BACHELOR THESIS IN MATHEMATICS / APPLIED MATHEMATICS

Single asset trading: a recurrent reinforcement learning approach

by

Marko Nikolić

Kandidatarbete i matematik / tillämpad matematik

DIVISION OF APPLIED MATHEMATICS
MÄLARDALEN UNIVERSITY
SE-721 23 VÄSTERÅS, SWEDEN

Bachelor thesis in mathematics / applied mathematics

Date:
2020-04-10

Project name:
Single asset trading: a recurrent reinforcement learning approach

Author:
Marko Nikolić

Supervisor:
Rita Pimentel

Reviewer:
Doghonay Arjmand

Examiner:
Ying Ni

Comprising:
15 ECTS credits

Course code:

MMA390

"In theory, there is no difference between theory and practice. In practice, there is"

Benjamin Brewster, 1882

Acknowledgements

I would like to express my deepest gratitude and thanks to the supervisor Rita Pimentel for her valuable comments, eye for a detail, guidance, support and most importantly for always setting the bar higher and suggesting improvements. Rita has truly been there every step of the way and always been available. I have truly been lucky to have her as a supervisor without whom this thesis would not been possible. Furthermore, all the people that have helped and made the thesis possible like the reviewer Doghonay Arjmand for the valuable comments and suggestions. Finally I would like to thank the examiner Ying Ni for the feedback and help during the process.

Additionally, I would like to send my appreciations and love to my amazing mother for her love, unwavering support and guidance. That resulted in me being in this position and me being the man I am today.

I would also like to thank my wonderful, loving and supporting girlfriend Sofia.A Kohler for bearing with me for some reason all these years and giving me the drive, support and friendship during our time and that has truly made me a better person as a result.

Marko Nikolic, April 16, 2020

Abstract

Asset trading using machine learning has become popular within the financial industry in the recent years. This can for instance be seen in the large number of daily trading volume which are defined by an automatic algorithm. This thesis presents a recurrent reinforcement learning model to trade an asset. The benefits, drawdowns and the derivations of the model are presented. Different parameters of the model are calibrated and tuned considering a traditional division between training and testing data set and also with the help of nested cross validation. The results of the single asset trading model are compared to the benchmark strategy, which consists of buying the underlying asset and hold it for a long period of time regardless of the asset volatility. The proposed model outperforms the buy and hold strategy on three out of four stocks selected for the experiment. Additionally, returns of the model are sensitive to changes in epoch, m, learning rate and training/test ratio.

Contents

Abbreviations

MVO	**M**ean **V**ariance **O**ptimizer
ML	**M**achine **L**earning
RL	**R**einforcement **L**earning
IA	**I**ntelligent **A**gent
UL	**U**nsupervised **L**earning
SL	**S**upervised **L**earning
FX	**F**oreign **E**xchange
S&P 500	**S**tandard **P**oor's **500**
NASDAQ	**N**ational **A**ssociation of **S**ecurities **D**ealers **A**utomated **Q**uotations
MSFT	**M**icrosoft
MS	**M**organ **S**tanley
DB	**D**eutsche **B**ank
NVDA	**N**vidia
JPM	**JP M**organ **Chase**
AXP	**A**merican **E**xpress
BRK	**B**erkshire **H**athaway
GE	**G**eneral **E**lectric
CV	**C**ross **V**alidation
RSI	**R**elative **S**trength **I**ndex

List of Figures

Chapter 1

Introduction

"What we want is a machine that can learn from experience." — Alan Turing, 1947

1.1 Background

Stock market prediction and behavior have been studied for years by individual investors, financial institutions and governments. There is much research on the subject but the results are rarely replicable [1]. There are hypothesis that are widely believed and debated. An example is the efficient market hypothesis coined by the famous Eugene Fama [2], which states that securities markets are extremely efficient in reflecting the available information about a specific asset or the stock market as a whole. Meaning that investors would be better of by investing in the benchmarks[1] than allocating resources themselves or paying a fund manager to do so.

Whenever there exists more than one option there will always be an optimization problem. The stock market is not an exception and Harry Markowitz realized this in the early 1950s when he wrote his PhD thesis on Portfolio Selection [3]. During this time handful people were aware of the portfolio optimization problem mainly Henry Mann (1943) [4] (cited in [5]) and Alfred Martin (1955) [6] (also cited in [5]). However, it is widely accepted that Harry Markowitz is the father of modern portfolio theory [5]. Prior to this point there was no considerations to risk to reward measures and the portfolio selection was focused on value investing[2]. Harry Markowitz opened the door for mathematics into the portfolio selection and portfolio management. Investors saw the benefits of mean variance optimization (MVO) even if the MVO had issues such as overweight's, constraints, transaction cost and a need for expected returns [5]. However, with all the

[1]A benchmark is a standard for which the performance of the investors, investment managers, mutual funds or individual security can be compared. These are usually big market segments of a specific market or combination of one or more segments (example of the most famous are S&P500, Russel 2000, NASDAQ). However, trading models use the underlying security that the model is trading as the benchmark. In this thesis the benchmarka are the respective stocks.

[2]Value investing is an investment strategy of picking assets that are trading below their book value.

shortcomings of the MVO investors saw the benefits and the simplicity in the model which became popular at that time. The MVO changed the investors tinkling that lead to an increased research in the portfolio theory field producing new models such as Black-Litterman [7] and Markowitz 2.0 [8].

Over the last decades the use of internet and the quantity of information about all factors related to a specific asset has increased the amount of data available[3]. This has created a new challenge for investors and institutions, such as handling big data[4]. The common problems with big data are dealing with the growing data, generating insights in a timely manner (common with high frequency trading[5] [1]), validating and securing data[6]. In comparison to two decades ago nowadays an investor can in a very short time download large data sets for 30-50 years about each specific asset. That includes financial statements, yearly, monthly, weekly, daily or hourly returns and even down to a second. The large data sets have given raise to high frequency trading and the need for automated systems. Particularity, machine learning (ML) techniques to handle the large data sets and to try to analyse the patterns in the data and predict future behavior with certain level of confidence. Michael Rechenthin [1] states that 55% of the trading volume on a given day (August 2014) is model based.

ML algorithms automatically build a mathematical model using data, to gain the capability to make predictions or decisions with minimal or no human intervention [9]. The increased use of ML and application to various fields has increased over the past years and there are no signs of slowing down [9]. Therefore, the need to follow the evolution of portfolio management and include ML techniques such as reinforcement learning (RL), supervised learning (SL) or unsupervised learning (UL) have become integral. This has become even more crucial since more and more of the daily trading volume is model based [1].

When thinking about RL learning one can consider an infant who waves its arms, and observes the environment, at that point the infant has no explicit teacher. But the infant has a direct connection to the environment through its sensory systems. Applying this connection generates a wealth of information about cause and effect, what to do in

[3]Over the last two years alone 90 % of the data in the world was generated (March 28, 2019). Source: https://blazon.online/data-marketing/how-much-data-do-we-create-every-day-the

[4]Big data is a concept that describes the large volumes of data, both structured a and unstructured.

[5]As highlighted in [1], "A model that needs thirty minutes to arrive at a prediction that is needed every minute in the future is of little value".

[6]Source: https://www.datamation.com/big-data/big-data-challenges.html

order to achieve a goal and consequence of action [10]. RL works in similar way, the models make a decision in its environment and based on the outcome of that decision there is either a positive reward or a negative reward. The cumulative rewards creates a set of information that simplifies future decision making for the models [9]. The key characteristic of RL is the existence of an intelligent agent[7] (IA) that has the ability to learn good behavior through experience. Meaning that the IA modifies or acquires new skills and behavior incrementally over time [9]. The only requirement is the ability to interact with the environment and leads to accumulation of information [9]. RL applies to problems that experience frequent decision making that relay on past experience [9]. The characteristics of the RL make it game changing for portfolio management and one of the reasons for recent popularity in the area of algorithmic trading [1].

This thesis focuses on the area of recurrent RL which is a subsection of RL. Recurrent meaning that the previous output are fed back into the models as part of the new input. The recurrent RL framework introduces a simple and elegant approach of problem presentation, avoiding Bellman's curse of dimensionality and provides advantages in efficiency [11]. The recurrent RL can be used to optimize performance functions such as return function, wealth or risk adjusted performance ratio like Sharpe ratio [11]. This thesis uses the recurrent RL trading system proposed by Moody and Saffel [11] and optimizes the Sharpe ratio in order to outperform the benchmark (ie. the respective stocks) on the basis of total return. Furthermore this thesis evaluates the impact of different training/test ratio on the performance of the trading system.

1.2 Literature review

Mody and Wu [12] introduced recurrent RL as an application for the markets in 1996 but the initial pioneers of recurrent reinforcement were Farley and Clark [13, 14] (cited in [11]). Mody and Saffell [15] and Moody, John E., et al. [16] later used recurrent RL model and applied it to currency markets and S&P 500 respectively. Moody, John E., et al.[15] showcased that RL model provides an elegant and significantly more efficient method for training trading systems when transactions cost is considered as a factor than other standardized supervised learning techniques. The results over the chosen

[7]IA is an algorithm that is autonomous in its actions, which are based upon the environment, user input and expectations. The IA can be used to autonomously gather information and perform action. Additionally, an IA can also learn or use the past experience in achieving their future goals which might be simple or complex.

time period were outperformed S&P 500 and proved that there is predictability in the S&P 500 and the currency market.

The results in [15] also illustrated that the recurrent RL model when compared to the Q-learning model outperformed the Q-learning model and one of the reasons for the outperformance was the high frequency of trades initiated by the Q-learner that in turn lead to higher transaction cost for the model. Later in 2001 Moody and Saffell [11] further developed recurrent RL model and compared it again with Q-learning, the study showcased the same conclusion as before, that the recurrent RL was more efficient than Q-learner and that S&P 500 contained predictability.

This thesis is based and expands the work of Moody and Saffell [11]. The main difference between the original paper [11] and this thesis is the approximation of actual position with respect to the weights (explained in chapter 2) where Moody and Saffell [11] use online learning[8] to approximate the position with respect to weights with the help of previous position with respect to the weights, and thereby making the trading model stochastic. Whereas, this thesis uses batch learning that uses the entire training data set at once to generate the best predictor. Secondly, the original paper [11] derived the differential Sharpe ratio to complement the online learning approach, while this thesis uses the standardized Sharpe ratio as the utility function. Lastly, Moody and Saffell [11] run the simulations on the Foreign Exchange (FX) and S&P 500 under a specific time period, learning rate, transaction cost and training/test ratio. Additionally, no motivation is given for the selection of the parameter values [11]. Therefore, there exists the possibility of overfitting the model and settings so that it becomes one case specific. This thesis implements different ranges of all the mentioned above and runs the simulation on variety of different stocks with different characteristics, in order to get a broader understanding of the models ability. The differences between this thesis and the original work [11] are mainly because there exists room for improvement on the original work such as mention above and the need for differentiation between the original work and this thesis.

Timmermann and Granger [17] (cited in [1]) argued that the lack of published works in RL models is due to the very little incentive for publishing such models in the academic literature. Indeed, there is much higher incentive to sell the models to trading firms and

[8]Online learning is used when it is computationally infeasible (large data set) to train on the entire data set. Therefore, online learning uses sequential order an updates the best predictor at each step.

receive a monetary gain. Furthermore, Timmermann and Granger [17] also consider the possibility of a "file drawer" bias in the published work due to the issues in publishing results that are barley statistically significant. However, the markets are partially driven by human emotion and exhibit large degree of error in contrast to the efficient market theory Eugene F. Fama [2].

An opposite side of the efficient market hypotheses argues that the market is inefficient and that there exists predictability in the stock market, Michel.D Rechenthin [1] shows this fact by using 22 million stock transactions. He continues, to state that widespread adoption of a trading strategy is enough to affect the price of the market and to eliminate the benefit of that model. Therefore, for these reasons it is best for the traders, banks, and trading firms to keep their models hidden to avoid widespread adoption. This fact needs to be under consideration if and when implementing a trading model. Because the theoretical results of the model and the real world test can differ because that strategy already exists in the real world. By adding additional participants with the same strategy leads to very different results in the theory and real world application.

RL have been applied in many other areas besides Finance. For example, Giannoccaro and Pontrandolfo [18] used RL to manage inventory decisions in all stages of a supply chain, in doing so optimizing the performance of the whole chain. Other notable examples where RL was applied successfully are elevator schedule [19], a space-shuttle payload scheduler [20], resource management[21], traffic signal control[22], robotics training [23], online web system auto-configurations[24] and chemical reactions [25].

1.3 Problem formulation and aim of the paper

The aim of this thesis is to find a RL model, that has the ability to outperform the respective benchmark under a set of specific constraints and assumptions. For this purpose, the following questions are of interest:

1. Does the chosen RL model outperform the respective benchmark under specific conditions and assumptions?
2. How does the performance of the chosen RL model depend on the training set?

These questions serve as a red line of this thesis in trying to understand the chosen RL model.

1.4 Outlining the method

The first step is to define and analyse the techniques of recurrent RL such as gradient ascent. Secondly, a model is created and fitted on financial time series datasets. The data is split in two parts one for training and another for testing, meaning that an "optimal" representatives of the overall data is chosen for training and then the trained IA is allowed to trade by itself on the remaining data available. In that process, there is selection of four stocks out of a set of eight stocks that are considered for the implementations. This is done, because these financial time series should incorporate different possible behaviors, such as flatness periods, rapid growth and decline or unexpected jumps.

1.5 Disposition of the paper

The thesis has the following outline: first chapter contains introduction, background and literature review of the problem. The second chapter is mainly devoted to the derivation and explanation of the model. The third chapter presents the data and the results of the implementations. The forth chapter gives an conclusion to the findings.

Chapter 2

Reinforcement learning model

"Patterns of price movement are not random. However, they're close enough to random so that getting some excess, some edge out of it, is not easy and not so obvious thank God." — James Harris Simons

Trader's or investor's main objective is to optimize the economic utility function, profit function, performance function or risk adjusted return of their model. This section presents and thoroughly describes the underlying trading model of the study and the derivations upon which the trading model is based. Starting by introducing the structure of the trading model, profit measure and wealth measure of the model, the utility function and the Sharpe ratio. Finally, the recurrent RL model is presented, explained and derived.

2.1 Structure of the trading model

The following model is proposed by Moody and Saffell [11]. This model trades fixed position weights in one asset or security[1]. Methods proposed in this section can be applied and generalized to increasingly complex IA such as varying trading position, continuously trading or managing multiple securities at the same time.

The model under consideration is only allowed to be long, short or neutral in its positioning, which is represented by $F_t \in \{1, 0, -1\}$. Initiation of the long position occurs when the model buys a specific amount of shares in a security, meaning that $F_t = 1$, while being neutral is not taking any decision at time t, i.e. $F_t = 0$. Moreover, shorting a security means borrowing shares and selling them to a third party, making money in the process if the security decreases in value or taking a loss if the security gains in value, in this case $F_t = -1$. In addition to paying a fee for the privilege of borrowing the shares, the short seller also has to pay the securities dividends on the stocks held. However, in

[1]Security is a certificate that has monetary value and is tradable. Examples of securities are stocks, bonds, debt securities and various derivatives.

this thesis this is not taken into consideration, i.e. it is assumed that borrowing stocks is free of charge.

The time series of the stock price being traded is denoted by $\{z_t : t \geq 0\}$. The actual position at time t is F_t and the position is entered or reallocated at the end of each period t. Meaning that a trade is possible at the end of each period. The trading costs are nonzero, which restrict the model from excessive trading. The return R_t (defined in Equation 2.6) is realized at the end of each period $(t-1, t]$ and taking into account profit or loss from the position F_{t-1} and the transaction cost that occurred at time t due to the position changes from F_{t-1} to F_t.

Properties that make this trading model interesting is the ability to incorporate the transaction costs, market impact and taxes into the decision making. The model must have internal information of the current state and for those reasons must be recurrent [11]. Based on those statements the function F is defined as

$$\begin{aligned} F_t &= F(\boldsymbol{\theta_t}; F_{t-1}, I_t), \text{where} \\ I_t &= \{z_t, z_{t-1}, z_{t-2}, ...; y_t, y_{t-1}, y_{t-2}, ...\}, \end{aligned} \tag{2.1}$$

$\boldsymbol{\theta_t}$ represents the (learned) model parameters at time t and I_t represents the information set at time t in which z_t, z_{t-1}, z_{t-2} represents the present and past values of the time series and an arbitrary number of other external variables that might have an impact on the model y_t [11]. A model which only allows long and short positions has the following definition:

$$\begin{aligned} F_t &= sign(uF_{t-1} + v_0 r_t + v_1 r_{t-1} + ... + v_m r_{t-m} + w), \text{where} \\ r_t &= z_t - z_{t-1} \end{aligned} \tag{2.2}$$

with r_t representing the absolute daily returns of the model parameters $\boldsymbol{\theta_t}$, and $\boldsymbol{\theta_t}$ is represented by the weights $\{u, v_i, w\}$ where $i = 0, 1, 2, .., m$. Parameter m represents the number of time series data points which are considered in the trading model. The model definition describes a discrete trading model which may imply problems because of the need to be differentiable. This problem is a result of the *sign* having discrete values such as $F_t = \{1, 0, -1\}$, but if *sign* is replaced with *tanh* this makes the model continuous and differentiable, therefore resulting in the following definition[11]:

$$\begin{aligned} F_t &= tanh(uF_{t-1} + v_0 r_t + v_1 r_{t-1} + ... + v_m r_{t-m} + w) \\ F_t &= tanh(\boldsymbol{\theta_t^T x_t}), \end{aligned} \tag{2.3}$$

with

$$\begin{aligned} \boldsymbol{\theta_t^T} &= [u, v_0, v_1, v_2, ..., v_m, w] \\ \boldsymbol{x_t^T} &= [F_{t-1}, r_t, r_{t-1}, ..., r_{t-m}, 1], \end{aligned} \tag{2.4}$$

where $\boldsymbol{x_t}$ represents the input vector[2].

2.2 Profit and wealth of the trading model

Every investor and trader need to have in the consideration the additive profit and the wealth at different time periods. Therefore, additive profits are imperial to consider whenever the trades are fixed number of securities or shares z_t. Furthermore, $r_t = z_t - z_{t-1}$ represents the difference in value of the stock from period $t-1$ to t, r_t^f is the risk free interest rate and δ is the transaction cost associated with trading. If $F_t = F_{t-1}$ then $\delta = 0$ otherwise there exists a penalty proportional to the difference value of the stock. The accumulated additive profit over all the trading periods T when the trading position size $\mu > 0$ can be represented in the following way [11]:

[2]There is a stochastic extension of this model that includes a noise ϵ_t variable into F_t. More details can be found in [11].

$$
\begin{aligned}
P_t =& \sum_{t=1}^{T} R_t \\
&\text{with} \\
R_t \equiv& \mu \left\{ r_t^f + F_{t-1}(r_t - r_t^f) - \delta |F_t - F_{t-1}| \right\},
\end{aligned}
\tag{2.5}
$$

where μ represents the maximum number of shares possible per transaction. Moreover, it is assumed that $F_T = F_0 = 0$. The equation (2.5) when $F_t = F_{t-1} = 1$ earns positive return if $r_t - r_t > 0$ and earns r_t^f if $r_t = r_t^f$. Negative return occur when $r_t - r_t^f < 0$ given that $|r_t - r_t| < r_t^f$. However, if shorting is taken into consideration (ie $F_t = F_{t-1} = -1$) the return is positive when $r_t - r_t < 0$ and returns r_t^f (assuming risk free rate is positive) when $r_t = r_t^f$. Negative returns are a result of $r_t - r_t^f > 0$ given that $|r_t - r_t^f| > r_t^f$. In both cases $F_t = F_{t-1} = 1$ and $F_t = F_{t-1} = -1$ the transaction cost can be disregarded since the positioning remains the same and $\delta = 0$, but if $F_t \neq F_{t-1}$ then the transaction cost needs to be taken into account. If, the risk free interest rate is null i.e. $r_t^f = 0$ this leads to a simplification in the (2.5) expression:

$$
R_t = \mu \left\{ F_{t-1} r_t - \delta |F_t - F_{t-1}| \right\}. \tag{2.6}
$$

The investor and trader are interested in the wealth of the portfolio at time t, which is defined as $W_T = W_0 + P_T$ [11]. Additionally, multiplicative profits are suitable when a fraction of accumulated wealth is invested meaning $v > 0$ is invested either long or short. In this case the percentage daily returns are defined as $\hat{r}_t = \frac{z_t - z_{t-1}}{z_{t-1}}$. However, if short sales are disallowed and the leverage factor v is set at $v = 1$, the wealth at time T is:

$$
\begin{aligned}
W_T =& W_0 \prod_{t=1}^{T} \{1 + R_t\}, \text{where} \\
\{1 + R_t\} \equiv& \{1 + (1 - F_{t-1}) r_t^f + F_{t-1} \hat{r}_t\} \{1 - \delta |F_t - F_{t-1}|\}.
\end{aligned}
\tag{2.7}
$$

Similarly, to the additive profits Equation (2.5), the simplification of (2.7) becomes:

$$\{1 + R_t\} = \{1 + F_{t-1}\hat{r}_t\}\{1 - \delta|F_t - F_{t-1}|\}, \tag{2.8}$$

if $r_t^f = 0$.

2.3 Utility function

Trading models can be optimized in variety of ways such as minimizing performance functions, such as the risk (volatility) of the portfolio or the transaction costs. Trading models can also be optimized in an effort to maximize the performance of functions such as profit, utility function or performance ratios such as the Sharpe ratio S_t [26]. In this thesis the performance criteria that is considered is the Sharpe ratio, S_t

After a sequence of trades in time periods $t \in \{1, 2, ..., T\}$ a generalized expression can be defined as $(S_1, S_2, ..., S_T)$ [11]. In the optimization of the trading model the interest is in the marginal increase in performance at each time period with respect to R_t. Additionally, it is important to note that S_t depends on the current trading return R_t while S_{t-1} does not. The trading strategy is to derive the Sharpe ratio differential ΔS_t at each time step, to capture the marginal utility of the trading return R_t at each time step [11].

2.3.1 The Sharpe ratio

The Sharpe ratio is a widely used performance ratio in modern portfolio theory [11]. It was developed by William Sharpe and presented in his Mutual fund performance publication in 1966 [26]. It represents the risk premium[3] by unit of standard deviation σ_{R_t} (the total risk for the same period). The Sharpe ratio is a risk adjusted return measure that gives the investor a better overview of the profits associated with specific risk taking activities. In this thesis the risk premium is given by R_t (defined in Equation 2.6).

[3]Risk premium is usually defined by $E[R_t] - r_t^f$.

$$\begin{aligned} S_T &= \frac{E[R_T]}{\sigma_{R_T}} \\ &= \frac{E[R_T]}{\sqrt{E[R_T^2] - (E[R_T])^2}} \\ &= \frac{A}{\sqrt{B - A^2}}, \end{aligned} \tag{2.9}$$

where $A = \frac{1}{T}\sum_{t=1}^{T} R_t$, $B = \frac{1}{T}\sum_{t=1}^{T} R_t^2$.

Modern portfolio theory states that adding assets to a portfolio that have low correlation to each other increases the diversification (and thus it decreases the risk σ_{Rt} of the portfolio) without the sacrifice of the return R_t [3]. Therefore, concluding that adding diversification would increase the Sharpe ratio in comparison to similar portfolios that have lower diversification. Additionally, Sharpe ratio gives a good overview in whether the excess return[4] were due to smart decisions or if the excess return was due to higher risk σ_{R_t}. Concluding that, excess return is acceptable as long as that increase in excess return did not come from additional risk. There exists a possibility of the Sharpe ratio being negative and in that case, it is either return is negative or the risk-free rate is greater than the return of the portfolio.

There are no perfect performance ratios without limitations and that statement is also true for Sharpe ratio. Sharpe ratio is widely accepted and adopted but nonetheless it has limitations. Firstly, Sharpe ratio uses σ_{R_t} as a proxy for the entire portfolio risk, thereby assuming that the returns are normally distributed, which is rarely the case. Secondly, choosing specific periods for the analysis of Sharpe ratio in order to get the best potential. This can be done by increasing the time period of measurement, meaning σ_{R_t} is higher in one day than it is if calculated over a week. Therefore, by increasing the measurement intervals one can decrease the σ_{R_t} and possibly increase the Sharpe ratio if the risk premium doesn't drop too much. Lastly, and the most important, the positive spikes in return have a negative effect on the Sharpe ratio. Considering that investors do not mind the positive spike no matter how large but in the event of large positive spikes, σ_{R_t} increases as well and suppresses the Sharpe ratio. In conclusion the Sharpe ratio works better for normally distributed returns and low variation in the volatility.

[4]Excess return is $E[R_t] - R_M$ where R_M is the benchmark.

It is understandable that investors would like to protect or avoid the negative spikes completely but it's concerning when the ratio is penalizing positive outcomes such as positive spikes.

2.4 Recurrent RL model

This section presents the recurrent RL algorithm and derives the algorithm. Gradient ascent is a first-order iterative optimization algorithm that finds the local maximum point of the differentiable function. In order to find the local maximum the gradient ascent takes steps proportional to the positive gradient of the function. However, if the goal is to find the local minimum point, then the gradient takes the negative steps proportional to the gradient of the function. The gradient algorithm was proposed by Cauchy in 1847 [27]. The interest of this thesis is to locate the local maximum of the Sharpe ratio with gradient ascent algorithm.

The algorithm derives the Sharpe ratio function with respect to $\boldsymbol{\theta_t}$. The algorithm is optimized by continuously computing the Sharpe ratio differential in order to adjust future trading decisions based on the results. The learning rate determines how fast the local maximum is reached, a high learning rate can lead to overshooting the highest point and a very low learning rate can lead to slow convergence to the highest point. The formulation of the learning rate (p) is:

$$\Delta\boldsymbol{\theta} = p\frac{\partial S_T}{\partial\boldsymbol{\theta}} \tag{2.10}$$

In order to achieve the maximum Sharpe ratio the following derivation (2.11) is proposed [11]:

$$\begin{aligned}\frac{\partial S_T}{\partial\boldsymbol{\theta}} &= \frac{\partial}{\partial\boldsymbol{\theta}}\left\{\frac{A}{\sqrt{B-A^2}}\right\} \\ &= \left\{\frac{\partial S_T}{\partial A}\frac{\partial A}{\partial\boldsymbol{\theta}} + \frac{\partial S_T}{\partial B}\frac{\partial B}{\partial\boldsymbol{\theta}}\right\},\end{aligned} \tag{2.11}$$

where the partial derivative of the Sharpe ratio is taken with respect to θ. The second line of the equation (2.11) represents the partial derivative taken of Sharpe ratio and the result of the chain rule applied. Taking $\frac{\partial R_t}{\partial \boldsymbol{\theta}}$ outside of the bracket in (2.11), the equation becomes:

$$\begin{aligned}\frac{\partial S_T}{\partial \boldsymbol{\theta}} &= \sum_{t=1}^{T}\left\{\frac{\partial S_T}{\partial A}\frac{\partial A}{\partial R_t} + \frac{\partial S_T}{\partial B}\frac{\partial B}{\partial R_t}\right\}\frac{\partial R_t}{\partial \boldsymbol{\theta}} \\ &= \sum_{t=1}^{T}\left\{\frac{\partial S_T}{\partial A}\frac{\partial A}{\partial R_t} + \frac{\partial S_T}{\partial B}\frac{\partial B}{\partial R_t}\right\}\cdot\left\{\frac{\partial R_t}{\partial F_t}\frac{\partial F_t}{\partial \boldsymbol{\theta}} + \frac{\partial R_t}{\partial F_{t-1}}\frac{\partial F_{t-1}}{\partial \boldsymbol{\theta}}\right\}.\end{aligned} \tag{2.12}$$

Thereafter, expanding the $\frac{\partial R_t}{\partial \boldsymbol{\theta}}$ and we get the equation (2.12). The next step is to derive the eight derivatives from (2.12), starting with the left bracket and $\frac{\partial S_T}{\partial A}$:

$$\begin{aligned}\frac{\partial S_T}{\partial A} &= \frac{\partial}{\partial A}\left\{\frac{A}{\sqrt{B-A^2}}\right\} \\ &= \frac{1}{\sqrt{B-A^2}} + A(-\frac{1}{2})(B-A^2)^{-3/2}(-2A) \\ &= \frac{1}{\sqrt{B-A^2}} + A^2(B-A^2)^{-3/2}.\end{aligned} \tag{2.13}$$

Secondly, the derivation of $\frac{\partial A}{\partial R_t}$:

$$\frac{\partial A}{\partial R_t} = \frac{\partial}{\partial R_t}\left\{\frac{1}{T}\sum_{t=1}^{T} R_t\right\} = \frac{1}{T}. \tag{2.14}$$

Continuing with the derivation of $\frac{\partial S_T}{\partial B}$:

$$\begin{aligned}\frac{\partial S_T}{\partial B} &= \frac{\partial}{\partial A}\left\{\frac{A}{\sqrt{B-A^2}}\right\} \\ &= A(-\frac{1}{2})(B-A^2)^{-3/2} \\ &= \frac{-A^2}{2}(B-A^2)^{-3/2}.\end{aligned} \tag{2.15}$$

The last partial derivative in the left bracket $\frac{\partial B}{\partial R_t}$ of the function (2.12):

$$\frac{\partial B}{\partial R_t} = \frac{\partial}{\partial R_t}\left\{\frac{1}{T}\sum_{t=1}^{T} R_t^2\right\} = \frac{2}{T}R_t. \tag{2.16}$$

The right bracket of the equation (2.12) is derived and starting with the partial derivative of the return function with respect to F_t:

$$\begin{aligned}\frac{\partial R_t}{\partial F_t} &= \frac{\partial}{\partial F_t}\left\{\mu(F_{t-1}\cdot r_t - \delta|F_t - F_{t-1}|)\right\} \\ &= \frac{\partial}{\partial F_t}\left\{-\mu\cdot\delta|F_t - F_{t-1}|\right\} \\ &= \begin{cases} -\mu\cdot\delta & \text{if } F_t - F_{t-1} > 0 \\ \mu\cdot\delta & \text{if } F_t - F_{t-1} < 0 \end{cases} \\ &= -\mu\delta\cdot\text{sign }(F_t - F_{t-1}),\end{aligned} \tag{2.17}$$

in addition, considering the r_t^f in R_t (2.5) and taking partial derivative with respect to F_t to see the effect of the r_t^f:

$$
\begin{aligned}
\frac{\partial R_t}{\partial F_t} &= \frac{\partial}{\partial F_t}\left\{\mu(r_t^f + F_{t-1}(r_t - r_t^f) - \delta|F_t - F_{t-1}|)\right\} \\
&= \frac{\partial}{\partial F_t}\left\{\mu r_t^f + \mu F_{t-1}(r_t - r_t^f) - \mu\delta|F_t - F_{t-1}|\right\} \\
&= \frac{\partial}{\partial F_t}\{-\mu\cdot\delta|F_t - F_{t-1}|\} \\
&= \begin{cases} -\mu\cdot\delta & \text{if } F_t - F_{t-1} > 0 \\ \mu\cdot\delta & \text{if } F_t - F_{t-1} < 0 \end{cases} \\
&= -\mu\delta\cdot\text{sign }(F_t - F_{t-1}).
\end{aligned}
\tag{2.18}
$$

The partial derivative $\frac{\partial R_t}{\partial F_t}$ when considering r_t^f and when dropping r_t^f is exactly the same. Concluding that r_t^f does not have any impact on the return function with respect to the positioning. Secondly tacking partial derivative of the return function with respect to F_{t-1}:

$$
\begin{aligned}
\frac{\partial R_t}{\partial F_{t-1}} &= \frac{\partial}{\partial F_{t-1}}\{\mu(F_{t-1}\cdot r_t - \delta|F_t - F_{t-1}|)\} \\
&= \mu\cdot r_t + \frac{\partial}{\partial F_{t-1}}\{-\mu\cdot\delta|F_t - F_{t-1}|\} \\
&= \begin{cases} \mu\cdot\delta & \text{if } F_t - F_{t-1} > 0 \\ -\mu\cdot\delta & \text{if } F_t - F_{t-1} < 0 \end{cases} \\
&= \mu\cdot r_t + \mu\delta\cdot\text{sign }(F_t - F_{t-1}).
\end{aligned}
\tag{2.19}
$$

Similarly to (2.18), r_t^f is taken into account in the return function when deriving with respect to F_{t-1} to see the effect of r_t^f. In contrast to $\frac{\partial R_t}{\partial F_t}$ where the r_t^f does not have any affect with respect to F_t. In the (2.20) it is showcased that the return function is affected by r_t^f with respect to F_{t-1}. The reason for this occurrence is to account for the short position.

$$\begin{aligned}\frac{\partial R_t}{\partial F_{t-1}} &= \frac{\partial}{\partial F_{t-1}}\left\{\mu(r_t^f + F_{t-1}(r_t - r_t^f) - \delta|F_t - F_{t-1}|)\right\} \\ &= \frac{\partial}{\partial F_{t-1}}\left\{\mu r_t^f + \mu F_{t-1}(r_t - r_t^f) - \mu\delta|F_t - F_{t-1}|\right\} \\ &= \mu(r_t - r_t^f) - \mu\delta|F_t - F_{t-1}|\frac{\partial}{\partial F_{t-1}} \\ &= \begin{cases} \mu \cdot \delta & \text{if } F_t - F_{t-1} > 0 \\ -\mu \cdot \delta & \text{if } F_t - F_{t-1} < 0 \end{cases} \\ &= \mu(r_t - r_t^f) + \mu\delta \cdot \text{sign}\ (F_t - F_{t-1})\end{aligned} \tag{2.20}$$

Lastly, the derivation of $\frac{\partial F_t}{\partial \boldsymbol{\theta}}$ is necessary further in order to have all the partial derivatives in equation (2.12):

$$\begin{aligned}\frac{\partial F_t}{\partial \boldsymbol{\theta}} &= \frac{\partial}{\partial \boldsymbol{\theta}}\left\{tanh(\boldsymbol{\theta^T x_t})\right\} \\ &= (1 - tanh^2(\boldsymbol{\theta^T x_t}))\left\{\frac{\partial}{\partial \boldsymbol{\theta}} \cdot \boldsymbol{\theta^T x_t}\right\} \\ &= (1 - tanh^2(\boldsymbol{\theta^T x_t}))\left\{\boldsymbol{x_t} + \boldsymbol{\theta^T}\frac{\partial F_{t-1}}{\partial \boldsymbol{\theta}}\right\}\end{aligned} \tag{2.21}$$

The second line of the equation (2.21), the F_t is derived with respect to $\boldsymbol{\theta}$ and using the chain rule $(1 - tanh^2(\boldsymbol{\theta^T x_t}))$ is multiplied by the inner derivative of $\boldsymbol{\theta^T x_t}\frac{\partial}{\partial \boldsymbol{\theta}}$. The partial derivative of $\boldsymbol{x_t}$ becomes partial derivative of F_{t-1} with respect to $\boldsymbol{\theta}$. Because, $\boldsymbol{x_t} = [F_{t-1}, r_t, r_{t-1}, ..., r_{t-m}, 1]$ and the only variable in $\boldsymbol{x_t}$ that is differentiable with $\boldsymbol{\theta}$ is the F_{t-1}.

Lastly, inclusion of all the derived partial equations into (2.12) except equations (2.18 and 2.20) that contained r_t^f which is not considered, ie . substituting (2.13, 2.14, 2.15, 2.16, 2.17, 2.19 and 2.21) into (2.12) the following expression is achieved:

$$\begin{aligned}\frac{\partial S_T}{\partial \boldsymbol{\theta}} &= \sum_{t=1}^{T}\left\{\frac{\partial S_T}{\partial A}\frac{\partial A}{\partial R_t}+\frac{\partial S_T}{\partial B}\frac{\partial B}{\partial R_t}\right\}\cdot\left\{\frac{\partial R_t}{\partial F_t}\frac{\partial F_t}{\partial \boldsymbol{\theta}}+\frac{\partial R_t}{\partial F_{t-1}}\frac{\partial F_{t-1}}{\partial \boldsymbol{\theta}}\right\}\\ &= \sum_{t=1}^{T}\left\{\left\{\frac{1}{\sqrt{B-A^2}}+A^2(B-A^2)^{-3/2}\right\}\left\{\frac{1}{T}\right\}+\left\{\frac{-A^2}{2}(B-A^2)^{-3/2}\right\}\left\{\frac{2R_t}{T}\right\}\right\}\times\\ &\left\{\{-\mu\delta\cdot\text{sign }(F_t-F_{t-1})\}\left\{(1-tanh^2(\boldsymbol{\theta^T x_t}))\left\{\boldsymbol{x_t}+\boldsymbol{\theta^T}\frac{\partial F_{t-1}}{\partial \boldsymbol{\theta}}\right\}\right\}\right.\\ &\left.+\{\mu\cdot r_t+\mu\delta\cdot\text{sign }(F_t-F_{t-1})\}\frac{\partial F_{t-1}}{\partial \boldsymbol{\theta}}\right\}\end{aligned} \tag{2.22}$$

The implementation of the 2.22 into Python is straightforward. Firstly, define the F_t and R_t functions. Secondly, define the gradient function that introduces the different parts of S_T such as A and B. The gradient function returns the S_T at different time t and the gradient at time t. Finally, define the training function that takes into account the epoch, m, transaction cost and learning rate.

Chapter 3

Reinforcement trading

"History does not repeat itself but it often rhymes" — Mark Twain

This part of the thesis provides a detailed account of the data gathering, underlying assets, assumptions and the different parameters under consideration. Specifically, this section provides the implementation process of the simulations.

3.1 Assumptions

Throughout the thesis there are assumptions about the stock market and constraints on the behaviour of the IA:

- Negligible Market Impact: meaning action of the IA have minimum impact on the stock market. In other words, actions of the IA won't impact the individual stock performance when buying or selling the stock, mainly the stock prices are given as input data and remains unaffected by IA actions.
- Infinitely differentiable: instead of discrete stock number, this thesis assumes that the agent can trade continuous amount of stocks (e.g., 0.27 of a stock).
- No fees associated with short selling: shorting stocks does not have the traditional cost associated with it such as cost of borrowing stocks.

3.2 Data set

The data is downloaded using Python package panda and pandas_datareader, from Yahoo finance. The downloaded data set is in a csv format and contains six columns namely date, open, high, low, close, adjusted close[1] and the daily trading volume. In

[1]Adjusted close is the close price when adjusted for the dividend and other factors such as splits. Source:https://help.yahoo.com/kb/SLN28256.html.

this thesis only the closing price and the date are used. The data set under consideration is from 01-01-2000 to 12-11-2019, almost twenty years, which means 4998 daily observations. The reason for the length of the data set is that the trading model is having the possibility to train and learn during different volatile times such as 2000 and 2008.

In the selection process of stocks, desired characteristics such as an upward long term trend, downward long term trend, unpredictable moves in either direction and the stock with low volatility were taken into account. In order to do this, eight stocks were selected from the S&P 500 index. The reason for choosing stocks from the S&P 500 and not OMX 30 or otherwise is the availability of large data sets for downloading from Yahoo finance. The following stocks were under consideration: BRK (Berkshire Hathaway), NVDA (Nvidia), DB (Deutsche Bank), GE (General Electric), AXP (American Express), JPM (JP Morgan Chase Co), MS (Morgan Stanley) and MSFT (Microsoft).

The stocks that were chosen for training and testing are BRK, NVDA, DB and GE. From the initial set of stocks, BRK has the lowest daily standard deviation 0.01394 (see Table 3.1) in comparison to the other stocks; DB is in a clear long term downtrend (see Figure 3.1); NVDA is the stock with highest daily standard deviation 0.038329 (see Table 3.1) in comparison to the other stocks and exhibits most unpredictable positive and negative moves (see Figure 3.1); and lastly GE was range-bound during the period in question and exhibits volatile moves in both directions (see Figure 3.1). The standard deviation and mean of the different stocks are calculated using daily returns for the entire data set ranging from 2000 to 2019.

Measure	**MSFT**	**MS**	**DB**	**NVDA**	**GE**	**JPM**	**AXP**	**BRK**
std	0.01904	0.0315	0.0272	0.03832	0.0198	0.0243	0.0220	0.0139
mean	0.0454	0.0504	-0.0020	0.1543	0.0023	0.0593	0.0479	0.0456

Table 3.1: Standard deviation and mean of daily returns of each stock. Mean is presented as percentage change.

The four selected stocks give a diverse range of characteristics desired when training and testing a trading model. The reason being, if the model is performing good on low volatility stocks and is only tested on those stocks, then the model appears good when showcasing the results. However, for the model to be robust and reliable there is a need to test it on a variety of stocks with different characteristics, to avoid the cherry picking bias[2].

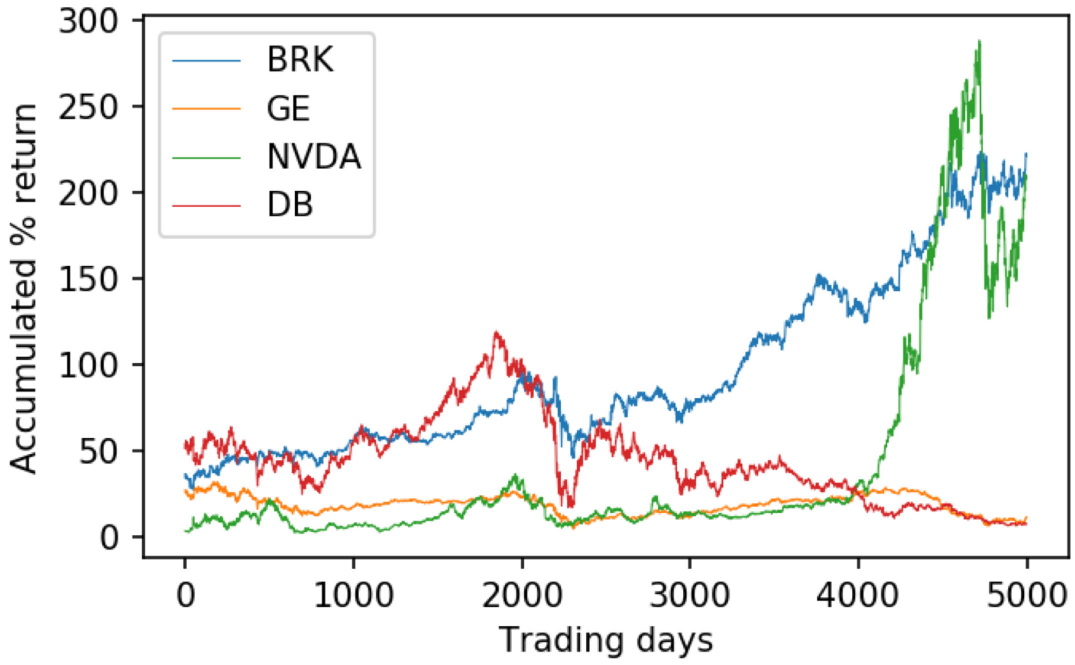

FIGURE 3.1: Evolution of the four selected stocks from 01-01-2000 to 12-11-2019.

3.2.1 Outliers and missing values

Outliers in a data set are common and they can have a positive or negative affect on the performance of the models [28]. Therefore, those outliers need to be addressed and this can be done by different methods. An outlier is a value that differs significantly from the normal distribution of the data set. In financial time series these outliers can occur as a consequence of information shocks (positive news coverage or negative earnings report) or a unexpected shocks (political or economic shocks) [29]. However, this thesis is attempting to showcase a trading model that is able to trade like a human trader and the environment needs to be close to real life as possible within the range of the assumptions. The exclusion of the extreme events from the data set would be considered as selecting data that is favorable for the model and that fits the model instead of the model adapting to the data. Therefore, this thesis includes the outliers and test the model as close to a live scenario as possible with regard to the initial assumptions.

[2]"Cherry picking is the act of pointing at individual cases or data that seem to confirm a particular position, while ignoring a significant portion of related cases or data that may contradict that position." Source: https://english.stackexchange.com/questions/70550/cherry-picking-what-is-the-correct-usage

Similarly to outliers, missing values are very common in financial time series which can reduce performance of the chosen model or create a bias within the model [30]. The cause of missing values can be attributed to holidays and weekends because there is no trading on those days. Another common source of missing values can be attributed to human error that can lead to unregistered values in the data set. Therefore, it is imperative to deal with the missing values in the data set.

There are numerous methods to deal with this problem such as replacing the missing value with the previous known value [31] (cited in [32]). However, this method has a number of draw-downs and is not recommended when the time series is non-stationary [31]. The missing values are removed at Yahoo Finance database, meaning that the downloaded data does not contain missing values. This solves the problem of the missing values but can create a loss of information and smaller data sets [31]. The chosen data sets are large enough, thus the impact of removing missing values from the data set is not significant.

3.3 Training, validation and test sets

When building a machine learning model it is not appropriate to train the model and test it on the same data set. Therefore, dividing the data into three sets, a training , a validation and a test set is the common approach, as seen in the figure (3.2).

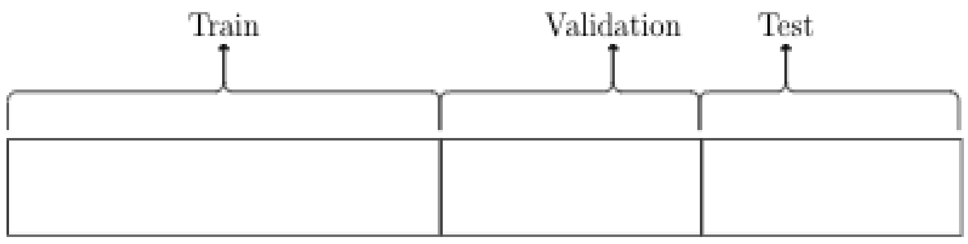

FIGURE 3.2: Train, validation, test ratio.

The training set is used for the model fitting, the fitted data is saved and feed back as input to the model and incorporated in future decision making process. Validation data set is used to evaluate the training of the model, to see how well the model learned and usually to make parameters adjustments.

The test set is used as a generalization of the performance of the model and therefore at the time of validation and training it should be assumed as if the test set did not exist.

Otherwise, the model trades on patterns that did not exist in the other two sets and for that reason it is important that the test set is left untouched until the end. All decisions that affect the model need to be based on the observations of the training and validation set. Initially the data set is split 40 percent for training, 30 percent for validation and 30 percent for testing.

3.4 Implementation

The implementation of a trading model is a thorough process and all variables that have a impact on the models performance must be kept in mind. This thesis considers the following variables: learning rate, commission rate, m (number of time series inputs), training/validation ratio and epochs. Based on the results of the variables the optimal variable values are selected for each stock and adjusted if necessary on the validation data set. The model is then simulated on the test data set with the optimal variable values.

3.4.1 Learning rate

The learning rate is a hyperparameter[3] that affects the incremental step size of the model [33].

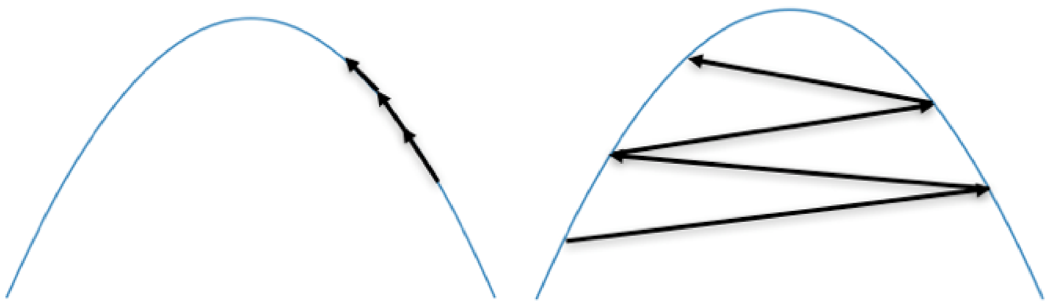

Figure 3.3: The difference between small learning rate (left) and big learning rate (right).

Selection of the learning rate is challenging because a small value results in long computing time, especially when using large data sets. However, if a large value is selected the potential to sub-optimal training could lead to unstable training process [33]. Therefore, the learning rate is one of the most important hyperparameter when training and

[3]Hyperparameter is a parameter that is determined before the learning begins.

validating a model. This thesis tests a range of learning rates to estimate its effect on the model performance and to select the optimal learning rate.

3.4.2 Epoch

Epoch is the selected data set passed through the trading model multiple times. Having only one epoch would mean that one runs the entire data set through the model and that would create a problem for the model especially if the data set is large [34]. Therefore, epoch are divided into smaller batch sizes and feed the model one by one in order to update the weights of the model at the end of each time step.

The model in this thesis is gradient ascent which is a iterative process and it updates and trains on the data provided at each pass of the data through the model. Therefore, one epoch is not enough because it leads to underfitting (see Figure 3.4) and a large number of epochs leads to overfitting (see Figure 3.4). Concluding, that the tuning of the epoch requires the ability to recognize when the model has reached the point of diminishing returns.

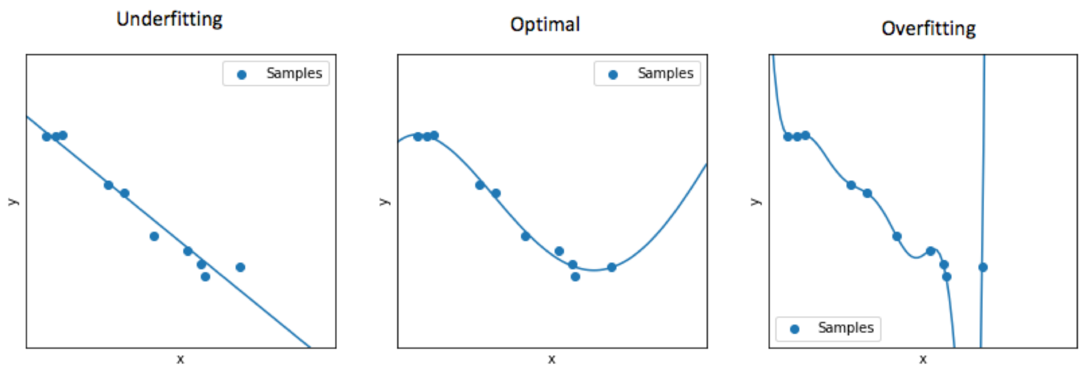

FIGURE 3.4: Epoch from underfitting to overfitting.

3.4.3 Nested Cross-Validation

In a continuation of Section 3.3 this section emphasizes on the training and validation data sets. This thesis is adapting nested Cross-Validation (CV) to prevent data leakage[4] and to simulate as close to the real world trading environment as possible. When the model is at time t, it needs past experience to be able to trade in $t+1$, meaning that

[4]"Data leakage refers to information outside of the training set being used in the creation of the model."
Source: https://machinelearningmastery.com/data-leakage-machine-learning/.

the trading model is in the present and must not have the information about the future. Therefore, the data set is dissected where training sets comes chronologically before the validation set and is used for fitting the model as illustrated in the Figure 3.5 [35].

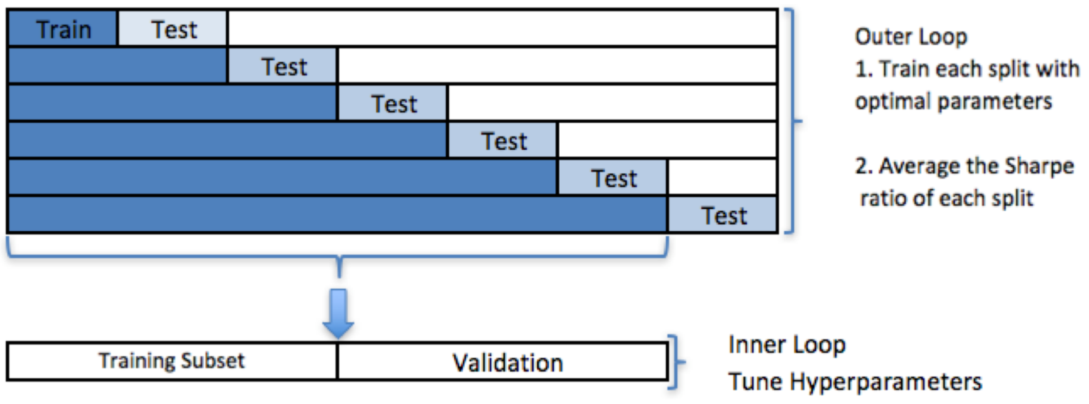

FIGURE 3.5: Nested cross-validation.

Looking at the Figure 3.2 the choice of the set sizes is fairly arbitrary, and this might lead to the set error being a poor estimate. The solution to this problem is the nested CV (see Figure 3.5), because it contains two loops, the inner and outer loop. The inner loop refines the hyperparameters and verifies them on the validation data set. The outer loop splits the data into multiple segments of training and test sets, and the error on each segment is averaged in order to calculate a robust estimator of the error [35]. Therefore. nested CV is an advantageous procedure that provides nearly unbiased estimate of the error [36].

3.4.4 Transaction cost

One of the main cost associated with trading is transaction cost and it has an impact on the profitability of the trading model. There are several different transaction fees such as stock trade fee (flat), meaning that the broker charges a single rate no matter the trade amount. Stock trade fee (per share), meaning that the broker charges per share that is traded. The trading cost have under the past several years been on decline because of the competition and cost reduction by the brokers. This has led to a number of brokers in 2019 dropping the trading cost to zero. Interactive Brokers was one of the first to drop the transaction cost to zero[5]. However, this thesis includes the simulation of the

[5]Source: https://www.cnbc.com/2019/10/13/battle-for-client-assets-heats-up-as-brokers-cut-fees-to-zero.html

transaction cost, because of the interest in seeing how the trading model adapts when the transaction cost increase and the impact of rising trading cost on the performance of the model. The implementation test a variety of trading costs from 0.25% to 25%.

3.4.5 Number of time series inputs

The last hyperparameter in consideration is the number of time series inputs. The model needs to be guided on how many data points from the past to consider when it is at time t in order to be able to have a superior performance over the training data set. This hyperparameter is of vital importance, if m is to large then the model considers a large number of past events that have lower probability of happening at time t. However, if a small number m is selected, that could also lead to lower performance because the trading model does not have enough information from the past to be able to make correct decisions in the future.

Chapter 4

Results

"Simplicity is the ultimate sophistication" — *Leonardo da Vinci*

This section displays results of the parameters referred to in the previous section. Initially, the parameters values are set at random with the commission rate equal to 0.25%, m equal 80, learning rate being 0.1, and epoch equal to 500. When the optimal parameter values are obtained for every stock, the optimal values replace the initial values in future cases. Whenever the results from different stocks retain similar characteristics, such as converging to a specific range over a number of iterations, they are not showcased.

4.1 Learning rate

This subsection presents the results of the selected stocks and the impact of the learning rate on the Sharpe ratio of those stocks. The learning rate is set to $i \cdot 0.05$ where i is a range from 1 to 2000 by increments of 40, meaning that the learning rate goes from 0.05 to 98.05. The reason for this parameter length is to showcase the trading model performance when the learning rate increases linearly, in order to capture the behavior of the learning rates impact on the Sharpe ratio over a large sample.

When looking at Figure 4.1 it is noticeable that, as the learning rate increases from 0.05 onward the Sharpe ratio of the validation data decreases until the learning rate approaches the 10th iteration. When this point is reached the Sharpe ratio for both training and validation data remains in a range. The desired outcome is to maximize the Sharpe ratio and the difference between the training and validation Sharpe ratio. This is achieved when the learning rate is 0.05 or smaller (i.e. at the beginning of the chart). The results of the rest of the stocks confirm this finding with all of the Sharpe ratios being larger at smaller learning rates.

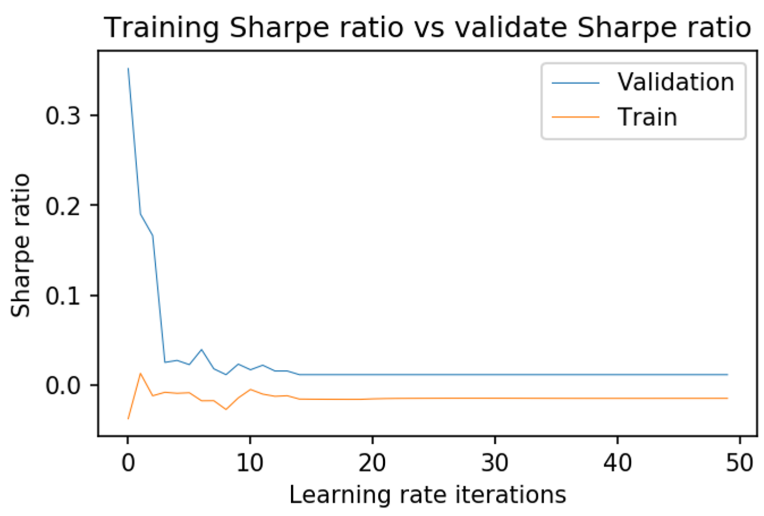

FIGURE 4.1: Learning rate simulations result from BRK.

However, as mentioned in the Section 3.4.1, choosing to small learning rate on a large data set it results in long computation time. Therefore, the conclusion is made that from this point forward in the rest of the cases the learning rate is set to 0.05 in order to avoid long computation time and to obtain better result.

4.2 Transaction cost

Intuitively, considering transaction cost lower is always better. This section is presenting the transaction cost impact on the trading model and the performance of the trading model as measured by Sharpe ratio. The transaction cost is set to $0.0025 \cdot i$ where i is a range from 10 to 100 by increments of 10, meaning that the transaction cost iterations simulates from 0.25% to 22,75% by 2.5%. Considering that the real world transition cost is zero or close to zero (see Section 3.4.4), the transaction cost under consideration here are exaggerated in comparison. The reason for the unrealistic parameter range is intentional in order to test the trading models adaptability to higher transaction cost.

The Figure 4.2 displayed below, confirms the intuition that higher costs would impact the performance of the trading model and put pressure on the Sharpe ratio. Figure 4.2 displays the Sharpe ratio of the GE at different transaction cost iterations starting from 0.25% to 22.75%. There is a clear downtrend when transaction costs increase linearly as the Sharpe ratio decreases. However between the 6th and the 7th iteration (see Figure 4.2), the transaction cost increases and the Sharpe ratio increases as well. This phenomenon is counter intuitive considering that between the 6th and the 7th period the transaction cost is increased by 2.5% and the expectation is for the Sharpe ratio to fall. In this period the trading model decreases the trading activity (see Figure 4.3) and by doing so decreases the volatility and risk premium remains same or bigger as a result of less trading activity. The simulations of the rest of the stocks exhibit the same downward sloping Sharpe ratio as transaction cost increases but with difference in slope. However, similarly to GE (see Figure 4.2) the rest of the stocks exhibit a spike in Sharpe ratio in a period or various periods.

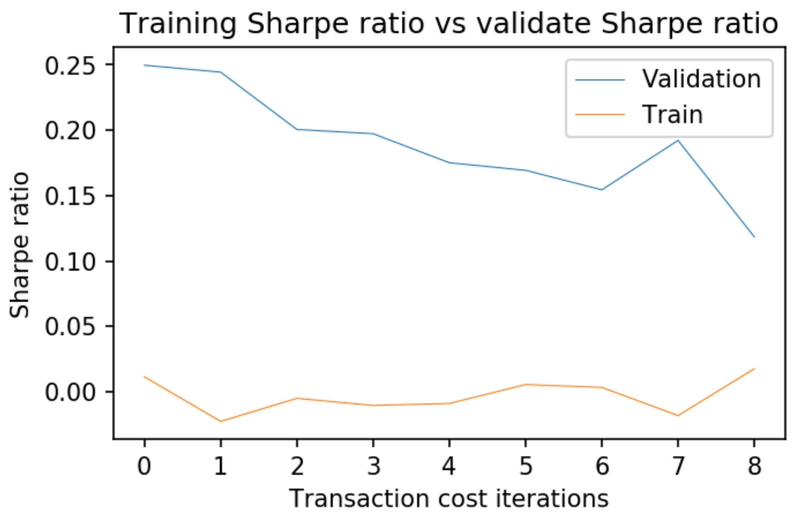

FIGURE 4.2: Transaction cost iteration result from GE.

Figure 4.3 confirms the decrease in trading activity as transaction cost increases. The first plot displays the positioning of the model when the transaction cost is set at 0.25% and the plot bellow displays the positioning when the rate is at 25%. These transaction costs are chosen because of the clarity at the extremes. The positioning is zero in both plots in the first 80 trading days. This is due to need of time series inputs of 80 days before the trading model can make a decision. The need for 80 data points in the minimal choice, and it's the same for other stocks.

The trading positionings after 80 trading days are very different (see Figure 4.3). The first plot is going from short to long positions in high frequency while the second plot keeps the position for lengthy periods of time. The case with the higher transaction cost executed 2303 long, 616 short and 80 neutral positions, while the lower cost executed 1615 long, 1304 short and 80 neutral. Figure 4.3 displays the ability of the trading model to adapt to rising trading cost and lower the frequency of trading. The findings are confirmed across the rest of stocks where the trading rate is lower when transaction cost increases.

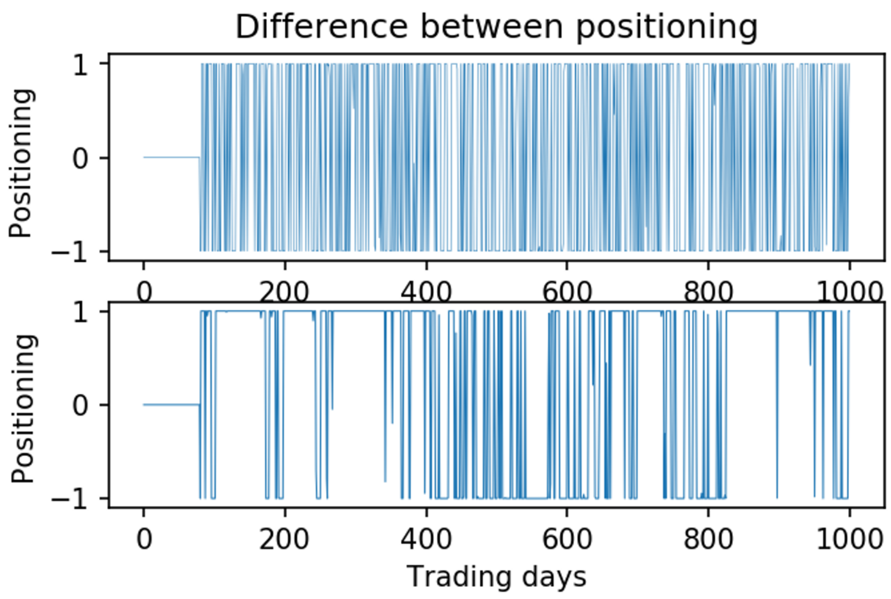

FIGURE 4.3: Positioning at different transaction costs for GE. In the top plot the transaction cost equals 0.25% and the bottom plot transaction cost equals 25%

4.3 Epoch

The epoch parameter is difficult to optimize because as mentioned in Section 3.4.2 if the epoch number is small the outcome is underfitting and large epoch leads to overfitting. The learning rate from Section 4.1 is applied in this section. The simulations are run on train data set and validated on the validation data set. The epoch number is set at i where i is a range from 1 to 500 by increments of one. On the rest of the stocks it is done in the same way, and this results in same shape of the curve as in Figure 4.4 with different slopes and optimal epoch points. The exception of Figure 4.4 can be found in DB where the shape of the training curve is upward sloping until 200 epoch, and after that point a dramatic decrease in the Sharpe ratio with 50% and the curve becomes flat after epoch 250 but the validation curve is upward sloping and levels of smoothly at 100 epochs.

When analyzing Figure 4.4 the training data set finds its maximum Sharpe ratio after the validation data set, with about additional 150 epochs from the 200 where validation is at maximum Sharpe ratio. This occurrence is consistent with the rest of the stocks. This can be attributed to the fact that the training data set has 3000 observations and validation data set has only 1000. The difference is three times and therefore applying the same number of epoch leads to different convergence to the maximum Sharpe ratio. However, considering that the test data has 1000 observations and if the test maximum number of epochs is selected it would lead to overfitting. Therefore, the prioritization lies on the validation maximum point while taking into consideration the training curve shape.

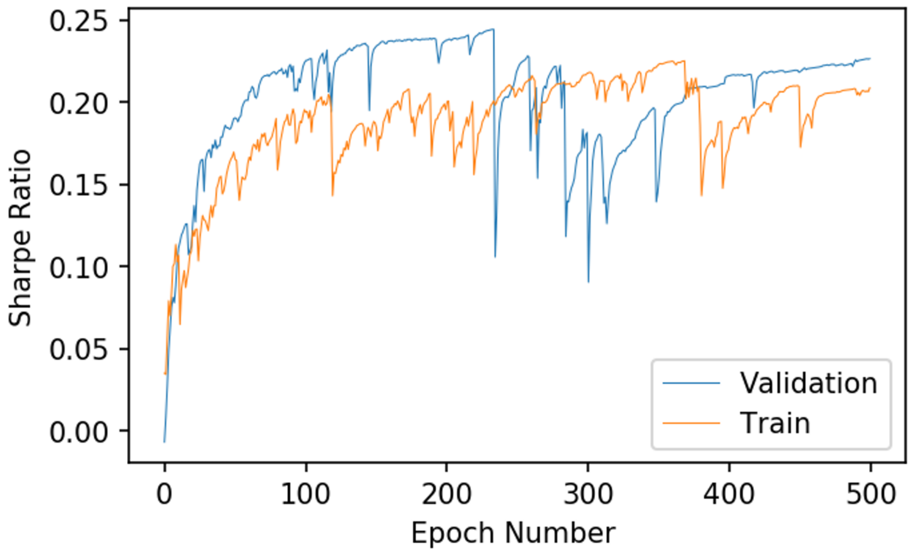

FIGURE 4.4: Sharpe ratio at each epoch number for NVDA.

In an effort not to overfit the model the selection of the epoch is done after the validation curve slope starts to level of. Looking at the Figure 4.4 the optimal epoch for NVDA is at approximately 150. Both the training and validation data has the maximum around this point and thereafter levels of the training data having slightly higher values. The rest of the stocks have the Sharpe ratio maximum point for validation at 100 epoch. The hyperparameter are tuned and further optimized in the Section 4.5.

4.4 Time series input

This section presents the findings of the optimal number of time series inputs for each of the four stocks. Additionally, the previous optimized parameters presented in Sections 4.1 and 4.4 are applied in this section (i.e. learning rate equal 0.05 and the new epoch values). For each stock two cases are considered. Firstly $m = i$ with i representing a range from 1 to 500 by increments of 50. Second, the case where i represents a range from 1 to 200 by increments of 5. The reasoning behind the first simulation is to locate the global maximum of the Sharpe ratio over a large range of m iterations. The second simulation pinpoints the exact location of the maximum Sharpe ratio for the specific stock within error range of plus minus 5.

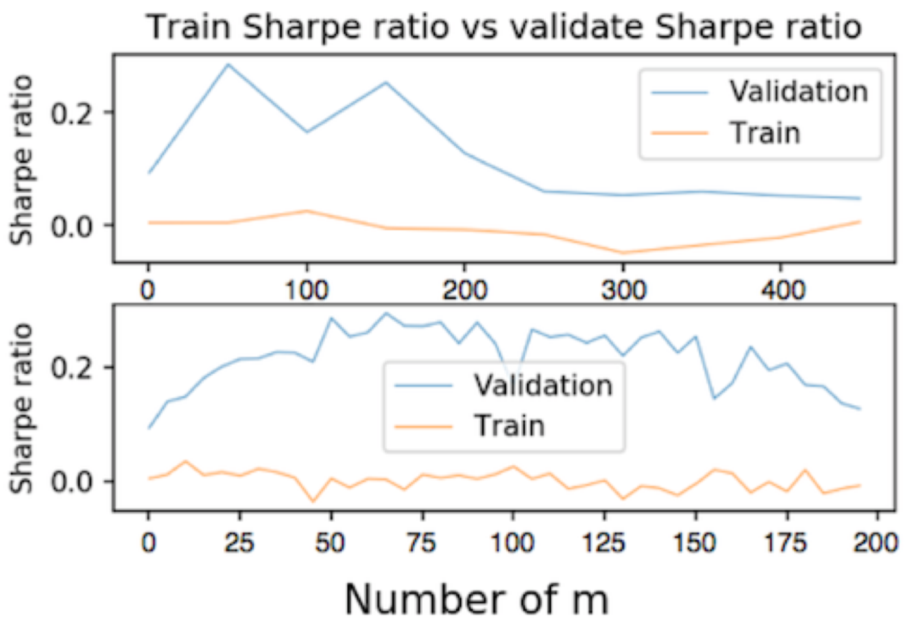

FIGURE 4.5: Sharpe ratio at different m DB.

Figure 4.5 contains two plots. The first (above) is the plot for first case that contains a large number of m and the second plot (below) contains the results of the second case for a smaller number of m . In the first plot there exists a possibility of information loss due to the increments of m being large and missing the global maximum Sharpe ratio. However, in the first plot after the first iteration the Sharpe ratio only falls and this can be confirmed from the simulation of the rest of the stocks. Concluding that when a maximum point is reached, by adding more time series inputs lead only to decreases in the performance of the model.

After locating the global maximum the second simulation (plot below) is done in order to pinpoint the exact location of the Sharpe ratio maximum point. The maximum Sharpe ratio of DB is reached at m equal 65 (i.e. m equal to 65 trading days), however the next highest optimal point when m equals 50 trading days. Taking into consideration that by adding 15 trading days the model only performs incrementally better, the decision is made to select the first point at with 50. Naming that the optimal time series inputs for DB is 50 trading days.

The rest of the stocks exhibited same concave shape of the Sharpe ratio over number of m iterations similar to the Figure 4.5. Over a large number of iterations this becomes very clear, therefore the statement can be made with high confidence that the maximum point is reached plus minus 5 trading days (the length of the iterations). The optimal time series inputs for the rests of the stocks: BRK optimal at m equal 70 trading days, NVDA optimal at m equal 115 trading days and GE optimal at m equal 65 trading days. Similarly to DB, when selecting the highest Sharpe ratio for the rest of the stocks, if the next highest Sharpe ratio was slightly lower but was obtained using significantly less time series inputs. The next highest point was selected as the optimal point for that stock.

4.5 Training validation ratio

This thesis has already concluded in Section 4.1 and 4.2 the effect of learning rate and transaction cost on the model performance. Concluding that the higher transaction cost leads to lower performance and lower learning rate leads to better performance. Having reached a conclusion on those two variables, there is no need to cross validate them. Instead, this section focuses on the tuning the number of epoch and time series inputs while the transaction cost and learning rate remains constant at 0.0025 and 0.05 respectively. The previous parameter optimizations were achieved using the standardized training/validation ratio from Section 3.3 while this section focuses on tuning parameters using nested CV.

The data set from 2000 to 2019 is chronologically dissected into portions. The training and validation data set are set up as follows: training set starts with the first 252 trading days and thereafter add chronologically 252 trading days which creates a new training set at each iteration. Validation set starts at the end of the training set (i.e. 256) and add 30 percent of the training set length (i.e. 75). This process creates a large number of training and validation sets with different lengths while maintaining the ratio 70/30 such as in the Figure 3.5. This process extends over the first 3998 data points and leaves 1000 data points untouched for the testing of the model.

The learning of the two parameters is done on each portion of the data in order to determine the performance of the model at a specific parameter value. The parameter values after the learning are averaged over the number of portions of data sets. This is done in order to test a specific parameter value on all the data set portions and then compare to the performance of the other values, as seen in Table 4.1.

The first case optimizes the number m while epoch value (previously optimized in Section 4.3) remain constant. The second case uses the new tuned m value and learn the epoch values for each of the stocks. The learning is done in the vicinity of the local maximum values, meaning if the optimal value of the parameter is 50 the iterations are from 25 to 75 (i.e. 50 simulations). The number of iterations and the approach applies for both learning of epochs and m. One of the issues when considering m bigger than 75 (in the case of NVDA m=115), is that the number m is larger than the smaller validation data set (i.e. 75). This creates a zero value and can cause the parameter to record a smaller average value for the parameter value because of a number of zeros resulting from small validation data set.

Stocks	m	**Average Sharpe**
BRK	52	0.3033
NVDA	95	0.1984
DB	58	0.2944
GE	66	0.2103

Table 4.1: Results of the nested CV tuning of m

The actual results of the nested CV can be seen in the Table 4.1 where the four stocks and the iterations run on them are presented. The values represent average Sharpe ratio of the iteration run over the portioned data sets when epoch numbers are constant (i.e. BRK = DB = GE = 100 and NVDA = 150). The middle column represents the new tuned value of m. The values presented are the ones that performed the best over all of different portions of data sets. In Section 4.4 the model optimized the parameter value, by the standardized training/validation ratio presented in Section 3.3. However when learning on a number of different data sets using nested CV, the outcome results in a robust parameter value. BRK new optimal m is 49 trading days compared to previous 70 obtained in Section 4.4, NVDA new optimal m is 95 compared to previous 115, DB new optimal m is 58 compared to previous 50 and GE new optimal m is 66 compared to previous 65. Conclusion can be made that that the method from Section (3.3) is biased and overfitted for a the specific length of the data set when compared to the results of nested CV.

Similarly to the learning of m, the learning of epoch is performed in the same way. If the previous optimal value of epoch from Section 4.3 is 100 then the learning is considered from 75 epoch to 125, and the number differs from stock to stock since BRK, DB and GE have optimal epoch value (Section 4.3) at 100 and NVDA at 150. NVDA learning runs from 125 epoch to 175 and the rest as mention above from 75 epoch to 125.

Stocks	Number of epoch	Average Sharpe
BRK	96	0.3115
NVDA	134	0.2001
DB	101	0.2945
GE	101	0.211

Table 4.2: Results of the nested CV tuning of epoch.

Comparing to previous results (see Section 4.3), BRK new optimal value for epoch is 96 compared to previous 100, NVDA new value equals 134 and the previous was 150, DB new value equals 101 and the previous was 100 and lastly GE new value equals 101 and previous equal 100. All of the stocks improved the average Sharpe ratio and most notably, BRK and NVDA did so with significantly less epoch numbers, whereas DB and GE previous optimizations were close to the true value. The conclusion is that nested CV produces superior results when taking into consideration different lengths of the data sets. Nested CV insures that the best parameter and unbiased value is selected for testing the data.

4.6 Test data

This section of the thesis incorporates the parameter optimization and parameter tuning done in previous section and tests the trading model using those parameter. Transaction cost and learning rate continue to be constant at 0.0025 and 0.05 respectively and the values for the stocks are as follows: BRK, epoch equal 96, m equal 52; NVDA, epoch equal 134, m equal 95; DB, epoch equal 101, m equal 58; and GE, epoch equal 101, m equal 66. The length of the test data is 1000 trading days (i.e. from approximately beginning of 2016 to the end of 2019).

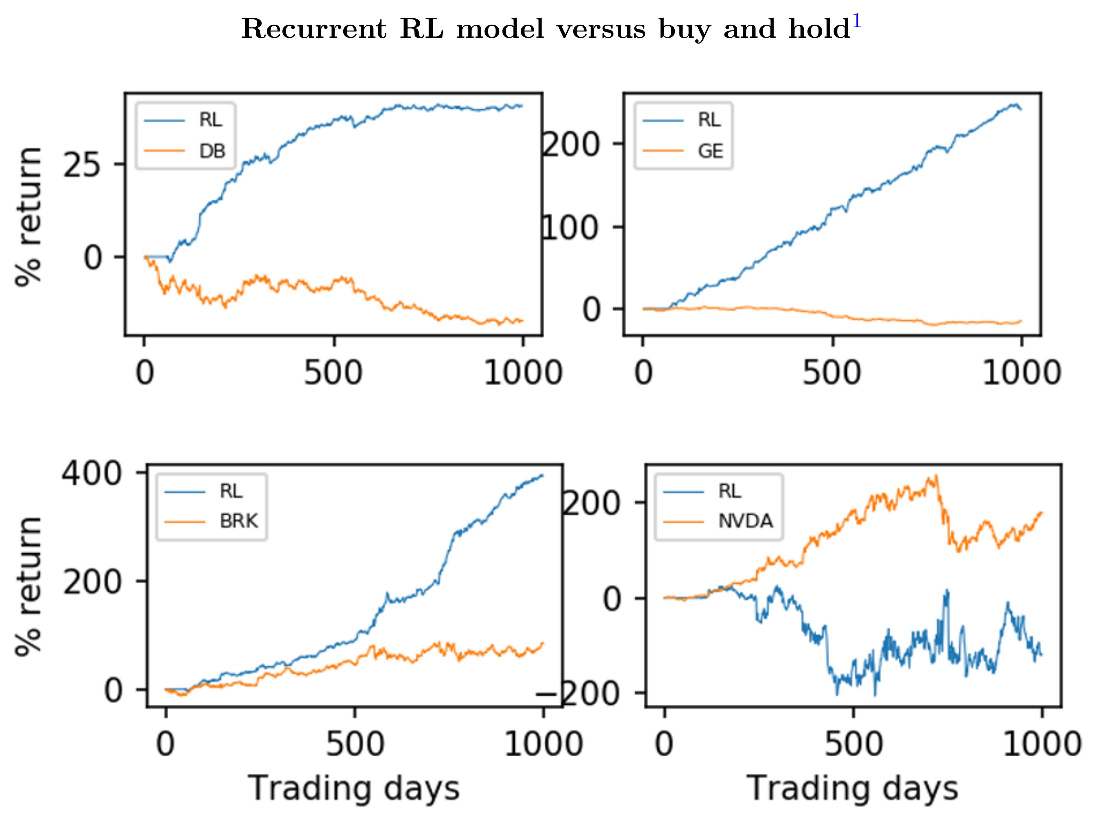

FIGURE 4.6: The result of the model on the test data set compared to the buy and hold strategy of the respective benchmarks. The y-axis represents the accumulated percentage return and x-axis represents the trading days.

Figure 4.6 showcases the result of the RL trading model tested on the test data set of the four stocks. The beginning of each chart the RL trader is flat for an extended period and that is due to the need of time series inputs and that varies between stocks. The return is calculated using cumulative returns of the stocks versus the cumulative return of the RL trader.

The results are very interesting considering the out performance in three of the four experiments. Most notably the GE and BRK where the trading model out performed the buy and hold strategy with several factors. The results of DB is interesting as well in a period where the buy and hold strategy return was negative the RL trader out performed the benchmark and made a positive return. However, NVDA buy and hold strategy managed a gain of approximately 200% while the RL trader returns were negative for the same period. This can be attributed to number of factors such as the high volatility in the NVDA stock which made it hard for RL trader learn and overcome or the change in characteristics of the underlying stock from training data set to testing. NVDA stock when looking over the entire period from 2000 to 2019, in the first 16 years it was range bound and not exhibiting volatility. The parameters of the model were tuned and selected using period (2000 - 2016). Then applied on a data set that has different characteristics and higher volatility. The trading model performance relative to the respective benchmarks is superior. If the training data does not differ too much in terms of volatility or characteristics from testing data as seen in NVDA.

4.7 Limitations

This section of the thesis presents limitation of data gathering and limitations impacting the model performance:

- The computation time is long when selecting large values for epochs or time series data points. The wide range of parameters that need to be optimized over a large number of iterations takes long time considering the large data set. The simulation time limits iterations ranges, for example it takes four hours and fifteen minutes to test one parameter with twenty iterations.
- Possibility of overfitting the data to fit a specific stock or underfitting.
- Reliance of the Yahoo Finance for delivery of accurate data and data that was cleaned properly by them.

Chapter 5

Conclusion and further research

After rigorous testing of the trading model, a number of conclusions are made. Firstly, the learning rate has a significant impact on the performance of the model, but excessively low learning rate slows down the model and leads to overfitting. Therefore, it's vital to choose the optimal value for learning rate, which in this body of work is set at 0.05. Secondly, the performance of the trading model is significantly impacted when trading cost increase. However, the models ability to adapt and change strategy to less trading activity is noteworthy. This can be attributed to the Sharpe ratio being affected negatively by higher transaction rates at each iteration.

The optimization of epochs and m increased the performance of the trading model. The tuning of the hyperparameters with nested CV validated the hyperparameter values over different data set lengths also increased performance. Nested CV process of selecting hyperparameter values increases the validity and reliability of the experiment. Because it decreases the possibility of cherry picking parameter values that perform well on a specific data set length. Additionally, the hyperparameter are optimized and tuned on the training data set to avoid data leakage. In contrast to the [15] that does not motivate how the parameters are chosen and how the model is tested, the results are only showcased which has the possibility of being biased for the reasons mentioned above. Therefore, concluding that the nested CV versus the standardized training testing ratio is superior and robust both in terms of avoiding bias and in terms of superior returns.

The model showcases significant sensitivity to selection of epoch, number of time series inputs and training validation ratio with respect to returns. This fact is showcased in Section 4.3, 4.4 and 4.5 where the sensitivity of different epoch and m values have significant impact on the Sharpe ratio. Furthermore, in Section 4.6 the NVDA stock shows how the selection of the training/test ratio impact on the trading model and the serious consequences follow as a result, when the test and training data have different characteristics. The training/test ratio is hard to overcome without creating a bias especially when trying to test the model on close to real world environment. The application of the nested CV helps with the problem but cannot solve it because inevitably the model parameters are optimized and tuned on the training data that unavoidable.

The results of this thesis are interesting and the performance achieved is superior with respect to the benchmark. However, this in only true under specific assumptions and conditions. If the stock exhibits unexpected drops which do not have a structure, the model can't cope with such an event and if the stock shifts from long period of low volatility to periods of high volatility. In those cases the model has problems out performing the benchmark. That is one of the biases of [15]. The experiment is done using the S&P 500 as a benchmark which is a segment of 500 stocks and exhibits lower volatility than a single asset because of the diversification of the 500 stocks within the index. Giving the trading model favorable conditions by selecting S&P 500 as a benchmark and not choosing several assert to test the robustness of the model in different environments.

The trading model suggested by [15] and optimized in this body of work has a lot of benefits and attributes that are beneficial and can be utilized when trading. But it has also a number of notable drawdowns that can make the trading model viewed as a momentum trader. Meaning that the model does well in a well-defined trend with lower volatility environment. The benefit of the model is the possibility to add features such as trading volume or other methods in order to make the model more dynamic and avoid the draw downs of the unexpected drops in the market. Moreover the model can be utilize for a different utility function that does not penalize positive spikes in the return.

When considering to implement this trading model in real time trading one needs to be cautious, because the data set that the model is trading with in this experiment might not been possible in the real market and would have created less return even possible negative return. Additionally a trading model would suffer additionally in low liquidity market and the inability to trade fractional stock would also be a factor weighing down the returns of the model.

In conclusion, the model is able to outperform the respective benchmark under specific conditions and is very sensitive to the parameters such as epoch, m, learning rate and training/test ratio.

Ideas for further research

- Selection of the utility function that does not penalize the positive spikes in return instead of the current Sharpe ratio. One proposition is the Sterling ratio that focuses on maximum drawdowns instead of standard deviation.
- Incorporation of other mechanisms such as the daily volume, the moving averages or the RSI (Relative Strength Index). In order to improve the models performance when trading volatile stocks that exhibit unstructured spikes.

Chapter 6

Contributions and Objectives

Objective 1: for Bachelor degree, student should demonstrate knowledge and understanding in the major field of study, including knowledge of the field's scientific basis, knowledge of applicable methods in the field, specialisation in some part of the field and orientation in current research questions.

The focus of this thesis is using recurrent RL model and applying it to securities trading. Model based trading has gained momentum in recent years and is a fundamental factor for everyday trading volume. The objective of the thesis is to optimize a RL model in order to outperform the benchmark under specific constraints and assumptions. The different fields of mathematics that have been applied are several variable calculus, vector algebra, methods of statistical inference, time series analysis and Python for financial engineering. The thesis describes in depth the background and introduces the subject. Derivation of the RL model and all the relevant variables are thoroughly described.

Objective 2: For Bachelor degree, the student should demonstrate the ability to search, collect, evaluate and critically interpret relevant information in a problem formulation and to critically discuss phenomena, problem formulations and situations.

This thesis is relying on previous research from Moody and Saffell [15] that present the recurrent RL. Moody and Saffell [15] were crucial in the understanding of the recurrent RL model and the implications of the different parameters of the model. This thesis differentiates itself from the original source and implements different approach to the utility function. Additionally, this thesis views critically the results obtained in [15] and points out the shortcoming of the results and tries to overcome that. Moreover, other sources were used to get a thorough overview and deeper understanding of the scientific area and uses of recurrent RL in other fields.

Objective 3: For Bachelor degree, the student should demonstrate the ability to independently identify, formulate and solve problems and to perform tasks within specified time frames.

This thesis formulates the research question based on previous work inspiration [15] and introduces new experiments. Most notably the results differ from the original inspiration and showcase the RL model in a different setting. In the pursuit of the results a number of coding and mathematical problems were overcome and solved. For the majority of the time the timeline was kept with exception of the presentation date which was pushed forward.

Objective 4: For Bachelor degree, the student should demonstrate the ability to present orally and in writing and discuss information, problems and solutions in dialogue with different groups.

The financial engineering field is full of unfamiliar notations and implications of the different variables/parameters. Some time is spent on the understanding, explanation and formulation of the chosen problem and the variables and notations surrounding the problem. In the process of this a vast knowledge about the problem is gained which makes it possible to present and answer questions about the topic. Even more time is spent making sure that the content, description, derivation, and experiments are adequately motivated.

Objective 5: For Bachelor degree, student should demonstrate ability in the major field of study make judgments with respect to scientific, societal and ethical aspects.

In the process of writing this thesis, the importance of ML techniques in finance and other industries that rely on decision making based on time series analysis were understood. Additionally, the implications of ML for the future and the ability of ML to be implemented in everyday life such as traffic signs, cars or process automation was realized. ML techniques can have large impact on societal well being, because of the need of corporations to cut costs and in some cases a ML robot or algorithm can perform the service at a lower cost than a number of humans can. Ethical question arise when the use of ML is incorporated into fields such as marketing, personal recruitment, or justice systems. Fields that have large degree of impact on the lives of everyday people. Therefore the implementation and creation of ML algorithm needs to consider the bias of the model and have a broader understanding of the impact of the algorithm on a broader scale before implementation. Even after implementation there needs to be constant oversight of the ML algorithm in order to make sure that decisions taken by algorithm are within acceptable parameters. If mistakes are made i the construction or implementation of the algorithm, there needs to exist some form of accountability and responsibility taking for the mistakes and subsequent learning of those mistakes in order that future ML algorithms can have a chance to improve lives and societies.

Bibliography

[1] Michael David Rechenthin. Machine-learning classification techniques for the analysis and prediction of high-frequency stock direction available at. Available at: https://doi.org/10.17077%2Fetd.fvui75i2, 2014.

[2] Burton G. Malkiel and Eugene F. Fama. Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417.

[3] Harry Markowitz. Portfolio selection. *The journal of finance*, 7(1):77–91, 1952.

[4] Henry B. Mann. Quadratic forms with linear constraints. *The American Mathematical Monthly*, 50(7):430–433, 1943.

[5] Sarah Perrin and Thierry Roncalli. Machine learning optimization algorithms & portfolio allocation. Available at: https://doi.org/10.2139%2Fssrn.3425827, 2019.

[6] Alfred D. Martin Jr. Mathematical programming of portfolio selections. *Management Science*, 1(2):152–166, 1955.

[7] Fischer Black and Robert B. Litterman. Asset allocation. *The Journal of Fixed Income*, 1(2):7–18, sep 1991.

[8] Paul D. Kaplan and Sam Savage. Markowitz 2.0. In *Frontiers of Modern Asset Allocation*. John Wiley & Sons, Inc, sep 2015.

[9] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11, 2018.

[10] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.

[11] John Moody and Matthew Saffell. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks*, 12(4):875–889, 2001.

[12] John Moody and Lizhong Wu. Optimization of trading systems and portfolios. In *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*, pages 300–307. IEEE, 1997.

[13] BWAC Farley and W Clark. Simulation of self-organizing systems by digital computer. *Transactions of the IRE Professional Group on Information Theory*, 4(4):76–84, 1954.

[14] WESLEY A Clark and Bernard G Farley. Generalization of pattern recognition in a self-organizing system. In *Proceedings of the March 1-3, 1955, western joint computer conference*, pages 86–91, 1955.

[15] John E Moody, Matthew Saffell, Y Liao, and L Wu. Reinforcement learning for trading systems and portfolios. In *KDD*, pages 279–283, 1998.

[16] John Moody, Matthew Saffell, W Lo Andrew, Yaser S Abu-Mostafa, Blake LeBaraon, and Andreas S Weigend. Minimizing downside risk via stochastic dynamic programming. *Computational Finance*, pages 403–415, 1999.

[17] Allan Timmermann and Clive WJ Granger. Efficient market hypothesis and forecasting. *International Journal of forecasting*, 20(1):15–27, 2004.

[18] Ilaria Giannoccaro and Pierpaolo Pontrandolfo. Inventory management in supply chains: a reinforcement learning approach. *International Journal of Production Economics*, 78(2):153–161, 2002.

[19] Robert H Crites and Andrew G Barto. Improving elevator performance using reinforcement learning. In *Advances in neural information processing systems*, pages 1017–1023, 1996.

[20] Wei Zhang and Thomas G Dietterich. High-performance job-shop scheduling with a time-delay td (λ) network. In *Advances in neural information processing systems*, pages 1024–1030, 1996.

[21] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 50–56, 2016.

[22] Itamar Arel, Cong Liu, Tom Urbanik, and Airton G Kohls. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4(2):128–135, 2010.

[23] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[24] Xiangping Bu, Jia Rao, and Cheng-Zhong Xu. A reinforcement learning approach to online web systems auto-configuration. In *2009 29th IEEE International Conference on Distributed Computing Systems*, pages 2–11. IEEE, 2009.

[25] Zhenpeng Zhou, Xiaocheng Li, and Richard N Zare. Optimizing chemical reactions with deep reinforcement learning. *ACS central science*, 3(12):1337–1344, 2017.

[26] William F Sharpe. Mutual fund performance. *The Journal of business*, 39(1):119–138, 1966.

[27] Claude Lemaréchal. Cauchy and the gradient method. *Doc Math Extra*, 251:254, 2012.

[28] Sang Kyu Kwak and Jong Hae Kim. Statistical data preparation: management of missing values and outliers. *Korean journal of anesthesiology*, 70(4):407, 2017.

[29] Ralph Grothmann. *Multi-agent market modeling based on neural networks.* PhD thesis, Universität Bremen, 2002.

[30] Marvin L Brown and John F Kros. Data mining and the impact of missing data. *Industrial Management & Data Systems*, 2003.

[31] Andries P Engelbrecht. *Computational intelligence: an introduction.* John Wiley & Sons, 2007.

[32] Magnus Andersson and Johan Palm. Forecasting the stock market: A neural network approch, 2009.

[33] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[34] Shivam Sinha, TN Singh, VK Singh, and AK Verma. Epoch determination for neural network by self-organized map (som). *Computational Geosciences*, 14(1):199–206, 2010.

[35] Leonard J Tashman. Out-of-sample tests of forecasting accuracy: an analysis and review. *International journal of forecasting*, 16(4):437–450, 2000.

[36] Sudhir Varma and Richard Simon. Bias in error estimation when using cross-validation for model selection. *BMC bioinformatics*, 7(1):91, 2006.

Python implementation

The following packages are nessasary.

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (5, 3) # (w, h)
plt.rcParams["figure.dpi"] = 150
import pandas as pd
import time
import statistics
```

The sharpe_ratio function returns the Sharpe ratio value at time t. The function positions returns the **Ft** (positions) which means that the model is fed back the last position and the historical time series in order to calculate the next position. This can be calculated using price series x and θ and the **Ft** vector values are {-1,0,1}.

```
def sharpe_ratio(rets):
    return rets.mean() / rets.std()


def positions(x, theta):
    M = len(theta) - 2
    T = len(x)
    Ft = np.zeros(T)
    for t in range(M, T):
        xt = np.concatenate([[1], x[t - M:t], [Ft[t - 1]]])
        Ft[t] = np.tanh(np.dot(theta, xt)) # sigmoid
    return Ft
```

The previous positions function gives us the position time series. The next step is to calculate the returns R and using $F_t = F_{t-1}r_t - \delta|F_t - F_{t-1}|$ formula.

```
def returns(Ft, x, delta):
    T = len(x)
    rets = Ft[0:T - 1] * x[1:T] - delta * np.abs(Ft[1:T] - Ft[0:T - 1])
    return np.concatenate([[0], rets])
```

In order to perform gradient ascent the calculation of the partial derivative of Sharpe ratio with respect to θ and the gradient function is presented.

```
def gradient(x, theta, delta):
    Ft = positions(x, theta)
    rets = returns(Ft, x, delta)
    T = len(x)
    M = len(theta) - 2

    A = np.mean(rets)
    B = np.mean(np.square(rets))
    S = A / np.sqrt(B - A ** 2)

    grad = np.zeros(M + 2)
    dFpdtheta = np.zeros(M + 2)

    for t in range(M, T):
        xt = np.concatenate([[1], x[t - M:t], [Ft[t-1]]])
        dRdF = -delta * np.sign(Ft[t] - Ft[t-1])
        dRdFp = x[t] + delta * np.sign(Ft[t] - Ft[t-1])
        dFdtheta = (1 - Ft[t] ** 2) * (xt + theta[-1] * dFpdtheta)
        dSdtheta = (dRdF * dFdtheta + dRdFp * dFpdtheta)
        grad = grad + dSdtheta
        dFpdtheta = dFdtheta


    return grad, S
```

After coding the gradient function the next step is to optimize the parameters using gradient ascent. The θ is updated by using $\theta = \theta - \alpha\frac{\partial S_T}{\partial\theta}$ ware α is the learning rate.

```
def train(x, epochs, M, commission, learning_rate):
    theta = np.ones(M + 2)
    sharpes = np.zeros(epochs) # store sharpes over time
    for i in range(epochs):
        grad, sharpe = gradient(x, theta, commission)
        theta = theta + grad * learning_rate
        sharpes[i] = sharpe

    print("Finished training!" )
    return theta, sharpes
```

Loading the data and visualizing the data sets.

```
MSFT = pd.read_csv("MSFT_00-19.csv")
MS = pd.read_csv("MS_00-19.csv")
```

```
DB = pd.read_csv("DB_00-19.csv")
NVDA = pd.read_csv("NVDA_00-19.csv")
JPM = pd.read_csv("JPM_00-19.csv")
AXP = pd.read_csv("AXP_00-19.csv")
BRK = pd.read_csv("BRK_00-19.csv")
GE = pd.read_csv("GE_00-19.csv")


plt.plot(MSFT['Adj Close']) # clear bull
rets_MSFT = MSFT['Adj Close'].pct_change() # % change
str(rets_MSFT.mean())   # mean = 0.0.04535741371884 % return over 19 years
str(rets_MSFT.std())    # stdv = 0.0190413893895 19 years

plt.plot(MS['Adj Close']) #  volatile
rets_MS = MS['Adj Close'].pct_change() # % change
str(rets_MS.mean())  # mean = 0.05040036339 % return over 19 years
str(rets_MS.std())    # stdv = 0.031559378545560686 19 years

plt.plot(DB['Adj Close']) # clear bear
rets_DB = DB['Adj Close'].pct_change() # % change
str(rets_DB.mean())  # mean = -0.002022819561718967 % return over 19 years
str(rets_DB.std())    # stdv = 0.02729055932219991 19 years

plt.plot(NVDA['Adj Close']) # unpredictable
rets_NVDA = NVDA['Adj Close'].pct_change() # % change
str(rets_NVDA.mean())  # mean = 0.15432360872299333 % return over 19 years
str(rets_NVDA.std())    # stdv = 0.03832909116429849 19 years

plt.plot(GE['Adj Close']) #  volatile
rets_GE = GE['Adj Close'].pct_change() # % change
str(rets_GE.mean())  # mean = 0.002287207771264611 % return over 19 years
str(rets_GE.std())    # stdv = 0.0198453888213898 19 years

plt.plot(JPM['Adj Close']) #  statistically unremarkable
rets_JPM = JPM['Adj Close'].pct_change() # % change
str(rets_JPM.mean())  # mean = 0.05938723308960855 % return over 19 years
str(rets_JPM.std())    # stdv = 0.0243428300362348 19 years

plt.plot(AXP['Adj Close']) #  statistically unremarkable
rets_AXP = AXP['Adj Close'].pct_change() # % change
str(rets_AXP.mean())  # mean = 0.04791624846365439 % return over 19 years
str(rets_AXP.std())    # stdv = 0.022042834625303 75 19 years

plt.plot(BRK['Adj Close']) #  lowest vol
rets_BRK = BRK['Adj Close'].pct_change() # % change
str(rets_BRK.mean())  # mean = 0.045610939922837435 % return over 19 years
str(rets_BRK.std())    # stdv = 0.013943468606808014 19 years
```

Setting up the range of the training, validation and test data sets.

```
rets = BRK['Adj Close'].diff()[1:]#  lowest vol
#rets = NVDA['Adj Close'].diff()[1:]# unpredictable
#rets= DB['Adj Close'].diff()[1:]# # clear bear
#rets = GE['Adj Close'].diff()[1:]#  volatile

N = 3000
P = 999
T = 1000

x = np.array(rets)

x_train = x[-(N+P+T):-(P+T)]
x_validate = x[-(P+T):-T]
x_test = x[-T:]

std = np.std(x_train)
mean = np.mean(x_train)

x_train = (x_train - mean) / std
x_test = (x_test - mean) / std
x_validate =(x_validate -mean) / std
```

Training the model using range of different parameter values for M , epoch, commission rate and learning rate. In order to find out optimal parameters for maximum absolute return of the model. The difference of the parameters is evaluated using Sharpe ratio over the number of the parameters.

```
start = time.time()
M_train = []
M_validate = []

S_train = []
S_validate = []

for i in np.arange(1,2000,40):

    """Implementation of the model and the loop function for validation of the different paramet
    and then piloting those parameters for test and validate over the different parameters
    """

    theta, sharpes=train(x_train, epochs=100, M=80, commission=(0.0025), learning_rate=(0.05*i))
    theta, sharpes=train(x_validate, epochs=100, M=80, commission=(0.0025), learning_rate=(0.05*
    train_returns = returns(positions(x_train, theta), x_train, 0.0025)
    validate_returns = returns(positions(x_validate, theta), x_validate, 0.0025)

    #x_test = pd.Series(x_test) # not bad function

    # transform from difference to %
    percent_train= []
    for i in range(1, len(train_returns)):
        percent_train.append(train_returns[i]/ 100)

    percent_validate = []
    for i in range(1, len(validate_returns)):
        percent_validate.append(validate_returns[i]/ 100)

    #plt.plot((pd.Series(percent_train)).cumsum()) # test
    #plt.plot((pd.Series(percent_validate)).cumsum()) # test

    percent_train = pd.Series(percent_train)
    mu_train = statistics.mean(percent_train)
    std_train = np.std(percent_train, axis=0)

    percent_validate = pd.Series(percent_validate)
    mu_validate =  statistics.mean(percent_validate)
    std_validate = np.std(percent_validate, axis=0)

    M_train = np.append(M_train, [mu_train], axis=0) # append to the matrix
    M_validate = np.append(M_validate, [mu_validate], axis = 0)

    S_train = np.append(S_train, [std_train], axis = 0)
    S_validate = np.append(S_validate, [std_validate], axis = 0)

end = time.time()
print(end -start)
```

Visualizing the results of the implementation above.

```
sharpe_train = ((M_train*100)/(S_train*100))
sharpe_validate = ((M_validate*100)/(S_validate*100))

plt.plot(sharpe_validate, label="Validation", linewidth=0.5)
plt.plot(sharpe_train,label="Train", linewidth=0.5)
plt.xlabel('Transaction cost iterations')# needs to be changed for different parameters
plt.ylabel('Sharpe ratio')
plt.legend()
plt.title("Train Sharpe ratio vs validate Sharpe ratio")
```

The next loop is designed to test different ratios of train/validate on the different parameters. The data is sliced 11 times and contains different ranges of train and validate and a range of the parameters are tested upon those sliced data sets in order to tune the parameters.

```
matrix_t = []
matrix_v =[]
a_t=[]
a_v =[]

for X in range (1, 12, 1): #suppose to be (1, 12, 1)
    X_train = x[:-4746 + X*252]
    X_validate = x[(-4746 + X*252) :-4746 + int((X*252)*1.3)]



#for t in range (1,6,1):
    a_t=[]
    a_v =[]

    for i in np.arange(1,20,1):

        theta, sharpes=train(X_train, epochs=150, M=140+i, commission=(0.0025), learning_rate=(0
        theta, sharpes=train(X_validate, epochs=150, M=140+i, commission=(0.0025), learning_rate

        train_returns = returns(positions(X_train, theta), X_train, 0.0025)
        validate_returns = returns(positions(X_validate, theta), X_validate, 0.0025)

        #x_test = pd.Series(x_test) # not bad function

        # transform from difference to %
        percent_train= []
        for i in range(1, len(train_returns)):
            percent_train.append(train_returns[i]/ 100)

        percent_validate = []
        for i in range(1, len(validate_returns)):
            percent_validate.append(validate_returns[i]/ 100)

        percent_train = pd.Series(percent_train)
        mu_train = statistics.mean(percent_train)
        std_train = np.std(percent_train, axis=0)

        percent_validate = pd.Series(percent_validate)
        mu_validate =  statistics.mean(percent_validate)
        std_validate = np.std(percent_validate, axis=0)
        sharpe_train = ((mu_train*100)/(std_train*100))
        sharpe_validate = ((mu_validate*100)/(std_validate*100))

        #if i == 5: #
        #    break;

        a_t.append(sharpe_train)
        a_v.append(sharpe_validate)

    matrix_t.append(a_t)
    matrix_v.append(a_v)

    print("---------------------------")
    print("------ Loop number {} ------ ".format(X))
    print("---------------------------")

matrix_v = np.array(matrix_v, dtype=np.float32)
matrix_v= matrix_v.round(decimals=4, out=None)#end product are the values at different parameter

matrix_t = np.array(matrix_t, dtype=np.float32)
matrix_t = matrix_t.round(decimals=4, out=None) #end product are the values at different paramet
```

The result of the previous iteration.

```
values_t =[]
values_v = []

for i in range (0,5,1):
    t=[row[i] for row in matrix_t] # perfection!

    v=[row[i] for row in matrix_v]
    #v=[row[1] for row in matrix_v]



    avrage_t = sum(t)/len(t)
    avrage_v = sum(v)/len(v)



    values_t.append(avrage_t) #### receive the average of that parameter for all sets
    values_v.append(avrage_v)

matrix_end_t = np.array(values_t, dtype=np.float32)
matrix_end_v = np.array(values_v, dtype=np.float32)
```