

Aalto University  
School of Science  
Master's Programme in Mathematics and Operations Research

Markus Wilkman

# Feasibility of a Reinforcement Learning Based Stock Trader

Master's Thesis  
Espoo, July 30, 2020

Supervisor: Professor Ahti Salo  
Advisors: Lasse Kiviluoto M.Sc. (Tech.), Yliaisti Oy  
Sakari Malka M.Sc. (Tech.&Econ.), Yliaisti Oy

The document can be stored and made available to the public on the open internet pages of Aalto University. All other rights are reserved.

<b>Author:</b>	Markus Wilkman	
<b>Title:</b>	Feasibility of a Reinforcement Learning Based Stock Trader	
<b>Date:</b>	July 30, 2020	<b>Pages:</b> vii + 76
<b>Major:</b>	Systems and Operations Research	<b>Code:</b> SCI3055
<b>Supervisor:</b>	Professor Ahti Salo	
<b>Advisors:</b>	Lasse Kiviluoto M.Sc. (Tech.) Sakari Malka M.Sc. (Tech.&Econ.)	
<p>Managing stock investments well is essential for both institutional and individual investors. The task is, however, so challenging that often a passive index may perform better than actively hand-picking the stocks to invest in. Machine Learning, which has successfully been applied to solve complex problems, could provide a solution.</p> <p>The Machine Learning approach called Reinforcement Learning is utilised to create an independent trading system for managing a portfolio consisting of eleven stocks and a cash option on a daily basis. The trading system is trained with a policy optimisation algorithm, using information on both the stock prices and fundamental information of the underlying companies.</p> <p>The trading system is evaluated on historical data, previously unseen in training. The results indicate that the constructed trading systems on average outperform simple strategies, some of which represent a passive index consisting of the same eleven stocks. The most advanced trading system constructed seems to outperform the passive index with a 97% probability based on the total return during a fixed testing period.</p> <p>The trading system appears to perform better than the passive index when markets are going up, but when the market drops the trading system performs no better than the index even though it could allocate everything to cash. Overall, the market tends to go up and therefore the trading systems seem to outperform the simple strategies. Despite promising results, some assumptions made on the hypothetical trades performed back-in-time may not hold in practice. One suggestion for improving the trading system is to use data of higher frequency, which would decrease the inconsistency between the testing environment and the real-world.</p>		
<b>Keywords:</b>	Reinforcement Learning, Policy Gradient, Portfolio Management, Trading System, Kelly Criterion, Risk-Adjusted Return	
<b>Language:</b>	English	

<b>Utfört av:</b>	Markus Wilkman		
<b>Arbetets namn:</b>	Portföljhantering baserat på förstärkningsinlärning		
<b>Datum:</b>	30 juli 2020	<b>Sidantal:</b>	vii + 76
<b>Huvudämne:</b>	Systems and Operations Research	<b>Kod:</b>	SCI3055
<b>Övervakare:</b>	Professor Ahti Salo		
<b>Handledare:</b>	DI Lasse Kiviluoto DI&EM Sakari Malka		
<p>Hantering av investeringar är essentiellt för både privata och professionella investerare. Aktiv aktieförvaltning är dock en så utmanande uppgift att många investerare istället föredrar en passiv aktiefond. Samtidigt har maskininlärning applicerats framgångsrikt på flera komplexa problem och övermänsklig prestanda uppnåtts.</p> <p>Förstärkningsinlärning, en kategori inom maskininlärning, används för att skapa ett autonomt portföljhanteringssystem, som dagligen förvaltar en portfölj bestående av elva aktier och ett kontantalternativ. Systemet baserar allokeringsbesluten på information om företagens aktiekurs samt kvartalsrapporter och tränas genom förstärkningsinlärningsalgoritmen Policy Gradient.</p> <p>Portföljhanteringssystemet utvärderas med hjälp av historisk data som inte använts i träningskedet. Resultaten indikerar att systemet i medeltal presterar bättre än ett passivt index bestående av samma aktier samt andra enkla strategier. Det mest avancerade systemet presterade bättre än det passiva indexet med 97% sannolikhet under en viss testningsperiod.</p> <p>Systemet verkar prestera väl då marknaden överlag går uppåt men då marknaden faller så försämras prestationen, trots att kontantalternativet existerar. Eftersom marknaden generellt sett stiger så verkar portföljhanteringssystemet ändå slå de enkla strategierna. Trots lovande resultat så kan vissa antaganden gällande de teoretiska affärerna bakåt i tiden ifrågasättas. Ett utvecklingsförslag för systemet är att använda mer frekvent data, vilket skulle minska på klyftan mellan experimenten och verkligheten.</p>			
<b>Nyckelord:</b>	Förstärkningsinlärning, Policy Gradient, Portföljhantering, Algoritmiskt Handelssystem, Kelly Kriteriet, Riskjusterad Avkastning		
<b>Språk:</b>	Engelska		

# Acknowledgements

I wish to thank Yliaisti Oy for giving me the opportunity to write this thesis. The fruitful discussions with the group have been invaluable in providing expertise on the topic and countless development ideas for the trading system. I would also like to thank my supervisor Ahti Salo for his valuable feedback and guidance throughout the process of writing this thesis.

Espoo, July 30, 2020

Markus Wilkman

# Abbreviations and Acronyms

ANN	Artificial Neural Network
CCM	CRSP/Compustat Merged
CRP	Constant Rebalanced Portfolio
CRSP	Center for Research in Securities Prices, LLC
Compustat	Compustat North America database
DD	Downside Deviation
DDPG	Deep Deterministic Policy Gradient
EDGAR	Electronic Data Gathering, Analysis and Retrieval
fAPV	final Accumulated Portfolio Value
FSD	First-order Stochastic Dominance
FTL	Follow the Leader
GPU	Graphics Processing Unit
IT	Information Technology
LSTM	Long Short-Term Memory
MDP	Markov Decision Process
ML	Machine Learning
MPT	Modern Portfolio Theory
OLMAR	On-line Moving Average Reversion
PG	Policy Gradient
PMPT	Post-Modern Portfolio Theory
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
S	Sharpe ratio
SAC	Soft Actor-Critic
SEC	U.S. Securities and Exchange Commission
SGD	Stochastic Gradient Descent
SR	Sortino ratio
UBAH	Uniform Buy and Hold
UCRP	Uniform Constant Rebalanced Portfolio
VaR	Value-at-Risk

# Contents

Abbreviations and Acronyms	v
<b>1 Introduction</b>	<b>1</b>
1.1 Problem statement . . . . .	2
1.2 Structure of the Thesis . . . . .	3
<b>2 Portfolio Management</b>	<b>4</b>
2.1 Modern Portfolio Theory . . . . .	4
2.2 Kelly Criterion . . . . .	6
2.3 Strategies . . . . .	7
2.3.1 Uniform Buy and Hold . . . . .	8
2.3.2 Uniform Constant Rebalanced Portfolios . . . . .	8
2.3.3 Follow the Leader . . . . .	9
2.3.4 On-line Moving Average Reversion . . . . .	9
2.4 Metrics . . . . .	10
2.4.1 Sharpe Ratio . . . . .	10
2.4.2 Sortino Ratio . . . . .	11
2.4.3 Omega Ratio . . . . .	12
<b>3 Reinforcement Learning</b>	<b>13</b>
3.1 Background . . . . .	13
3.2 Markov Decision Process . . . . .	14
3.3 Deep Learning . . . . .	17
3.4 Algorithms . . . . .	19
3.5 Policy Gradient . . . . .	22
3.6 Reinforcement Learning Based Trading Systems . . . . .	25
<b>4 Data</b>	<b>28</b>
4.1 Database . . . . .	28
4.1.1 Combining the Data . . . . .	29
4.2 Selection of Stocks . . . . .	30

4.3	Initial Feature Selection . . . . .	34
<b>5</b>	<b>Implementation and Training</b>	<b>37</b>
5.1	Problem Formulation . . . . .	37
5.2	Network Layout . . . . .	40
5.2.1	Technical Network . . . . .	40
5.2.2	Fundamental Network . . . . .	42
5.2.3	Time-lagged Return Network . . . . .	43
5.2.4	Output Layer . . . . .	43
5.3	Training of the Trading System . . . . .	45
<b>6</b>	<b>Results</b>	<b>49</b>
6.1	Comparison via Backtesting . . . . .	49
6.2	Allocations of the Trading System . . . . .	53
6.3	Robustness of Trading Systems . . . . .	57
<b>7</b>	<b>Discussion</b>	<b>62</b>
<b>8</b>	<b>Conclusions</b>	<b>66</b>
<b>A</b>	<b>Hyperparameters</b>	<b>72</b>
<b>B</b>	<b>Network Specifics</b>	<b>73</b>
<b>C</b>	<b>Trading Systems</b>	<b>74</b>

# Chapter 1

## Introduction

According to the efficient-market hypothesis (EMH), share prices reflect all information available to the market participants. The hypothesis states that when adjusting for the risk of an investment, it is impossible to consistently get excess returns. In other words, the only way to increase ones returns is by investing in riskier assets. This means that stocks are neither under- or overvalued, making most stock analysis obsolete.

In support of the EMH, Malkiel [2007] shows that more than two-thirds of professionally and actively managed portfolios have been outperformed by the passive S&P 500 Index, thought to represent the large companies of the United States stock exchange. Malkiel also argues that there was little correlation annually between the outperformers, meaning that the (roughly) third that managed to beat the market did not do it consistently.

There are however many critics of the EMH. For instance, Siegel [2010] argue that market crashes are poorly explained by the EMH. According to the hypothesis, the stock should be fairly priced at every time-instance, which is not possible when stock prices fall rapidly. He also notes that by “2007-2009, you had to be a fanatic to believe in the literal truth of the EMH”. Further, Coval et al. [2005] found that there are some investors who consistently and robustly beat the market. They also found investors who consistently underperform, demonstrating the complexity of the market.

Be as it may, adequate investment performance is essential for pension funds, portfolio managers and many others. Highlighted by the two opposing views on EMH, predicting returns in the stock market is an extremely challenging task.

Solving challenging problems is an area where Machine Learning (ML) has gained a lot of attention. ML and Reinforcement Learning (RL) have been around for a long time. However, due to improvements in data, hardware and algorithms, tasks of increasing complexity are being solved. At the moment,

some systems trained using RL match or even exceed human capabilities in a variety of tasks. Examples of these tasks are playing games [Mnih et al., 2015; Silver et al., 2017] and robotics [Levine et al., 2016].

## 1.1 Problem statement

Consequently, we take on the challenging problem of outperforming the market by utilising a sophisticated ML algorithm. For the attained model to be significant it should outperform simple portfolio management strategies that any investor could employ. Therefore, the feasibility of outperforming the basic strategies, including a passive index, using the proposed trader is examined. The problem is quantified as follows.

The portfolio is formed by hand-picking a number of stocks from the Information Technology sector of the S&P500 index. A risk-free asset, cash, is also added to the portfolio. The reason for limiting the selection of stocks to ones listed on the United States stock market, is to ensure a high trading volume and liquidity. The stocks are chosen from a specific sector to have similar underlying companies with interesting correlations and the IT sector is specifically chosen because of its generally higher volatility. The portfolio is further limited by only allowing long positions in the stocks, meaning that the value of the stocks is expected to increase. Consequently, during an extensive drop in stock prices, cash may be the only option with a non-negative profit and hence the best allocation as the trader naturally strives to maximise the value of its portfolio.

The portfolio is managed on a daily basis, utilising daily information of the market and allocating the assets of the portfolio accordingly. As transaction costs are included when selling and buying stocks, the optimal action in every time-step depends on the current allocation of the portfolio assets. This makes the problem a dynamic one, motivating the usage of RL techniques compared to other types of ML approaches.

A RL algorithm called Policy Gradient is utilised when training the portfolio management system. Following the convention of Moody et al. [1998], the system is referred to as the *trading system*. The trading system is trained on historical data and evaluated using *backtesting*, meaning that the performance is tested on historical data. In practise, the data is split into a training set, roughly consisting of data from years 2001-2017, and a test set, roughly with the data from 2018-2019. The backtesting is performed using the test set, and the performance of the trading system is evaluated by a risk-adjusted return of the portfolio.

There are two goals for this thesis. First, obtain an understanding of

currently used predictive variables when modelling stock behaviour. The predictive variables include both technical indicators, derived from trading information of the stocks, and fundamental indicators, derived from the quarterly reports of the companies. The second objective is to test the performance of the trained RL trading system compared to market indices and commonly used strategies, to see whether the automatic trader is able to outperform them. If this seems feasible, the research simultaneously works as a proof-of-concept for a similar trading system incorporating *intraday* market information, meaning information from within each trading day. Therefore, the trading system is designed to scale well with respect to the frequency of the information.

The scope of the thesis excludes derivation of new indicators for financial modelling, instead relying on existing research to provide relevant ones. Further, closer examination of the relevance for the documented indicators is excluded. Similarly, an existing algorithm in RL is relied upon when constructing and training the trading system.

## 1.2 Structure of the Thesis

In the more theoretical part, Chapter 2 presents relevant research and techniques in portfolio management. Chapter 3 looks at RL, starting from the basics, briefly introducing deep learning, discussing different algorithms and finally presenting research done on utilising RL in the financial markets.

In the experimental part, Chapter 4 presents the data, discusses the sources used to obtain the data, explores the stocks selected to the portfolio and examines the indicators found in the scientific literature. Chapter 5 presents the implementation in detail. First, the environment is characterised in mathematical terms suitable for RL. Next, the network architecture and the training of the trading system is presented. In Chapter 6, the performance of the trading system is compared to basic portfolio management strategies, the allocations of the trading system are examined and the robustness of the trading system is evaluated. The experiments performed in the thesis are discussed in Chapter 7 and the thesis is concluded in Chapter 8.

## Chapter 2

# Portfolio Management

### 2.1 Modern Portfolio Theory

This Section presents one of the most influential theories in portfolio management, Modern Portfolio Theory (MPT), introduced by Harry Markowitz [1952]. This thesis and the eventual assessment of the trading system is based largely on the ideas of MPT.

MPT is a mathematical framework for allocating assets in a portfolio. One key argument in MPT is that it is possible to form a portfolio such that the expected return of the portfolio is maximised for a given level of risk. The theory assumes that investors are risk averse so that if two portfolios provide the same expected return, the one with the lower risk will be preferred. As a result, to accept an increase in risk investors will require compensation in form of an increased expected return. Therefore also the converse holds: to obtain a portfolio with a higher expected return one must also take on more risk.

What is considered risk in practice, depends largely on the investor. A long-term investor might consider the risk of ruin, meaning that money is permanently lost, to be the real risk. For that investor a fluctuating portfolio value is of lesser concern, as long as the general direction is positive. Conversely, a short-term investor that plans to liquidate the portfolio within a couple of years, may consider a fluctuating portfolio value to be risk. The MPT takes the latter approach and quantifies risk as the variance of returns for the portfolio.

The task can be written as an optimisation problem

$$\begin{aligned}
 \min \quad & \sum_{i=1}^m \sum_{j=1}^m w_i w_j \sigma_{ij} \\
 \text{s.t.} \quad & \sum_{i=1}^m w_i \mathbb{E}[r_i] = \bar{r} \\
 & \sum_{i=1}^m w_i = 1 \\
 & w_i \geq 0 \quad \forall i \in \{1, \dots, m\},
 \end{aligned}$$

where  $m$  is the number of assets,  $w_i$  the proportion of the portfolio allocated in asset  $i$ ,  $\sigma_{ij}$  the covariance between assets  $i$  and  $j$ ,  $\mathbb{E}[r_i]$  the expected return of asset  $i$ , and  $\bar{r}$  the desired expected return for the investor. The objective function is the portfolio variance, which one aims to minimise while holding the expected return of the portfolio at the desired level. The portfolio weights are constrained to the feasible region, meaning that they sum to one and are positive, as we do not allow short positions, which would benefit from decreasing asset prices.

Another key argument advocated by MPT is that assets should not be evaluated separately, but together as a portfolio. The investor can reduce the risk of the portfolio, if the underlying assets are not perfectly positively correlated, meaning that the correlation coefficient  $\rho_{ij} < 1$ . Consider two assets with the same expected return and the same variance,  $\sigma_i^2 = \sigma_j^2$ . If one examines the assets separately, any combination of them is optimal, even one where everything is allocated to one of the assets. Conversely, utilising the MPT, we notice that the portfolio covariance decreases, as  $\sigma_{ij} = \sigma_i \sigma_j \rho_{ij} < \sigma_i^2$ . Investing in multiple assets, or diversifying ones investments, is therefore encouraged by MPT.

The solutions obtained from solving the optimisation problem for multiple different desired expected returns, form an *efficient frontier* of possible portfolios. These are so called Pareto optimal solutions, as one cannot choose a portfolio that would both have a higher expected return and a lower risk than any portfolio in the efficient frontier.

Figure 2.1 shows the efficient frontier, the possible portfolios and the individual assets, in the mean-standard deviation diagram. One notices the Pareto optimality of the efficient frontier when comparing it to the set of possible portfolios.

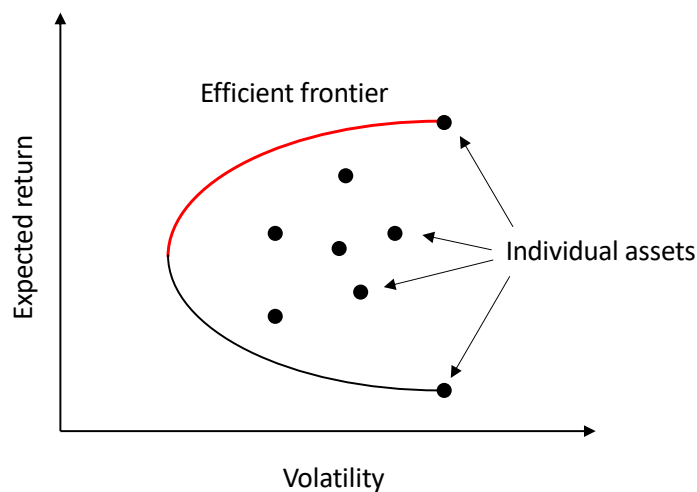


Figure 2.1: The individual assets (points), and the set of possible portfolios one can obtain from different combinations of them (area within curve) form a portfolio management problem. With Modern Portfolio Theory, one can solve the set of optimal solutions, called the efficient frontier (red curve). Depending on the investors desired expected return, a portfolio on the efficient frontier will be optimal.

## 2.2 Kelly Criterion

In recent years the Kelly Criterion, also called the Capital Growth Theory, has become part of the mainstream investment theory [Hagstrom, 2013]. While MPT focuses on portfolio selection in a single period, the Kelly Criterion does so in multiple periods. The Kelly Criterion is therefore suitable for portfolio management, where allocations are periodically managed.

The Kelly Criterion, presented by Kelly Jr. [1956], determines the optimal size of an investment, in order to maximise the expected value of the logarithm of wealth. Using the logarithm to quantify the value of money, was already suggested in 1738 by Daniel Bernoulli, later translated in Bernoulli [1954]. This essentially means that as the wealth increases, the value of increasing the wealth with a unit, decreases. Kelly revisited this idea in his paper, showing that it had notable properties. Maximising the one-period expected logarithm of wealth is equivalent to maximising the long run growth rate of wealth, or in terms of investments: the return, for each period. Therefore the optimal bet now does not depend on the past or the future, it is still optimal in the long run.

We define  $b_{win} > 0$  as the net odds for a bet. This implies that if the bet is won, the bettor receives  $b_{win} + 1$  times the bet-size, and if lost, the bettor loses the bet-size. The optimal bet according to the Kelly Criterion is then

$$f^* = \frac{b_{win}p_{win} - (1 - p_{win})}{b_{win}}, \quad (2.1)$$

where  $p_{win}$  is the probability of winning. The bet-size  $f^* \in (-\infty, 1]$  is expressed as the proportion of the total wealth. However, a negative bet-size indicates that the bettor should bet on the complement, which in some cases is impossible.

Let us consider a bet made on a coin toss where one receives 3 times the bet. Using the Equation above with the values  $b_{win} = 2$  and  $p_{win} = 0.5$ , the optimal bet becomes  $f^* = 25\%$  of the wealth.

The bets produced by the Kelly Criterion are superior in the long horizon. In practice however, the bets can be rather risky, especially in a shorter time-frame. This can happen because the bets produced can be a large fraction of the total wealth, and should that bet fail it takes longer to recover. Therefore, the fractional Kelly Criterion is often used. This means wagering a proportion  $\alpha \in [0, 1]$  according to the Kelly bet and  $1 - \alpha$  in cash, to decrease the riskiness while settling for a suboptimal growth rate. According to MacLean et al. [2011], many high profile investors act according to some fractional Kelly strategy. Furthermore, MacLean et al. [1992] argues that betting more than the Kelly bet is never advisable, as the growth rate decreases while the risk of bankruptcy increases.

## 2.3 Strategies

This Section presents some commonly used strategies, when sequentially managing a portfolio. The strategies are surveyed by Li and Hoi [2014].

The proportion of the portfolio invested in the  $m$  different assets at time  $t$  is denoted by  $\mathbf{w}_t$ . In mathematical terms,

$$\mathbf{w}_t = \{w_{it}\}_{i=1}^m \in [0, 1]^m, \text{ such that } \sum_{i=1}^m w_{it} = 1,$$

meaning that the portfolio weights are non-negative and sum to one, for each  $t$ . Furthermore,  $\mathbf{r}_t \in \mathbb{R}^m$  is considered to be the return of the assets at time  $t$ . This, is defined as the price-relative vector, in other words, the prices of assets at time  $t + 1$  divided by their prices at time  $t$ . Therefore, the return of

the portfolio from an investment made at time  $t$  is  $\mathbf{r}_t^T \mathbf{w}_t$  and the cumulative return from investing in  $n$  periods becomes

$$R_{[1:n]} = \prod_{t=1}^n \mathbf{r}_t^T \mathbf{w}_t. \quad (2.2)$$

The cumulative return is equivalent to the portfolio value at time  $n$  divided by the initial portfolio value. Possible transaction costs may occur that reduces the return, but for now they are only mentioned and in Section 5.1 introduced explicitly.

### 2.3.1 Uniform Buy and Hold

The Uniform Buy and Hold (UBAH) strategy consists of investing an equal portion in every asset and then holding the positions until the end.

The strategy starts off by investing  $w_{i1} = \frac{1}{m}$  into every of the  $m$  assets. Because the asset prices move, the allocation at time 2 will be slightly different. In mathematical terms, it will be

$$\mathbf{w}_2 = \frac{\mathbf{r}_1 \odot \mathbf{w}_1}{\mathbf{r}_1^T \mathbf{w}_1},$$

where  $\odot$  is the Hadamard, or elementwise:  $(A \odot B)_{ij} = A_{ij} B_{ij}$ , product. The division ensures that the new portfolio weights sum to one.

No rebalancing of allocations is done, meaning that the cumulative return for the portfolio will be

$$R_{[1:n]} = \left( \bigodot_{t=1}^n \mathbf{r}_t \right)^T \mathbf{w}_1.$$

As all positions are held until the end, no transaction costs need to be paid, apart from when obtaining the initial allocation.

### 2.3.2 Uniform Constant Rebalanced Portfolios

If one instead, chose to rebalance the weights at every time to match the initial allocation, one is employing the Constant Rebalanced Portfolios (CRP) strategy. Passive equity funds that track some market index, for instance the S&P500 index, are CRP that are rebalanced periodically.

In the special case, when one chooses to use the Uniform CRP (UCRP), the cumulative return becomes

$$R_{[1:n]} = \prod_{t=1}^n \mathbf{r}_t^T \left[ \frac{1}{m}, \dots, \frac{1}{m} \right]^T.$$

Naturally, constantly rebalancing the portfolio imposes transaction costs as one actively trades during the time period under examination.

Utilising a CRP based solution can be shown to be optimal, when the returns of the assets are independent and identically distributed [Cover and Thomas, 1991].

### 2.3.3 Follow the Leader

We move away from strategies that use some kind of case-insensitive allocation scheme. The idea of the Follow the Leader (FTL) approach is to increase the weights of successful assets. The approach can therefore be classified as a *Follow-the-Winner* type approach, where assets that have performed successfully in the past are expected to do so also in the future.

Following FTL, the allocation at time  $t$  is done according to the optimal CRP allocation at time  $[t_0, t - 1]$ . In other words, the allocation is done according to the constant allocation that would have resulted in the highest return in a historical time-interval. Mathematically,

$$\mathbf{w}_t = \mathbf{w}_{t-1}^* = \arg \max_{\mathbf{w}} \prod_{i=t_0}^{t-1} \mathbf{r}_i^T \mathbf{w},$$

where  $\mathbf{w}$  is still non-negative and sums to one. Thus assets that have performed well in the past will get a greater weight in the FTL inspired portfolio. The cumulative return can be calculated using Equation (2.2) and transaction costs will occur.

### 2.3.4 On-line Moving Average Reversion

The underlying assumption that suggests using an CRP based approach does, however, not always hold for the stock market. The *Follow-the-Loser* type approach instead transfers allocation from winners to losers. The approach is largely based on the assumption of mean reversion, meaning that deviations from the mean tend to correct themselves. For instance, a stock that trades over the average price is believed to decline in price in the future, while a stock trading under the average price is believed to increase in price.

An example of such an approach is the On-line Moving Average Reversion (OLMAR), published by Li et al. [2015]. The idea of mean reversion is apparent, as OLMAR predicts the following return  $\mathbf{r}_t$ , at time  $t$ , using a moving average, as

$$\hat{\mathbf{r}}_t = \frac{\frac{1}{l} \sum_{i=t-l+1}^t \mathbf{p}_i}{\mathbf{p}_t},$$

where the hat on  $\hat{\mathbf{r}}_t$  denotes that it is a prediction,  $l$  is the window size for the moving average and  $\mathbf{p}_j$  is the vector of closing prices at time  $j$ .

The allocation is then updated according to

$$\mathbf{w}_t = \arg \min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_{t-1}\|^2, \text{ such that } \hat{\mathbf{r}}_t^T \mathbf{w} \geq \epsilon,$$

where  $\epsilon$  is a threshold for the predicted return. The idea is to keep the previously allocated weights,  $\mathbf{w}_t = \mathbf{w}_{t-1}$ , if such a predicted return is over the threshold  $\epsilon$ . Conversely, if not, then find a new allocation that satisfies the condition not too far from the earlier weights. The cumulative return will again be calculated using Equation (2.2) and transaction costs will occur. Because of the careful updating of the portfolio weights, these costs should be more modest.

## 2.4 Metrics

From different theories and strategies of portfolio management, this Section presents metrics used to compare portfolios against each other. The idea of MPT, that risk is defined from the perspective of a short-term investor and returns should be evaluated together with risk, is incorporated in all the following risk-adjusted return metrics.

### 2.4.1 Sharpe Ratio

One of the most common performance metric of portfolio managers is the Sharpe ratio, developed by William Sharpe [1966]. It compares the returns of an investment to a risk-free asset and adjusts for riskiness.

Sharpe defined the ratio as

$$S = \frac{\bar{r} - r_f}{\sqrt{\text{Var}[r]}}, \quad (2.3)$$

where  $\bar{r}$  is the average return of the investment,  $r_f$  is the return of the risk-free asset and  $\text{Var}[r]$  is the variance of the returns.

The Sharpe ratio has its weaknesses. One is the assumption that the returns of investments are normally distributed, which is assumed when calculating the standard deviation of the returns. However, the distributions are often skewed and with abnormal kurtosis, making the standard deviation less efficient and the Sharpe ratio biased. Furthermore, as the ratio defines the risk as the standard deviation of the returns, both negative and positive volatility is seen as bad. Large positive swings in returns are therefore discouraged, even though a rational investor would welcome them.

In summary, the Sharpe ratio is an immensely popular metric, with many drawbacks, the most crucial being that both positive and negative volatility are seen as bad.

### 2.4.2 Sortino Ratio

In an effort to improve the Sharpe ratio, the Sortino ratio was introduced by Sortino and Price [1994]. It addresses the main weakness of the Sharpe ratio, by using Downside Deviation (DD), or negative volatility, instead of overall volatility as denominator.

The Sortino ratio is defined as

$$SR = \frac{\bar{r} - T}{\sqrt{\int_{-\infty}^T (T - r')^2 f(r') dr'}}$$

where  $T$  is the annual target return,  $r'$  a random variable and  $f(r')$  the distribution of annual returns.

The authors preferred the continuous version of the denominator over the simpler discrete version. The reason for that is that the continuous version is considered to attain the uncertainty of the realisation of returns, while the discrete version does not. Essentially, all the outcomes that could have happened, but did not, are better captured by the fitting of the continuous probability distribution. In practice, however, many that use the Sortino ratio employ the discrete, more simple, version for the downside risk. [Sortino and Forsey, 1996]

Compared to the Sharpe ratio, the assumption of normally distributed returns is no longer required. The distribution  $f$  can be fitted freely, thus allowing asymmetrical returns that often occur in the stock market. The fitting of the distribution does, however, not make the Sortino ratio bullet-proof as one needs to find a distribution that explains the returns properly. Furthermore, the problem with a negative numerator, that is shared but not as probable with the Sharpe ratio, exists. If the target return  $T$  is higher than the obtained one, the Sortino ratio becomes negative. In that case, a portfolio with a higher Downside Deviation is preferred to one with a lower DD, making no sense.

The Sortino ratio was the first element in Post-Modern Portfolio Theory (PMPT), designed to fill the gap between MPT and the realities of the investment market. The two major limitations of MPT are the ones explained above: the assumption of symmetric returns and the penalisation of positive volatility, both successfully overcome by the Sortino ratio.

In summary, the Sortino ratio is a definite improvement from the Sharpe ratio, overcoming most of its weaknesses, provided that the portfolio performs better than the target return  $T$ .

### 2.4.3 Omega Ratio

In an attempt to find an even better performance measurement, the Omega ratio was introduced by Keating and Shadwick [2002]. The Omega ratio retains all the information that the Sharpe ratio and Sortino ratio discards. Both those ratios contain only the first and second moment, mean and variance, of returns. Utilising the entire distribution, the Omega ratio contains all moments. Furthermore, the Omega ratio belongs to the PMPT, overcoming the same earlier limitations as the Sortino ratio did. In addition, the Omega ratio works consistently, even though the return of the portfolio does not reach the target return  $T$ .

The Omega ratio is defined as

$$\Omega(T) = \frac{\int_T^\infty [1 - F(r)] dr}{\int_{-\infty}^T F(r) dr},$$

where  $T$  is the target return and  $F$  is the cumulative distribution function of the returns. Similar as with the Sortino ratio, the distribution of returns is to be fitted and the corresponding cumulative distribution function used in the calculations.

One can examine the Omega ratio as a function of  $T$  and such compare different investments with one another, depending on what target return the individual investor has. The Omega ratio is also consistent with First-order Stochastic Dominance (FSD) of returns. For an investments to FSD another, it has to have a larger Omega ratio, regardless of the target return  $T$ . If an investment has FSD over another, then the first investment should always be selected by the investor, assuming that the investor prefers more return to less. According to Klar and Müller [2017], the Sharpe ratio is, however, not consistent with FSD, meaning that the ratio can prefer a portfolio that surely leads to smaller returns.

In summary, the Omega ratio is in some ways a further improvement of the Sortino ratio, by retaining all the information of the return distribution and being consistent regardless of target return  $T$ . The improvement is, however, somewhat marginal.

## Chapter 3

# Reinforcement Learning

### 3.1 Background

This Section presents the history of RL and some recent advances which have contributed to the popularity of the field. Even though the term *reinforcement learning* did not appear in engineering literature until the 1960s, its history stretches much further [Sutton and Barto, 2018]. RL as seen today stems mainly from two fields.

One of the fields is trial-and-error learning. The idea originates from the 1850s, from the psychology of animal learning. The principle is that decisions followed by a good or bad response have a tendency to be adjusted accordingly, also called the *Law of Effect*. Even though stemming from psychology, the idea of using trial-and-error slowly reached the field of engineering. In one of the first thoughts of Artificial Intelligence (AI), Alan Turing presented a computer system that worked similarly as the Law of Effect, in 1948.

The other field that RL stems from, optimal control, focuses on designing a controller for a dynamic system over time, such that some measure is optimised. One approach in the field was developed by Richard Bellman in the 1950s, called the Bellman equation. He later also introduced the Markov Decision Process, a discrete stochastic version of the optimal control problem. Both of these approaches are essential to modern RL.

The two aforementioned fields remained separate for a while, probably due to belonging to rather different disciplines. Finally, in 1980s the fields were fully combined to form RL. During the last decade, advances in the amount of data available and computational power and cost has laid the foundation to the soaring interest and recent advances in AI, ML and RL, taking place today.

Especially, Mnih et al. [2015] attracted a lot of attention towards RL, by

combining Deep Learning and RL successfully on the Atari 2600 games. They surpassed human-level control on 49 of the games, using the same algorithm and network architecture. In other words, they did not only perform well in one challenging task, but in a somewhat diverse range of task, starting from the same machine.

Another advancement occurred with a popular board game. The game Go was long considered to be a too challenging task for current AI technology, due to its vast search space and difficulty in evaluating moves and positions. However, Silver et al. [2016] managed to train an agent, called AlphaGo, using a combination of RL and supervised learning, that mastered the game. Having beaten one of the best Go players in the world, they decided to retire AlphaGo. Instead Silver et al. [2017] developed a new version, AlphaGo Zero. While the previous version was trained using human “expertise”, consisting of observations of human experts playing the game, the newer version was trained without any such information. AlphaGo Zero won against the champion-defeating AlphaGo 100-0.

The advances in RL are not limited to games, even though they may have received most attention. For example personalised web services, robotics and optimising memory control are other domains with recent promising advances [Sutton and Barto, 2018; Levine et al., 2016].

## 3.2 Markov Decision Process

This Section presents the mathematical framework used to describe a RL problem, namely a finite Markov Decision Process (MDP). MDPs are used to characterise sequential decision making, with both immediate and future rewards, into mathematical terms.

In a MDP the decision maker is called the *agent* and its counterpart, the *environment*. They interact at discrete time steps, the agent takes action  $a_t \in \mathcal{A}$  and the environment responds with a new situation, or state,  $s_{t+1} \in \mathcal{S}$  and reward  $r_{t+1} \in \mathcal{R} \subset \mathbb{R}$  for the agent to take action  $a_{t+1} \in \mathcal{A}$  based upon. The set of possible states, actions, and rewards are  $\mathcal{S}$ ,  $\mathcal{A}$  and  $\mathcal{R}$ , respectively, where  $\mathcal{A}$  is shorthand for  $\mathcal{A}(s_t)$ , as the actions available may depend on which state the agent is in. The finiteness, in finite MDP, comes from the assumption that the set of states, actions and rewards are of finite size. The system is visualised in Figure 3.1.

Quantifying the goal of the task, as a reward is one of the distinct features of RL. The agent simply strives to maximise the cumulative reward in the long run. Therefore, the reward must be defined such that maximising the reward accomplishes the exact goal of the task. Let us, for instance, consider

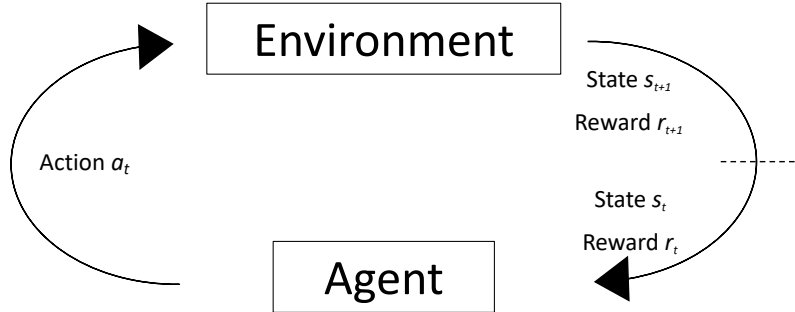


Figure 3.1: The interaction between the agent and the environment in a Markov Decision Process. The dashed line moves clock-wise to elicit new responses from the agent and the environment.

the game of chess. If the reward was quantified as taking the opponent's pieces, the agent may learn to do just that, but still lose the game. Instead the reward must be defined from the end result of the game, win or lose, and then mapped to the individual moves that lead to that result.

The agent decides which action to take in a specific state using a *policy*  $\pi$ . The policy can be either deterministic  $a_t = \pi(s_t)$  or stochastic, when the probability of selecting action  $a_t$  is  $\pi(a_t|s_t)$ . Likewise, the transition from state  $s_t$  and action  $a_t$  to the next state  $s_{t+1}$  can be deterministic  $s_{t+1} = f(s_t, a_t)$  or stochastic, when the probability of arriving at state  $s_{t+1}$  is  $P(s_{t+1}|s_t, a_t)$ .

If assumed that the MDP reaches a terminal state where the sequence ends at time  $T$ , the trajectory, or path, can be written as

$$\tau = s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_T,$$

where  $s_0$  is randomly sampled from a state-start distribution  $P_0(\cdot)$ . If assumed that both the policy and the transition are stochastic, the probability of said trajectory can be written as

$$P(\tau|\pi) = P_0(s_0) \prod_{t=0}^{T-1} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t), \quad (3.1)$$

given that policy  $\pi$  is followed.

The return of the trajectory needs to be a finite value and is therefore

defined as

$$R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t, \quad (3.2)$$

where  $\gamma \in [0, 1)$  is the *discount factor*. The discount factor is introduced for mathematical convenience but can be motivated, for instance by the assumption that getting money now is preferred to getting money in the future. This is reflected in the formula, as the reward now is worth the full amount while future rewards are discounted. The expected return thus becomes

$$\mathbb{E}_{\tau \sim \pi} [R(\tau)] = \int P(\tau|\pi) R(\tau) d\tau := J(\pi) \quad (3.3)$$

where the expectation is over  $\tau$  and  $\tau \sim \pi$  denotes that the trajectory is obtained following policy  $\pi$ . The central problem for RL to solve

$$\pi^* = \arg \max_{\pi} J(\pi),$$

is essentially finding the policy that leads to the highest expected cumulative return.

Other key functions in RL are the *value functions*. The action-value function gives the expected return when starting from state  $s$ , selecting action  $a$  and then acting according to policy  $\pi$

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s, a_0 = a]. \quad (3.4)$$

Another value function, the optimal action-value function, is similar except one acts according to the optimal policy after the initial action

$$Q^*(s, a) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s, a_0 = a]. \quad (3.5)$$

Using the optimal action-value function, the optimal action in every state can be computed as

$$a^*(s) = \arg \max_a Q^*(s, a),$$

resulting in the agent following an optimal policy.

All the value functions obey the self-consistent Bellman equations, essentially stating that the action-value now is the current reward plus the value of the next state. The action-value function at time  $t$  becomes

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim P(\cdot | s, a)} [r_t + \gamma \mathbb{E}_{a' \sim \pi(\cdot | s')} [Q^\pi(s', a')]],$$

which is obtained by combining Equations (3.2) and (3.4), if the terminal time  $T$  approaches infinity. Similarly, the optimal action-value function becomes

$$Q^*(s, a) = \mathbb{E}_{s' \sim P(\cdot|s,a)} [r_t + \gamma \max_{a'} [Q^*(s', a')]], \quad (3.6)$$

when combining Equations (3.2) and (3.5).

### 3.3 Deep Learning

When the set of possible states  $\mathcal{S}$  is large, finding the true optimal policy or optimal action-value function becomes impossible. The problem is solved via *function approximation*, meaning that desired function is approximated using examples from it. This Section introduces Deep Learning, a function approximation method.

Approximating the desired function can be done using an Artificial Neural Network (ANN), which has contributed to some of the most impressive recent advances in ML. The architecture of ANNs are inspired by the neurons in the brain, with synapses connecting them, thus creating vast and complex systems. When building complex ANNs and having them learn a task, it is called Deep Learning.

Figure 3.2 shows a simple example of an ANN. Every circle depicts a node containing a real number and the arrows are links between nodes. The network consists of an input layer, two hidden layers and the output layer. Every link is associated with a weight, and therefore the ANN is said to be parameterised by a set of weights,  $\theta$ , that together with the input of the network determines the output.

All the layers presented in the Figure are called *fully-connected* layers. Every node in the previous layer is connected to the next, and therefore the value in each node becomes a weighted sum of the previous layer. There are also no loops in the layer. Generally, there are many different types of layers, some are feed-forward like the previously introduced, while others contains loops and are therefore called recurrent layers. Another example of a feed-forward layer is the *convolutional* layer, which essentially connects only a subset of the previous layer to nodes in the following one. If one considers an image as the input, then only nearby pixels are connected to form a node in the following layer. This mapping from a subset of nodes to the next layer, is then applied to multiple different subsets of nodes, with the same weights. This means that spatial positions of the input are utilised in convolutional layers, making them well suited for image recognition, and other similar tasks. Long short-term memory (LSTM) layer is an example of

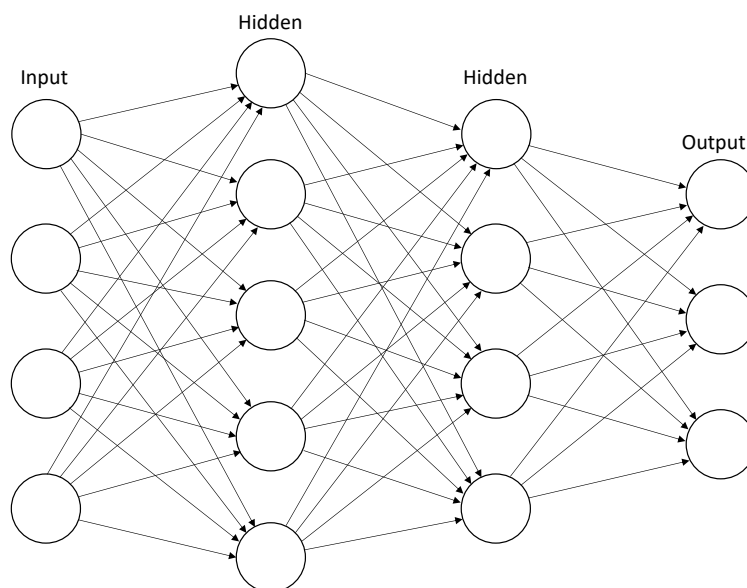


Figure 3.2: Example of an Artificial Neural Network. The network consists of 4 input nodes, connected to 5 nodes in a hidden layer, connected to 4 nodes in another hidden layer, finally connected to the output layer consisting of 3 nodes. All layers are fully connected so that each node has a path from every node in the previous layer.

a recurrent layer. LSTM layers can process arbitrary sequences of data with a loop connecting the output of the previous input, as an additional feature for the following input. One example of a use-case is interpreting sentences where the loop enables some kind of memory, connecting earlier information of words to the current input word.

Common for all kind of layers is to use *activation functions*. The activation function of a layer operates on each node in that layer, usually introducing non-linearity to the network. The activation functions are essential for the approximation of complex functions. In the simple example of Figure 3.2, the activation function would be applied after the weighted sum in each layer and without them the network would be a linear mapping from input to output. Commonly used activation functions are the Rectifier Linear Unit (ReLU)  $f(x) = \max(x, 0) \in [0, \infty)$  and a *sigmoid* function  $f(x) = \frac{1}{1+e^{-x}} \in (0, 1)$ . The ranges of the activation functions make the network act somewhat similar to the biological neurons that are either activated or not.

Sutton and Barto [2018] argue that the complex functions to be approx-

imated in RL may need multiple simple layers in a hierarchical structure to succeed. That means building deep structures, of many hidden layers, potentially combining different type of layers and activation functions.

Increasing the complexity of the network, however introduces a problem. Networks that contain a large number of weights, suffer from *overfitting*, meaning that the model learns too well the specifics of the training data, resulting in a model that does not perform well on new data. Several methods exist to mitigate this risk. For instance, penalising networks that use large weights, called *regularisation*, is one method. Examples of regularisation include l1 and l2-regularisation. The former induces sparse weights while the latter simply reduces weights towards origin. Another mitigation technique is to randomly remove nodes during the training, called *dropout*, that forces the model to only learn robust dependencies and not the rarely-occurring cases. Further, when training the model using small random subsets of the train data, overfitting usually occurs less.

Typically, the ANN learns using some version of Stochastic Gradient Descent (SGD). This means adjusting each weight in a directions such that some objective function is optimised. To do so, derivatives are utilised, which indicate how a small change in the variable differentiated with respect to changes the objective function. As the ANN is parameterised by multiple weights, the objective function depends on multiple variables and partial derivatives are used. Each weight is therefore updated in its own direction that optimises the objective function. Let us consider an objective  $f(\boldsymbol{\theta})$  that depends on weights  $\boldsymbol{\theta}$ , that is to be minimise. The weights are updated via

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}),$$

where  $\alpha$  is the learning rate, deciding the magnitude of each update, and  $\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta})$  is the gradient of function  $f(\boldsymbol{\theta})$  with respect to  $\boldsymbol{\theta}$ , in other words, the vector of partial derivatives with respect to each weight.

### 3.4 Algorithms

The taxonomy of RL algorithms is not straightforward, but the different algorithms can be divided roughly as in Figure 3.3, which covers relevant groupings and algorithms presented in this Section.

The hierarchically highest grouping is whether the algorithm is model-free or model-based. An algorithm is the latter if it has access to or tries to learn a model for the environment. This allows the agent to think ahead, as it can model what the consequences of actions are, and then select the optimal one. Furthermore, the sample efficiency, and as a result, training time is

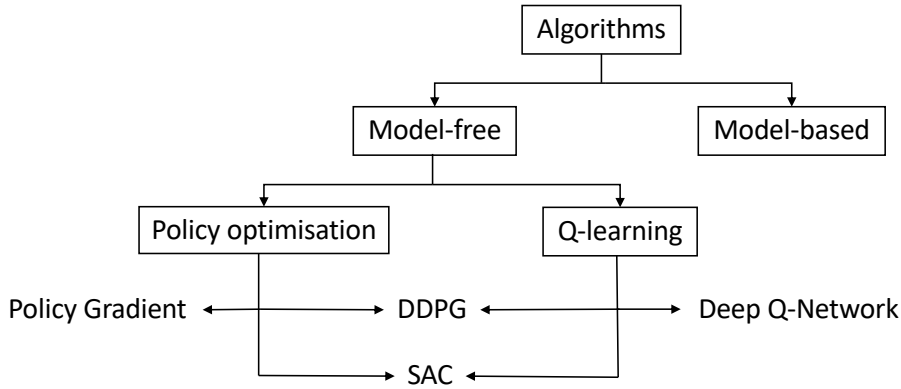


Figure 3.3: A taxonomy of RL algorithms. The boxes form the groupings and the free text represents the algorithms. DDPG stands for Deep Deterministic Policy Gradient and SAC for Soft-Actor-Critic.

improved. A major disadvantage is that a real model of the environment is rarely available, and therefore the agent must first learn the model. As a result, the agent may perform well with the learned model but poorly in the real situation. Algorithms are model-free if they do not attempt to utilise a model for the environment. Even though they have poorer sample efficiency, they are easier to implement and therefore currently more popular. This survey therefore focuses on model-free algorithms.

The two main approaches for algorithms in model-free RL are *policy optimisation* and *Q-learning*. The two approaches tie together the two previous Sections, as the different functions related to MDPs from Section 3.2 are now approximated with ANNs presented in Section 3.3.

Let us begin by presenting the policy optimisation approach. The policy  $\pi(\cdot|s_t)$  discussed in Section 3.2 can be rewritten using some ANN function approximator presented in Section 3.3, as  $\pi_{\theta}(\cdot|s_t)$  now parameterised by weights  $\theta$ . With the policy optimisation approach some version of the expected return of trajectories, defined as  $J(\pi)$  in Equation (3.3), is directly optimised. More specifically, the parameters  $\theta$  are updated using SGD with, for instance,  $J(\pi_{\theta})$  as the objective function. This update is always performed *on-policy*, meaning that all data used is obtained using the most recent policy  $\pi_{\theta}$ .

The Q-learning approach instead uses an ANN to model  $Q_{\theta}(s, a)$  parameterised by  $\theta$  that approximates the optimal action-value function  $Q^*(s, a)$ . The objective function to use with SGD is based on the Bellman equation for the optimal action-value function, presented in Equation (3.6). This can be done *off-policy*, meaning that data collected using any policy can be utilised.

The action in every state can then be obtained using

$$a(s) = \arg \max_a Q_{\theta}(s, a).$$

While policy optimisation directly optimises the desired policy, the approach is not sample efficient. The Q-learning approach on the other hand is, but it is less stable, as one is only indirectly optimising the performance of the policy. The Policy Gradient algorithm, which will be presented in the next Section, is an example of a policy optimisation algorithm while the Deep Q-Network, the algorithm Mnih et al. [2015] developed to master the Atari games, is an example of a Q-learning algorithm.

In an attempt to combine the strengths of the two approaches presented above, new algorithms have been presented. These, so called *actor-critic*, approaches consist of an actor, the policy network  $\pi_{\theta_1}$  and a critic, the Q-network  $Q_{\theta_2}$ , meaning at least two ANNs are used. Examples of actor-critic algorithms are the Deep Deterministic Policy Gradient (DDPG) algorithm, presented by Lillicrap et al. [2015], and the Soft Actor-Critic (SAC) algorithm, presented by Haarnoja et al. [2018]. Both algorithms gained a lot of popularity at respective release. The DDPG algorithm obtains good performance in multiple tasks, but, it is still rather unstable, and in high-dimensional tasks basic on-policy algorithms still perform better. According to Haarnoja et al. [2018], the SAC algorithm outperforms prior on-policy and off-policy algorithms, reaching state-of-the-art performance. Furthermore, the approach seems stable and sample efficient, successfully combining the strengths of the policy optimisation and Q-learning approach, while overcoming their weaknesses.

Another way to group algorithms is by whether the learning is done *off-line* or *on-line*. The former treats the data as a static dataset, while the latter allows for a dynamic one, where new observations arrive periodically. Off-line learning means dividing the data into a train and test set, collecting batches of observations from the train set and updating the parameters, to finally test performance on the test set. With on-line learning the train-test grouping becomes less clear. In its most extreme form, the algorithm sequentially predicts an observation, updates parameters on that observation and then discards the observation. The algorithm first processes the train set in this manner, and then the test set similarly, meaning that the model also updates between every observation in the test set. The gradients used to update the weights of the models are estimated using samples and therefore the off-line learning that uses more samples should attain a better estimation of the true gradient. Therefore, off-line learning usually results in a smoother training process. On the other hand, on-line learning estimates the gradient

using a single sample, resulting in a more volatile learning process that can potentially avoid getting stuck in local minima more efficiently. The approach can also adapt to a changing environment and more data can be utilised for training as opposed to testing. The generalisation estimate in on-line learning is however not nearly as reliable, as the model continuously changes.

### 3.5 Policy Gradient

This Section presents the Policy Gradient (PG) algorithm, which is a simple policy optimisation based approach. The PG algorithm is also called REINFORCE, after the algorithms presented by Williams [1992].

As stated in the previous Section, algorithms using the policy optimisation approach update the parameters of the neural network via SGD using some variant of the expected return of trajectories as the objective function. The PG algorithm being one of the simpler approaches uses the expected return of trajectories exactly as presented in Equation (3.3).

An expression for the policy gradient  $\nabla J(\pi_{\theta})$  is therefore needed. In the Equation for the expected return of trajectories, the expected value is taken over entire trajectories so that the gradient of the transition probabilities  $P(s_{t+1}|s_t, a_t)$  will be needed. PG, however, is a model-free approach and no model for the environment that could describe the transition from one state to another is available. Yet, the gradient can be rewritten so as not to include the gradient of the transition probabilities.

**Proposition 1.** *Assuming that the set of states, actions and rewards are finite, the state transitions are stochastic, the RL agent follows a stochastic policy  $\pi_{\theta}$  and the objective function  $J(\pi_{\theta})$  is as defined in Equation (3.3), then*

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi} \left[ R(\tau) \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right].$$

*Proof.* The gradient with respect to  $\boldsymbol{\theta}$  is

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}}) &= \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\tau \sim \pi} [R(\tau)] \\
&= \nabla_{\boldsymbol{\theta}} \sum_{\tau'} R(\tau') \cdot P(\tau' | \boldsymbol{\theta}) \\
&= \sum_{\tau'} R(\tau') \cdot \nabla_{\boldsymbol{\theta}} P(\tau' | \boldsymbol{\theta}) \\
&= \sum_{\tau'} R(\tau') \cdot P(\tau' | \boldsymbol{\theta}) \cdot \frac{\nabla_{\boldsymbol{\theta}} P(\tau' | \boldsymbol{\theta})}{P(\tau' | \boldsymbol{\theta})} \\
&= \sum_{\tau'} R(\tau') \cdot P(\tau' | \boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} \log P(\tau' | \boldsymbol{\theta}) \\
&= \mathbb{E}_{\tau \sim \pi} [R(\tau) \nabla_{\boldsymbol{\theta}} \log P(\tau | \boldsymbol{\theta})] \\
&= \mathbb{E}_{\tau \sim \pi} \left[ R(\tau) \nabla_{\boldsymbol{\theta}} \left( \log P_0(s_0) + \sum_{t=0}^{T-1} (\log P(s_{t+1} | s_t, a_t) + \log \pi_{\boldsymbol{\theta}}(a_t | s_t)) \right) \right] \\
&= \mathbb{E}_{\tau \sim \pi} \left[ R(\tau) \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t | s_t) \right].
\end{aligned}$$

The assumption of a finite number of states, actions and rewards, makes it possible to write the expected value as a summation over all the possible realisations  $\tau'$  of the random  $\tau$ . The linearity of the derivative and sum function, allow interchanging the differentiation and summation. Furthermore, the formula for the derivative of a logarithmic function is used before writing the expression as an expected value again. The last two lines comes from using the formula for the probability of a trajectory presented in Equation (3.1) and removing all terms that do not depend on  $\boldsymbol{\theta}$  as their derivative is zero, which finishes the proof.  $\square$

Calculating the exact gradient in Proposition 1 via the expected value over all possible trajectories becomes impossible as the number of states increases. Therefore, the gradient is usually estimated using the sample average

$$\frac{1}{S} \sum_{i=1}^S \left[ R(\tau'_i) \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t | s_t) \right],$$

where  $S$  is the size of the sample and  $\tau'_i$  is the realised trajectory of the  $i^{\text{th}}$  sample. Statistically this estimator is an unbiased and consistent estimator of the gradient. Optimising the parameters using an estimation of the gradient, instead of the actual gradient is what separates SGD from its exact variant:

Gradient Descent. All model-free algorithms are, by the same reasoning, forced to use some SGD approach as the environment is often too complex for the exact version.

As described in Section 3.3, the ANN basically consists of sequential operations applied to the input to finally produce the output of the network, like a function composition. Moving from input to output is called the *forward-pass*. Now that the network is to learn and  $\nabla_{\theta} \log \pi_{\theta}(a_t|s_t)$  is needed, one essentially looks at those operations in the opposite order to construct the partial gradients with respect to the parameters. It is called *backpropagation* when moving backwards in the network to calculate the effect of individual parameters. The chain rule can be applied when taking the partial derivative of these function compositions, ending up with a sequence of operations that ultimately comprises the partial derivative.

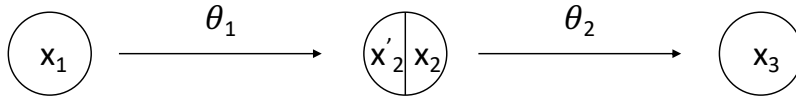


Figure 3.4: Simple example of network consisting of a fully connected layer of one node, with a ReLU activation function, followed by a fully connected layer with one node. The arrows show the direction of the forward-pass, meaning the direction from input to output.

Figure 3.4 shows a very simple ANN, where the circles denote nodes and the arrows operations. The node divided into two parts uses the ReLU activation function, making the input and output of the node different. In this example the forward-pass is

$$x_3 = x_2\theta_2 = \max(0, x'_2) \cdot \theta_2 = \max(0, x_1\theta_1) \cdot \theta_2.$$

To calculate the partial derivative with respect to  $\theta_1$  the operations are examined in reverse order and the chain rule used to obtain

$$\begin{aligned} \frac{\partial x_3}{\partial \theta_1} &= \frac{\partial x_3}{\partial x_2} \frac{\partial x_2}{\partial x'_2} \frac{\partial x'_2}{\partial \theta_1} \\ &= \theta_2 \mathbb{1}_{x'_2 > 0} x_1, \end{aligned}$$

where  $\mathbb{1}_{x'_2 > 0}$  is an indicator function that is equal to one, if  $x'_2$  is positive and zero otherwise. Worth to notice is that the the ReLU activation function is

not differentiable at zero, but it is defined to be zero at that point. Consequently, the expression presented is not the true derivative, but this does not affect its functionality.

Thus the tools needed to make the ANN learn can be combined. The parameters are updated using SGD, as

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{1}{S} \sum_{i=1}^S \left[ R(\tau'_i) \sum_{t=0}^{T-1} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t | s_t) \right],$$

where  $\alpha$  again is the learning rate and the backpropagation is done with respect to every parameter in the network, according to the idea presented above.

### 3.6 Reinforcement Learning Based Trading Systems

Most academic research on ML in finance utilises supervised learning techniques. This forces a two-step approach, where first predictions of the future are made and then these predictions are used to derive the actual trading action. This approach poses several limitations that may lead to suboptimal performance. RL instead conveniently merges the steps into a single one, where other important aspects also can be taken into account. As a result, RL has recently made considerable advances in the financial domain. In this Section the research done at the intersection of RL and portfolio management is presented.

Moody et al. [1998] presents a trading system using RL algorithms, applied to a single asset. They train the trading system using a novel objective function called the *differential Sharpe ratio*, for on-line learning. Moody et al. prefer on-line learning because of the speed of the convergence and because that allows the model to adapt to changing market conditions during trading. To calculate the differential Sharpe ratio, Moody et al. estimate the first and second moment of the return distribution using an exponential moving average

$$\begin{aligned} A_t &= A_{t-1} + \eta(r_t - A_{t-1}) \\ B_t &= B_{t-1} + \eta(r_t^2 - B_{t-1}), \end{aligned}$$

where  $\eta$  is the smoothing factor that controls the influence of new versus old observations and  $r_t$  is the return at time  $t$ . The authors initialise the estimated moments,  $A_0$  and  $B_0$ , as zero. Assuming that the risk-free rate is

zero, the Sharpe ratio, presented in Equation (2.3), at time  $t$  can be written as

$$S_t = \frac{A_t}{K_\eta(B_t - A_t^2)^{\frac{1}{2}}},$$

where  $K_\eta$  is a proportionality constant. When expanding to first order in  $\eta$ , the ratio becomes

$$S_t \approx S_{t-1} + \eta \frac{dS_t}{d\eta},$$

where only  $\frac{dS_t}{d\eta}$  depends on the return at time  $t$ . The differential Sharpe ratio is thus defined as

$$D_t := \frac{dS_t}{d\eta} = \frac{B_{t-1}(r_t - A_{t-1}) - \frac{1}{2}A_{t-1}(r_t^2 - B_{t-1})}{(B_{t-1} - A_{t-1}^2)^{\frac{3}{2}}}$$

and maximising it is equivalent to maximising the original Sharpe ratio. As the estimated moments were arbitrarily initialised to zero, one should let the estimates converge before using the formula. The authors find that maximising the differential Sharpe ratio leads to more consistent results than maximising profits. In support for using RL for these kind of tasks, they find that trading systems trained using either of the objective functions outperform trading systems based on supervised learning.

Jiang et al. [2017] also presents an on-line RL trading system, for which the performance is evaluated on multiple assets of the cryptocurrency market. The network architecture they propose for the trading system shares weights between assets, making it linear in the number of assets in the portfolio. The trading system therefore scales well and learns patterns that work for all assets, making the approach more robust. The authors find their trading system to outperform traditional portfolio-selection strategies, including those presented in Section 2.3, in three separate backtests.

Liang et al. [2018] implement the DDPG, PG and another policy optimisation approach called Proximal Policy Gradient, to a portfolio management task. They propose adding random noise to the input state, in order to obtain a more robust trading system, which they find to improve the return and Sharpe ratio, with a 95% confidence level, backtesting the trading systems on the Chinese stock market. They also find that the basic PG approach works better than the more advanced Proximal PG and DDPG approaches. They reason that the explicit and instantaneous reward of the actions: the change in portfolio value, make approximating the value function useless, and consequently the approaches that utilise the “critic” part ineffective.

The PG trading system also seems to outperform some basic trading strategies, including UCRP, a Follow-the-Winner and a Follow-the-Loser approach presented in Section 2.3.

Yu et al. [2019] apply the DDPG algorithm to the dynamic portfolio optimisation task. Furthermore, they add different modules to the trading system to improve performance. The first module provides a prediction of tomorrows returns, that can be used as an additional input to the RL model. The second module is a data augmentation module, that generates synthetic data of the market. The authors motivate this module by the scarcity of data available in the stock market, and hypothesise that it should increase the robustness of the model. The third module provides the model with the greedily calculated one-step optimal allocation, that the actor can be nudged towards. The authors provide a realistic backtest setting were the decisions are executed on following prices and not instantaneously as is usually done. The authors argue that their model outperforms a CRP and the model of Jiang et al. [2017] discussed above, in robustness and risk-adjusted returns. Their implementation of the model is however done using DDPG and not PG, which Jiang et al. originally employ. With the DDPG approach, the trading system does not even appear to beat the CRP strategy, which it does by a margin in the original paper by Jiang et al.

## Chapter 4

# Data

### 4.1 Database

This Section presents the databases used to obtain the data and the combining of the databases, to obtain an accurate trading environment in terms of information available. Two kinds of data are used in this research:

1. data related to the trading of the stock, provided by the stock exchanges,
2. data related to the underlying company of the stock, reported by the companies.

To obtain the historical information of the stocks, data from the Center for Research in Security Prices, LLC (CRSP) is used. This data contains daily information on, for instance, prices and volume of stock trades. Stocks included are ones listed on the New York Stock Exchange (NYSE) and National Association of Securities Dealers Automated Quotations (NASDAQ), among others.

To get fundamental information about the companies, Standard & Poor's Compustat North America database is used. The coverage of companies in this database is even wider than that in the CRSP database. Examples of information included in the database are income statements, balance sheets and cash flow statements released in the quarterly reports.

The CRSP/Compustat Merged (CCM) database from [CRSP/Compustat, 2020] consists of these two databases, with a highly accurate linkage to match companies and stock with one another. Though the coverage of the two databases starts before 1950, the data used in this research is from 2000-2019.

### 4.1.1 Combining the Data

Even though CCM provides the linkage between stock and company, the time matching still needs consideration.

When a company is traded on the stock exchange, it is required to follow the rules of the U.S. Securities and Exchange Commission (SEC). This means filing quarterly earnings reports of the financial state of the company, to the SEC. These reports are public and from where the Compustat database draws its data. In practice however, companies tend to make a public earnings announcement, before filing the reports to the SEC. These announcements are less regulated and the company can choose rather freely what to include. The announcements often contain a portion of the information that goes into the report.

When examining earning announcements during 2003-2004, Barron et al. [2016] found that 86% and 50.4% of earning announcements contained the balance sheet and cash flow information, respectively. They also noted that during the first quarter of 2008, 98% of the earning announcements contained the balance sheet information. This is in line with earlier results obtained by Chen et al. [2002], who also found that once a firm disclosed one balance sheet in combination with the earnings announcement, they tended to continue.

The Compustat dataset contains the complete information of the company's fundamentals and when the data was finalised, but not of the time when the information first became public. If only the Compustat data was utilised to merge with the CRSP data, the merge would be done using the date when the data was finalised to ensure that the data was indeed available to the public at that time. There may however be significant delays between the dates when the information was first published, then reported and finally finalised.

In order to not delay vital information from our model, the Electronic Data Gathering, Analysis, and Retrieval (EDGAR) system of SEC from [EDGAR, 2020] is utilised. With this system, one can access filings submitted by the companies.

By examining the timestamps of the quarterly reports filed to the SEC, using Form 10-K or Form 10-Q, the finalisation date found in the Compustat database can be improved upon. As the reports filed are public all information was available at that date. However, the stock market is not always open and therefore the matching of the data is done to the following day if the reports were filed after market close.

Moreover, since 28.03.2003 all public companies making a quarterly earnings announcement must submit Form 8-K to the SEC, containing the announcement [SEC, 2003]. Therefore, the original announcements are down-

loaded and scraped to observe what information was published already at the quarterly announcement. This information can then be merged earlier, at the date of announcement instead of at the date of the quarterly report.

As a result of the merge of the CRSP and Compustat databases, with the help of the information from the SEC, the data includes all information and contains a more accurate timestamp of when the information became public.

## 4.2 Selection of Stocks

This Section presents the process of selecting the stocks and simultaneously some characteristics of the stocks are explored.

The selection process is started from the S&P 500 stocks to ensure that the stocks have higher volume. This supports two big assumptions made on the market. The first one is that the investments made by the trading system has no influence on the market. The second one is that no *slippage* occurs, meaning that the price the trading system makes its decision based upon is still the same when it executes according to that decision. More specifically, the trading system makes the allocation decision based on the closing prices of a day but is still able to buy and sell stocks with those exact prices shortly after. Higher volume indicates a higher liquidity of the stock, which mitigates the risk that the two assumptions do not hold. It is critical that these assumptions hold, as they enable our backtesting procedure, explained in Section 5.3.

It is reasoned that since fundamental information of the companies is used as input to the trading system, comparable companies should be used. This is based on the assumption that some fundamentals are only important for certain companies. For instance, the R&D expense might be relevant for pharmaceutical companies, but less so for hotel chains. The taxonomy of the Global Industry Classification Standard is therefore used to select companies from a single sector. The sector chosen is the Information Technology (IT) sector, due to its larger volatility and well-known companies. The initial selection is further limited to companies that contain complete data from both the CRSP and Compustat database, for the time-interval 2001-2019.

We strive for a diversified selection of stocks, within the IT sector. Therefore stocks with the highest volatility and the lowest volatility, stocks with strong direct correlation and with weak direct correlation, and stocks with strong time-lagged correlation are selected to the portfolio.

All the returns examined in this thesis are adjusted for dividends, stock splits, et cetera, to derive comparable returns. Table 4.1 shows the selected stocks and some statistics of their rate of returns. For instance, Advanced

name	mean	mean-Y	min	max	sd
FISERV INC	0.098%	28.186%	-5.536%	6.164%	0.0116%
INTERNATIONAL BUSINESS M	0.001%	0.305%	-7.628%	8.864%	0.0126%
ORACLE CORP	0.037%	9.778%	-9.432%	10.18%	0.0131%
MICROSOFT CORP	0.115%	33.721%	-9.253%	10.452%	0.0143%
CORNING INC	0.054%	14.573%	-9.297%	11.331%	0.0153%
APPLE INC	0.106%	30.669%	-9.961%	8.198%	0.0153%
JUNIPER NETWORKS INC	0.028%	7.235%	-15.373%	10.245%	0.0168%
QUALCOMM INC	0.043%	11.578%	-15.251%	23.207%	0.0189%
XILINX INC	0.077%	21.473%	-17.077%	18.437%	0.0189%
WESTERN DIGITAL CORP	0.024%	6.194%	-18.182%	15.348%	0.0244%
ADVANCED MICRO DEVICES I	0.234%	80.605%	-24.229%	52.29%	0.0384%
Average for IT sector	0.082%	22.992%	-11.613%	13.63%	0.0172%

Table 4.1: Descriptive statistics of the rate of returns of the selected stocks. All values are reported in daily rate of returns, except the mean-Y column that is the daily average, converted to annual rate of return. The returns of the stocks are examined for the period 2014-2019 and sorted by the standard deviation. The last row contains the averaged values over all the stocks in the IT sector, contained in our dataset.

Micro Devices and Western Digital were selected due to their higher volatility while Fiserv and International Business Machines were selected because of their lower volatility.

Figure 4.1 presents the smoothed direct correlations between the stocks selected. For instance, Qualcomm Inc and Juniper Networks Inc were selected due to their lower direct correlations with the other stocks, while Oracle Corp and Microsoft Corp were selected because of their higher correlation. The rest of the stocks seemed to have interesting time-lagged correlations with the other stocks and were therefore included in the selection. These time-lagged correlations are presented in Figure 4.2. In addition to the eleven stocks presented above, the portfolio also consists of a twelfth asset: cash.

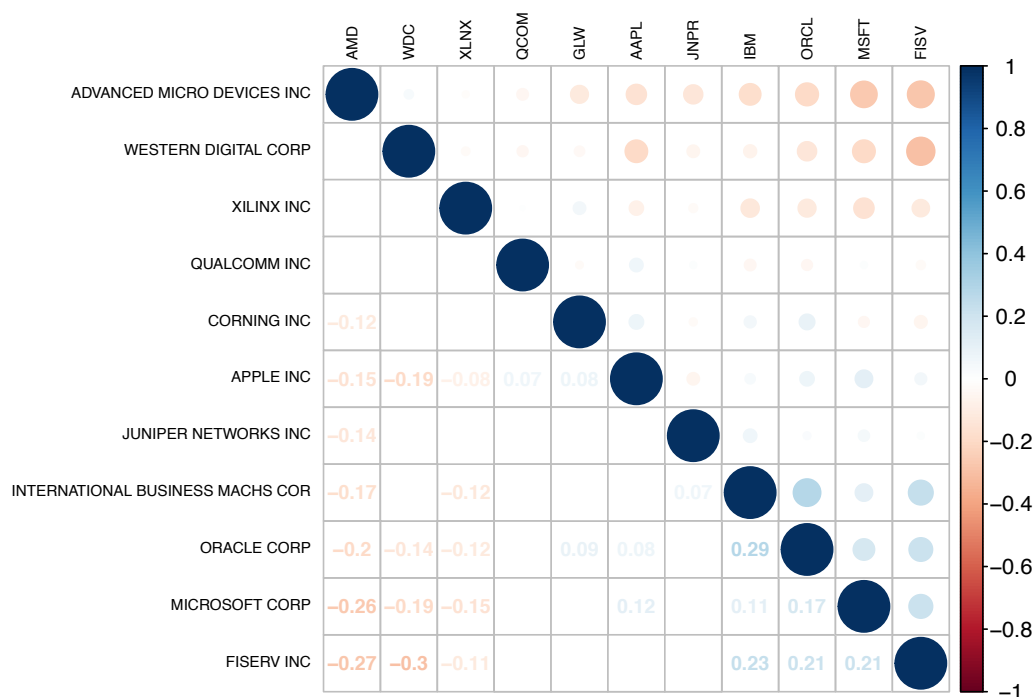


Figure 4.1: Correlation plot for the returns of the stocks selected. To the left the complete company name is presented, while above the ticker symbol is used. The correlations are calculated using the Pearson method, for data from 2014-2019 and the average return for the IT sector is subtracted from the returns of the individual stocks. The correlations are examined without time-lagging, meaning that instantaneous correlations are examined, and to smooth the results the correlations are calculated between the 3-day cumulative returns. Above the diagonal, all correlations are presented in glyph format, while below, the correlation coefficient is presented only for the significant correlations, at a 99% confidence level. As comparison, the average correlation, calculated similarly as above, between the companies in the IT sector is  $-0.019$ .

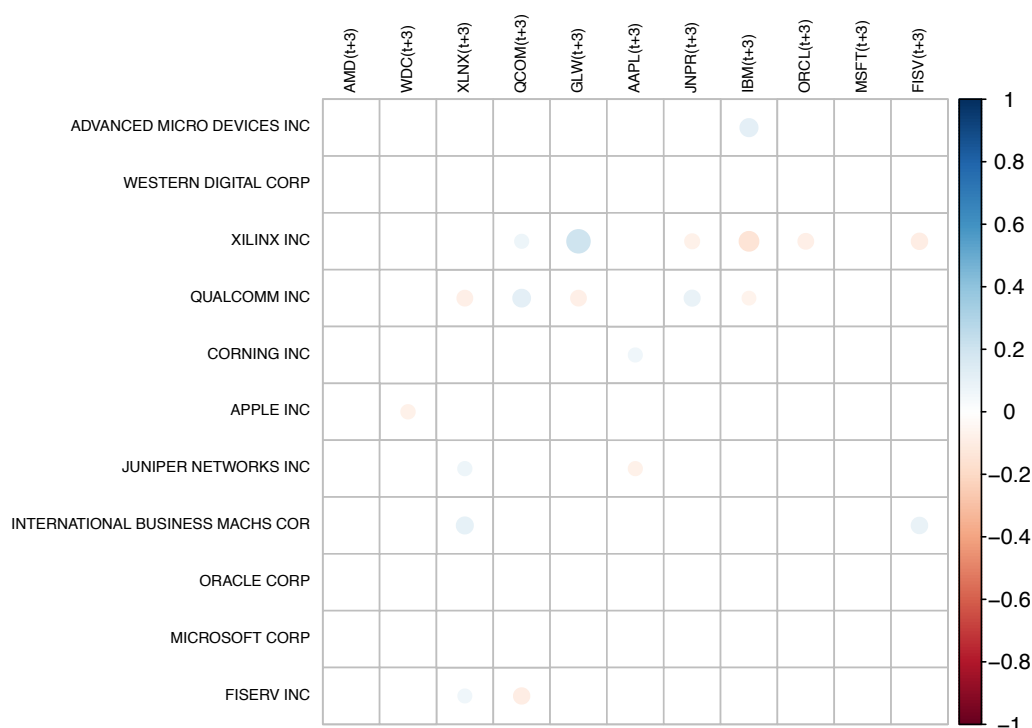


Figure 4.2: Time-lagged correlation plot for the returns of the stocks selected. The correlations are calculated using the Pearson method, for data from 2014-2019 and the average return for the IT sector is subtracted from the returns of the individual stocks. The correlations are examined with time-lagging and to smooth the results the correlations are calculated between the 3-day cumulative returns. More specifically, the correlation is calculated between the cumulative return from days  $t - 2$  to  $t$  for the stock in the row, and the cumulative return from days  $t + 1$  to  $t + 3$  for the stock in the column. A glyph representing the magnitude and direction of the correlation is presented only for the significant correlations, at a 99% confidence level.

### 4.3 Initial Feature Selection

This Section presents the initial set of features used as input to the trading system. These features are used to capture as much information about the market as possible, for the trading system to interpret.

Earlier approaches to use RL in portfolio management, for instance the research presented in Section 3.6, often use only basic information about the stocks such as closing price, highest price, lowest price and volume of the interval. However, in this thesis more complex features, including both fundamental and technical features, are used.

Fundamental features pertain to the fundamental information of the companies, for instance, the balance sheet. The majority of stock analyst believe that the stock prices reflect the fundamentals of the company, and therefore examining the fundamentals is the most fitting way to analyse stocks. As a consequence, much literature about fundamentals in predicting returns exists. The extensive replication study by Hou et al. [2018] is relied upon when selecting which features to use. The authors find that most of the 447 variables presented in earlier research are not significant. The fundamental variables use in this thesis are therefore selected from the set of variables deemed significant by Hou et al. The fundamental features used and their category is in Table 4.2.

Technical features can on the other hand be derived from the information about the stock. The stock analyst who use technical analysis when predicting the stock prices are therefore called chartists, as they study the past movements in stock prices. Even though technical analysis is not as popular as fundamental analysis, there is plenty of literature on the subject. Furthermore, these kind of features are better suited for shorter time trading systems, as the frequency of new information arrival is much larger. Along with the features with an asterisk (\*) in Table 4.2, an arbitrary selection of technical features, selected due to their general popularity, is also used. These technical features are in Table 4.3, along with a categorisation of what the feature attempts to identify. Most of the features are more thoroughly covered in Murphy [1999].

<b>Name</b>	<b>Category</b>
Standardised Unexpected Earnings	Momentum
*Cumulative Abnormal Returns	Momentum
Around Earnings Announcement Dates	Momentum
*Prior 11-month Returns	Momentum
Book-to-market Equity	Value-versus-growth
Earnings-to-price	Value-versus-growth
Cash Flow-to-price	Value-versus-growth
Enterprise Multiple	Value-versus-growth
Net Operating Assets	Investment
Net Stock Issues	Investment
Changes in Net Non-cash Working Capital	Investment
Changes in Financial Assets	Investment
Changes in Return on Equity	Profitability
Changes in Return on Assets	Profitability
Operating Profits-to-lagged Assets	Profitability
Cash-based Operating Profits-to-lagged Assets	Profitability
R&D Expense-to-market	Intangibles
*Seasonality	Intangibles
*Total Skewness	Trading frictions

Table 4.2: A subset of the variables found to be significant in predicting returns according to Hou et al. [2018], used in this research. The asterisk (\*) denotes that the variable is not actually a fundamental variable, as it only uses information related to the stock. The features are categorised by what the features attempts to capture.

<b>Name</b>	<b>Category</b>
Relative Strength Index	Momentum
Commodity Channel Index	Momentum
Stochastic Oscillator	Momentum
Directional Movement Index	Momentum
Aroon Indicator	Trend
Directional Movement Index	Trend
Moving Average Convergence/Divergence	Trend
Exponential Moving Average	Trend
Chaikin Accumulation/Distribution Line	Volume
On Balance Volume	Volume
Bollinger Band Width	Volatility

Table 4.3: Technical features used and selected due to their popularity. The features are categorised according to a taxonomy of 4 categories, by what the feature attempts to identify.

## Chapter 5

# Implementation and Training

### 5.1 Problem Formulation

This Section presents the problem in mathematical terms, the objective function for training the model and the model updating. We distinguish between the RL model and the trading system. We refer to the model when considering a model that produces allocation weights based on market information at a point in time, and to the trading system when considering the procedure how the model produces allocation weights at any point in time.

We define the number of stocks as  $m$  and the number of assets, meaning stocks and the cash option, as  $m^+$ . We repeat the definitions of Section 2.3, with  $\mathbf{w}_t$  being the allocation of the  $m^+$  assets at time  $t$ . Only long positions in the investments are allowed and all weights are therefore required to be non-negative. The weights should also sum to one for the portfolio allocation to be feasible. Furthermore,  $\mathbf{r}_t$  denotes the returns of the  $m^+$  assets at time  $t$ . This is the daily return of the stocks, defined as the closing prices at day  $t+1$  divided by  $t$ . As a result, if one were to invest according to  $\mathbf{w}_t$  at closing time on day  $t$  the portfolio would be worth  $\mathbf{r}_t^T \mathbf{w}_t$  at closing time on day  $t+1$ . Furthermore, the allocation weights at that time would be

$$\mathbf{w}'_{t+1} = \frac{\mathbf{r}_t \odot \mathbf{w}_t}{\mathbf{r}_t^T \mathbf{w}_t},$$

where the apostrophe (') denotes that the new allocation weights have not been decided yet.

Next, transaction costs are introduced into the return calculation. When the decision for the new allocation is made at closing time on day  $t$ , the allocation weights change from  $\mathbf{w}'_t$  to  $\mathbf{w}_t$  and transaction costs are charged. The convention of Moody et al. [1998] is followed and transaction costs at

time  $t$  approximated as

$$\mu_t = c \sum_{i=1}^m |w'_{it} - w_{it}|, \quad (5.1)$$

where  $c$  is the commission rate and the summation is over all stocks. Cash is not included in the summation, as one does not pay to solely change the cash allocation. Even though many brokers nowadays offer commission-free trading, a commission rate is imposed in this research. The average transaction cost that would occur with the selected stocks is estimated for a broker that offers an API. To allow some slippage, the estimate is slightly incremented to obtain the commission rate  $c = 10^{-4}$  used in this thesis.

The return from a decision made on day  $t$  then becomes  $\mathbf{r}_t^T \mathbf{w}_t - \mu_t$ . Furthermore, the cumulative return becomes

$$R_{[1:n]} = \prod_{t=1}^n (\mathbf{r}_t^T \mathbf{w}_t - \mu_t),$$

when investing for  $n$  days. For the trading strategies in Section 2.3, the transaction cost  $\mu_t$  is similarly subtracted from the cumulative returns defined.

Unlike games and robotics which tend to provide rewards sparsely, the portfolio management problem offers an instantaneous reward. We therefore agree with Liang et al. [2018] in that approximating a value function may be fruitless. We therefore discard Q-learning and actor-critic approaches in favour of policy optimisation approaches. Even many policy optimisation approaches estimate a value function and consequently, we select the PG algorithm in Section 3.5 that does not attempt to estimate a value function. The instantaneous reward can therefore be utilised directly when training the model.

Inspired by the differential Sharpe ratio presented in Section 3.6, a differential Sortino ratio is used as the objective function. The model is trained using batches to increase the stability of the training process. This means that some average of the daily returns over a batch is used when calculating the differential Sortino ratio. Therefore, we change the notation from time  $t$  to batch number  $k$ . The calculations are done similarly as in the differential Sharpe ratio, except that the second moment,  $B_k$ , is estimated using Downside Deviation only and the denominator changes slightly. The differential Sortino ratio becomes

$$DSR_k := \frac{B_{k-1}(r_k - A_{k-1}) - \frac{1}{2}A_{k-1}(r_k^2 - B_{k-1})}{(B_{k-1})^{\frac{3}{2}}}, \quad (5.2)$$

where

$$r_k = \left( \prod_{t=t_1}^{t_2} (\mathbf{r}_t^T \mathbf{w}_t - \mu_t) \right)^{\frac{1}{t_2-t_1+1}},$$

is the geometric mean of the returns from the decisions made in a batch that starts at time  $t_1$  and ends at time  $t_2$ . Batches consisting of 20 consecutive data points are used, so  $t_2 = t_1 + 19$ . This and the other variables that are predetermined and not optimised, for instance the commission rate  $c$ , are gathered in Appendix A.1.

Following Jiang et al. [2017], batches with different starts are valid and distinctive, even though they may overlap. This is due to the time-order of the data. The start time of the batch  $t_1$  is therefore drawn from a geometric distribution so that the probability of  $t_1$  is

$$P(t_1) = \beta(1 - \beta)^{t_n - t_1},$$

where  $t_n$  is the latest data-point in the training set and  $\beta$  is the distribution parameter. One can consider the selection process to start from the latest observation, picking it with a probability  $\beta$  and therefore not picking it with a probability  $1 - \beta$ . If the observation is not picked the same Bernoulli trial is performed on the previous observation, and so on. The probability density function is visualised in Figure 5.1.

The training of the model is done as follows. The batch start is drawn from the geometric distribution, as explained above. The allocation that the model starts from is always cash only. The model is then allowed to sequentially allocate the portfolio for every time-step in the batch. However, inspired by Liang et al. [2018] a small random noise is added to each observation in the batch before allocation to obtain a more stable and better performing model. The performance of the model is estimated on that batch, using the differential Sortino ratio presented in Equation (5.2). In MDP terms, this is the return of the trajectory,  $R(\tau)$ , and the parameters of the ANN are updated accordingly.

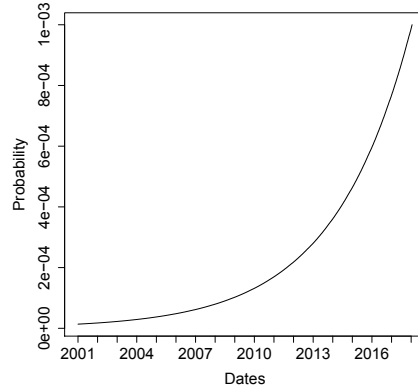


Figure 5.1: The probability density function for the random start date of a batch, if the latest observation in the training set is 20.02.2018 and the geometric distribution parameter  $\beta = 10^{-3}$ .

## 5.2 Network Layout

This Section presents the architecture of the Artificial Neural Network used to transform the inputted market information into allocation decisions for the portfolio. To quantify the information of the market, the trading system is fed with information about technical and fundamental features. Furthermore, to allow the model to utilise the time-lagged correlations between the stocks, examined in Section 4.2, the system is also fed with time-lagged returns of all the stocks. As a result three different types of information about the market are fed into the model.

We strive for a light-weight model that converges faster and does not require too much computation time. To obtain a light-weight model, the network is constructed so that the three input types have rather separate paths from input to output. We call these paths sub-networks. The entire network could, on the other hand, be called an *ensemble* network, because the outputs of the sub-networks are combined in a sort of voting procedure, to produce the output of the entire network. The ensemble network and the sub-networks are visualised in Figure 5.2. The sub-networks are defined as the path from an input until the  $m$  vector, that is combined with the other paths to become a vector of size  $m \times 1 \times 4$ . Given that the sub-networks are mapped independently with regards to the other inputs, the sub-networks can be trained separately. Furthermore, the training of the ensemble network will require less training when the sub-networks are pretrained. Consequently, the sub-networks are trained first and the pretrained weights then used in initialising the training of the ensemble network. Furthermore, four separate trading systems are obtained, amongst which performance can be compared.

As can also be seen in Figure 5.2, a fourth input is fed to the model, which is the current allocation weights for the stocks, in other words a subset of  $\mathbf{w}'_t$ . This allows the model to control the amount of transaction costs paid. Given the transaction cost in Equation (5.1), only the allocation for the stocks, not cash, is needed when calculating the transaction cost imposed. This fourth input is concatenated with the outputs of the sub-networks to eventually form the final output.

### 5.2.1 Technical Network

The sub-network transforming technical features into allocation weights is largely inspired by Jiang et al. [2017]. The technical features are fed in as 3-dimensional vectors. The first dimension is the number of stocks:  $m$ , the second is the number of past values for a feature and stock, denoted by  $n_{tech}$ ,

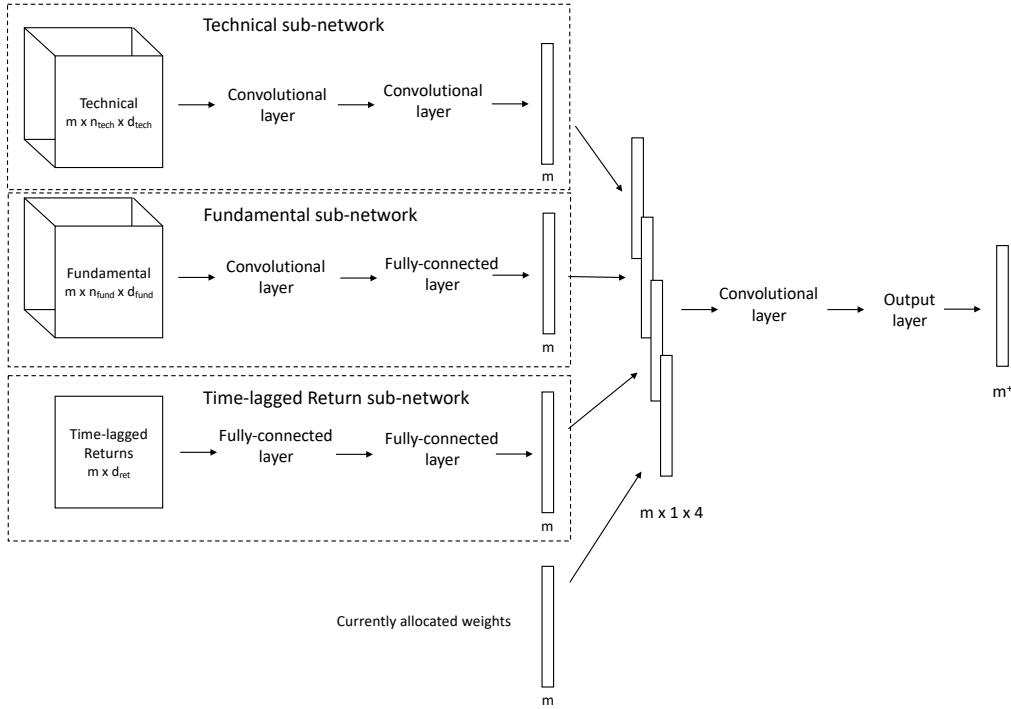


Figure 5.2: The ensemble network visualised. The inputs are to the left and the output to the right. In the middle of the Figure, the outputs of the sub-networks and the currently allocated weights for the stocks, all of size  $m$ , are concatenated together to form a vector of size  $m \times 1 \times 4$ . The number of stocks is denoted with  $m$ , the number of assets with  $m^+$ , the number of technical, fundamental and return features with  $d_{tech}$ ,  $d_{fund}$  and  $d_{ret}$ , respectively, and the number of past values used for every technical and fundamental feature by  $n_{tech}$  and  $n_{fund}$ , respectively. The sub-networks are visualised by dashed rectangles.

and the third,  $d_{tech}$ , is the number of features. By providing the network with multiple past values, the trading system can learn to interpret changes in features that seem to affect future returns. As visualised in Figure 5.2, the technical sub-network consists of two convolutional layers. The convolutional layer connects different subsets of nodes from one layer to the next using the same mapping, as described in Section 3.3. In this sub-network, the information from the different stocks are mapped separately. This means that all information regarding a stock remains separated from the information of the other, all the way from input to the output of the sub-network. Furthermore, the mapping from input to output uses the same weights regardless of stock. Consequently the model interprets certain combinations of features to influ-

ence prices of all stocks in the same manner. More in-depth details of the sub-network are in Appendix B, where the specifics of the entire network are presented. The output of the sub-network is the number of stocks,  $m$ , long.

Because ANNs usually learn better with normalised data, the normalisation is applied as follows. The technical indicators with an unconstrained range are normalised relative to their most recent value. For instance the closing price for a specific stock would be

$$\left[ 1 \quad \frac{p_{t-1}}{p_t} \quad \frac{p_{t-2}}{p_t} \quad \dots \quad \frac{p_{t-n_{tech}+1}}{p_t} \right].$$

In contrast, the intrinsically constrained technical indicators are transformed onto the range  $[0, 1]$ . For instance the Relative Strength Index that can obtain values between 0 and 100, is simply divided by 100 to obtain the desired range.

## 5.2.2 Fundamental Network

The sub-network for the fundamental features is presented next. The fundamental features are again fed in as a 3-dimensional vector. The first dimension is the number of stocks, the second,  $n_{fund}$ , is the number of last reported fundamental values plus one and the third,  $d_{fund}$ , is the number of features. The four previous values are used for the fundamental features. As the values are reported quarterly, this means values over a year. The plus one feature in that dimension is time since the latest fundamental value was reported, calculated in days. The second dimension,  $n_{fund}$ , for the input vector is therefore five. By providing the model with how recent the fundamental information is, the model can both separate fresh information from stale and learn to anticipate when new information is imminent.

The fundamental features are normalised by subsector, or industry, according to the Global Industry Classification Standard. Consequently, when a company releases new information of some fundamental feature, that information is compared to the other companies in the same industry, and via an affine transformation mapped to the interval  $[0, 1]$ . Value one means that the fundamental feature is the largest in the industry at that time, while zero means that it is the smallest. Furthermore, if another company in the same industry releases new information, causing the affine transformation to change, the values likewise change. For instance, a competitor releasing exceptional fundamental information can have an effect on the market's view of a company in the portfolio. The three older feature values are normalised with the same mapping so that the model can utilise how that fundamental feature value has evolved over time. The days since the last reported value

is divided by 70, roughly mapping it to the interval  $[0, 1]$ , as there are on average 62 trading days between quarterly earnings reports.

As visualised in Figure 5.2 of the ensemble network, the fundamental sub-network consists of a convolutional layer and a fully connected layer. The convolutional layer is thought to capture characteristics that work for all stocks equally, while the fully connected one can capture specifics of individual stocks. The former follows from the same reasoning as for the technical network, as information regarding the different stocks are examined separately. The fully connected layer can then draw from information regarding all the stocks, making it able to capture intermediate effects between the stocks via the fundamental values. To prevent the second layer from over-fitting, l1-regularisation and dropout are used, both presented in Section 3.3. The specifics of the sub-network are presented in Appendix B. The output of the sub-network is again the number of stocks,  $m$ , long.

### 5.2.3 Time-lagged Return Network

The transformation of the time-lagged returns, into allocation weights is presented next. The returns are fed into the trading system as a 2-dimensional vector. The first dimension is the number of stocks and the second,  $n_{ret}$ , is the number of interval lengths for the time-lagged returns. The initial selection of intervals are: three previous days, two previous weeks, the previous month and three previous months. More specifically, the values denote the cumulative return over that specific interval, for that specific stock, minus the average cumulative return for the IT sector. As the values describe returns, they are already adequately centred and scaled.

As visualised in Figure 5.2, the time-lagged return sub-network consists of two fully connected layers, and over-fitting is prevented by both l1-regularisation and dropout. This allows the network to fully exploit information stemming from all stocks, while still finding only robust and generalising characteristics in the data. More in-depth details of the sub-network are again presented in Appendix B and the output of the sub-network is  $m$  long.

### 5.2.4 Output Layer

The output layer finalising the allocation weights is presented next. The same output layer is used regardless if training one of the sub-networks or the ensemble network. However, the outputs of the sub-networks and the current allocation weights need to be combined first when training the ensemble, as can be seen in Figure 5.2. This combining is done using a convolutional layer,

meaning that the same weights, and therefore mapping, is used for all stocks. Consequently, the convolutional layer works as a weighted sum, or voting, of the four different  $m$  vectors. The resulting vector from the convolutional layer is of length  $m$ , the number of stocks.

If, on the other hand, one of the sub-networks is being trained, the output layer is directly applied to the output of size  $m$  from the sub-networks, in order to attain the allocation weights.

The output layer employs the Kelly Criterion, presented in Section 2.2. First the sigmoid activation function in Section 3.3 is applied to the vector of length  $m$ . The resulting vector belongs to  $[0, 1]^m$ . This can be interpreted as the probability of “winning” if one made the bet, or  $p_{win}$  as denoted in Equation (2.1). Furthermore, the net odds  $b_{win}$  are estimated for every stock by tracking earlier investments made by the trading system and the resulting win or loss obtained from them. The estimate is denoted by  $\hat{b}_{win}$  and estimated using an exponential moving average because the model continuously changes. If the model often performs well with a given stock, its  $\hat{b}_{win}$  will increase, resulting in larger investments in the future. The converse also holds: if the model invests poorly in a stock, its  $\hat{b}_{win}$  will decrease and future investments in that stock will be smaller or non-existent. As the model is trained to maximise the Sortino ratio, it will try to maximise the return and minimise Downside Deviation. This means that the allocation weights and further backwards, the probabilities of “winning” will be optimised accordingly. Therefore, by construction, the model will try to produce probabilities to accurately match whether or not wealth should be allocated to each asset.

The Equation for the optimal bet according to the Kelly Criterion is used to calculate the unnormalised allocation weights. As only long positions in the portfolio are allowed, negative allocations are converted to zero. Furthermore, the total allocation weights produced can exceed one, which in practice means investing with borrowed money, which we do not allow. Moreover, as described in Section 2.2, the Kelly Criterion bets can be rather risky. Consequently, to reduce risk, we do not allow the model to allocate more than 40% of the portfolio value to a single asset. Finally, the cash allocation must be deduced.

The allocation weights are finalised as follows. The unnormalised weights are limited to the range from 0 to 0.4. If the corresponding weights sum to below one, the remaining part of the portfolio value is allocated into cash. If, on the other hand, the capped weights sum to over one, they are scaled down so that they sum to one and consequently nothing is allocated in cash. As a result, the trading system never invests over the bets suggested by the Kelly Criterion and it only invests in cash if the other assets collectively do not appear appealing enough. The output layer therefore takes in a vector

of size  $m$  and outputs one of size  $m^+$ , the new allocation weights.

By construction, the policy of the trading system is deterministic, because it directly outputs the allocation weights. On the other hand, the PG algorithm described in Section 3.5 assumed a stochastic policy. This is however not a problem, as Silver et al. [2014] show that the stochastic policy gradient,  $\nabla J(\pi_\theta)$ , is a special case of its deterministic counterpart. When the randomness of the stochastic policy approaches zero, the two policy gradients are equivalent. Silver et al. [2014] also argue that in high dimensional action-spaces, the deterministic policy optimisation approaches may significantly outperform the stochastic ones. Consequently, our deterministic policy gradient approach is considered to be theoretically valid and having similar strengths and weaknesses as the stochastic PG algorithm.

Another key difference to the theoretical foundation is that our environment is continuous and set of states, actions and rewards are therefore infinite. However, the theory in Section 3.2, assumes a finite MDP. Sutton and Barto [2018] argue that even though most of the RL theory is restricted to the finite case, the ideas and methods apply more generally. Furthermore, many of the RL use-cases presented in Section 3.1 have successfully overcome this discrepancy.

### 5.3 Training of the Trading System

This Section describes the training process for the trading system and the programming used to carry out the experiments.

We hypothesise that the stock market changes dynamically, meaning that a strategy that is successful during some time period will most likely lose its edge over time. Therefore, an on-line approach that can adapt to changing market conditions is used when training the trading system, as is also done in the earlier research presented in Section 3.6. Compared to the on-line learning procedure presented in Section 3.4, a less extreme form of on-line learning is used. In the proposed approach the model is first trained as if using an offline approach until a good model is obtained. The performance is then tested on previously unseen data in an on-line manner, by sequentially retraining the model when new data becomes available. The on-line approach is the reason for distinguishing between the model, that is a point-in-time realisation of the trading system, and the trading system, that involves the retraining procedure of the model.

The entire process can be divided into two phases consisting of both training and testing. The first phase is the *hyperparameter* search in which parameters that control the training process are optimised. The second phase

	Static Training Set	On-line Testing Set
Hyperparameter Search	01.02.2001 - 21.09.2016	28.10.2016 - 28.03.2018
Backtesting	01.02.2001 - 20.02.2018	29.03.2018 - 30.12.2019

Table 5.1: Training and testing ranges used during the hyperparameter search phase and the backtesting phase.

is the backtesting phase, that supposedly estimates how well the trading system would perform with unseen, for instance real-world, data, also called the *generalisation* estimate. The intervals used for the training and testing in the different phases is in Table 5.1.

In the hyperparameter search the parameters that control the learning process are optimised. These hyperparameters are predetermined and static, meaning that they are not iteratively updated during the training process, as the ANN parameters are. The hyperparameters include, for instance, the learning rate used in SGD, the regularisation parameters used to improve generalisation as discussed in Section 3.3, and the feature selection process. The feature selection is done to distinguish the relevant features that should be used as input, from the set of initially selected features in Section 4.3. This is because irrelevant input features complicate and slow down the learning process.

As mentioned above, the sub-networks are first trained separately. A major benefit from this is that the hyperparameter search is remarkably simplified. This is because, for instance, the subset of those technical features in the feature selection process that work best for the technical sub-network will work best for the ensemble network as well. Consequently, the hyperparameter search is performed separately for all the sub-networks and then the separately obtained optimal hyperparameters are combined to train the ensemble network. Furthermore, the ANN parameters of the ensemble network can be initialised to assume the same values as for the trained sub-networks to speed up the training of the ensemble network.

In the hyperparameter search phase, the model is first trained on the static training set as described in Section 5.1. This training is done off-line until a model that performs well is found. We call 50 batches an epoch and allow the model to be trained for up to 400 epochs. The performance is then sequentially tested on the testing set, in an on-line manner.

Before presenting the proposed on-line testing, the gap between the training and testing sets is examined. A gap of 26 trading days is deliberately left between the two sets. When performing on-line testing, the observations are similarly moved from the testing range to the training range with a time-lag of 26 days, more thoroughly presented in Figure 5.3. This is done to miti-

gate risks involved in on-line learning. For instance *catastrophic interference*, when the ANN updates parameters such that previously learned information is completely and abruptly forgotten, can occur in an on-line setting. This is due to the model being continuously updated with random observations that may suddenly result in a below-par model.

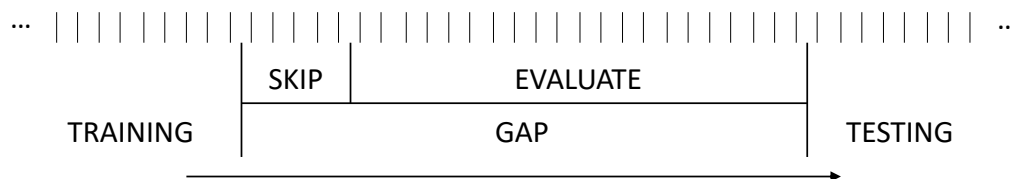


Figure 5.3: The data handling scheme, in on-line testing. The small vertical lines represent observations in their sequential order. Below these are the data partitions. The leftmost partition denotes the training set, SKIP the data discarded to unbiased the model performance estimate, EVALUATE the data used to evaluate the model performance on and TESTING the set used for testing the model. The arrow depicts that the partition splits are sequentially moved to the right.

Our procedure for the on-line testing is to start from the model that was trained in the off-line manner on the static training set. The model first produces allocation weights for the first observation in the testing set. The partition splits presented in Figure 5.3 are then moved one step to the right and the model is trained for 30 random batches on the new training set, as described in Section 5.1. Next, the resulting model is evaluated using the trading days in the gap before allowing it to produce the next allocation. As seen in Figure 5.3, days 1 to 5 of the trading days in the gap are discarded to prevent *leakage* when information in the train partition appears in the test partition too. Leakage can bias the test performance metrics slightly and discarding data from the split should prevent that. The remaining approximate month, days 6 to 26 of the trading days in the gap, can be used for the model evaluation. If the evaluation suggests a poor model, the model can be reverted back to the previously used one or simply retrained with random batches until a model with a good evaluation score on the gap is obtained. When the new model performs well enough, the allocation for the new first observation in the testing set is done, the partition splits shifted again and so on. The entire testing set is sequentially processed in this manner. The performance on the testing set is then evaluated using the allocations made for every observation in the testing set, stemming from equally many models.

Even though the model changes between every allocation in the testing

set the procedure enables more control, by allowing supervision and guidance of the on-line testing via the model evaluation on the trading days in the gap. This control adds to the reliability of the trading system, compared to the normal on-line procedure where no current model estimation is available. Furthermore, the procedure allows for more data to be used for training, compared to off-line approaches. For instance, when managing the portfolio today the trading system can use all historical data available, except for the preceding 26 days, to train the model used to allocate the portfolio with, for which the one-day return is obtained tomorrow.

In practice, one finds the optimal hyperparameters by training multiple trading systems using different hyperparameters, and comparing them with one another. The hyperparameters that produce the best trading system on the testing set are deemed to be optimal.

After obtaining the optimal hyperparameters as described above, the second phase of backtesting is begun. Using the optimal hyperparameters, the trading systems are trained as before, with the corresponding data intervals in Table 5.1. Based on the optimal hyperparameters four trading systems are trained, namely the three sub-networks and the ensemble network. Because all the data in the testing set is completely unseen until now, the performance on the test range serves as a generalisation estimate of the trading systems. Naturally, the generalisation estimate is for the on-line testing procedure of the model, and not only for the single model.

The experiments are implemented in the R software with the help of the software library Tensorflow. The former is used to control the process for training the trading system on a higher level and to visualise the results. The latter is used for training the model, therefore on a lower level. Tensorflow, for instance, performs automatic differentiation and therefore handles the backpropagation of the parameters in the ANN, covered in Section 3.5. The calculations of Tensorflow are performed on a Graphics Processing Unit (GPU), because the model has many input variables and a fair amount of parameters, in which case, the training is usually faster on a GPU than the normally used Central Processing Unit. Training and testing a trading system takes roughly 10 minutes, when using the GPU.

## Chapter 6

# Results

### 6.1 Comparison via Backtesting

This Section compares the performance of the proposed trading systems to the commonly used strategies in Section 2.3. All results are computed for the testing set in the backtesting phase in Table 5.1. Transaction costs are considered in all the calculations and the absolute returns are reported, meaning that the returns are compared to idle cash. The technical, fundamental and time-lagged return sub-networks are referred to as tech, fund and ret, respectively, and the ensemble network as ensemble, all described in Section 5.2.

	S	SR	Omega	DD	VaR	median	fAPV
tech	0.0846	0.1104	1.2012	1.13%	2.64%	<b>0.22%</b>	77.4%
fund	<b>0.0968</b>	<b>0.1275</b>	<b>1.2336</b>	0.98%	2.28%	0.17%	79.9%
ret	0.0921	0.1183	1.2162	0.92%	2.16%	0.15%	69.2%
ensemble	0.0830	0.1101	1.2004	1.27%	2.98%	0.14%	<b>86.6%</b>
UBAH	0.0816	0.1029	1.1871	0.98%	2.30%	0.19%	62.5%
UCRP	0.0766	0.0930	1.1687	<b>0.91%</b>	<b>2.12%</b>	0.16%	52.2%
FTL	0.0686	0.0853	1.1543	1.16%	2.71%	0.17%	58.2%
OLMAR	0.0337	0.0347	1.0613	1.24%	2.86%	0.11%	22.5%

Table 6.1: Performances of the trading systems and other trading strategies presented in Section 2.3. The best value is highlighted for every performance metric. The trading systems exhibit better values for all the risk-adjusted return metrics and for the final Accumulated Portfolio Value. No clear distinction between the trading systems and the strategies can be made based on the remaining metrics.

Table 6.1 shows the comparison between the strategies and the trading

systems in compact form. The strategies are the Uniform Buy and Hold, Uniform Constant Rebalanced Portfolio, Follow the Leader and On-line Moving Average Reversion strategy. The metrics, or columns, are divided into three categories, the first being risk-adjusted returns described in Section 2.4, with the Sharpe, Sortino and Omega ratio. Zero is used as the risk-free rate of return and a daily return that would yield an annual rate of return of 5% is used as the daily target return. The second category measures only risk, via Downside Deviation, the denominator used in the Sortino ratio, and Value-at-Risk ( $\text{VaR}_{5\%}$ ). The latter value depicts the lower bound for a daily loss that occurs with a 5% probability, based on the testing data. Conversely, a daily gain of more than the negative  $\text{VaR}_{5\%}$  values is 95% certain. The third category focuses on the rate of returns, via the daily median and the final Accumulated Portfolio Value (fAPV), or cumulative rate of return.

Based on the risk-adjusted return measurements in the Table above, the fundamental sub-network seems to perform best. In fact, all the trading systems seem to perform better than the strategies based on risk-adjusted return metrics. However, based on the second category the trading systems appear to be no less risky than the strategies, especially not the ensemble network that exhibits the poorest values in both the risk measurements. Finally, based on the fAPV, the trading systems seem to outperform the basic strategies, the best overall being the ensemble network while the OLMAR strategy seems to perform the poorest.

Figure 6.1 shows the cumulative returns of the trading systems and strategies during the backtest. The paths of the trading system largely resembles the average movement of the selected stocks, depicted by the Uniform Buy and Hold strategy and the Uniform Constant Rebalanced Portfolio. This can partly be explained by the restriction used in the output layer, to not allow a larger allocation than 0.4 for a single stock that forces the trading system to spread out the investments more. In contrast, the OLMAR strategy, which did not have a cap for the allocation weight often seems to deviate from the market movement in both directions. Furthermore, the ensemble network seems to perform well regardless of what time the period ends. The same does not apply for the time-lagged return sub-network as it seems to perform worse than most of the strategies for a large part of the interval.

The allocation weights for the trading systems during the backtest, is presented in Figure 6.2. The fundamental sub-network seems to change weights more seldom than the other trading systems. This is logical, as new fundamental information becomes available much more seldom, than the other type of information. The fundamental sub-network also uses a larger cash allocation compared to the other trading systems. As described in Section 5.2, the output of the ensemble network can be considered to be the result

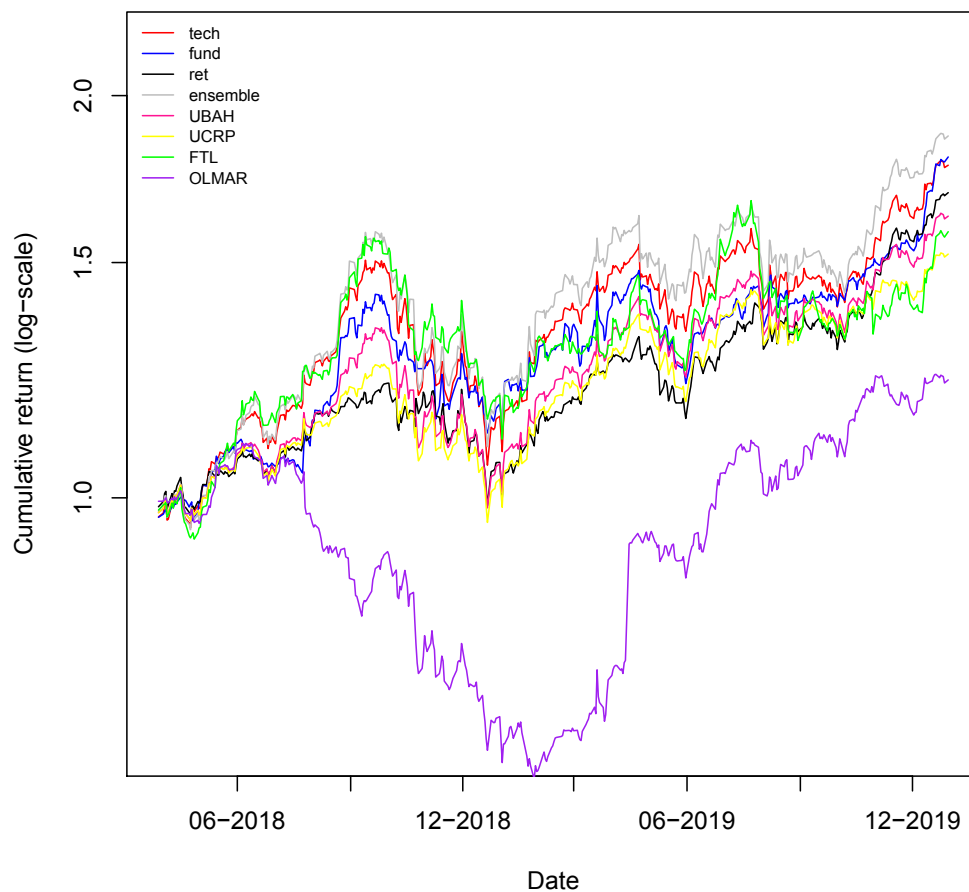


Figure 6.1: The cumulative returns of the different algorithms and strategies, for the backtest. The ensemble trading system seems to follow the market average in direction, while obtaining a slightly higher return.

of a voting procedure involving the sub-networks. Judging by the weight allocations, it appears that the vote of the technical sub-network is more dominant than the vote of the other sub-networks as the allocations made by the technical sub-network and the ensemble network resemble one another most.

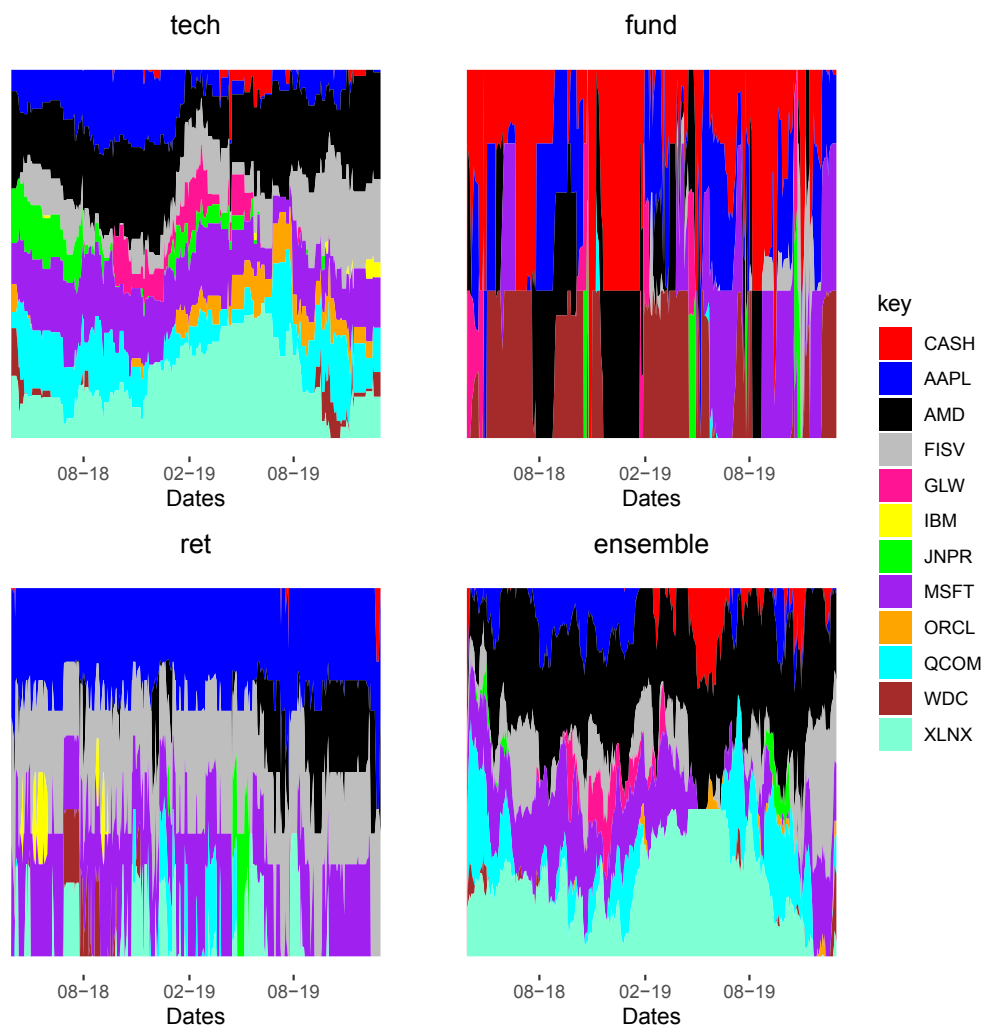


Figure 6.2: The allocation weights for the different trading systems. The weights are stacked on top of each other and the allocation weights always sum up to one.

## 6.2 Allocations of the Trading System

This Section examines the performance and the allocation weights during the backtest period for the ensemble network closer. The ensemble network in particular is chosen because it utilises all the information used to quantify the market and its performance based on the fAPV seems to be the best.

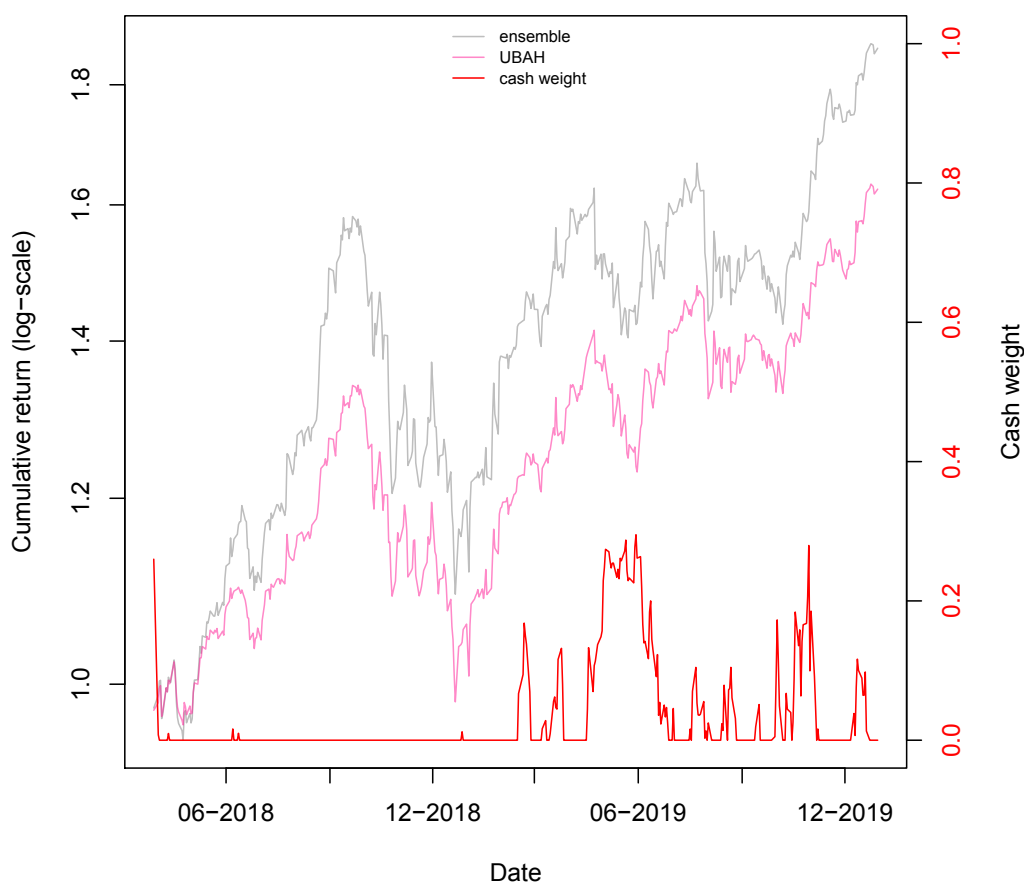


Figure 6.3: The cumulative returns of the ensemble network and the Uniform Buy and Hold strategy are plotted with the axis to the left. The cash allocation is plotted with the axis to the right. The ensemble network seems to generate excess return when the market is moving upwards, but market falls are not dampened using a larger cash allocation.

Figure 6.3 shows the cumulative return of the ensemble network in combination with its cash allocation weight. To be able to compare the return with some estimate of how the market, or a passive index, moves the return of the UBAH strategy is also visualised. The backtesting is started from an all-cash portfolio, similarly as in training described in Section 5.1. The small allocation in cash in the beginning of the period is therefore explained by the trading system not immediately buying stocks for all cash, but instead during two consecutive trading days. As seen in the Figure, the market dropped at the end of 2018, in which case the optimal allocation for the trading system would have been to allocate everything in cash. The system is however unsuccessful in doing this as nothing is allocated in cash during this time-period, which can largely be explained by the on-line scheme proposed in Section 5.3 involving the trading gap. Observations that exhibit a dropping market do not appear in the training set until after more than a month. If one however examines the following smaller drop in May 2019, the trading system allocates some money to cash. At this time, observations from the previous drop is in the training set and probably frequently selected to the batches, still drawn from a geometric distribution that favours more recent observations to earlier ones. Comparing the market with the ensemble trading system, it appears that when the market is bullish, i.e. going up, the trading system performs well and perhaps even better than the market. On the other hand, when the market is bearish, i.e. drops, the trading system seems to drop at least as much as the market does.

	min	max	mean
CASH	0.0000	0.2950	0.0333
AAPL	0.0000	0.2376	0.0544
AMD	0.0786	0.4000	0.3183
FISV	0.0000	0.4000	0.1021
GLW	0.0000	0.2416	0.0223
IBM	0.0000	0.0000	0.0000
JNPR	0.0000	0.1457	0.0083
MSFT	0.0000	0.3242	0.1455
ORCL	0.0000	0.0839	0.0052
QCOM	0.0000	0.4000	0.0954
WDC	0.0000	0.1830	0.0057
XLNX	0.0000	0.4000	0.2096

Table 6.2: Key measurements of the allocation stock-wise, for the ensemble network. Higher volatility stocks seem to be favoured by the model, while some stocks are not invested in at all.

Table 6.2 shows key statistics of the allocation weights during the backtest period. The only stock the model always invests in is AMD, which is also the stock with the highest average allocation, followed by XLNX and MSFT. The stocks that at some point reach the maximal allocation limit 0.4, are AMD, FISV, QCOM and XLNX. As discussed in Section 4.2, FISV was selected due to its lower volatility while AMD was selected due to its higher volatility. The two remaining, QCOM and XLNX, are both higher than average volatility stocks in Table 4.1. Consequently, there is only one higher volatility stock that the trading system does not allocate the maximal amount to, WDC, indicating that the trading system may prefer stocks with a higher volatility.

Figure 6.4 shows the trades that the trading system performs during the backtest. Some stocks are more actively traded and some less, or in the case of IBM: not at all traded. The trading system appears to buy and sell very quickly, as a green arrow is often immediately followed by a red one. The quantities of the orders are however not visualised in this Figure, making it possible for one of the orders to be significantly larger than the other. The stock with the highest cumulative return is AMD, which the model seems to capture based on the previous Table. Also MSFT and XLNX that were popular with the model offer decent cumulative returns. The number of orders the trading system makes per month is roughly 100, corresponding to approximately 5 per day, all executed at closing time of the market.

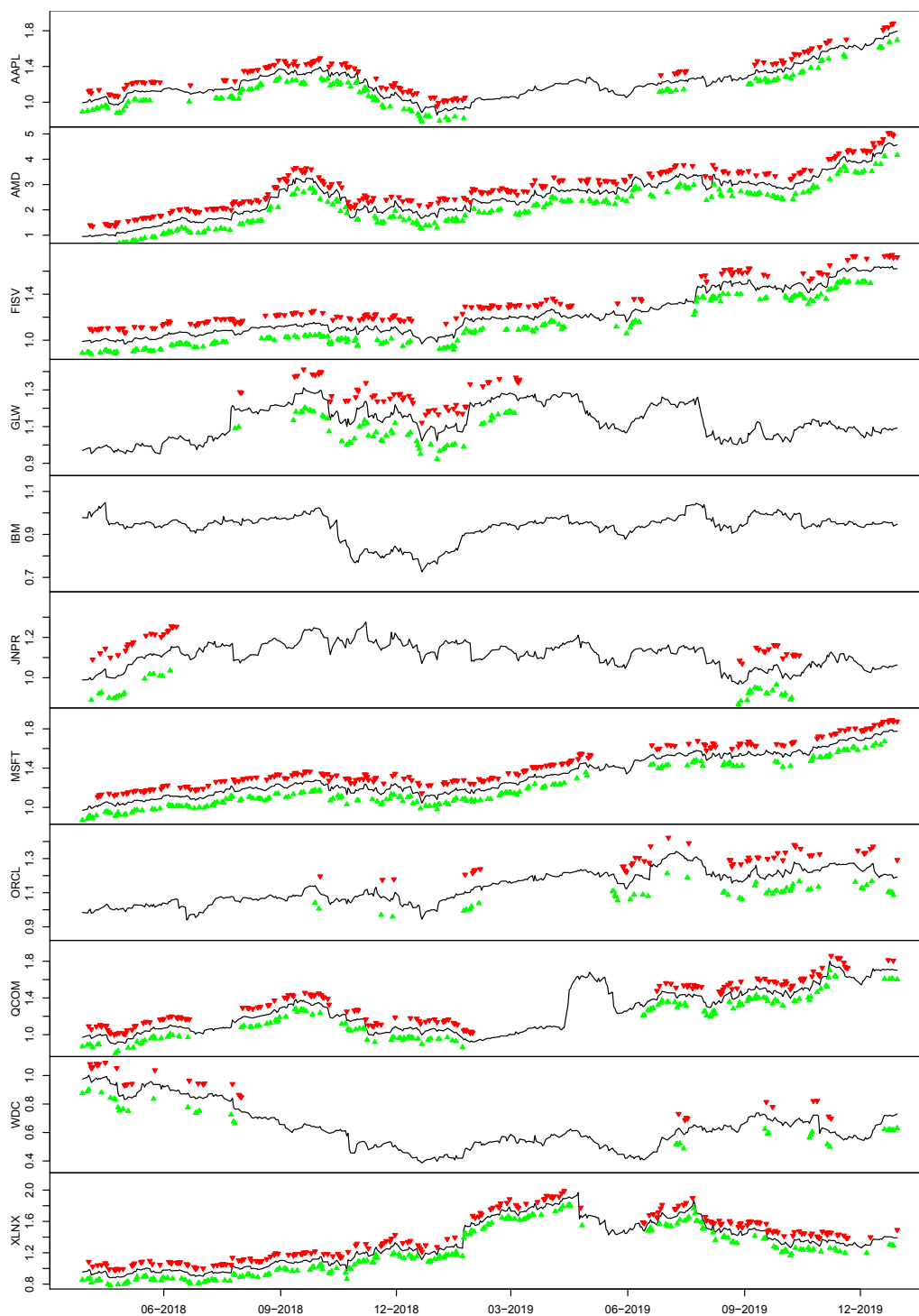


Figure 6.4: The cumulative return of the individual stocks, with green and red triangles denoting when the ensemble trading system increases and decreases the allocation in that specific stock, respectively. Due to the possible down-scaling in the output layer described in Section 5.2, the change in allocation need not stem from a change in the trading system’s evaluation of the stock.

### 6.3 Robustness of Trading Systems

This Section provides an estimate of the robustness of the trading systems. As described in Sections 5.1 and 5.3, there is plenty of randomness involved in the training of a trading system. The four trading systems in the previous Sections were only single realisations of all the possible trading systems. Consequently, 100 trading systems were trained for each of the technical, fundamental, time-lagged return and ensemble network, to obtain an understanding of the robustness of the proposed trading systems. Each trading system presented in the previous Sections was the trading system that obtained the median performance on the fAPV from the 100 trading systems trained, respectively.

	measure	S	SR	Omega	DD	VaR <sub>5%</sub>	median	fAPV
tech	min	0.0479	0.0527	1.0943	1.00%	2.34%	<b>0.08%</b>	33.3%
	max	0.1014	0.1372	1.2520	1.18%	2.76%	0.26%	99.3%
	mean	0.0847	0.1100	1.2006	1.08%	2.53%	<b>0.19%</b>	74.5%
	median	0.0870	0.1135	1.2071	1.09%	2.54%	0.19%	77.5%
fund	min	0.0095	-0.0017	0.9971	0.86%	1.99%	0.00%	0.3%
	max	<b>0.1504</b>	<b>0.2235</b>	<b>1.4204</b>	1.33%	3.12%	<b>0.30%</b>	<b>198.8%</b>
	mean	0.0849	0.1121	1.2053	1.12%	2.61%	0.14%	82.3%
	median	0.0882	0.1166	1.2127	1.12%	2.60%	0.14%	80.3%
ret	min	0.0524	0.0550	1.0987	<b>0.73%</b>	<b>1.70%</b>	0.04%	30.4%
	max	0.1288	0.1787	1.3325	<b>1.07%</b>	<b>2.49%</b>	0.25%	105.0%
	mean	<b>0.0921</b>	0.1188	1.2176	<b>0.93%</b>	<b>2.17%</b>	0.16%	70.3%
	median	<b>0.0924</b>	0.1186	1.2166	<b>0.93%</b>	<b>2.18%</b>	0.16%	69.3%
ensemble	min	<b>0.0544</b>	<b>0.0637</b>	<b>1.1143</b>	0.89%	2.07%	<b>0.08%</b>	<b>42.2%</b>
	max	0.1171	0.1639	1.3032	1.36%	3.18%	0.29%	131.7%
	mean	0.0904	<b>0.1203</b>	<b>1.2199</b>	1.15%	2.68%	<b>0.19%</b>	<b>88.0%</b>
	median	0.0908	<b>0.1205</b>	<b>1.2201</b>	1.14%	2.68%	<b>0.20%</b>	<b>86.8%</b>
UBAH	-	0.0816	0.1029	1.1871	0.98%	2.30%	0.19%	62.5%

Table 6.3: Key measurements of the trading systems obtained when running each training process 100 times. The best value in a column per measure is highlighted and the UBAH strategy is appended for reference. The entire list of trading systems can be found in Appendix C. The time-lagged return trading system seems to perform best based on the riskiness measures while the ensemble network seems to be the best trading system based on the returns and most risk-adjusted returns.

Key statistics of all the trading systems trained are in Table 6.3. The ensemble network seems to be the best based on minimum, average and median of the Sortino ratio, the Omega ratio and the final Accumulated Portfolio Value. The maximal value in those columns is however obtained

by the fundamental trading system. Nevertheless, the fundamental trading system appears to be a rather risky trading system in view of its minimal values. Performing both exceptionally well and poorly makes it suitable for a risk-seeking investor. In terms of risk, the time-lagged return trading system seems superior. Interestingly, the Sharpe ratio and the other risk-adjusted returns favour different trading systems based on the mean and the median measurement. This could be connected to one of the weaknesses of the Sharpe ratio described in Section 2.4.1.

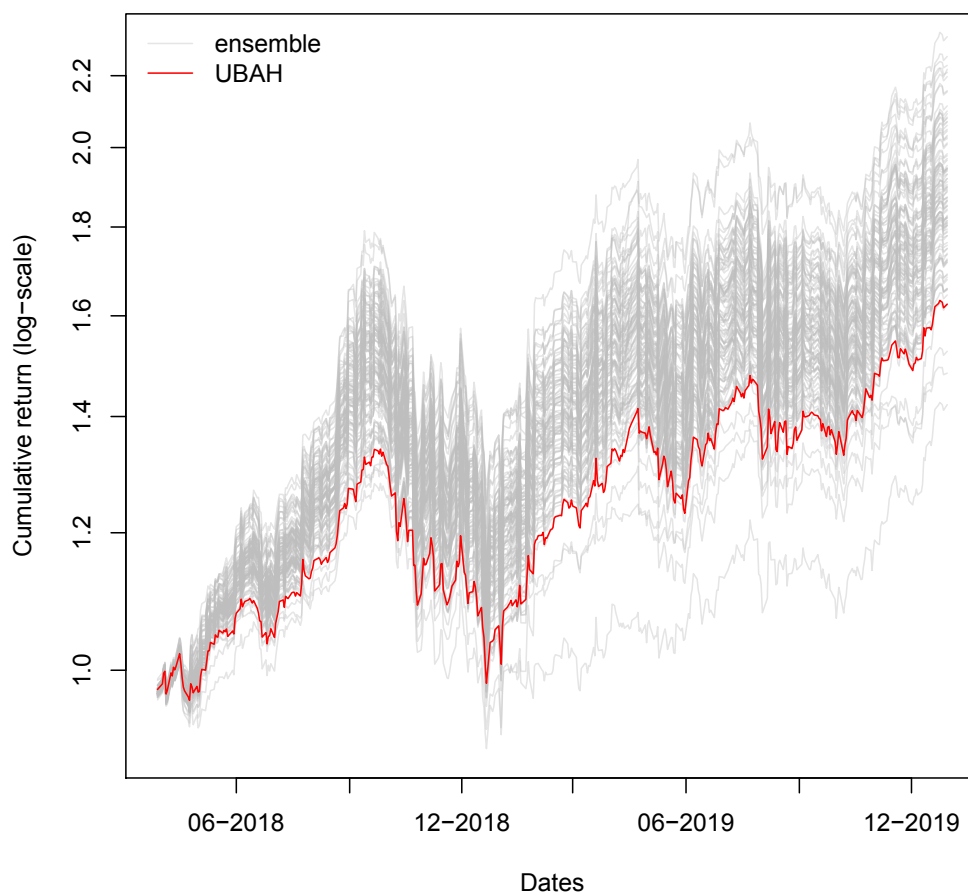


Figure 6.5: The cumulative returns of the 100 ensemble trading systems and the Uniform Buy and Hold strategy for the backtest. The ensemble networks seem to generate excess return in most of the runs.

The ensemble trading system seems to be the most robustly well performing trading system and therefore Figure 6.5 shows the cumulative returns of the ensemble trading systems. The returns seem to follow the market return in direction while gaining some additional return compared to the market. Some trading systems, however, perform worse than the market.

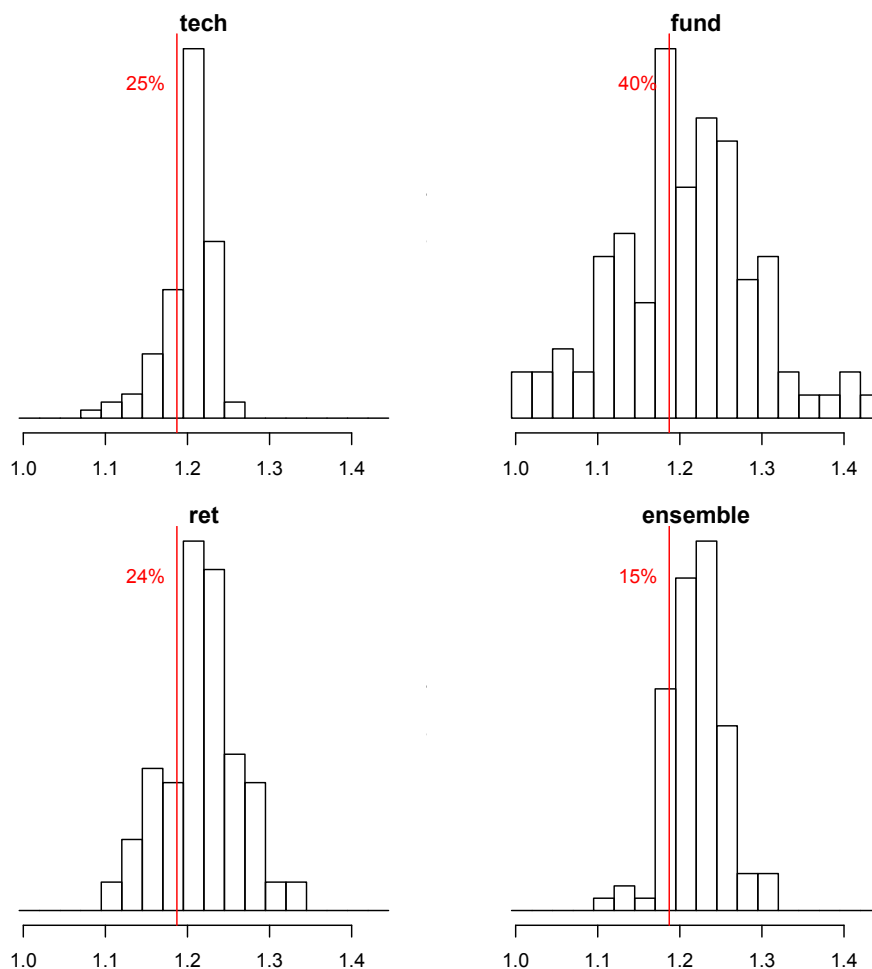


Figure 6.6: The distribution of the Omega ratio calculated with a daily target return corresponding to an annual rate of return of 5%, for the different trading systems. The red vertical line corresponds to the Omega ratio obtained using the UBAH strategy and the corresponding percentage depicts the proportion of trading systems that performed worse than that. The volatility for the fundamental trading system is the largest while the ensemble network seems to most robustly beat the UBAH.

Figure 6.6 shows the Omega ratio of the trading systems compared to the

UBAH strategy. Compared to the others, the fundamental trading system seems more volatile in performance and to less frequently obtain a better Omega ratio than the UBAH strategy. The technical trading system appears to be more robust but its distribution is negatively skewed meaning that larger Omega ratios are not reached. The Omega ratio of the ensemble network seems to most robustly obtain a better Omega ratio than the UBAH strategy, but the larger Omega ratios are still absent.

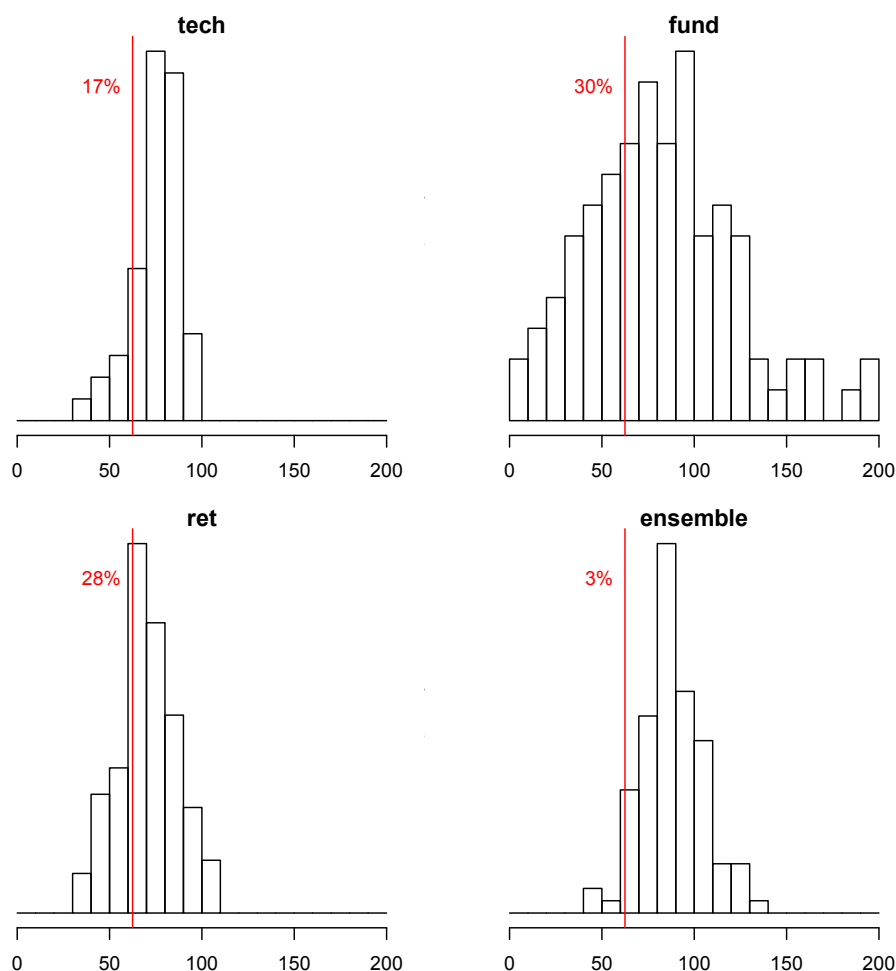


Figure 6.7: The distribution of the final Accumulated Portfolio Value, for the different trading systems. The red vertical line corresponds to the cumulative return obtained using the UBAH strategy and the percentage depicts the proportion of trading systems performing worse than that. The fundamental trading system appears to be volatile while the ensemble trading system obtains a better fAPV than UBAH in 97% of the runs.

Figure 6.7 shows the fAPV of the trading systems compared to the UBAH. All the trading systems except the time-lagged return are more frequently beating the UBAH based on this metric compared to the Omega ratio discussed above. This is due to the time-lagged return system being the least risky one in Table 6.3, even less so than the UBAH strategy on average. The fundamental trading system appears to be the most volatile based on this performance metric, reaching both the lowest and highest cumulative returns. Of the sub-networks, the technical trading system seems to be the most robust in obtaining a larger fAPV than the UBAH strategy but its values again seem negatively skewed. Only 3 out of the 100 ensemble networks perform worse than the UBAH, during the backtest period.

## Chapter 7

# Discussion

This Chapter discusses the implementation and the results of the trading system. Shortcomings of the analysis and suggestions for further development are also addressed.

Even though the trading system is fed with both technical and fundamental information, plenty of information regarding the market situation remains unseen by the system. For instance, news, social media and macroeconomic indicators, like interest rates and Gross Domestic Product, may influence the stock market as a whole. The inability of the trading system to predict larger swings in the stock market may stem from this omission. The trading system does not receive information about the swings until the stock prices in the training set reflects it. Furthermore, the trading gap proposed delays the arrival of new information to the training set used in training the trading system. In the trade-off between training with the newest information or having a more stable trading system, stability is, however, preferred.

Another remark on the implementation is that even though the light-weight ensemble network architecture described in Section 5.2 provides many benefits, there are also drawbacks. A major one is that more complex combinations of the three different input types, namely technical, fundamental and time-lagged return information, cannot be learned. This is because the only information shared between the sub-networks comes from their outputs, which essentially describes the stock evaluation of the respective sub-network. For instance, if an increase in both a technical value and a fundamental value has some effect on the return, but the same increase in only one of them does not, then this pattern cannot be learned by the ensemble network.

During the period under examination, 2001-2019, the stock market exhibited mostly steady increase in stock prices and a few abrupt market declines. The market behaviour partly explains why the trading system seems to perform well in bull markets but poorer in bear markets. The trading system

simply observes more bullish market days than bearish ones. Furthermore, the market corrections, meaning that the stock index falls more than 10%, only happened eight times during the entire time period. As a result, a trading system that performs well on bullish market days will in the long run be preferred to one that performs well when the market falls. Another explanation is that the on-line approach combined with the geometrically distributed batches guide the updating of the model to recent observations. Consequently, the previous market correction is often forgotten before the next one occurs. The assumption that the market changes, however, strongly favours utilising recent observations compared to stale, older ones.

Even though the performance on the backtest serves as a generalisation estimate of how the model would perform on previously unseen data, the on-line procedure makes this estimate less robust. Using an off-line procedure is however not possible due to two reasons. First, the market is dynamic and consequently the performance of a static model would decrease over time. The second reason is that the amount of relevant data is not enough. Given that the dataset contains daily data from 19 years the total number of trading day observations is approximately 5000, which is not enough to train a model and obtain a reliable generalisation estimate. The on-line procedure instead allows usage of data-points in both training and testing, and the multiple models obtained during the procedure provide a significantly larger set of test observations, by the reasoning that the same observation predicted by multiple different models can be regarded as multiple different test observations.

The results in Chapter 6 indicate that all the proposed trading systems on average perform better for this particular set of stocks and backtesting period than the strategies on the Sharpe ratio, Sortino ratio, Omega ratio and fAPV. Especially the ensemble network seems to robustly beat the market in both return and risk-adjusted return, in 97% and 85% of the runs, respectively. The median fAPV for the ensemble system is 24.3 percentage points higher than for the UBAH strategy, corresponding to an excess annual return of approximately 11 percentage points compared to a sort of market index.

Even though the results seem promising, there are a number of assumptions that could affect the performance if the trading system was used in real-time. Perhaps the biggest assumption is that the model could make the decision for the following allocation based on the closing price of a day, and then still be able to execute the orders at that same price even though the market is closed. In practise one could implement the trading system to read the current prices some minutes before the market closes and then allocating the portfolio with the following prices. Given that there is a minimal time difference between those events, the slippage should be small. However, a

large proportion of the volume per day occurs just before closing time, which can cause the price to deviate, only minutes apart.

Another assumption is that the generalisation estimate obtained using this particular testing set is reliable for new data. Even though a fairly large proportion of the data is used in testing, the number of observations is not too large. Furthermore, with the assumption that the market continuously changes, there is always the risk that at some point in the future the market changes so that the trading system is unable to produce models that capture the new market behaviour, due to either lack of information fed to the system or a too simple network architecture.

A shortcoming of the analysis is that some values for the hyperparameters were selected based on the earlier research in Section 3.6, for instance, the batch size and the number of previous feature values for the technical sub-network are not optimised in the hyperparameter search. It is possible that these selections could be improved upon, potentially leading to a better trading system. Another decision based on the previous research was to approximate the transaction cost as described in Section 5.1. If the initial portfolio value is huge, this approximation corresponds to reality. However, the transaction cost would often be larger in reality, as most brokers have a fixed minimal transaction cost per trade that would make small allocation changes more expensive.

The limitation of only allowing long positions, is a potential subject for future research. The potential loss in a short position is infinite, which greatly increases the risk of ruin for the trading system. However, allowing short positions would enable the trading system to gain money even on decreasing markets. The change in the output layer described in Section 5.2 would be minimal as the Kelly Criterion naturally outputs negative bet-sizes when the complement bet is suggested, which in this case is betting that the stock price decreases. Performing adequately also in decreasing markets is highly important for the usability of the trading system and therefore an important issue to address in the future development of the system.

One of the larger problems with this task is the lack of data. The robustness of the trading system could therefore be improved upon by either using a synthetic data generator similar as Yu et al. [2019] or by increasing the dataset in another way. The latter could be done by involving more stocks and therefore obtaining a more robust technical sub-network, because it uses the same mapping regardless of stock as described in Section 5.2. Another option would be to utilise intraday data and allow the model to trade, for instance, every 30 minutes increasing the amount of data by a factor of 13.

If the intraday data was used, one could also allocate the portfolio in way that corresponds to reality better. The trading system could, for instance,

read the asset prices at one minute, produce the allocation decision and then execute according to that decision on the prices for the following minute. Another improvement that would make the trading system correspond to reality more is to model the transaction costs exactly. That would require feeding the trading system the current portfolio value and selecting the broker whose pricing scheme to follow. The system would most likely learn that if the portfolio value is small, then only larger allocation changes should be performed, while a larger portfolio value allows the model to make smaller changes as well.

Another suggestion for further research is to use a different selection of stocks, perhaps the entire S&P 500 index, which could reveal stocks better suited for this type of algorithmic trading. Furthermore, using another or even multiple other testing periods when running the training procedure for the trading systems could provide more insight to the robustness of the solution.

## Chapter 8

# Conclusions

Performing adequately in one's investments is essential for professional money managers and aspiring individual investors. The task, however, seems so complex that many recommend investing in a passive stock index compared to actively selecting the stocks to invest in. Machine Learning, on the other hand, is a field that has recently exceeded human capabilities in many challenging tasks. This thesis seeks to combine the two, in constructing a ML based portfolio manager that independently allocates funds into stocks and cash, on a daily basis. This automated portfolio is compared with basic strategies, some depicting how the market moves, to see whether the ML based portfolio manager is relevant.

The eleven stocks used are selected from the Information Technology sector of stocks in the S&P 500 index. Stocks with both high and low volatility, that have interesting intermediate correlations are selected. Data on fundamental values for companies and the prices of their stocks, is combined to match the public information back-in-time as accurately as possible. The initial selection of features consists of fundamental indicators found to be significant when predicting returns and technical indicators that are commonly used by technical analysts. A subset of these features are used as input to the portfolio manager, modelled by an Artificial Neural Network that directly outputs the following allocation for the portfolio. A light-weight network architecture that converges quickly and learns robust patterns in the data is constructed. Four different portfolio managers are produced using different combinations of the input information.

After finding the optimal parameters that control the learning process, the data is split into a training and testing set. The training set is used in training the model using a Reinforcement Learning algorithm called the Policy Gradient algorithm. An on-line approach is used when testing the performance on the testing set, meaning that the model is retrained with all

available data at that time, between every prediction. Consequently, we call the procedure of sequentially updating the model to allocate the portfolio for the following day, the trading system.

The results indicate that based on return and risk-adjusted return, the trading systems outperform simple strategies and the market, on average. That edge seems to originate solely from the excess return that the trading systems generate as no clear difference can be seen in the measures of risk between the trading systems and the strategies. The resulting trading system appears to trade actively, performing roughly five trades per day while favouring more volatile stocks. Furthermore, the system seems to perform better than the market when the general direction is upwards but when the market is falling, the system is unsuccessful in outperforming the market even though it has the option of investing in cash.

The reliability of the results is evaluated by training each trading system multiple times. Due to the randomness of the training and testing procedure, a different trading system will be obtained at every run. The four trading systems seem to perform better than the market on average and median but their worst run is still worse than the market. Based on the Omega ratio, a risk-adjusted return, the most advanced trading system appears to perform better than the market with a probability of 85%. The corresponding percentages for the less advanced trading systems indicate a probability of 60%-76%. The corresponding probabilities for the the final Accumulated Portfolio Value, or cumulative return, is 97% for the advanced system and 70%-83% for the less advanced. The most advanced trading system earns an excess annual return of eleven percentage points, on median.

Even though the results indicate that a RL based trading system could outperform the market, the amount of data available is not too extensive, potentially making the generalisation estimate biased. Furthermore, the environments where the test was performed and the real-world differ slightly, as it is assumed in the testing that the price of a stock does not deviate much during the last minutes of trading before the market closes. One way to improve in both aspects would be to use intraday data, resulting in a larger dataset and a more realistic trading environment.

# Bibliography

- Barron, O. E., Byard, D. A., and Yu, Y. (2016). Earnings Announcement Disclosures that Help Analysts Forecast Earnings. *Contemporary Accounting Research*, 34(1):343–373.
- Bernoulli, D. (1954). Exposition of a New Theory on the Measurement of Risk. *Econometrica*, 22(1):23–36.
- Chen, S., DeFond, M. L., and Park, C. W. (2002). Voluntary Disclosure of Balance Sheet Information in Quarterly Earnings Announcements. *Journal of Accounting and Economics*, 33(2):229–251.
- Coval, J., Hirshleifer, D., and Shumway, T. (2005). Can Individual Investors Beat the Market? *SSRN Electronic Journal*. Available at [https://papers.ssrn.com/sol3/papers.cfm?abstract\\_id=364000](https://papers.ssrn.com/sol3/papers.cfm?abstract_id=364000). [Accessed 21.02.2020].
- Cover, T. and Thomas, J. (1991). *Elements of Information Theory*. Wiley.
- CRSP/Compustat (2020). *CRSP/Compustat Merged*. Center for Research in Security Prices. [Online]. Available at: WRDS <https://wrds-www.wharton.upenn.edu/>. [Accessed: 20.02.2020].
- EDGAR (2020). *Electronic Data Gathering, Analysis, and Retrieval system*. U.S. Securities and Exchange Commission. [Online]. Available at: SEC <https://www.sec.gov/edgar/search-and-access>. [Accessed: 20.02.2020].
- Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. Available at <https://arxiv.org/abs/1801.01290v2>. [Accessed 10.03.2020].
- Hagstrom, R. (2013). *Investing: The Last Liberal Art (2nd edition)*. Columbia University Press.
- Hou, K., Xue, C., and Zhang, L. (2018). Replicating Anomalies. *The Review of Financial Studies*, 33(5):2019–2133.

- Jiang, Z., Xu, D., and Liang, J. (2017). A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem. Available at <https://arxiv.org/abs/1706.10059v2>. [Accessed 21.02.2020].
- Keating, C. and Shadwick, W. (2002). A Universal Performance Measure. *Journal of Performance Measurement*, 6(3):59–84.
- Kelly Jr., J. L. (1956). A New Interpretation of Information Rate. *Bell System Technical Journal*, 35(4):917–926.
- Klar, B. and Müller, A. (2017). On Consistency of the Omega Ratio with Stochastic Dominance Rules. *Innovations in Insurance, Risk- and Asset Management*, pages 367–380. Available at [https://www.worldscientific.com/doi/pdf/10.1142/9789813272569\\_0014](https://www.worldscientific.com/doi/pdf/10.1142/9789813272569_0014). [Accessed 24.03.2020].
- Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end Training of Deep Visuomotor Policies. *The Journal of Machine Learning Research*, 17(1):1334–1373.
- Li, B. and Hoi, S. C. (2014). Online Portfolio Selection: A Survey. *ACM Computing Surveys (CSUR)*, 46(3):1–36.
- Li, B., Hoi, S. C., Sahoo, D., and Liu, Z.-Y. (2015). Moving Average Reversion Strategy for On-Line Portfolio Selection. *Artificial Intelligence*, 222:104–123.
- Liang, Z., Chen, H., Zhu, J., Jiang, K., and Li, Y. (2018). Adversarial Deep Reinforcement Learning in Portfolio Management. Available at <https://arxiv.org/abs/1808.09940v3>. [Accessed 21.02.2020].
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous Control with Deep Reinforcement Learning. Available at <https://arxiv.org/abs/1509.02971>. [Accessed 18.03.2020].
- MacLean, L. C., Thorp, E. O., and Ziemba, W. T. (2011). *The Kelly Capital Growth Investment Criterion*. World Scientific.
- MacLean, L. C., Ziemba, W. T., and Blazenko, G. (1992). Growth Versus Security in Dynamic Investment Analysis. *Management Science*, 38(11):1562–1585.
- Malkiel, B. G. (2007). *A Random Walk Down Wall Street: The Time-Tested Strategy for Successful Investing (9th Edition)*. W. W. Norton & Company.

- Markowitz, H. (1952). Portfolio Selection. *The Journal of Finance*, 7(1):77–91.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level Control through Deep Reinforcement Learning. *Nature*, 518(7540):529–533.
- Moody, J., Wu, L., Liao, Y., and Saffell, M. (1998). Performance Functions and Reinforcement Learning for Trading Systems and Portfolios. *Journal of Forecasting*, 17(5-6):441–470.
- Murphy, J. (1999). *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. New York Institute of Finance.
- SEC (2003). Final Rule: Conditions for Use of Non-GAAP Financial Measures. Available at <https://www.sec.gov/rules/final/33-8176.htm>. [Accessed 24.02.2020].
- Sharpe, W. F. (1966). Mutual Fund Performance. *The Journal of Business*, 39(1):119–138.
- Siegel, L. (2010). Black Swan or Black Turkey? The State of Economic Knowledge and the Crash of 2007-2009. *Financial Analysts Journal*, 66:6–10.
- Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529:484–489.
- Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic Policy Gradient Algorithms. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 387-395, Beijing, China. PMLR.
- Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L. R., Lai, M., Bolton, A., Chen, Y., Lillicrap, T. P., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis,

- D. (2017). Mastering the Game of Go without Human Knowledge. *Nature*, 550:354–359.
- Sortino, F. A. and Forsey, H. J. (1996). On the Use and Misuse of Downside Risk. *The Journal of Portfolio Management*, 22(2):35–42.
- Sortino, F. A. and Price, L. N. (1994). Performance Measurement in a Downside Risk Framework. *The Journal of Investing*, 3(3):59–64.
- Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction (2nd edition)*. A Bradford Book, Cambridge, MA, USA.
- Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3):229–256.
- Yu, P., Lee, J. S., Kulyatin, I., Shi, Z., and Dasgupta, S. (2019). Model-based Deep Reinforcement Learning for Dynamic Portfolio Optimization. Available at <https://arxiv.org/abs/1901.08740>. [Accessed 18.03.2020].

# Appendix A

## Hyperparameters

Hyperparameter	Value	Description
batch size	20	Size of minibatch when training
batches per epoch	50	Number of batches per epoch
epochs	400	Number of epochs when training
train test gap	26	Gap between train set and set
discarded days	5	Discarded days in train test gap
$n_{tech}$	20	Number of previous technical feature values
$n_{fund}$	5	Number of previous fundamental feature values
$\beta$	$10^{-3}$	Geometric distribution parameter
$c$	$10^{-4}$	Commission rate
$m$	11	Number of stocks
$m^+$	12	Number of assets

Table A.1: Predetermined hyperparameters in this thesis. These parameters control the learning process but are not optimised in the hyperparameter search. The data is examined daily and therefore, the corresponding units are days.

## Appendix B

# Network Specifics

Path	Layer	Filter	Padding	Stride	Activation	Regularisation
tech	convolutional	$(1 \times 3)$	No	1	ReLU	l2
tech	convolutional	$(1 \times (n_{tech} - 2))$	No	1	linear	l2
fund	convolutional	$(1 \times n_{fund})$	No	1	ReLU	l2
fund	fully-connected	full	-	-	linear	l1&dropout
ret	fully-connected	full	-	-	ReLU	l1&dropout
ret	fully-connected	full	-	-	linear	l1&dropout
ensemble	convolutional	$(1 \times 4)$	No	1	ReLU	l2
ensemble	output	$(1 \times 1)$	-	-	sigmoid	-

Table B.1: The specifics of the ensemble network. The specifics for the different sub-networks are deducible from the Table and Section 5.2. The Table is created from the perspective of convolutional layers. *Padding* means to surround the data with zeros in the calculations and *stride* controls which all subsets are used in the mapping of the convolutional layer.

# Appendix C

## Trading Systems

	S	SR	Omega	DD	VaR	median	FAPV
tech21	0.0894	0.1174	1.2144	1.08%	2.51%	0.21%	79.6%
tech27	0.0896	0.1177	1.215	1.08%	2.53%	0.19%	80.3%
tech30	0.0862	0.1133	1.2065	1.13%	2.65%	0.18%	80.1%
tech33	0.0895	0.1178	1.2152	1.1%	2.57%	0.19%	81.7%
tech36	0.0662	0.081	1.1463	1.11%	2.58%	0.2%	52.6%
tech39	0.0885	0.116	1.2116	1.09%	2.54%	0.23%	79.1%
tech40	0.0838	0.1084	1.1974	1.08%	2.52%	0.17%	72.4%
tech51	0.0818	0.1056	1.1921	1.11%	2.58%	0.18%	71.7%
tech52	0.0916	0.1209	1.221	1.09%	2.54%	0.15%	83.3%
tech55	0.0837	0.1085	1.1976	1.1%	2.57%	0.21%	73.8%
tech57	0.0746	0.0937	1.1699	1.09%	2.55%	0.16%	61.5%
tech60	0.0753	0.0949	1.1721	1.1%	2.56%	0.19%	62.6%
tech61	0.0912	0.1211	1.2213	1.11%	2.6%	0.21%	85.6%
tech64	0.0741	0.0933	1.1692	1.11%	2.6%	0.18%	62.3%
tech71	0.0784	0.0987	1.1793	1.03%	2.39%	0.15%	61.8%
tech72	0.093	0.1232	1.2254	1.08%	2.53%	0.18%	85%
tech74	0.0939	0.1243	1.2274	1.06%	2.48%	0.22%	84.3%
tech77	0.0891	0.1172	1.214	1.1%	2.57%	0.2%	81.3%
tech80	0.0772	0.0986	1.179	1.14%	2.65%	0.16%	67.6%
tech81	0.0773	0.098	1.1779	1.08%	2.52%	0.14%	64.2%
tech98	0.0882	0.1139	1.2079	1%	2.34%	0.2%	71.6%
tech107	0.0776	0.0981	1.1781	1.07%	2.5%	0.18%	63.9%
tech110	0.0784	0.0986	1.1791	1.02%	2.38%	0.19%	61.5%
tech113	0.0828	0.1072	1.1951	1.11%	2.6%	0.19%	73.4%
tech115	0.0934	0.123	1.225	1.04%	2.43%	0.21%	81.4%
tech118	0.0921	0.1219	1.2229	1.07%	2.51%	0.24%	83.3%
tech128	0.0653	0.0782	1.1412	1.05%	2.44%	0.19%	48.5%
tech131	0.0927	0.1229	1.2248	1.09%	2.54%	0.19%	85.1%
tech132	0.0824	0.1063	1.1935	1.1%	2.57%	0.2%	71.9%
tech134	0.0659	0.0786	1.1419	1.01%	2.36%	0.08%	47.5%
tech140	0.0867	0.1122	1.2046	1.04%	2.42%	0.19%	72.6%
tech142	0.0885	0.1164	1.2125	1.11%	2.6%	0.24%	81.5%
tech143	0.0913	0.1203	1.2198	1.07%	2.5%	0.17%	81.5%
tech153	0.0833	0.1076	1.1959	1.09%	2.54%	0.16%	72.1%
tech163	0.0569	0.0652	1.1173	1.05%	2.43%	0.1%	39.8%
tech167	0.0902	0.1188	1.217	1.09%	2.54%	0.17%	81.5%
tech180	0.0851	0.1113	1.2027	1.13%	2.65%	0.21%	78.4%
tech183	0.0859	0.1114	1.203	1.07%	2.49%	0.21%	74%
tech191	0.0712	0.088	1.1593	1.08%	2.52%	0.17%	56.7%
tech192	0.0793	0.1002	1.182	1.02%	2.39%	0.17%	62.7%
tech195	0.0909	0.1197	1.2188	1.07%	2.5%	0.23%	81%
tech198	0.0896	0.1181	1.2157	1.1%	2.57%	0.17%	81.8%
tech201	0.0894	0.118	1.2154	1.1%	2.58%	0.26%	82.2%
tech204	0.0852	0.1097	1.2	1.04%	2.42%	0.2%	70.7%
tech208	0.0797	0.1017	1.1849	1.08%	2.53%	0.17%	67.3%
tech209	0.0924	0.1224	1.2239	1.09%	2.55%	0.18%	84.9%
tech217	0.0906	0.12	1.2193	1.11%	2.6%	0.18%	84.6%
tech222	0.0839	0.1093	1.199	1.13%	2.63%	0.19%	76.2%
tech224	0.0851	0.1102	1.2008	1.07%	2.5%	0.2%	73.3%
tech229	0.0827	0.1067	1.1942	1.09%	2.55%	0.21%	71.9%
tech230	0.0929	0.1232	1.2252	1.08%	2.52%	0.2%	84.7%
tech231	0.0875	0.1147	1.2092	1.09%	2.55%	0.21%	78.2%
tech233	0.0867	0.1131	1.2062	1.09%	2.55%	0.22%	77%
tech236	0.0842	0.1091	1.1987	1.09%	2.54%	0.2%	73.4%
tech241	0.0954	0.1266	1.2318	1.05%	2.46%	0.19%	85.5%
tech244	0.0879	0.1142	1.2085	1.05%	2.45%	0.19%	75%
tech253	0.0572	0.0666	1.1197	1.1%	2.56%	0.15%	42.2%
tech255	0.0928	0.1225	1.224	1.06%	2.49%	0.18%	82.9%
tech258	0.0948	0.128	1.2342	1.18%	2.76%	0.21%	97.5%
tech261	0.0891	0.1169	1.2133	1.08%	2.53%	0.2%	79.5%
fund1	0.0798	0.1056	1.1918	1.27%	2.97%	0.09%	81.8%
fund2	0.0402	0.043	1.0767	1.23%	2.86%	0.09%	28.3%
fund3	0.0748	0.0959	1.1739	1.18%	2.76%	0.17%	67.9%
fund4	0.1	0.1376	1.2524	1.24%	2.91%	0.17%	114%
fund5	0.0839	0.1095	1.1993	1.13%	2.63%	0.17%	76.3%
fund6	0.0718	0.0894	1.1618	1.08%	2.52%	0.06%	57.7%
fund7	0.101	0.1369	1.2513	1.08%	2.53%	0.09%	97%
fund8	0.0767	0.0943	1.1712	0.93%	2.17%	0.07%	54%
fund9	0.0812	0.1046	1.1902	1.1%	2.57%	0.1%	70.6%
fund10	0.0566	0.0672	1.1208	1.2%	2.79%	0.14%	45.2%
fund11	0.0943	0.1267	1.2318	1.14%	2.68%	0.3%	93.3%
fund12	0.0813	0.1046	1.1902	1.08%	2.53%	0.09%	69.5%
fund13	0.0815	0.1071	1.1947	1.2%	2.79%	0.17%	78.5%
fund14	0.0973	0.1318	1.2414	1.11%	2.6%	0.1%	95%
fund15	0.1036	0.1415	1.26	1.1%	2.57%	0.17%	103.1%
fund16	0.0782	0.0998	1.1813	1.1%	2.56%	0.08%	76.6%
fund17	0.0968	0.1275	1.2336	0.98%	2.28%	0.17%	69.9%
fund18	0.09	0.1207	1.2204	1.19%	2.78%	0.16%	91%
fund19	0.1018	0.1388	1.2549	1.08%	2.52%	0.16%	98.5%
fund20	0.0884	0.117	1.2135	1.13%	2.64%	0.13%	83.2%
fund21	0.0833	0.1095	1.1993	1.2%	2.8%	0.2%	80.9%
fund22	0.0896	0.1174	1.2143	1.07%	2.49%	0.12%	78.8%
fund23	0.0666	0.0809	1.1461	1.05%	2.43%	0.14%	59.2%
fund24	0.1054	0.1421	1.2615	0.99%	2.32%	0.18%	93%
fund25	0.1012	0.1397	1.2564	1.2%	2.82%	0.1%	112.3%
fund26	0.0788	0.1016	1.1846	1.13%	2.64%	0.09%	69.9%
fund27	0.054	0.0603	1.1082	1.02%	2.37%	0.12%	36.2%
fund28	0.0965	0.1297	1.2375	1.1%	2.57%	0.19%	92.2%
fund29	0.0375	0.0363	1.0646	1.08%	2.49%	0.02%	22.7%
fund30	0.0594	0.0702	1.1263	1.11%	2.58%	0.1%	44.9%
fund31	0.0757	0.0964	1.1749	1.13%	2.63%	0.09%	65.5%
fund32	0.1167	0.164	1.3033	1.08%	2.55%	0.17%	124.4%
fund33	0.0926	0.123	1.2248	1.11%	2.59%	0.13%	86.8%
fund34	0.0509	0.0569	1.1021	1.11%	2.58%	0.01%	35.9%
fund35	0.106	0.1451	1.267	1.06%	2.48%	0.17%	102.7%
fund36	0.0561	0.0657	1.1181	1.14%	2.65%	0.06%	42.6%
fund37	0.1153	0.1618	1.299	1.14%	2.67%	0.13%	129.3%
fund38	0.0924	0.1251	1.2286	1.26%	2.94%	0.22%	101.2%
fund39	0.0224	0.0165	1.0291	1.23%	2.85%	0.08%	11.2%
fund40	0.1453	0.2143	1.402	1.12%	2.66%	0.28%	194.4%
fund41	0.0721	0.0918	1.1663	1.21%	2.83%	0.12%	65.8%
fund42	0.1166	0.1672	1.3091	1.29%	3.03%	0.21%	158.3%
fund43	0.0926	0.121	1.2213	1.01%	2.36%	0.14%	77.4%
fund44	0.09	0.1186	1.2167	1.1%	2.57%	0.17%	82.3%
fund45	0.1312	0.1899	1.3537	1.14%	2.68%	0.26%	163.1%
fund46	0.1184	0.1652	1.3058	1.04%	2.44%	0.16%	119.4%
fund47	0.0879	0.1162	1.212	1.14%	2.66%	0.19%	83.1%
fund48	0.0782	0.1003	1.1821	1.1%	2.58%	0.12%	67.3%
fund49	0.1171	0.1654	1.306	1.15%	2.71%	0.19%	135.4%
fund50	0.1089	0.153	1.2819	1.25%	2.93%	0.22%	133.1%
fund51	0.0871	0.113	1.206	1.04%	2.43%	0.1%	73.3%
fund52	0.0198	0.0111	1.0196	1.13%	2.61%	0.07%	8.7%
fund53	0.0705	0.0856	1.1549	1.01%	2.36%	0.1%	52.1%
fund54	0.038	0.0371	1.0662	1.08%	2.5%	0.04%	23.2%
fund55	0.0782	0.1	1.1816	1.1%	2.56%	0.14%	66.6%
fund56	0.0905	-0.0017	0.9971	1.29%	2.97%	0.11%	0.3%
fund57	0.0928	0.1221	1.2233	1.05%	2.46%	0.16%	81.8%
fund58	0.0536	0.0616	1.1106	1.13%	2.62%	0.07%	39.4%
fund59	0.0566	0.0669	1.1203	1.19%	2.76%	0.16%	44.7%
fund60	0.0559	0.0679	1.1219	1.3%	3.03%	0.16%	48.4%

	S	SR	Omega	DD	VaR	median	FAPV
tech270	0.0998	0.1353	1.2482	1.12%	2.62%	0.22%	99.3%
tech275	0.0852	0.1101	1.2006	1.06%	2.48%	0.19%	72.7%
tech281	0.0873	0.1145	1.2089	1.11%	2.59%	0.2%	79.5%
tech283	0.0479	0.0527	1.0943	1.12%	2.6%	0.15%	33.3%
tech286	0.0668	0.0813	1.1469	1.09%	2.55%	0.12%	52.4%
tech288	0.0925	0.1229	1.2246	1.09%	2.55%	0.2%	85.5%
tech289	0.0844	0.1102	1.2007	1.13%	2.64%	0.21%	77%
tech292	0.0909	0.12	1.2192	1.09%	2.54%	0.24%	82.6%
tech296	0.0858	0.1101	1.2007	1.01%	2.36%	0.12%	69.3%
tech300	0.0721	0.0894	1.1619	1.07%	2.5%	0.16%	57.4%
tech303	0.0828	0.1064	1.1937	1.07%	2.49%	0.22%	70.1%
tech307	0.0815	0.105	1.191	1.1%	2.58%	0.2%	71.1%
tech308	0.085	0.1107	1.2018	1.11%	2.59%	0.19%	76.3%
tech312	0.0707	0.0864	1.1564	1.03%	2.4%	0.17%	53.3%
tech316	0.0876	0.1151	1.21	1.12%	2.61%	0.18%	80.6%
tech317	0.0988	0.1325	1.243	1.07%	2.51%	0.23%	92.5%
tech328	0.0767	0.0967	1.1754	1.07%	2.5%	0.21%	62.7%
tech329	0.0878	0.1137	1.2075	1.03%	2.41%	0.21%	73.5%
tech346	0.0928	0.1236	1.226	1.12%	2.63%	0.22%	88.8%
tech349	0.0867	0.1138	1.2076	1.12%	2.61%	0.23%	79.6%
tech353	0.0772	0.0971	1.1764	1.05%	2.44%	0.17%	61.9%
tech363	0.0889	0.1163	1.2122	1.07%	2.5%	0.23%	78.2%
tech374	0.0965	0.1292	1.2368	1.08%	2.54%	0.21%	90.5%
tech377	0.0874	0.1143	1.2085	1.09%	2.55%	0.18%	78%
tech379	0.0591	0.0698	1.1257	1.11%	2.59%	0.17%	44.8%
tech381	0.0772	0.0986	1.179	1.13%	2.64%	0.12%	67.5%
tech384	0.0957	0.128	1.2345	1.09%	2.56%	0.23%	90.3%
tech390	0.0878	0.1132	1.2066	1%	2.34%	0.15%	71.1%
tech393	0.0846	0.1104	1.2012	1.13%	2.64%	0.22%	77.4%
tech396	0.0864	0.1134	1.2067	1.14%	2.65%	0.16%	80.4%
tech397	0.0903	0.1191	1.2175	1.09%	2.56%	0.23%	82.3%
tech401	0.0966	0.1291	1.2365	1.06%	2.49%	0.17%	88.5%
tech403	0.0906	0.1192	1.2178	1.08%	2.52%	0.19%	81.3%
tech406	0.0973	0.1312	1.2405	1.13%	2.64%	0.24%	96.3%
tech413	0.0956	0.1273	1.2332	1.07%	2.51%	0.23%	87.9%
tech417	0.0969	0.13	1.2382	1.09%	2.55%	0.21%	91.6%
tech428	0.1014	0.1372	1.252	1.08%	2.54%	0.23%	97.7%
tech433	0.0892	0.1165	1.2127	1.06%	2.47%	0.22%	77.5%
tech439	0.0927	0.1227	1.2245	1.08%	2.52%	0.21%	84.2%
tech441	0.0701	0.0872	1.1577	1.12%	2.61%	0.16%	57.7%
ret1	0.0938	0.1227	1.2244	0.97%	2.26%	0.12%	75.6%
ret2	0.1031	0.1369	1.2516	0.94%	2.2%	0.22%	83.7%
ret3	0.114	0.1549	1.2862	0.93%	2.19%	0.18%	97.3%
ret4	0.0901	0.1185	1.2165	1.07%	2.49%	0.19%	79.8%
ret5	0.0867	0.1119	1.2041	1.01%	2.37%	0.17%	70.8%
ret6	0.0989	0.1278	1.2344	0.87%	2.04%	0.19%	71.8%
ret7	0.0779	0.0948	1.1721	0.9%	2.1%	0.17%	52.7%
ret8	0.0941	0.1216	1.2226	0.94%	2.19%	0.17%	72.4%
ret9	0.1288	0.1787	1.3325	0.85%	2%	0.17%	105%
ret10	0.0771	0.0962	1.1747	1.01%	2.35%	0.14%	59.1%
ret11	0.0928	0.118	1.2158	0.86%	2.02%	0.18%	64.6%
ret12	0.1066	0.1442	1.2655	0.98%	2.3%	0.13%	93.7%
ret13	0.0765	0.0949	1.1722	0.99%	2.3%	0.15%	57.1%
ret14	0.0957	0.1265	1.2317	0.99%	2.31%	0.15%	80.1%
ret15	0.0901	0.1155	1.211	0.93%	2.18%	0.08%	67.9%
ret16	0.1007	0.1341	1.2461	0.98%	2.29%	0.15%	85.1%
ret17	0.0801	0.1004	1.1826	0.99%	2.31%	0.18%	61%
ret18	0.0828	0.1039	1.189	0.95%	2.21%	0.16%	60.9%
ret19	0.1127	0.1529	1.2822	0.92%	2.15%	0.19%	93.9%
ret20	0.0725	0.0887	1.1066	1%	2.34%	0.12%	53.8%
ret21	0.0921	0.1183	1.2162	0.92%	2.16%	0.15%	69.2%
ret22	0.1109	0.1488	1.2745	0.89%	2.08%	0.19%	87.6%
ret23	0.0896	0.1149	1.2099	0.95%	2.22%	0.17%	68.8%
ret24	0.1062	0.1424	1.2621	0.96%	2.25%	0.22%	90.1%
ret25	0.0948	0.121	1.2215	0.85%	1.98%	0.18%	65.4%
ret26	0.0978	0.128	1.2347	0.94%	2.2%	0.16%	77.3%
ret27	0.0951	0.1231	1.2253	0.91%	2.13%	0.2%	71.7%
ret28	0.096	0.1249	1.2288	0.93%	2.16%	0.16%	73.9%
ret29	0.0685	0.0821	1.1485	0.99%	2.31%	0.11%	49%
ret30	0.1001	0.1334	1.2447	1%	2.34%	0.19%	86.7%
ret31	0.0916	0.1186	1.2168	0.95%	2.23%	0.18%	71.5%
ret32	0.1003	0.1336	1.2453	0.98%	2.29%	0.15%	85%
ret33	0.0524	0.055	1.0987	0.89%	2.07%	0.08%	30.4%
ret34	0.0762	0.0923	1.1674	0.9%	2.1%	0.13%	51.3%
ret35	0.0912	0.1178	1.2153	0.97%	2.28%	0.15%	72.5%
ret36	0.0664	0.0789	1.1425	0.97%	2.27%	0.12%	46.2%
ret37	0.0566	0.0616	1.1107	0.9%	2.1%	0.11%	34.1%
ret38	0.0842	0.1068	1.1945	0.98%	2.29%	0.14%	65%
ret39	0.0718	0.0873	1.1581	0.98%	2.29%	0.13%	52%
ret40	0.0899	0.1155	1.2109	0.94%	2.19%	0.14%	68.1%
ret41	0.0907	0.1161	1.2122	0.93%	2.17%	0.17%	68%
ret42	0.1152	0.1582	1.2925	0.96%	2.26%	0.18%	103.5%
ret43	0.0789	0.0987	1.1793	1%	2.34%	0.16%	60.7%
ret44	0.0664	0.0777	1.1403	0.94%	2.2%	0.07%	44.4%
ret45	0.085	0.1077	1.1962	0.94%	2.18%	0.16%	62.9%
ret46	0.0855	0.1087	1.1981	0.96%	2.24%	0.19%	65.1%
ret47	0.1088	0.1455	1.2681	0.91%	2.12%	0.14%	87%
ret48	0.0853	0.1086	1.1979	0.97%	2.26%	0.16%	65.5%
ret49	0.1202	0.1643	1.3045	0.89%	2.09%	0.2%	99.6%
ret50	0.1141	0.1558	1.2879	0.93%	2.19%	0.16%	97.8%
ret51	0.0957	0.1246	1.2282	0.92%	2.16%	0.18%	73.5%
ret52	0.0913	0.1111	1.203	0.73%	1.7%	0.04%	52%
ret53	0.088	0.1104	1.2014	0.88%	2.06%	0.16%	61.1%
ret54	0.0915	0.1178	1.2153	0.93%	2.18%	0.2%	69.5%
ret55	0.0877	0.1116	1.2036	0.93%	2.17%	0.15%	65.1%
ret56	0.1052	0.1407	1.2588	0.94%	2.21%	0.25%	87.1%
ret57	0.0885	0.1128	1.2058	0.94%	2.19%	0.17%	66.4%
ret58	0.1003	0.1324	1.243	0.91%	2.13%	0.09%	78%
ret59	0.0915	0.1154	1.2109	0.83%	1.93%	0.06%	60.6%
ret60	0.099	0.1301	1.2386	0.94%	2.21%	0.17%	79.2%
fund61	0.0494	0.0544	1.0975	1.08%	2.5%	0.06%	33.7%
fund62	0.1012	0.1397	1.2564	1.22%	2.86%	0.24%	114.2%
fund63	0.1076	0.1502	1.2766	1.23%	2.89%	0.18%	127.6%
fund64	0.0919	0.1231	1.225	1.16%	2.71%	0.13%	91%
fund65	0.0467	0.0466	1.0833	0.91%	2.1%	0.12%	26.4%
fund66	0.1376	0.2018	1.3772	1.18%	2.78%	0.19%	188.1%
fund67	0.0505	0.0564	1.1012	1.11%	2.58%	0.07%	35.6%
fund68	0.0894	0.1184	1.2162	1.11%	2.6%	0.12%	83.2%
fund69	0.0823	0.1069	1.1945	1.15%	2.67%	0.24%	75.3%
fund70	0.0776	0.1	1.1815	1.18%	2.76%	0.11%	71.4%
fund71	0.076	0.0934	1.1695	0.96%	2.24%	0.12%	55%
fund72	0.1446	0.211	1.3957	1.02%	2.42%	0.28%	167%
fund73	0.0901	0.1188	1.2169	1.08%	2.52%	0.17%	80.7%
fund74	0.1171	0.1645	1.3044	1.08%	2.55%	0.08%	124.9%
fund75	0.0564	0.0603	1.1083	0.86%	1.99%	0%	32.2%
fund76	0.1086	0.1523	1.2806	1.19%	2.8%	0.14%	125.8%
fund77	0.031	0.0265	1.0472	1.07%	2.48%	0.11%	17.3%
fund78	0.1044	0.1425	1.2619	1.06%	2.49%	0.15%	100.5%
fund79	0.0823	0.1096	1.1994	1.33%	3.12%	0.26%	90.1%
fund80	0.079	0.0975	1.1771	0.93%	2.17%	0.15%	56%
fund81	0.0593	0.0711	1.128	1.18%	2.74%	0.14%	47.6%
fund82	0.0955	0.1291	1.2363	1.17%	2.74%	0.24%	97.9%
fund83	0.109	0.1502	1.2768	1.09%	2.55%	0.16%	110.7%
fund84	0.0609	0.0725	1.1307	1.12%	2.6%	0.14%	46.8%
fund85	0.1152	0.1615	1.2985	1.13%	2.65%	0.15%	127.5%
fund86	0.0788	0.1016	1.1846	1.16%	2.71%	0.13%	71.7%
fund87	0.127	0.181	1.3364	1.09%	2.58%	0.18%	145.1%
fund88	0.0218	0.0143	1.0254	1.15%	2.66%	0.05%	10.4%
fund89	0.1114	0.1553	1.2866	1.09%	2.56%	0.18%	116.4%
fund90	0.1005	0.1387	1.2545	1.2%	2.82%	0.12%	111%
fund91	0.1092	0.1502	1.2768	1.07%	2.5%	0.18%	108.4%
fund92	0.0644	0.0784	1.1415	1.12%	2.6%	0.09%	51.1%
fund93	0.1268	0.1822	1.3386	1.14%	2.68%	0.14%	153.9%
fund94	0.0747	0.0954	1.173	1.17%	2.72%	0.19%	96.7%
fund95	0.1594	0.2235	1.4204	1.09%	2.58%	0.22%	198.8%
fund96	0.1074	0.1464	1.2806	1.02%	2.38%	0.17%	99.2%
fund97	0.0979	0.1319	1.2418	1.09%	2.55%	0.16%	93.3%
fund98	0.0966	0.1258	1.2304	0.91%	2.13%	0.02%	73.5%
fund99	0.1016	0.1392	1.2556	1.12%	2.63%	0.2%	103.5%
fund100	0.0625	0.0758	1.1367	1.15%	2.68%	0.12%	50.3%
ensemble1	0.079	0.1024	1.1861	1.19%	2.77%	0.14%	73.8%
ensemble6	0.0903	0.1196	1.2184	1.12%	2.63%	0.15%	85.1%
ensemble9	0.0901	0.1184	1.2162	1.07%	2.51%	0.17%	80.3%
ensemble10	0.0837	0.1081	1.1968	1.07%	2.51%	0.19%	71.8%
ensemble11	0.0923	0.1242	1.2271	1.2%	2.8%	0.2%	95.3%
ensemble13	0.0896	0.1195	1.2182	1.19%	2.78%	0.17%	90%
ensemble16	0.0775	0.0993	1.1803	1.15%	2.69%	0.18%	69.1%
ensemble25	0.0745	0.0975	1.1767	1.36%	3.18%	0.2%	78.7%
ensemble43	0.0931	0.1249	1.2284	1.15%	2.69%	0.22%	91.9%
ensemble51	0.105	0.1437	1.2643	1.1%	2		

	S	SR	Omega	DD	VaR	median	FAPV		S	SR	Omega	DD	VaR	median	FAPV
ret61	0.1155	0.1556	1.2876	0.86%	2.01%	0.19%	89.1%	ensemble277	0.0765	0.0974	1.1767	1.12%	2.6%	0.17%	65.6%
ret62	0.088	0.1117	1.2037	0.92%	2.16%	0.17%	64.8%	ensemble278	0.1069	0.1431	1.2635	0.93%	2.17%	0.24%	87.3%
ret63	0.0743	0.0894	1.1621	0.9%	2.00%	0.13%	49.5%	ensemble279	0.0883	0.1142	1.2084	1.01%	2.37%	0.21%	72.5%
ret64	0.1002	0.1292	1.2372	0.85%	1.98%	0.2%	70.5%	ensemble284	0.0946	0.1247	1.2282	1.02%	2.39%	0.21%	81.5%
ret65	0.086	0.1098	1.2002	0.96%	2.24%	0.17%	65.9%	ensemble289	0.0964	0.1312	1.2403	1.2%	2.82%	0.15%	103.3%
ret66	0.1044	0.1327	1.2439	0.74%	1.72%	0.12%	63.3%	ensemble299	0.0941	0.1255	1.2296	1.11%	2.59%	0.23%	89.2%
ret67	0.0686	0.0819	1.1481	0.97%	2.26%	0.16%	48%	ensemble302	0.0929	0.127	1.2321	1.31%	3.07%	0.2%	107.9%
ret68	0.1272	0.1752	1.3258	0.85%	1.99%	0.21%	102.3%	ensemble307	0.0951	0.1279	1.2342	1.14%	2.67%	0.24%	94.2%
ret69	0.0992	0.131	1.2402	0.94%	2.2%	0.17%	79.6%	ensemble316	0.098	0.1313	1.2407	1.07%	2.51%	0.24%	91.1%
ret70	0.1008	0.1306	1.2396	0.83%	1.95%	0.16%	70.2%	ensemble319	0.0909	0.1152	1.2105	0.89%	2.07%	0.16%	64.5%
ret71	0.1059	0.1413	1.2599	0.92%	2.15%	0.15%	84.9%	ensemble322	0.1023	0.1378	1.2531	1.04%	2.44%	0.22%	94.4%
ret72	0.0905	0.1157	1.2113	0.9%	2.11%	0.14%	66%	ensemble324	0.0995	0.1363	1.2501	1.21%	2.83%	0.19%	109.2%
ret73	0.0957	0.121	1.2216	0.82%	1.91%	0.13%	63.1%	ensemble329	0.076	0.099	1.1796	1.26%	2.95%	0.18%	75%
ret74	0.1051	0.1393	1.2563	0.9%	2.1%	0.12%	81.6%	ensemble344	0.0892	0.1177	1.2149	1.11%	2.59%	0.2%	82.2%
ret75	0.0879	0.1115	1.2035	0.92%	2.15%	0.16%	64.6%	ensemble345	0.0887	0.1155	1.2108	1.03%	2.41%	0.15%	74.9%
ret76	0.0755	0.0895	1.1623	0.85%	1.98%	0.16%	47.4%	ensemble348	0.1017	0.1383	1.254	1.1%	2.59%	0.15%	100.9%
ret77	0.0985	0.1304	1.2391	1%	2.34%	0.2%	84.1%	ensemble350	0.0814	0.1051	1.1911	1.11%	2.6%	0.21%	71.8%
ret78	0.097	0.1268	1.2323	0.93%	2.18%	0.16%	75.6%	ensemble351	0.0837	0.1098	1.2	1.17%	2.74%	0.14%	79.5%
ret79	0.0763	0.0945	1.1715	0.99%	2.31%	0.17%	57%	ensemble362	0.0935	0.1237	1.2264	1.06%	2.49%	0.2%	84%
ret80	0.0727	0.0864	1.1565	0.89%	2.06%	0.15%	47.2%	ensemble365	0.0942	0.1266	1.2317	1.16%	2.71%	0.23%	94.4%
ret81	0.1026	0.1376	1.2528	0.99%	2.32%	0.19%	89.1%	ensemble372	0.0809	0.1044	1.1898	1.13%	2.64%	0.16%	72.1%
ret82	0.1138	0.1528	1.2821	0.87%	2.04%	0.17%	88.7%	ensemble374	0.0959	0.1284	1.2351	1.09%	2.56%	0.24%	90.4%
ret83	0.095	0.1239	1.2268	0.94%	2.2%	0.14%	74.3%	ensemble375	0.0912	0.1215	1.2221	1.15%	2.68%	0.21%	88.6%
ret84	0.0758	0.0925	1.1678	0.94%	2.19%	0.11%	53.2%	ensemble377	0.0763	0.0969	1.1759	1.13%	2.63%	0.18%	65.9%
ret85	0.1035	0.139	1.2556	0.98%	2.29%	0.15%	89.2%	ensemble378	0.0872	0.1141	1.208	1.09%	2.55%	0.16%	77.9%
ret86	0.108	0.1469	1.2707	0.97%	2.28%	0.21%	94.8%	ensemble379	0.0849	0.1141	1.2077	1.31%	3.06%	0.18%	93%
ret87	0.0952	0.1232	1.2255	0.91%	2.13%	0.12%	71.7%	ensemble381	0.081	0.1071	1.1946	1.26%	2.94%	0.08%	82.4%
ret88	0.0625	0.0702	1.1266	0.9%	2.09%	0.14%	38.6%	ensemble384	0.0917	0.1204	1.2201	1.05%	2.46%	0.16%	80.4%
ret89	0.0743	0.0897	1.1627	0.93%	2.16%	0.16%	50.9%	ensemble389	0.0944	0.1275	1.2333	1.19%	2.79%	0.2%	98.3%
ret90	0.0642	0.0747	1.1348	0.95%	2.22%	0.11%	42.9%	ensemble397	0.0797	0.1022	1.1857	1.1%	2.58%	0.18%	68.9%
ret91	0.0945	0.1235	1.226	0.93%	2.18%	0.11%	73.4%	ensemble399	0.0651	0.0795	1.1435	1.14%	2.65%	0.18%	52.6%
ret92	0.0928	0.119	1.2176	0.89%	2.09%	0.1%	67.4%	ensemble401	0.0898	0.1192	1.2176	1.14%	2.68%	0.16%	86.2%
ret93	0.0905	0.1173	1.2143	0.97%	2.28%	0.14%	72.1%	ensemble405	0.0819	0.1047	1.1905	1.05%	2.46%	0.18%	67.8%
ret94	0.0806	0.0996	1.1811	0.92%	2.14%	0.2%	56.5%	ensemble410	0.0913	0.1219	1.2228	1.15%	2.68%	0.16%	88.9%
ret95	0.1244	0.1713	1.3181	0.85%	2.01%	0.15%	100%	ensemble420	0.077	0.0987	1.1791	1.15%	2.68%	0.2%	68.4%
ret96	0.1012	0.1309	1.2403	0.82%	1.91%	0.13%	68.9%	ensemble438	0.0544	0.0637	1.1143	1.18%	2.75%	0.09%	42.2%
ret97	0.0881	0.112	1.2043	0.92%	2.15%	0.16%	64.8%	ensemble440	0.0906	0.1206	1.2203	1.14%	2.67%	0.18%	87.2%
ret98	0.1172	0.1589	1.294	0.86%	2.01%	0.16%	91.5%	ensemble441	0.1164	0.1624	1.3004	1.07%	2.51%	0.24%	121%
ret99	0.0931	0.1211	1.2215	0.97%	2.26%	0.18%	74.3%	ensemble442	0.0908	0.1205	1.2202	1.13%	2.64%	0.25%	86.4%
ret100	0.067	0.0796	1.1438	0.98%	2.28%	0.13%	46.8%	ensemble459	0.0835	0.1101	1.2004	1.21%	2.83%	0.22%	82.5%

Table C.1: The trading systems obtained when running the training process many times. Due to the randomness of the training procedure, different trading systems are obtained for every run. A threshold for the performance during the off-line training part described in Section 5.3 was used and models not overcoming the threshold were discarded before the on-line testing. That is the reason why the numbering of the 100 trading systems of each type is not exhaustive. The metrics tracked are Sharpe ratio, Sortino ratio, Omega ratio, Downside Deviation, Value-at-Risk, daily median value and final Accumulated Portfolio Value.