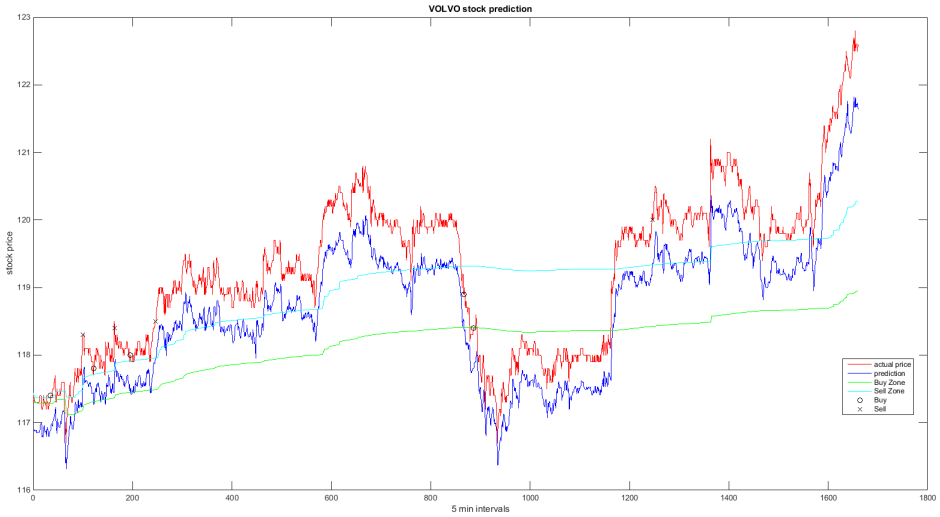


Figure 5.14: Stock Price prediction of VOLVO. Training from 23rd of November 2016 to 16th of December 2016

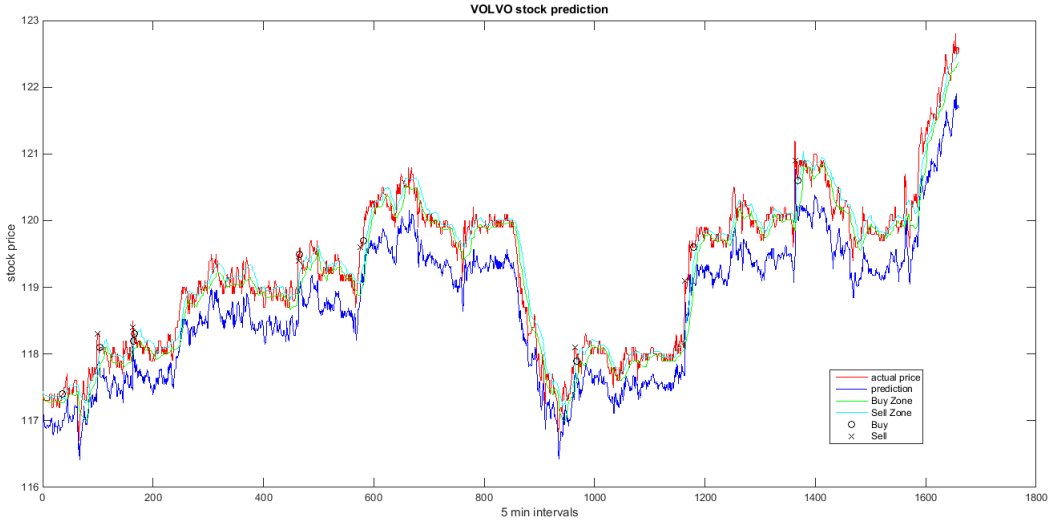


Figure 5.15: Stock Price prediction of VOLVO. Training from 23rd of November 2016 to 13th of January 2017

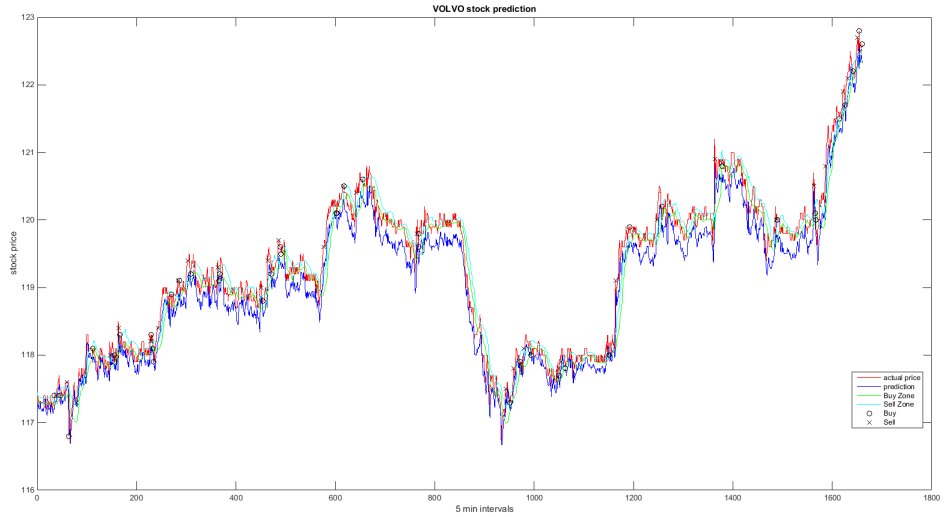


Figure 5.16: Stock Price prediction of VOLVO. Training from 1st of February 2017 to 10th of February 2017

In the Figures above we can easily see that MAPE and MXPE while testing are really big. However the errors while train and validation are really low and the neural networks converged.

In figures 5.14 and 5.15 we can see that our prediction follows the trend of actual price but the error is high. Those two experiments have one thing in common. The training set is chronologically long time before the testing set and also the stock price had a 20% increase in price in between. However the training in figure 5.16 used data just before the testing and still we had high error. The reason is that we did not use enough data for the network to generalize.

This method tries to approximate the function that models the stock market. If the train data and the test data have a huge difference in the stock price then the testing will have huge error. Also if we train the neural network with a low amount of data we will still have high error.

Furthermore in figure 5.17 we can see that MAPE is low and specifically below 0.07%. However the MXPE is high and to be precise we can identify the points in the figure where the prediction is wrong and the blue line diverges from the red line.

Finally in figure 5.18 we can see that the blue curve fits the red line almost perfectly. The main difference between figures 5.17 and 5.18 is that we used more data to train the network in figure 5.18

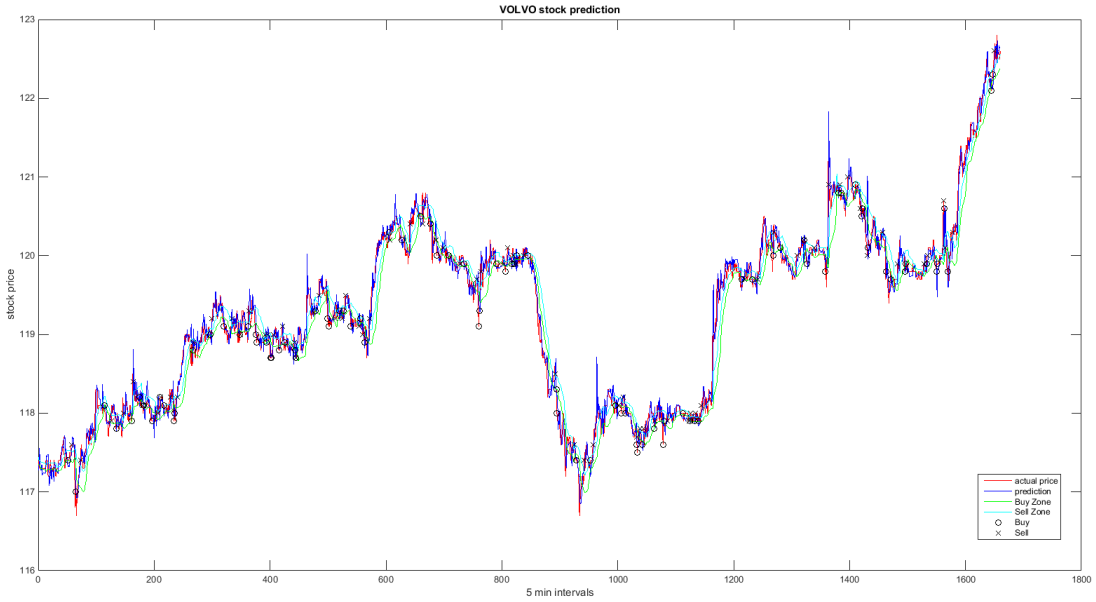


Figure 5.17: Stock Price prediction of VOLVO. Training from 16th of January 2017 to 10th of February 2017

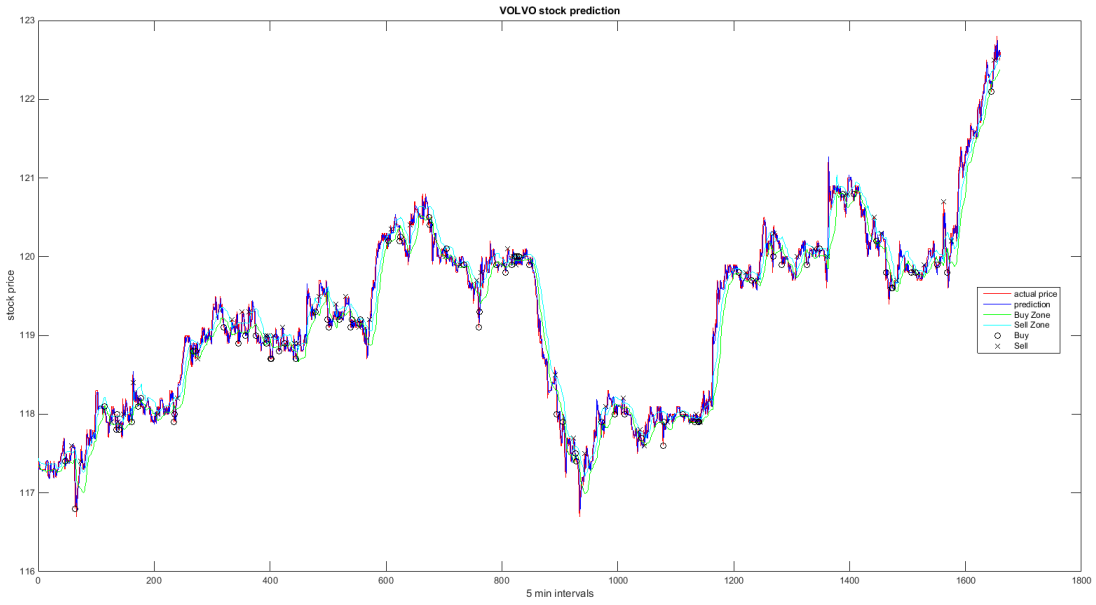


Figure 5.18: Stock Price prediction of VOLVO. Training from 10th of November 2016 to 10th of February 2017

5.3.5.1 Discussion of Results

Through the experiment we were able to conclude that the more data we have to train our network with, the better results we get. Also an important factor is the

quality of the data we have.

What we mean with that? For example if we have data for one year and the stock price increase by 50% within a single night then we will have similar results as in figures 5.14 , 5.15 and 5.16. The reason for that is when we try to approximate the behavior of a function we do it with in the range of our data. We basically have no idea what happens outside of that range because we don't have information about the behaviour of the function there. As a result when the stock price diverges from that range our trained neural network error will increase.

As a result of the observation above a new questions rise.

1. What do we do when the stock price moves out of the train range?
2. Is it possible to prevent situations like above?
3. How long can a trained neural network be used and perform with low error?

We concluded that the neural network can be used as long as the stock price remains in the trained range. The moment the stock price moves outside the trained range we will start observing increase on the testing error. We can deal with this by training a new network, add more recent data to the train set and this is something that will expand neural networks trained range and it will able to predict the price with low error. We can prevent this problem by occasionally training a new network every fixed time and adding the recent data to the new train set.

5.3.6 How often do we have train a new network?

We tried three experiments with retraining the neural network. Those are listed below:

1. Train the network once and run the simulation.
2. Train a new network at the end of every week by include new data in the train set.
3. Train a new network at the end of every day and include the new data in the train set.

In the initial training set we had data from 10th of November 2016 until 10th of February 2017. The testing took place from 13th of February until the 10th of March 2017. We were able to run this simulation only with the VOLVO stock and the main reason is the 3rd option we had to train twenty neural networks which is something that takes so much time.

We were able to see an increase of 0.3% in the winnings by retraining our network at the end of each week. However by training each day we did not see any significant increase in winnings from training each week.

The need to retrain our network comes when the testing price range diverges out

from the testing price even if it is every day, every week or every month. We are treating stock market prediction as a function approximation and when the data points are in the training range we can identify huge error. We should consider retraining our network with including more recent data in the train set only when the stock price is close to the bounds of the training range. This concept it is explained in section 5.3.5

5.4 Reinforcement Learning model and trading strategy

We managed to create a prototype of the environment and the agent using tensorflow. We also run few simulations for a couple of weeks trying to train the agent. However we didn't have the time to tune the Reinforcement Learning trading strategy so we left it as future work. There was a high amount of hyper-parameters that we needed to optimize and due to time limitation we decided we will look it in the future. The preliminary results were promising and we believe that this model has the potential to surpass any other model came up with.

6
Conclusion

6.1 General Comments

We were able to come with a way to successfully predict the stock market and in combination with a good trading strategy we were able to profit from stock trading using historical data. The reason we used historical data and not real time data for testing was time efficiency but also the ability to compare models and trading strategies using the same testing data.

How would the model behave with real-time data?
We treated our historical data as real time data. We can use the same methods and be able to predict the stock price in real time. We can achieve this by collecting the transactions in real time and converting them into 5-min intervals and just passing them forward to our network, point by point. One of the goals of this thesis was that we should be able to utilize the stock market on real time and all the simulations were done in way that would make it easy to transition from historical to real time data.

Also the prediction models can easily work with different time intervals as well. We choose 5-min intervals as a representation of short term-trading. Just by changing the way we process our data input to a different time interval we can train a network that predicts the stock price on a different time gaps. We can train our network based in 2-min, 10-min, 30-min intervals or even try 1-hour or 2-hour intervals and it should be able to predict the stock price with the same accuracy.

The average winnings in our experiments were about 8% increase of the initial capital with in a month of testing. As an investment it can be characterized as really profitable. However the neural network cannot predict sudden changes in the price that happen during the time that the stock market is closed. An example is when a company or their direct competitors announce their term results. Those kinds of events can skyrocket the stock price or make it lose considerable value.

6.2 Limitations

The main limitation we had during this thesis was the time constraints due to the lack of available data. We had to wait at least three months to run experiments as we had to collect our own data. Also trying to train a neural network takes quite long. If we were lucky a neural network would converge in a couple of hours but sometimes could take up to 15 hours. Especially in the Reinforcement Learning (RL) models training would take quite a few days. The whole process it was a trial and error in order to come up with the optimal hyper parameters and we had to train numeral different networks just to be able to select them.

Another limitation we have to face was the computing power. To train complex models like Reinforcement Learning and Recurrent neural networks, someone would need GPU clusters in order to speed up the training process. However we did not have this computing power and we were not able to investigate in depth those models.

6.3 Future Work

Finally we talk about the work we can do in the future expanding the work done in this thesis.

6.3.1 Test our existing models with more data

We have to test the existing methods with more data as the keep coming. We want to ensure that the results we got it is not just a random event that happened as result of the time period. We have to test with even more data as time passes and make sure that our model can generalize.
An other thing we can do is check how our model works with stocks outside the OMSX30 index. We can try train and evaluate out model with stocks that belong to smaller companies that do not have as many transactions as the big ones. In that case we will be able to see if we can expand our work to other stocks and maybe even other stock markets as well.

6.3.2 Advanced decision based trading strategy using different artificial neural networks

The concept behind this idea is that we will have few neural networks trained in different time intervals. For example, we can have the prediction of the stock price in 5-min, 10-min and 30-min intervals. Using this information we can decide when is the best time to place our action. If we know how the stock will move with in the next thirty minutes we will be able to increase our profit even more. The more information we have, the more profit we can achieve.

6.3.3 Reinforcement Learning tuning

Reinforcement learning is one of the most promising areas of machine learning. The perfect way that stock market can be modeled with RL makes it one of the best ways to maximize our profit. The initial results were promising, although much more work is needed for the model to be ready for use. The time and the computing power limitation were the main reason that we left Reinforcement learning as future work.

6.3.4 Recurrent neural network mode.

We can also explore a model using recurrent neural networks. Stock market can be modeled as time series and traditional methods tried to approximate them. Most famous traditional models are the Monte Carlo models. It was shown by studies that time series can be approximated effectively using recurrent neural networks. Some applications of them are text classification, text-to-speech, speed-to-text and even translation from a language to another. We could easily use those models and try to approximate stock movement by treading stocks as a time-series problem.

Time limitations and computational power constrains were the main reasons we did not have time to develop a prototype and test a recurrent network model. In contrast to feed forward networks, recurrent neural networks are more complicated and computationally more expensive to compute. We believe that it is possible to increase the performance of our presented models but we might check this out in the future.

Bibliography

[1] Information about tensorflow.

[2] G. Appel. *Technical Analysis: Power Tools for Active Investors*. FT Press, illustrated edition edition, 2005.

[3] E. M. Azoff. *Neural network time series forecasting of financial markets*. John Wiley & Sons, Inc., 1994.

[4] A. Bernemann, R. Schreyer, and K. Spanderen. Pricing structured equity products on gpus. In *High Performance Computational Finance (WHPCF), 2010 IEEE Workshop on*, pages 1–7. IEEE, 2010.

[5] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[6] R. Breen. The accelerated binomial option pricing model. *Journal of Financial and Quantitative Analysis*, 26(02):153–164, 1991.

[7] N. Chriss. *Black Scholes and beyond: option pricing models*. McGraw-Hill, 1996.

[8] J. C. Cox, S. A. Ross, and M. Rubinstein. Option pricing: A simplified approach. *Journal of financial Economics*, 7(3):229–263, 1979.

[9] B. Detollenaere and P. Mazza. Do japanese candlesticks help solve the trader's dilemma? *Journal of Banking & Finance*, 48:386–395, 2014.

[10] J. H. George B. Thomas, Maurice D. Weir. *Thomas' Calculus, Multivariable (12th Edition)*. 12 edition, 2009.

[11] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[12] P. Goodwin and R. Lawton. On the asymmetry of the symmetric mape. *International journal of forecasting*, 15(4):405–408, 1999.

[13] S. Haykin. *Neural Networks - A Comprehensive Foundation, Second Edition*. Prentice Hall, 2 edition, 1998.

[14] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] A. Lapedes and R. Farber. How neural nets work. In *Proceedings of the 1987 International Conference on Neural Information Processing Systems*, pages 442–456. MIT Press, 1987.

[16] B. G. Malkiel. *A random walk down Wall Street*. W. W. Norton, revised and updated edition, 1999.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

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

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[20] R. Sharda and R. B. Patil. Connectionist approach to time series prediction: an empirical test. *Journal of Intelligent Manufacturing*, 3(5):317–323, 1992.

[21] D. Srinivasan, A. Liew, and C. Chang. A neural network short-term load forecaster. *Electric Power Systems Research*, 28(3):227–234, 1994.

[22] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[23] J. L. Ticknor. A bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications*, 40(14):5501–5506, 2013.

[24] W. J. Wilder. *New Concepts in Technical Trading Systems*. Trend Research, 1 edition, 1978.

[25] G. Zhang, B. E. Patuwo, and M. Y. Hu. Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1):35–62, 1998.

A

Appendix 1