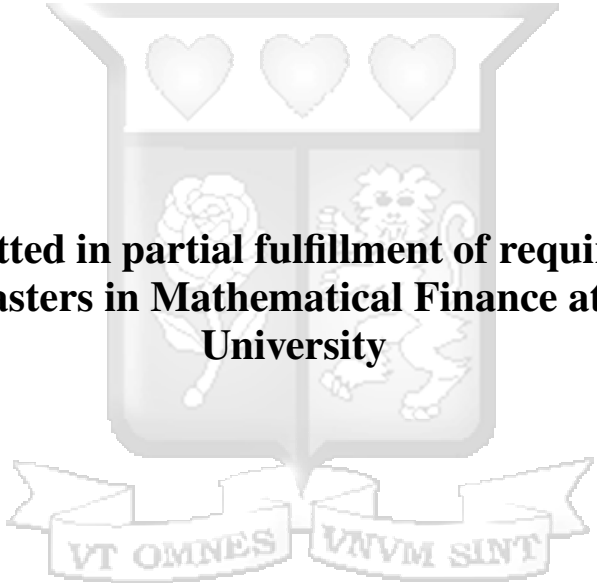
2021

Dynamic portfolio optimization using reinforcement learning.

Yegon, Donald Kibet
Strathmore Institute of Mathematical Sciences
Strathmore University

Dynamic Portfolio Optimization Using Reinforcement Learning

Yegon Donald Kibet
60169



A Thesis submitted in partial fulfillment of requirements for the Degree of Masters in Mathematical Finance at Strathmore University

Strathmore Institute of Mathematical Sciences
Strathmore University

December 2021

Declaration and Approval

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Signature ..... Date .....

Donald Yegon

Approval

The thesis of Donald Yegon was reviewed and approved by the following:

Ferdinand Othieno

Senior Lecturer, Strathmore Institute of Mathematical Sciences

Strathmore University

Dr. Godfrey Achono Madigu

Dean, Strathmore University Institute of Mathematical Sciences

Strathmore University

Dr. Bernard Shibwabo

Director of Graduate Studies

Strathmore University

Abstract

This study uses machine learning in the development of a dynamic investment strategy for portfolio optimization. We aim to explore the efficiency of this approach over a passively managed portfolio and assess the whether transaction costs erode the gains in the dynamically managed portfolio. To this end we explore the application of recurrent reinforcement learning for optimal asset allocation of a portfolio consisting of stock prices for six companies in different sectors. We develop an environment based on monthly historic prices of these stocks and a re-balancing agent that acts on the environment. The risk and return factors of the individual stock are taken as the state of the environment. Using a modified version of Sterling Ratio as the performance measure, we select model parameters through direct recurrent reinforcement learning from historical data and test the efficacy of the strategy on unseen data. From the analysis we find that the regularly re-balanced portfolio out performs the market portfolio based on buy and hold strategy based on both the terminal wealth and the risk adjusted return measure.

Keywords *Reinforcement learning, downside deviation, portfolio optimization*

Table of Contents

List of Figures

List of Tables

LIST OF SYMBOLS AND ABBREVIATIONS

DP	Dynamic Programming
$TDLearning$	Temporal difference Learning
RRL	Recurrent Reinforcement Learning
$MDPL$	Markov Decision Process

1. Introduction

1.1 Background

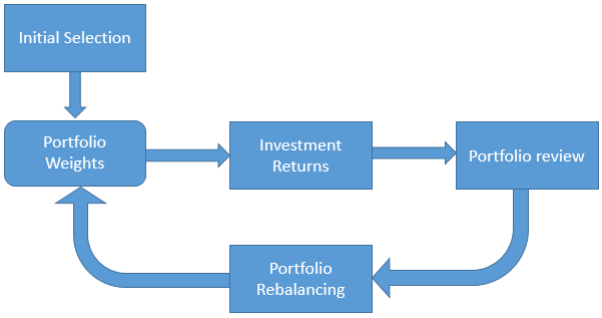
Quantitative investment management consists of two dominant schools of thoughts, passive and active portfolio management. In passive asset management the belief is that the market is efficient Fama (1965). Research has shown that despite markets being predictable and containing inefficiencies, exploitation of these does not lead to profitable strategy Malkiel (2003).

The second school of thought is active asset management. These methods are motivated with the belief that the portfolio can be re balanced at regular time intervals to yield an even greater return. This introduces a new complexity due to the market frictions that are incurred whenever the portfolio is adjusted. Shukla (2004) showed that this methods do not on average generate much return net of transaction costs. In this study we test this assertion by researching the efficacy of an actively managed portfolio using a novel approach.

One of the main tasks faced by an active portfolio manager is optimal asset allocation. This consists of the problem of initial selection and portfolio re-balancing. The investment decision, known as a strategy is typically made in terms of weights that are proportions of the total wealth. In subsequent time steps, the problem of Portfolio re-balancing is then faced. This involves adjusting the portfolio weights to maintain its optimal state. The result is a feedback loop between the market and the investor shown in Figure 1-1.

From this vantage point, active investment management can be viewed as a dynamic system which can be optimized through optimal control theory. Optimal control theory is a branch

Figure 1-1: Investment as a feedback loop



of optimization theory concerned with actions applied on dynamic systems to achieve desired outcomes Neisy and Abkenar (2014) . Here a dynamic system is any system whose state changes with time. To influence this system, a goal is set in terms of an objective function which we aim to maximise. This is achieved through control parameters which are the means of influencing the system behaviour.

As markets consist of assets whose value changes over time, they can be viewed as dynamic systems. In dynamic portfolio management, the investors desired utility forms the objective function of the control system with the investment decisions, typically in form of portfolio weights representing the control parameters.

A prominent approach to optimizing control in dynamic systems is through dynamic programming. Introduced by Bellman (1952), it involves breaking down complex problems into simpler sub problems recursively. This is predicated on the assumption that the problem has optimal substructure.

Elton and Gruber (1971) introduced dynamic programming as an optimal control methodology applicable to many areas in finance including asset allocation. This approach lends itself nicely to the dynamic asset allocation problem but encounters challenges in higher dimensional problems. It also requires a model of the underlying to be known which is not

always feasible.

Developments in computing offer opportunities at solving the challenges faced by dynamic programming through Reinforcement Learning. Reinforcement learning is a branch of machine learning that is similar to optimal control. It is an umbrella term used to describe problems, their applicable class of solutions and the study of these problems and solutions. The basic structure consists of an agent and the environment. The agent reads the state of the environment and takes action on the environment at each time step. The environment provides reward based on the action taken by the agent. The agent therefore aims to maximize the rewards from the environment by adjusting actions towards those that give more reward through trial and error without explicitly being informed of the optimal actions. Filos (2019).

From this formulation we can see four main components of a reinforcement learning system: a policy, a reward signal, a value function, and, in some cases, a model of the environment. Sutton, Barto, and Williams (1992) The policy can be viewed as the agent's behaviour when acting in an environment. Is is essentially the mapping of external stimuli to an action. In portfolio management this is the investment strategy. The reward signal is the environments response to the agent's actions. In our case the reward signal is in form of risk and returns from the investment portfolio. The agent then determines how desirable this rewards are through a value function. The formulation for value function in a portfolio management scenario can be through utility functions or risk adjusted return functions. In some cases, we may have the actual model of the environment which is a mathematical formulation of the system upon which the agent acts. Filos (2019)

In order to optimize a portfolio, a performance measure is required which we view as

the value function in reinforcement learning environment. Since portfolio managers are concern with both the risk as well as the return Markowitz (1952), a risk adjusted return measure is suitable. On such strategy is the Sharpe ratio. This ratio indicates the average return per unit of risk in excess of the risk free rate of return. Sharpe (1994)

Intuitively however, investors are only concern with negative risk, that is those leading to losses Roy (1952). In order to avoid penalizing large positive variability, downside risk measures are considered. Here only the downside deviation of risk is measured against the expected return. Harlow (1991)

Many investment computation assume friction-less markets. This assumption may lead to overstating of the expected returns from investments. In reality however, financial markets are subject to transaction costs such as fees, commissions and taxes. When taken into account, these costs may limit the arbitrage opportunities observed in the market as well as wipe out the expected gains from the investments.Frazzini, Israel, and Moskowitz (2012)

This research focuses on the active portfolio management through application of reinforcement learning to the selection of a dynamic investment portfolio strategy that maximises against risk adjusted returns. We use a risk adjusted return measure that focuses on downside deviation as well as the impact of the current portfolio re-balancing decision on the returns. In order to test efficacy in real world applications, we incorporate the effects of transaction costs in the portfolio performance. We then assess whether this dynamically re balanced portfolio yields better risk adjusted return as compared to a static portfolio whos weights are selected at the initial investment period and held to maturity.

1.2 Problem statement

Studies on portfolio optimization through dynamic optimal control focus on friction-less markets. Introduction of transaction costs to such solutions lead to increased complexity making the solution of such problems cumbersome. In addition, approaches that have been applied to this problems typically assume a well-defined model of the underlying asset which is not always feasible.

These challenges have resulted in limited practical application of these methods. Novel approaches such as reinforcement learning have been shown to offer promising results in a trading setting and can be extended to address the above challenges in a portfolio management setting.

In this study, we use direct reinforcement learning to find an optimal investment strategy that takes into account the resultant market frictions when selecting desired portfolio weights. By employing the use of a modified risk measure suitable for optimizing against downside risk, we construct an actively managed portfolio that takes positions in a multi-asset environment.

1.3 Research objectives

- Develop reinforcement learning framework that learns from historic data the optimal investment strategy.

- Incorporate effect of transaction costs on a dynamic investment portfolio

1.4 Research Question

Does an actively managed portfolio using Reinforcement Learning yield better results than buy and hold strategy in the presence of transaction costs

2. Literature Review

The development of quantitative methods in portfolio management can be traced back to 1952 with the publication of the seminal paper by Harry Markowitz. In his paper, the relationship between risk and return was for the first time formally defined rather than making separate considerations for this two aspects of an investment Markowitz (1952).

This idea was revolutionary because of two key ideas. The first is that it unified two components that had before been considered separately, the risk and return relationship as well as their resultant diversification benefits of their correlation. Secondly, the decision making process was formalized as an optimization problem. Fabozzi, Kolm, Pachamanova, and Focardi (2007)

The method developed by Markowitz does provide a great foundation for portfolio analysis but suffers from the fact that it is focused on one period appraisal of portfolios. In reality portfolio managers are often taking considerations over multiple periods. When making considerations over multiple time periods, the problem becomes a dynamic optimal control problem whose solution requires the application of more complex approaches. Researchers have attempted to solve the resultant optimal control problem through dynamic programming Samuelson (1969) and reinforcement learning approaches such as TD learning Van Roy, Abu-Mostafa, LeBaron, Lo, and Weigend (2001) and Q learning Neuneier (1996).

2.0.1 Dynamic asset allocation using Dynamic Programming

The need for multi-period consideration led Samuelson (1969) to formulate and solve for a framework that considers the lifetime consumption and investment using dynamic stochastic programming. Building on this view that previous work focused on static analysis while in

reality, relationship between current and future decisions should be considered, Elton and Gruber (1971) undertook a survey of the application of dynamic programming in finance.

In the time since several approaches to optimizing asset allocation using dynamic programming have found application by practitioners and academia. Wallace and Høyland (2007) developed a stochastic asset allocation approach for both tactical and strategic asset allocation within a Norwegian insurance firm. Musumeci and Musumeci (1999) applied dynamic programming to retirement asset allocation problem with different investor preferences while Kung (2008) set up a multi period asset allocation model for long term and short term bonds.

The application of dynamic programming gives a solution to the temporal problem of dynamic asset allocation. It however requires a model of the system to exist which is not always feasible. In addition, the dynamic programming approach requires the solution of the Hamillton Jacobi Bellman (HJB) Yong and Zhou (1999) equation which becomes complex when considering higher dimensional problems such as multi-asset environments.

2.0.2 Dynamic asset allocation using Reinforcement Learning

From Brealey, Myers, Marcus, Wang, and Zhu (2007) we know that the investor seeking to build an optimal portfolio needs to solve an optimization for two problems. The first is the expected yields for all the available assets simultaneously and the second being portfolio construction based on their risk appetites. In multi-period models, this is further complicated by transaction costs if the investor intends to revise their decision at regular time intervals.

Neuneier (1996) developed a framework for combining these two tasks, modelling the asset

yields and search for the optimal portfolio through combining them into a Markov Decision Process (MDP). MDP provides a model for multistage decision making in stochastic environments. Neuneier (1996) defines the MDP as a finite state set $S = 1, 2...n$, a finite set of admissible control actions for each state $i \in S$, a set of transition probabilities p_{ij}^{π}, which describe the system dynamics and a return function $r(i, j, u(i))$ with $i, j \in S$, $u(i) \in U(i)$.

For MDPs, when the discrete state space is small and and if an accurate model of the system exists, dynamic programming can be used to obtain the solution. On the other hand for large state spaces without explicit models, reinforcement learning approaches such as Q learning can be applied to obtain the solution. Singh (1994)

2.0.2.1 Dynamic asset allocation using Q-Learning

Neuneier (1996) then proceeds to apply an iterative solution on the Bellman Equation to find an optimal policy. Using an algorithm known as Q learning as the optimization strategy and neural networks as the value function approximate, he was able to obtain superior results in the German stock market. Q-Learning (QL), is a reinforcement-learning method that does not require a model of the system but optimizes the policy by sampling state-action pairs and returns while interacting with the system Barto, Sutton, and Watkins (1989).

Two years later, he built on his work developing a new formulation that would only use one value function for multiple assets and allow for model free policy iteration. This new framework allowed for relaxation of one of the main assumptions from his previous work, that of friction-less markets and allows for several constraints to be considered in the asset allocation processes through the introduction of an artificial deterministic step. Neuneier (1998).

The Q-learning approach therefore extends the solution of the dynamic optimal control prob-

lem to cases where dynamic programming encounters challenges. Its principle is founded on discounted future rewards. A key question that arises is whether it is more advantageous in the long run to focus on expected future discounted returns or to optimise for immediate returns. The Q learning approach focuses on discounted future returns whereas immediate returns approach can be achieved through Direct Recurrent Reinforcement Learning.

2.0.2.2 Dynamic asset allocation using Recurrent Reinforcement

Around the same time Moody and Wu (1997) was developing a direct reinforcement approach to solving the problem of asset allocation. This work compared methods optimized by a forecast of future rewards, Supervised Learning to one optimised directly through Direct Reinforcement. The framework developed takes the prior positions taken in the assets available as input in the current decision hence resulting in a Recurrent Reinforcement Learning (RRL) as follows

$$F_t = F(\theta_t; F_{t-1}, I_t) \tag{2.1}$$

where F_t is the positions taken at time t, θ_t denotes the system parameters at time t and I_t denotes the information set at time t which includes the present and past values of the price series.

They introduced a new measure of portfolio performance, the differential Sharpe ratio. This measure considers exponential weighted moving average of the returns and can therefore be used in determining the impact of the current returns on a trade or investment portfolio. When taken as the optimization problem, this was found to yielded better empirical results than profit optimization. The measure was shown to be more suited for online optimizing trading systems as compared to running and moving Sharpe Ratios. One key innovation of

this research was the use of recurrent reinforcement learning to capture path dependence of trading decisions.

Moody, Wu, Liao, and Saffell (1998) then compared the RRL approach to a Q-learning approach. The key comparison was that whereas RRL optimizes for immediate rewards, Q learning approximates discounted future rewards with the following formulation of the Q learning version Barto et al. (1989) of the Bellman Equation

$$Q^*(x,a) = U(x,a) + \Upsilon \sum_{y=0}^{n} P_{xy}(a) max_b Q^*(y,b) \tag{2.2}$$

where n is the number of system states, $P_{xy}(a)$ is the transition probability from x to y given action a and Υ is the discount factor. The empirical results showed that Q-learning performs better on a single asset environment but worse on multi-asset allocation problem due to excessive changing of position hence more incurred transaction costs.

Given the favourable results over both supervised learning approaches and Q learning, Moody and Saffell (2001) set out to assess the performance of a recurrent reinforcement learning model in different settings. Here a new performance measure that optimizes against downside risk was introduced. This is the differential Sterling ratio also used in this research.

Most of the works on recurrent reinforcement learning in finance focused on trading problems. Suárez, Moody, and Saffell (2009) modified the RRL trading solution to be applied to a portfolio management setting. This was motivated by the view that Markowitz and other methods end up finding corner solutions to the portfolio optimization problem. These kind of solutions are prone to switching behaviour. The key innovation in this was to shrink the

current weights towards the prior portfolio weights through the following formulation:

$$F_n(\lambda, w) = \lambda F_n + (1 - \lambda) F_n^s \tag{2.3}$$

where F_n is the composition of the portfolio currently, F_n^s is the prior portfolio composition and λ is a hyper-parameter that determines the relative importance of the two positions in the final output.

More recently, Maringer and Ramtohul (2010) built on the work done by Moody and Wu (1997) developing a means of capturing the effects of regimes in the market through a threshold recurrent reinforcement learning algorithm. This was aimed at addressing the concerns with the simplistic nature of solutions previously developed that were unlikely to cater for the non linearity and structural breaks observed in the market.

Recurrent Reinforcement learning has also been shown to yield promising results in foreign exchange markets. Gold (2003) showed that a single layered Recurrent Reinforcement Learning outperforms a two layered network in the foreign exchange market. Dempster and Leemans (2006) developed a three layered Recurrent Reinforcement Learning that entailed a machine learning layer, a risk management layer and a utility optimization layer. Spooner, Fearnley, Savani, and Koukorinis (2018) showed the application of reinforcement learning in market making yields promising results.

Bertoluzzo and Corazza (2007) tested Recurrent Reinforcement Learning on data from 9 financial Markets around the world. Their implementation was using an adaptive neural network and the results showed positive returns in all but one of the tested applications.

Almahdi and Yang (2017) employed the use of expected maximum draw down as the

performance measure. They further used various transaction cost consideration to validate the results of the developed recurrent reinforcement learning model.

Pendharkar and Cusatis (2018) researched the application of reinforcement learning in personal retirement investment optimization. The research they considered portfolio returns or differential Sharpe ratio to measure performance. Results indicated that the high-learning frequency trading agent consistently beats both the single asset stock and bond cumulative returns by a significant margin.

3. Research Methodology

3.1 Structure of portfolios

The portfolio developed can take varying long and short positions in an asset. This can be built up starting from a simple trading agent assumes discrete fixed size trading positions in a given asset.

$$F_t \in [-1, 1] \tag{3.1}$$

where F_t is the position assumed in each asset varying between -1 and 1 where negative values constitute short selling and positive values constitute long positions.

In order to properly take into account the gains resultant from transitioning from previous position F_{t-1} to F_t effect of transaction costs must be considered. The position F_t is therefore a result of

$$F_t = F(\theta; F_{t-1}, I_t) \tag{3.2}$$

where I_t is the information at time t, F_{t-1} is the prior positions taken in the portfolio and θ is the current asset state.

We adjust the amount invested in each asset through a model parameter θ and an activation function that takes θ and an asset return x as it's inputs. The activation function then gives varying quantities of f_t as it's output which corresponds to a trading agent that takes varying quantities of wealth in multiple assets available to it. These θ parameters are initialized randomly and optimized through gradient ascent.

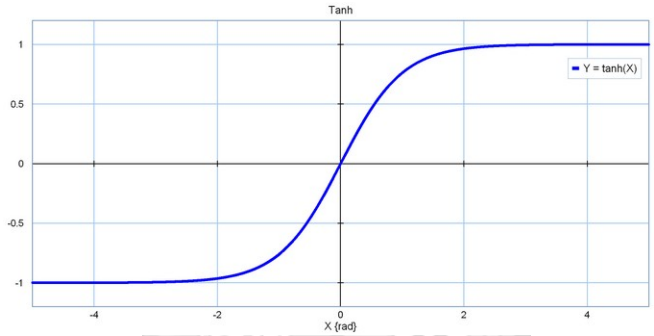
For this analysis we use desire an activation function that gives varying long and short

positions in the available assets. Therefore the tanh function given as follows is selected:

$$f_t = tanh(x^{it}\theta) \tag{3.3}$$

Since we require the weights to sum up to 1 as they are proportions of the overall wealth,

Figure 3-1: Plot of Tanh function



this introduces a constraint in the optimization. A simple way to solve this is by using softmax Moody and Saffell, 2001 outputs outside the training hence the expression for F_t becomes

$$F_t = softmax(f_t(\theta_{t-1}; F_{t-1}I_t)) \tag{3.4}$$

Based on these positions taken, we are able to compute returns at time T as

$$R_T = F_{T-1}r_t - \rho|F_{T-1} - F_T| \tag{3.5}$$

where ρ is the transaction cost, r_t is the return of the portfolio at time t and F is the positions taken in the position. Here we take the transaction costs to be 0.6% Frazzini et al. (2012) We aim to develop a means to assess risk adjusted returns of different portfolio mixes at each time step. These returns are used as the basis of the investment strategy. The return

informs the proportion of wealth invested in each asset.

3.2 Financial performance functions

3.2.1 Profit and wealth

The portfolio's ultimate aim is to maximise a measure in form of a function that can be utility functions of wealth, return or a risk adjusted return. In this analysis, we first use multiplicative profits to assess whether the strategy is profitable. This is due to the portfolio being constructed as a fixed fraction of the accumulated wealth in long and short positions. For our multi asset, regularly re-balanced portfolio, the wealth at the end of the investment period is given by

$$W_T = W_0 \prod_{t=1}^{T} 1 + R_t = W_o \prod_{t=1}^{T} (\sum_{a-1}^{m} F_{t-1}^{a} \frac{Z_t^a}{Z_{t-1}^a})(1 - \sigma \sum_{a=1}^{m} |F_t^a - F_{t-1}^a|) \tag{3.6}$$

where F_{t-1}^a is the effective portfolio weight of asset a before re-balancing

3.2.2 Risk adjusted returns: Differential Sharpe Ratio

To assess the performance of our portfolio at the initial selection as well as in subsequent portfolio re-balancing, we require a performance measure. This measure should capture both the portfolio expected returns and its associated risk. Since the 1960s, Sharpe ratio Sharpe, 1994 has been used as a risk weighted measure for investment performance.

$$S_t = \frac{average(R_t) - R_f}{StandardDeviation(R_t)} \tag{3.7}$$

Here we can see the main problem with this measure is that large positive returns are penalized in the performance. It is noteworthy that even Harry Markowitz, when developing his Modern Portfolio Theory, recognized that since only downside deviation is relevant to investors, using downside deviation to measure risk would be more appropriate than using standard deviation measure Rollinger and Hoffman, 2015.

For the reason above we chose to use a risk measure that only penalizes for downside risk. One such measure is the Sterling Ratio. For simplicity of computation, we use a modified version concerned with the downside deviation of returns.

$$SR = \frac{AverageReturn}{downsidedeviation} \tag{3.8}$$

For the purpose of this analysis we assume the absence of a risk free asset so as to assess the effects of trading in the stock market purely hence all risk measures are adjusted to assume a risk free rate of 0.

The *downsidedeviation* is defined as the root mean-square of the deviations of the realized return's under-performance from the target return where all returns above the target return are treated as under-performance of 0. It is mathematically defined as

$$downsidedeviation = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(min(0, R_t - T))^2} \tag{3.9}$$

where R_t is the i^{th} return, N = total number of returns, T = target return taken as 0 for this analysis.

The aim of an online model is to able to compute the effect of the current returns on the

performance measure. To do this we first define an exponential weighted moving average for the return and the downside deviation

Taking the first moment of returns as A given by

$$A = \frac{1}{T}\sum_{1}^{T}(R_t) \tag{3.10}$$

the exponential moving average for returns becomes:

$$A_t = A_{t-1} + \eta(R_t - A_{t-1}) \tag{3.11}$$

and for squared downside we have

$$DD_t^2 = DD_{t-1}^2 + \eta(min(0, R_t - T)^2 - DD_{t-1}^2) \tag{3.12}$$

taking the first order expansion of moving average returns and the squared downside deviation, we get:

$$SR \approx SR_{t-1} + \eta\frac{dSR}{d\eta} + O(\eta^2) \tag{3.13}$$

Since only the first order term depends on the return R_t at time t we define this as our Differential Risk Ratio (DRR) which is given by Moody and Saffell, 2001

$$\begin{aligned} DRR &= \frac{dSR}{d\eta} \\ &= \frac{R_t - 0.5A_{t-1}}{DD_t - 1}, R_t > 0 \\ &= \frac{DD_{t-1}^2(R_t - 0.5A_{t-1} - 0.5A_{t-1}R_t^2)}{DD_t^3 - 1}, R_t < 0 \end{aligned} \tag{3.14}$$

3.3 Recurrent Reinforcement Learning Approach

3.3.1 Maximising Immediate Return

The DRR above becomes our measure of the impact of return R at time T on the performance. We aim to optimize the performance function Differential Risk Ratio through optimal choice of θ. The is done by computing DRR in foward passes and adjusting θ through gradient ascent

$$\Delta\theta = \rho \frac{dDSR_T(\theta)}{d\theta} \tag{3.15}$$

where ρ is the learning rate selected through experimentation and the rate of change of change the performance measure with θ is given by:

$$\frac{dDSR_T(\theta)}{d\theta} = \sum_{t=1}^{T} \frac{dDSR_T}{dR_t}\left(\frac{dR_t}{dF_t}\frac{dF_t}{d\theta} + \frac{dR_t}{dF_{t-1}}\frac{dF_{t-1}}{d\theta}\right) \tag{3.16}$$

It should also be noted that $\frac{dF_t}{d\theta}$ is path dependent and therefore approximated using back propagation through time as

$$\frac{dF_t}{d\theta} = \frac{dF_t}{d\theta} + \frac{dF}{dF_{t-1}}\frac{dF_{t-1}}{d\theta} \tag{3.17}$$

As we are interested in an online learning model that is considers the effect of the current returns on performance, the equation can be written in terms of the Differential Risk ratio as

$$\Delta\theta = \rho \frac{dDSR_T(\theta)}{d\theta} \tag{3.18}$$

$$\approx \rho \frac{dDSR}{dR_t}\left(\frac{dR_t}{dF_t}\frac{dF_t}{d\theta} + \frac{dR_t}{dF_{t-1}}\frac{dF_{t-1}}{d\theta t - 1}\right) \tag{3.19}$$

3.4 Data description

For this research we will use monthly stock market data of six companies operating in different market segments in the New York Stock exchange. These are Microsoft, General Electric, JP Morgan, Walmart, Johnson and Johnson and Exonmobil. These companies are from different economic sectors hence a diversified portfolio.

The Capital Asset Pricing Model (CAPM) (Luenberger, 1997) and the Efficient Market Hypothesis (EMH) (Fama, 1970) expect that a market index made up of the underlying stocks is efficient and portfolio derived by regular re-balancing of its constituent assets cannot perform better. We therefore use this index performance to compare with the results of our trading agent under the assumption that CAPM and EMH are not exactly satisfied in reality and arbitrage opportunities can be exploited via proper strategies.

4. Empirical Results

4.1 Descriptive data analysis

The analysis period used to test the efficacy of this trading strategy consists of three epochs. The first is the bullish market prior to the 2008 financial crisis. The second period consists of the bear run that led to sharp decline in return following the crash. The last period consists of the post crash recovery. These give multiple scenarios to assess the efficacy of our trading strategy and can be observed in the cumulative assets wealth below;

Figure 4-1: Terminal wealth for each asset over time

The figure above shows the cumulative wealth as a result of the returns for each asset. JP Morgan is seen to perform well based on the cumulative wealth however it carries significant risk due to the sharp downside observed. GE is seen to have low returns while still bearing significant risk. We also observe that while Exxon Mobil has the second lowest average returns, it's downside deviation measure gives it the best risk adjusted return. These are

Measure	GE	JNJ	JPM	MSFT	WMT	XOM
Mean Returns	0.0007	0.0065	0.0094	0.0083	0.0045	0.0061
Downside Deviation	0.0574	0.0243	0.0539	0.0427	0.0307	0.03
Risk Adjusted Return	0.0125	0.2690	0.1751	0.1954	0.1461	0.2023

Table 4.1: Risk and return measures for each individual asset.

quantified through our modified downside risk adjusted measure as per the Table above.

4.2 Portfolio performance

In the analysis, both terminal wealth and the risk adjusted returns are used to appraise the portfolios. We obtain a visual representation of the terminal wealth over time and summarise the mean returns, downside risk and risk adjusted returns in table 4.3.

4.2.1 Buy and hold strategy on market portfolio

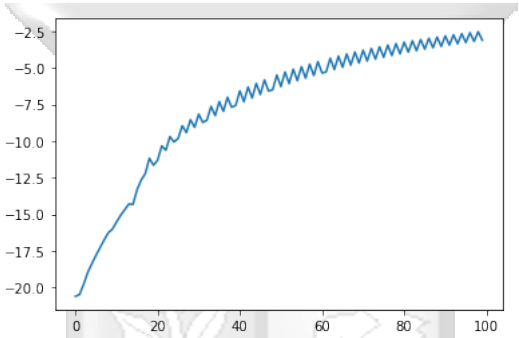
We compare the risk adjusted returns of a buy and hold strategy whose portfolio weights is equal to the market capitalization of the assets at the start of the investment period. The resultant portfolio returns over time is captured by Figure 4-2. The buy and hold portfolio does well in the initial period but experiences very sharp decline during the financial crisis. The recovery there after does reach the original highs showing but with significant variability along the way.

Figure 4-2: Terminal wealth over time for a buy and hold portfolio

4.2.2 Recurrent Reinforcement Learning portfolio

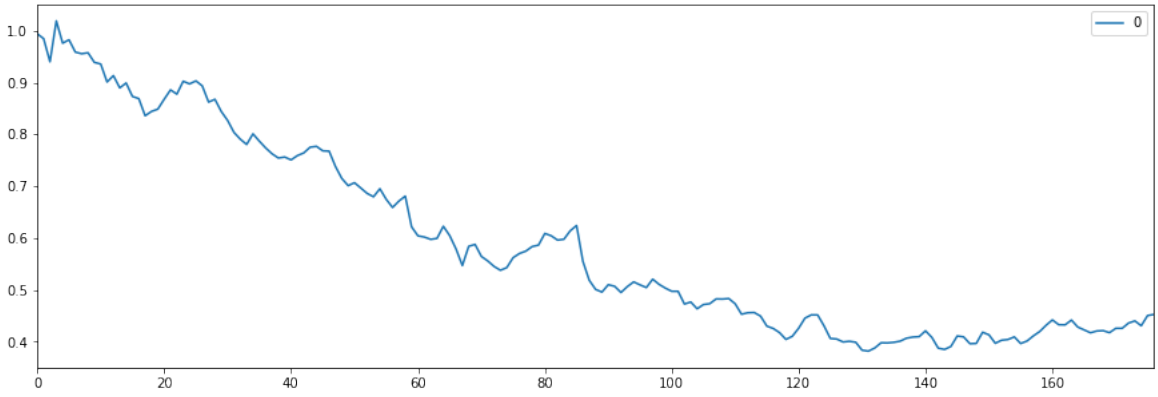
For the portfolio optimized through reinforcement learning we use the first 15 years of data, 1987 to 2002 for developing the strategy. At the start, initialization of the model parameters is done using random portfolio weights. The model is run 100 times with the previous runs selected weights being used to initialize the current run. These weights are then adjusted subsequently on a monthly basis using gradient accent as seen in the figure below.

Figure 4-3: Change in Differential Risk measure against number of runs



We first attempt to apply the initial learned model parameters to the entire test period to similar to the buy and hold strategy. Here we find that the portfolio returns start out profitably but quickly decline into losses as per the figure 4-4.

Figure 4-4: Returns of reinforcement learning where parameters are trained once

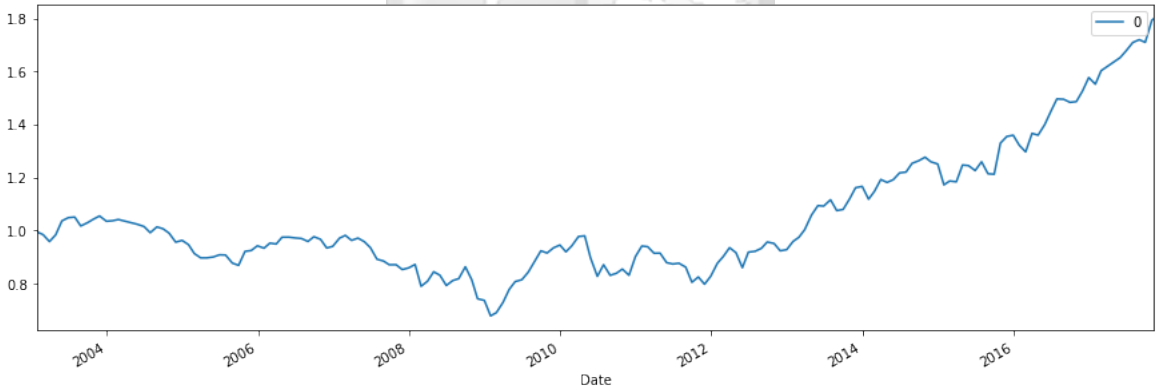


This is due to the model parameters having been set on old information that has since changed. This highlights the need of an actively managed portfolio updated on the latest information in the market.

To confirm the efficacy of an actively managed portfolio, we use a sliding window of the individual stocks risk and return information for parameter optimization. Here we use the previous 15 years data to adjust the model parameters for each portfolio readjustment date. This is done so as to take into account the recent information in the market as information from the far past is not relevant and leads to poor investment strategies.

The resultant portfolio has not only better downside risk characteristics but also higher risk adjusted return and terminal wealth. We also observe lower variability in the portfolio performance over the volatile period of the financial crisis as shown by figure 4-5 .

Figure 4-5: Terminal wealth over time for a reinforcement optimized portfolio



In comparing the three strategies above, a buy and hold strategy, a passive RRL approach and an active RRL approach, we find that the active RRL approach yields better performance as measured by both terminal wealth and risk adjusted return as seen in table 4.2

Measure	Buy and Hold Portfolio	RRL Passive	RRL Active
Mean Returns	0.0042	-0.0041	0.0038
Downside Deviation	0.031	0.0213	0.0216
Risk Adjusted Return	0.1362	-0.1931	0.1778
Terminal wealth	1.776	0.4528	1.8

Table 4.2: Risk and return comparison of RRL Portfolios.

4.2.3 Transaction costs

The initial analysis has been carried out with transaction costs of 0.6% of the transaction value. We also tested the efficacy of this strategy on varying values of transaction costs from 1% to 0.1%. Here we see that the portfolio returns favourable risk adjusted return and profitable wealth even for uncharacteristically high transaction costs of 1%. This is illustrated by the Table 4.3

Transaction cost	Mean Returns	Downside	Risk adjusted return
0.001	0.004092	0.021612	0.189349
0.002	0.003995	0.021649	0.184551
0.003	0.003898	0.021690	0.179734
0.004	0.003801	0.021733	0.174900
0.005	0.003704	0.021779	0.170059
0.006	0.003836	0.021580	0.177771
0.007	0.003738	0.021624	0.172872
0.008	0.003621	0.021696	0.166914
0.009	0.003409	0.021805	0.156322

Table 4.3: Risk and return measures for the Buy and hold portfolio.

5. Conclusion and Recommendation

5.1 Discussion of results

The application of recurrent reinforcement learning to portfolio optimisation is seen to yield profitable trading strategy as measured by both risk adjusted returns of 18% and terminal wealth of 180% at the end of the investment period as compared to a buy and hold strategy, whose risk adjusted return is 13% and terminal wealth is 177% which represents the market portfolio. The results obtained here show that contrary to Malkiel (2003), market predictability and inefficiencies can be profitably exploited. It is also observed that the actively managed portfolio gives significantly smoother returns as per figure 4-5 over the investment period with little variability being observed even in turbulent periods like the 2008 financial crisis. One key observation is that despite not shrinking the current portfolio weights towards those in the previous period as done by Suárez et al. (2009) we still obtain favourable investment strategy with switching of positions not causing excessive transaction costs. We also measure the effects of different transaction costs on the portfolio returns and find it to be inversely proportional to the returns as expected from literature. Here the RRL portfolio also outperforms the market portfolio within the reasonable range of transaction costs, between 0.3% and 0.6% of the transaction value Frazzini et al. (2012) as shown in table 4.3.

The computation time for determining the readjustment of the portfolio takes less than five minutes to run and therefore can be applied to the portfolio optimization problem but may not be appropriate for real time trading without further optimization.

5.2 Conclusion

In this paper we applied Recurrent Reinforcement Learning to the problem of optimal asset allocation with transaction costs to assess the efficacy of this approach. We used a risk measure that considers downside deviation as the performance measure due to its advantage in only penalizing downside deviation. We were able to apply this to the problem of initial selection and portfolio re-balancing on a monthly basis.

From the analysis we find that this investment strategy yields a trading strategy with minimized risk properties and outperforms a buy and hold approach centred on market efficiency and passive investment management.

5.3 Recommendations

The application used in this research extends previous studies that focused RRL in trading to portfolio management. The developed strategies can be tested on larger portfolios containing more assets as well as the risk free asset in order to extend its practical application. In addition, this study uses a monthly review of the portfolio to re-balance the portfolio. Further research can focus on optimal time for portfolio re-balancing to also be learned through recurrent reinforcement learning by making it a part of the decision problem. In this study, it is implicitly assumed that the investment strategy has zero impact on the market. A relaxation of this assumption is an area for further research to assess whether the strategies developed remain viable.

These investment strategy can be applied by fund managers looking to minimise downside risk for a given return. The portfolio can be trained with different targets of minimum

downside risk to suit different investor risk preferences.

Bibliography

Almahdi, S. & Yang, S. Y. (2017). An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, *87*, 267–279.

Barto, A. G., Sutton, R. S., & Watkins, C. (1989). *Learning and sequential decision making*. University of Massachusetts Amherst, MA.

Bellman, R. (1952). On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, *38*(8), 716.

Bertoluzzo, F. & Corazza, M. (2007). Making financial trading by recurrent reinforcement learning. In *International conference on knowledge-based and intelligent information and engineering systems* (pp. 619–626). Springer.

Brealey, R. A., Myers, S. C., Marcus, A. J., Wang, H., & Zhu, L. (2007). Fundamentals of corporate finance.

Dempster, M. A. & Leemans, V. (2006). An automated fx trading system using adaptive reinforcement learning. *Expert Systems with Applications*, *30*(3), 543–552.

Elton, E. J. & Gruber, M. J. (1971). Dynamic programming applications in finance. *The Journal of Finance*, *26*(2), 473–506.

Fabozzi, F. J., Kolm, P. N., Pachamanova, D. A., & Focardi, S. M. (2007). *Robust portfolio optimization and management*. John Wiley & Sons.

Fama, E. F. (1965). The behavior of stock-market prices. *The journal of Business*, *38*(1), 34–105.

Filos, A. (2019). Reinforcement learning for portfolio management. *arXiv preprint arXiv:1909.09571*.

Frazzini, A., Israel, R., & Moskowitz, T. J. (2012). Trading costs of asset pricing anomalies. *Fama-Miller working paper*, 14–05.

Gold, C. (2003). Fx trading via recurrent reinforcement learning. In *2003 ieee international conference on computational intelligence for financial engineering, 2003. proceedings.* (pp. 363–370). IEEE.

Harlow, W. V. (1991). Asset allocation in a downside-risk framework. *Financial analysts journal*, *47*(5), 28–40.

Kung, J. J. (2008). Multi-period asset allocation by stochastic dynamic programming. *Applied mathematics and computation*, *199*(1), 341–348.

Malkiel, B. G. (2003). Passive investment strategies and efficient markets. *European Financial Management*, *9*(1), 1–10.

Maringer, D. & Ramtohul, T. (2010). Threshold recurrent reinforcement learning model for automated trading. In *European conference on the applications of evolutionary computation* (pp. 212–221). Springer.

Markowitz, H. (1952). Portfolio selection. *The journal of finance*, *7*(1), 77–91.

Moody, J. & Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE transactions on neural Networks*, *12*(4), 875–889.

Moody, J. & Wu, L. (1997). Optimization of trading systems and portfolios. In *Proceedings of the ieee/iafe 1997 computational intelligence for financial engineering (cifer)* (pp. 300–307). IEEE.

Moody, J., Wu, L., Liao, Y., & Saffell, M. (1998). Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, *17*(5-6), 441–470.

Musumeci, J. [Jim] & Musumeci, J. [Joe]. (1999). A dynamic-programming approach to multiperiod asset allocation. *Journal of Financial Services Research*, *15*(1), 5–21.

Neisy, A. & Abkenar, T. G. (2014). International journal of operational research and decision science studies.

Neuneier, R. (1996). Optimal asset allocation using adaptive dynamic programming. In *Advances in neural information processing systems* (pp. 952–958).

Neuneier, R. (1998). Enhancing q-learning for optimal asset allocation. In *Advances in neural information processing systems* (pp. 936–942).

Pendharkar, P. C. & Cusatis, P. (2018). Trading financial indices with reinforcement learning agents. *Expert Systems with Applications*, *103*, 1–13.

Rollinger, T. N. & Hoffman, S. T. (2015). Sortino: A 'sharper'ratio. *CA, Newport Beach: Red Rock*.

Roy, A. D. (1952). Safety first and the holding of assets. *Econometrica: Journal of the econometric society*, 431–449.

Samuelson, P. A. (1969). Lifetime portfolio selection by dynamic stochastic programming. *The review of economics and statistics*, 239–246.

Sharpe, W. F. (1994). The sharpe ratio. *Journal of portfolio management*, *21*(1), 49–58.

Shukla, R. (2004). The value of active portfolio management. *Journal of Economics and Business*, *56*(4), 331–346.

Singh, S. P. (1994). *Learning to solve markovian decision processes* (Doctoral dissertation, University of Massachusetts at Amherst).

Spooner, T., Fearnley, J., Savani, R., & Koukorinis, A. (2018). Market making via reinforcement learning. In *Proceedings of the 17th international conference on au-*

tonomous agents and multiagent systems (pp. 434–442). International Foundation for Autonomous Agents and Multiagent Systems.

Suárez, A., Moody, J., & Saffell, M. (2009). Dynamic portfolio management with transaction costs. In *Multidisciplinary symposium on reinforcement learning*.

Sutton, R. S., Barto, A. G., & Williams, R. J. (1992). Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems Magazine*, *12*(2), 19–22.

Van Roy, B., Abu-Mostafa, Y., LeBaron, B., Lo, A., & Weigend, A. (2001). *Temporal-difference learning and applications in finance*. Cambridge, MA: MIT Press.

Wallace, S. W. & Høyland, K. (2007). Stochastic programming models for strategic and tactical asset allocation: A study from norwegian life insurance.

Yong, J. & Zhou, X. Y. (1999). *Stochastic controls: Hamiltonian systems and hjb equations*. Springer Science & Business Media.

6. APPENDICES

6.1 Implementation Code

Python Code

September 27, 2020

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.special import softmax
import math
from scipy.special import expit
from datetime import datetime, timedelta
import statistics

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

[2]:
```
price = pd.read_csv("SnP5_month.csv")
price.head()
price['Date'] = pd.to_datetime(price["Date"])
names = price.set_index('Date').columns
```

[3]:
```
logreturns = price.apply(lambda x: np.log(x) - np.log(x.shift(1)) if x.name in␣
 ↪['GE', 'JNJ', 'JPM', 'MSFT', 'WMT', 'XOM'] else x).dropna()
# Split the data into test and train datasets
logreturns.set_index('Date', inplace=True)
returns_array_all = logreturns.to_numpy()
returns_array_train = logreturns.loc['01/01/1987':'31/12/2002'].to_numpy()
returns_array_test = logreturns.loc['01/01/2003':'31/12/2017'].to_numpy()
```

[4]:
```
returns_mean = np.mean(returns_array_test,axis=0)
returns_downside = np.sqrt(np.mean(np.
 ↪where(returns_array_test<0,returns_array_test,0)**2,axis=0))
risk_adjusted_return = returns_mean/returns_downside
perfomance = pd.DataFrame(np.
 ↪stack((returns_mean,returns_downside,risk_adjusted_return)))
perfomance.columns = names
perfomance["Measure"] = ["Mean Returns","Downside Deviation","Risk Adjusted␣
 ↪Return"]
perfomance[['Measure','GE', 'JNJ', 'JPM', 'MSFT', 'WMT', 'XOM']]
```

[4]:

	Measure	GE	JNJ	JPM	MSFT	WMT	XOM
0	Mean Returns	0.000714	0.006546	0.009441	0.008344	0.004487	0.006062
1	Downside Deviation	0.057359	0.024325	0.053908	0.042687	0.030705	0.029965
2	Risk Adjusted Return	0.012456	0.269094	0.175140	0.195466	0.146141	0.202292

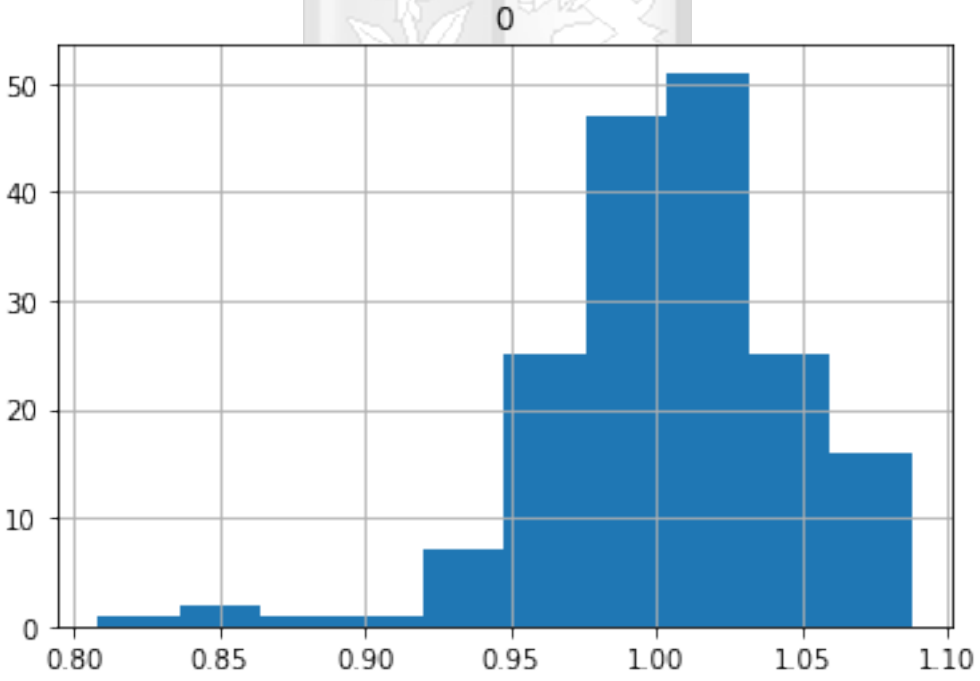
[5]:
```
wealth_each_T = 1
wealth_each_t = []
for row in returns_array_test:
    wealth_each_T = wealth_each_T*(row+1)
    wealth_each_t.append(wealth_each_T)

wealth_each_t_df = pd.DataFrame(wealth_each_t)
wealth_each_t_df.index = logreturns.loc['01/01/2003':'31/12/2017'].index
wealth_each_t_df.columns = names
wealth_each_t_df.plot(figsize=(15,10))
print(names)
print(wealth_each_T)
```

```
Index(['GE', 'JNJ', 'JPM', 'MSFT', 'WMT', 'XOM'], dtype='object')
[0.65849907 2.78680506 3.03375215 3.01645527 1.82011732 2.39384069]
```

[6]:
```
# Market capitalization start of 2016
marketcap_weights_2 =  [0.3848,0.1351,0.0187,0.0215,0.1205,0.3193]
portfolio_weights = np.multiply(np.ones(shape=(returns_array_test.
 ↪shape[0],returns_array_test.shape[1])),marketcap_weights_2)
wealth_buy_hold = 1+ np.sum(marketcap_weights_2*returns_array_test,axis=1)[1:]
returns_buy_hold = np.sum(marketcap_weights_2*returns_array_test,axis=1)[1:]
returns_buy_hold_mean = np.mean(returns_buy_hold)
returns_buy_hold_downside = np.sqrt(np.mean(np.
 ↪where(returns_buy_hold<0,returns_buy_hold,0)**2))
print("Mean Returns: " ,returns_buy_hold_mean)
print("Downside: " ,returns_buy_hold_downside)
print("Risk adjusted return: " ,returns_buy_hold_mean/returns_buy_hold_downside)
wealth_buy_hold_df = pd.DataFrame(wealth_buy_hold)
#wealth_buy_hold_df.plot(figsize=(15,5))
wealth_buy_hold_df.hist()
```

```
Mean Returns:  0.004219638153008188
Downside:  0.030962087571388318
Risk adjusted return:  0.13628403263439845
```

[6]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001F8022DEF48>]],
      dtype=object)
```



[7]:
```
wealth_buy_hold_T = 1
wealth_buy_hold_t = np.zeros(shape=(wealth_buy_hold.shape))
i = 0
for row in wealth_buy_hold:
    wealth_buy_hold_T = wealth_buy_hold_T * row
    wealth_buy_hold_t[i] = wealth_buy_hold_T
    i+=1
wealth_buy_hold_t_df = pd.DataFrame(wealth_buy_hold_t)
wealth_buy_hold_t_df.index = logreturns.loc['01/01/2003':'31/12/2017'].index[1:]
wealth_buy_hold_t_df.plot(figsize=(15,5))
print(wealth_buy_hold_T)
```

```
1.7761660216018917
```

1 2.0 Active portfolio management

[8]:
```
def train(returns,p,n,tx_cost,target,minR,num,init,init_matrix,weights_matrix):
    init = init
    init_matrix =init_matrix

    DDRs_list = []

    for number in range(num):
        weights_w = weights_matrix[0,:returns_array_train.shape[1]]
        weights_u = weights_matrix[0,returns_array_train.shape[1]:
 ↪returns_array_train.shape[1]*2]
        weights_v = weights_matrix[0,returns_array_train.shape[1]*2:
 ↪returns_array_train.shape[1]*3]

        Xt = weights_w*returns
        rt_prev = np.vstack((init,returns))[:-1,:]
        i = 0
```

```
        A_t = 0
        DD_t = 0

        f_prev = np.zeros(6)
        a_prev = 0
        change_i = 0
        for X in Xt:
            F_t_prev = softmax(f_prev)
            r_matrix = np.stack((X,f_prev,rt_prev[i,:]))
            f = np.tanh(np.sum(weights_matrix*r_matrix,axis=0))
            F_t = softmax(f)*np.sign(f)
            R =  np.sum(F_t_prev*X - tx_cost*np.abs(F_t - F_t_prev))

            A_t = (A_t + n * (R - A_t))
            DD_t = np.sqrt((DD_t**2)+n*((np.minimum(R,0)**2) - DD_t**2))

            a = (1.0 - np.tanh(f)**2)*r_matrix
            b =  tx_cost * np.abs(f - f_prev)
            c = a_prev
            d = X - tx_cost * np.abs(f - f_prev)
            DD_ = np.where(DD_t==0,1,DD_t)
            e = np.where(X >0 ,1/DD_,(np.square(DD_t) - A_t*0.5 )/(np.
↪power(DD_,3)))
            change_i += e * (a*b + c*d)
            f_prev=f
            a_prev = a
            i+=1
        DDRs_list.append(A_t/DD_t**2)
        changes = change_i/len(Xt)
        weights_matrix = p*changes +  weights_matrix

    return weights_matrix,DDRs_list
```

[9]:
```
def invest(returns,weights):

    init = np.zeros((1,returns.shape[1]))
    weights_w = weights[0]

    Ft = weights_w*returns
    Ft_prev = np.vstack((init,Ft))[:-1,:]
    rt_prev = np.vstack((init,returns))[:-1,:]
    r_matrix = np.concatenate((returns,Ft_prev,rt_prev),axis=1)

    f_list = []

    for r in r_matrix:
```

```
        f_list.append(np.tanh(np.sum(weights*r.reshape(3,returns.
↪shape[1]),axis=0)))

    f_t = np.array(f_list)

    F_t = softmax(f_t,axis=1)*np.sign(f_t)

    F_t_prev = np.vstack((init,F_t))[:-1,:]

    R_t = np.array(np.sum(F_t_prev*returns - tx_cost*np.abs(F_t -␣
↪F_t_prev),axis=1))
    r_t = F_t_prev*returns - tx_cost*np.abs(F_t - F_t_prev)
    return R_t, F_t, r_t
```

1.1 2.2 Use learned positions to invest in the entire test set.

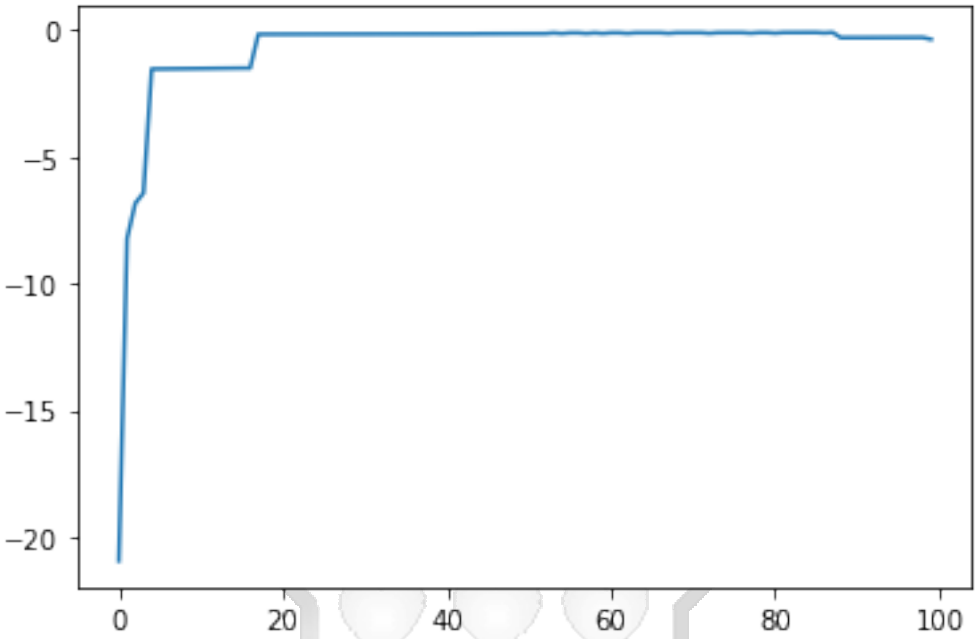
[10]:
```
p=0.05
n = 0.2
target = 0.00
minR=0
tx_cost = 0.006

init = np.zeros((1,returns_array_train.shape[1]))
init_matrix = np.zeros((1,returns_array_train.shape[1]*3))
```

[11]:
```
weights_matrix = np.random.rand(3,returns_array_train.shape[1])
```

[12]:
```
investment_weights, DRs_list =␣
↪train(returns_array_train,p,n,tx_cost,target,minR,100,init,init_matrix,weights_matrix)
```

[13]:
```
plt.plot(DRs_list)
```

[13]:
```
[<matplotlib.lines.Line2D at 0x1f80234a7c8>]
```



[14]:
```
returns, positions, rts = invest(returns_array_test,investment_weights)
```

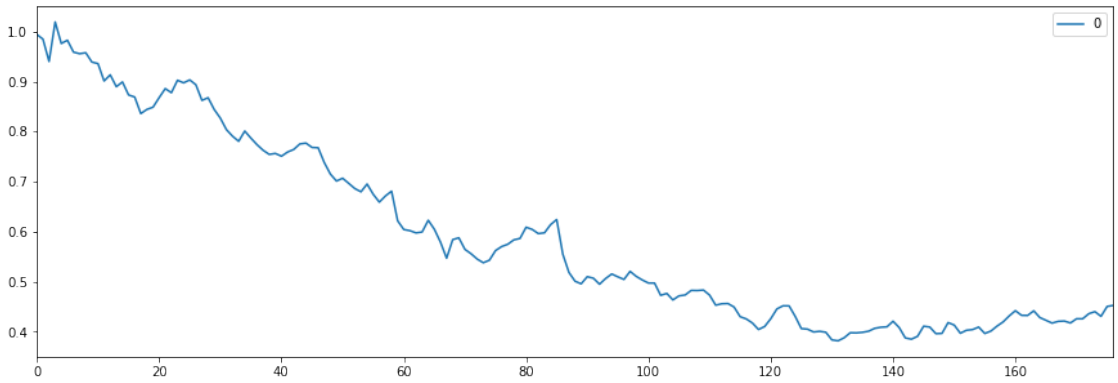
[15]:
```
returns_buy_hold_mean = np.mean(returns)
returns_buy_hold_downside = np.sqrt(np.mean(np.where(returns<0,returns,0)**2))
print("Mean Returns: " ,returns_buy_hold_mean)
print("Downside: " ,returns_buy_hold_downside)
print("Risk adjusted return: " ,returns_buy_hold_mean/returns_buy_hold_downside)
```

```
Mean Returns:  -0.004119166076425904
Downside:  0.021328503300311753
Risk adjusted return:  -0.1931296358880323
```

[16]:
```
wealth = 1+returns
wealth_T = 1
wealth_t = np.zeros(shape=(wealth.shape))
i=0
for row in wealth:
    wealth_T = wealth_T * row
    wealth_t[i] = wealth_T
    i+=1
```

[17]:
```
wealth_df = pd.DataFrame(wealth_t)
wealth_df.plot(figsize=(15,5))
print(wealth_T)
```

```
0.4528201373089252
```



[18]:
```
np.sum(np.where(returns>0,1,0))/len(returns)
```

[18]: 0.4519774011299435

2 Sliding Window

[19]:
```
train_length = len(returns_array_train)
start = 0
end = train_length-1
now = train_length
returns_sliding = logreturns
```

[20]:
```
p=0.05
n = 0.2
target = 0.00
minR=0
tx_cost = 0.006

init = np.zeros((1,returns_array_train.shape[1]))
init_matrix = np.zeros((1,returns_array_train.shape[1]*3))

weights_matrix = np.random.rand(3,returns_array_train.shape[1])
```

[21]:
```
F_t_prev = np.zeros((1,returns_array_train.shape[1]))
rt_prev = F_t_prev
returns_list = []
positions_list = []
```

[22]:
```
sliding_returns_array = returns_sliding.iloc[start:end].to_numpy()
trained_weights = investment_weights
```

[23]:
```
pd.DataFrame(investment_weights)
```

[23]:

	0	1	2	3	4	5
0	0.089367	3.360950	-0.361912	40.866709	0.063003	0.501685
1	0.388614	54.033316	1.979154	231.415295	0.337686	0.770520
2	0.357218	5.826284	-0.258558	15.695471	0.619221	0.488586

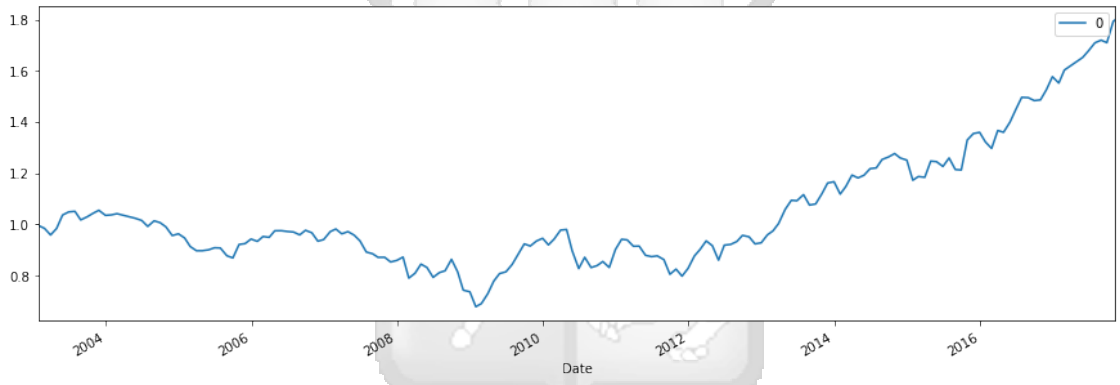
[24]:
```
while now < len(returns_sliding):
    sliding_returns_array = returns_sliding.iloc[start:end].to_numpy()
    runs = 10

    trained_weights, DDRs =␣
 ↪train(sliding_returns_array,p,n,tx_cost,target,minR,runs,init,init_matrix,trained_weights)

    current_returns = np.array(returns_sliding.iloc[now]).reshape(1,6)

    weights_w = trained_weights[0]
    Ft = weights_w*current_returns
    r_matrix = np.concatenate((current_returns,F_t_prev,rt_prev),axis=1)
    f_t = (np.tanh(np.sum(trained_weights*r_matrix.reshape(3,current_returns.
 ↪shape[1]),axis=0)))
    F_t = softmax(f_t)*np.sign(f_t)

    R_t = np.sum(F_t_prev*current_returns - tx_cost*np.abs(F_t-F_t_prev))
    returns_list.append(R_t)
    positions_list.append(F_t)

    F_t_prev = F_t.reshape(1,6)
    rt_prev = current_returns

    init = trained_weights[0]
    init_matrix = trained_weights.reshape(1,returns_sliding.shape[1]*3)

    start += 1
    end  +=1
    now = end+1
```

[26]:
```
returns  = np.array(returns_list).reshape(len(returns_list),1)
returns_buy_hold_mean = np.mean(returns)
returns_buy_hold_downside = np.sqrt(np.mean(np.where(returns<0,returns,0)**2))
print("Mean Returns: " ,returns_buy_hold_mean)
print("Downside: " ,returns_buy_hold_downside)
print("Risk adjusted return: " ,returns_buy_hold_mean/returns_buy_hold_downside)
```

```
Mean Returns:  0.0038362509380287994
Downside:  0.021579725343564515
Risk adjusted return:  0.17777107340121187
```

[27]:
```
wealth_sliding_window = 1+np.array(returns_list).reshape(len(returns_list),1)
wealth_T_sliding = 1
wealth_t_sliding = np.zeros(shape=(wealth_sliding_window.shape))

i=0
for row in wealth_sliding_window:
    wealth_T_sliding = wealth_T_sliding * row
    wealth_t_sliding[i] = wealth_T_sliding
    i+=1

wealth_df = pd.DataFrame(wealth_t_sliding)
wealth_df.index = logreturns.loc['01/01/2003':'31/12/2017'].index
wealth_df.plot(figsize=(15,5))
print(wealth_T_sliding)
```

```
[1.7990456]
```



[84]:
```
returns_all = np.array(returns_list)
returns_mean = np.mean(returns_all)
returns_downside = np.sqrt(np.mean(np.where(returns_all<0,returns_all,0)**2))
risk_adjusted_return = returns_mean/returns_downside
perfomance = pd.DataFrame(np.
 ↪stack((returns_mean,returns_downside,risk_adjusted_return)))
perfomance["Measure"] = ["Mean Returns","Downside Deviation","Risk Adjusted␣
 ↪Return"]
perfomance
```

[84]:

	0	Measure
0	0.003836	Mean Returns
1	0.021580	Downside Deviation
2	0.177771	Risk Adjusted Return

3 Different transaction costs

[20]:
```
for i in list(np.arange(0.001,0.01,0.001)):

    p=0.05
    n = 0.2
    target = 0.00
    minR=0
    tx_cost = 0.006

    init = np.zeros((1,returns_array_train.shape[1]))
    init_matrix = np.zeros((1,returns_array_train.shape[1]*3))

    #weights_matrix = np.random.rand(3,returns_array_train.shape[1])
    #print(weights_matrix)
    weights_matrix = np.array(
        [
            [0.078290,0.722538,0.313503,0.478544,0.049602,0.367066],
            [0.358434,0.202546,0.796813,0.865089,0.312694,0.444018],
            [0.227023,0.707982,0.100362,0.601250,0.293784,0.345326]
        ])

    investment_weights, DRs_list =␣
↪train(returns_array_train,p,n,tx_cost,target,minR,100,init,init_matrix,weights_matrix)

    train_length = len(returns_array_train)
    start = 0
    end = train_length-1
    now = train_length
    returns_sliding = logreturns

    p=0.05
    n = 0.2
    target = 0.00
    minR=0

    init = np.zeros((1,returns_array_train.shape[1]))
    init_matrix = np.zeros((1,returns_array_train.shape[1]*3))

    weights_matrix = np.random.rand(3,returns_array_train.shape[1])

    F_t_prev = np.zeros((1,returns_array_train.shape[1]))
    rt_prev = F_t_prev
    returns_list = []
    positions_list = []

    sliding_returns_array = returns_sliding.iloc[start:end].to_numpy()
```

```
    trained_weights = investment_weights

    tx_cost = i
    print("Transaction cost is: ", i)
    while now < len(returns_sliding):
        #print("Now is ",now)
        sliding_returns_array = returns_sliding.iloc[start:end].to_numpy()
        runs = 10

        trained_weights, DDRs =␣
↪train(sliding_returns_array,p,n,tx_cost,target,minR,runs,init,init_matrix,trained_weights)

        current_returns = np.array(returns_sliding.iloc[now]).reshape(1,6)

        weights_w = trained_weights[0]
        Ft = weights_w*current_returns
        r_matrix = np.concatenate((current_returns,F_t_prev,rt_prev),axis=1)
        f_t = (np.tanh(np.sum(trained_weights*r_matrix.reshape(3,current_returns.
↪shape[1]),axis=0)))
        F_t = softmax(f_t)*np.sign(f_t)

        R_t = np.sum(F_t_prev*current_returns - tx_cost*np.abs(F_t-F_t_prev))
        returns_list.append(R_t)
        positions_list.append(F_t)

        F_t_prev = F_t.reshape(1,6)
        rt_prev = current_returns

        init = trained_weights[0]
        init_matrix = trained_weights.reshape(1,returns_sliding.shape[1]*3)

        start += 1
        end  +=1
        now = end+1

    start = 0
    end = train_length-1
    now = train_length

    returns  = np.array(returns_list).reshape(len(returns_list),1)
    returns_buy_hold_mean = np.mean(returns)
    returns_buy_hold_downside = np.sqrt(np.mean(np.
↪where(returns<0,returns,0)**2))
    print("Mean Returns: " ,returns_buy_hold_mean)
    print("Downside: " ,returns_buy_hold_downside)
    print("Risk adjusted return: " ,returns_buy_hold_mean/
↪returns_buy_hold_downside)
```

```
    ␣
↪print("**************************************************************************")
```

```
Transaction cost is:  0.001
Mean Returns:  0.004092279821685278
Downside:  0.02161232152307252
Risk adjusted return:  0.1893493865208562
**************************************************************************
Transaction cost is:  0.002
Mean Returns:  0.003995414726330927
Downside:  0.02164941626484336
Risk adjusted return:  0.18455069076477176
**************************************************************************
Transaction cost is:  0.003
Mean Returns:  0.003898480740143183
Downside:  0.021690249222995613
Risk adjusted return:  0.17973425293841638
**************************************************************************
Transaction cost is:  0.004
Mean Returns:  0.0038011581331409724
Downside:  0.021733347643210853
Risk adjusted return:  0.1748997989423131
**************************************************************************
Transaction cost is:  0.005
Mean Returns:  0.003703687946951759
Downside:  0.021778843781934724
Risk adjusted return:  0.17005897944058543
**************************************************************************
Transaction cost is:  0.006
Mean Returns:  0.0038362509380287994
Downside:  0.021579725343564515
Risk adjusted return:  0.17777107340121187
**************************************************************************
Transaction cost is:  0.007
Mean Returns:  0.0037381498809164705
Downside:  0.021623839125381305
Risk adjusted return:  0.17287170234857885
**************************************************************************
Transaction cost is:  0.008
Mean Returns:  0.0036214222814384208
Downside:  0.021696353143364544
Risk adjusted return:  0.16691387061728252
**************************************************************************
Transaction cost is:  0.009000000000000001
Mean Returns:  0.0034086606644930756
Downside:  0.021805412472409323
Risk adjusted return:  0.15632176959762165
```

6.2 Similarity Checker Report

Curiginal

Document Information

Analyzed document	Masters_Thesis.pdf (D117945602)
Submitted	2021-11-09 09:10:00
Submitted by	
Submitter email	donald.kibet@strathmore.edu
Similarity	6%
Analysis address	library.strath@analysis.urkund.com

Sources included in the report

W	URL: https://faculty.comm.virginia.edu/sdb7e/files/mcintireSeminars/cong_AlphaPortfolio.pdf Fetched: 2021-11-09 09:11:00	2
W	URL: https://www.cmegroup.com/education/files/rr-sortino-a-sharper-ratio.pdf Fetched: 2021-11-09 09:11:00	3
W	URL: https://arxiv.org/pdf/1909.09571 Fetched: 2021-11-09 09:11:00	3
W	URL: https://www.researchgate.net/publication/317622713_An_adaptive_portfolio_trading_system_A_risk-return_portfolio_optimization_using_recurrent_reinforcement_learning_with_expected_maximum_drawdown Fetched: 2021-11-09 09:11:00	6
W	URL: https://scholarsmine.mst.edu/cgi/viewcontent.cgi?article=8909&context=masters_theses Fetched: 2021-11-09 09:11:00	3
W	URL: https://arxiv.org/pdf/2007.06460 Fetched: 2021-11-09 09:11:00	1
W	URL: https://dspace.cuni.cz/bitstream/handle/20.500.11956/121295/120371753.pdf?sequence=1 Fetched: 2021-11-09 09:11:00	2
W	URL: https://2021.ecmlpkdd.org/wp-content/uploads/2021/07/sub_657.pdf Fetched: 2021-11-09 09:11:00	1
W	URL: https://arxiv.org/pdf/1808.09940 Fetched: 2021-11-09 09:11:00	1

Curiginal

Entire Document

DYNAMIC PORTFOLIO OPTIMIZATION USING REINFORCEMENT LEARNING by YEGON DONALD KIBET 60169 Thesis presented in fulfillment of the academic requirement for the degree of Master of Science in Mathematical Finance at Strathmore University JUNE 2020

Declaration and Approval Declaration I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself. © No part of this thesis may be reproduced without the permission of the author and Strathmore University Signature ..... Date ..... Donald Yegon Approval The thesis of Donald Yegon was reviewed and approved by the following: Ferdinand Othieno Signature ..... Date ..... Dean, Strathmore Institute of Mathematical Sciences Strathmore University Dr. Caroline Kariuki Signature ..... Date ..... Director, ofice of Educational Development Strathmore University ii

ABSTRACT This study uses machine learning in the development of a dynamic investment strategy for portfolio optimization. We aim to explore the efficiency of this approach over a pas- sively managed portfolio and assess the whether transaction costs erode the gains in the dynamically managed portfolio. To this end we explore the application of recurrent rein- forcement learning for optimal asset allocation of a portfolio consisting of stock prices for six companies in different sectors. We develop an environment based on monthly historic prices of these stocks and a re-balancing agent that acts on the environment. The risk and return factors of the individual stock are taken as the state of the environment. Using a modified version of Sterling Ratio as the performance measure, we select model parameters through direct recurrent reinforcement learning from historical data and test the efficacy of the strategy on unseen data. From the analysis we find that the regularly re-balanced portfolio out performs the market portfolio based on buy and hold strategy based on both the terminal wealth and the risk adjusted return measure. Keywords Reinforcement learning, downside deviation, portfolio optimization iii

TABLE OF CONTENTS LIST OF FIGURES vi LIST OF TABLES vii LIST OF SYMBOLS AND ABBREVIATIONS viii 1 INTRODUCTION 1 1.1 Background . . . . . 1 1.2 Problem statement . . . . . 5 1.3 Research objectives . . . . . 5 1.4 Research Question . . . . . 6 2 LITERATURE REVIEW 7 2.0.1 Dynamic asset allocation using Dynamic Programming . . . . . 7 2.0.2 Dynamic asset allocation using Reinforcement Learning . . . . . 8 2.0.2.1 Dynamic asset allocation using Q-Learning . . . . . 9 2.0.2.2 Dynamic asset allocation using Recurrent Reinforcement 10 3 RESEARCH METHODOLOGY 14 3.1 Structure of portfolios . . . . . 14 3.2 Financial performance functions . . . . . 16 3.2.1 Profit and wealth . . . . . 16 3.2.2 Risk adjusted returns: Differential Sharpe Ratio . . . . . 16 iv

3.3 Recurrent Reinforcement Learning Approach . . . . . 19 3.3.1 Maximising Immediate Return . . . . . 19 3.4 Data description . . . . . 20 4 EMPIRICAL RESULTS 21 4.1 Descriptive data analysis . . . . . 21 4.2 Portfolio performance . . . . . 22 4.2.1 Buy and hold strategy on market portfolio . . . . . 22 4.2.2 Recurrent Reinforcement Learning portfolio . . . . . 23 4.2.3 Transaction costs . . . . . 25 5 Conclusion and Recommendation 27 5.1 Discussion of results . . . . . 27 5.2 Conclusion . . . . . 28 5.3 Recommendations . . . . . 28 6 APPENDICES 34 v

LIST OF FIGURES 1-1 Investment as a feedback loop . . . . . 2 3-1 Plot of Tanh function . . . . . 15 4-1 Terminal wealth for each asset over time . . . . . 21 4-2 Terminal wealth over time for a buy and hold portfolio . . . . . 22 4-3 Change in Differential Risk measure against number of runs . . . . . 23 4-4 Returns of reinforcement learning where parameters are trained once . . . . 23 4-5 Terminal wealth over time for a reinforcement optimized portfolio . . . . . 24 vi

LIST OF TABLES 4.1 Risk and return measures for each individual asset. . . . . 22 4.2 Risk and return comparison of RRL Portfolios. . . . . 25 4.3 Risk and return measures for the Buy and hold portfolio. . . . . 26 vii

LIST OF SYMBOLS AND ABBREVIATIONS DP Dynamic Programming TDLearning Temporal difference Learning RRL Recurrent Reinforcement Learning MDPL Markov Decision Process viii

1. INTRODUCTION 1.1 Background Quantitative investment management consists of two dominant schools of thoughts, passive and active portfolio management. In passive asset management the belief is that the market is efficient Fama

(1965). Research has shown that despite markets being predictable and containing inefficiencies, exploitation of these does not lead to profitable strategy Malkiel (2003). The second school of thought is active asset management. These methods are motivated with the belief that the portfolio can be re balanced at regular time intervals to yield an even greater return. This introduces a new complexity due to the market frictions that are incurred whenever the portfolio is adjusted. Shukla (2004) showed that this methods do not on average generate much return net of transaction costs. In this study we test this assertion by researching the efficacy of an actively managed portfolio using a novel approach. One of the main tasks faced by an active portfolio manager is optimal asset allocation. This consists of the problem of initial selection and portfolio re-balancing. The investment decision, known as a strategy is typically made in terms of weights that are proportions of the total wealth. In subsequent time steps, the problem of Portfolio re-balancing is then faced. This involves adjusting the portfolio weights to maintain its optimal state. The result is a feedback loop between the market and the investor shown in Figure 1-1. From this vantage point, active investment management can be viewed as a dynamic system which can be optimized through optimal control theory. Optimal control theory is a branch 1

Figure 1-1: Investment as a feedback loop of optimization theory concerned with actions applied on dynamic systems to achieve desired outcomes Neisy and Abkenar (2014) . Here a dynamic system is any system whose state changes with time. To influence this system, a goal is set in terms of an objective function which we aim to maximise. This is achieved through control parameters which are the means of influencing the system behaviour. As markets consist of assets whose value changes over time, they can be viewed as dynamic systems. In dynamic portfolio management, the investors desired utility forms the objective function of the control system with the investment decisions, typically in form of portfolio weights representing the control parameters. A prominent approach to optimizing control in dynamic systems is through dynamic pro- gramming. Introduced by Bellman (1952), it involves breaking down complex problems into simpler sub problems recursively. This is predicated on the assumption that the problem has optimal substructure. Elton and Gruber (1971) introduced dynamic programming as an optimal control method- ology applicable to many areas in finance including asset allocation. This approach lends itself nicely to the dynamic asset allocation problem but encounters challenges in higher dimensional problems. It also requires a model of the underlying to be known which is not 2

always feasible. Developments in computing offer opportunities at solving the challenges faced by dynamic programming through Reinforcement Learning. Reinforcement learning is a branch of machine learning that is similar to optimal control. It is an umbrella term used to describe problems, their applicable class of solutions and the study of these problems and solutions. The basic structure consists of an agent and the environment. The agent reads the state of the environment and takes action on the environment at each time step. The environment provides reward based on the action taken by the agent. The agent therefore aims to maximize the rewards from the environment by adjusting actions towards those that give more reward through trial and error without explicitly being informed of the optimal actions. Filos (2019). From this formulation we can see

in some cases, a model of the environment. Sutton, Barto, and Williams (1992) The policy can be viewed as the agent's behaviour when acting in an environment. Is is essentially the mapping of external stimuli to an action. In portfolio management this is the investment strategy. The reward signal is the environments response to the agent's actions. In our case the reward signal is in form of risk and returns from the investment portfolio. The agent then determines how desirable this rewards are through a value function. The formulation for value function in a portfolio management scenario can be through utility functions or risk adjusted return functions. In some cases, we may have the actual model of the environment which is a mathematical formulation of the system upon which the agent acts. Filos (2019) In order to optimize a portfolio, a performance measure is required which we view as 3

the value function in reinforcement learning environment. Since portfolio managers are concern with both the risk as well as the return Markowitz (1952), a risk adjusted return measure is suitable. On such strategy is the Sharpe ratio. This ratio indicates the average return per unit of risk in excess of the risk free rate of return. Sharpe (1994) Intuitively however, investors are only concern with negative risk, that is those leading to losses Roy (1952). In order to avoid penalizing large positive variability, downside risk measures are considered. Here only the downside deviation of risk is measured against the expected return. Harlow (1991) Many investment computation assume friction-less markets. This assumption may lead to overstating of the expected returns from investments. In reality however, financial markets are subject to transaction costs such as fees, commissions and taxes. When taken into account, these costs may limit the arbitrage

opportunities observed in the market as well as wipe out the expected gains from the investments.Frazzini, Israel, and Moskowitz (2012) This research focuses on the active portfolio management through application of reinforce- ment learning to the selection of a dynamic investment portfolio strategy that maximises against risk adjusted returns. We use a risk adjusted return measure that focuses on down- side deviation as well as the impact of the current portfolio re-balancing decision on the returns. In order to test efficacy in real world applications, we incorporate the effects of transaction costs in the portfolio performance. We then assess whether this dynamically re balanced portfolio yields better risk adjusted return as compared to a static portfolio whos weights are selected at the initial investment period and held to maturity. 4

1.2 Problem statement Studies on portfolio optimization through dynamic optimal control focus on friction-less markets. Introduction of transaction costs to such solutions lead to increased complexity making the solution of such problems cumbersome. In addition, approaches that have been applied to this problems typically assume a well-defined model of the underlying asset which is not always feasible. These challenges have resulted in limited practical application of these methods. Novel approaches such as reinforcement learning have been shown to offer promising results in a trading setting and can be extended to address the above challenges in a portfolio management setting. In this study, we use direct reinforcement learning to find an optimal investment strategy that takes into account the resultant market frictions when selecting desired portfolio weights. By employing the use of a modified risk measure suitable for optimizing against downside risk, we construct an actively managed portfolio that takes positions in a multi-asset environment. 1.3 Research objectives • Develop reinforcement learning framework that learns from historic data the optimal investment strategy. • Incorporate effect of transaction costs on a dynamic investment portfolio 5

1.4 Research Question Does an actively managed portfolio using Reinforcement Learning yield better results than buy and hold strategy in the presence of transaction costs 6

2. LITERATURE REVIEW The development of quantitative methods in portfolio management can be traced back to 1952 with the publication of the seminal paper by Harry Markowitz. In his paper, the relationship between risk and return was for the first time formally defined rather than making separate considerations for this two aspects of an investment Markowitz (1952). This idea was revolutionary because of two key ideas. The first is that it unified two components that had before been considered separately, the risk and return relationship as well as their resultant diversification benefits of their correlation. Secondly, the decision making process was formalized as an optimization problem. Fabozzi, Kolm, Pachamanova, and Focardi (2007) The method developed by Markowitz does provide a great foundation for portfolio analysis but suffers from the fact that it is focused on one period appraisal of portfolios. In reality portfolio managers are often taking considerations over multiple periods. When making considerations over multiple time periods, the problem becomes a dynamic optimal control problem whose solution requires the application of more complex approaches. Researchers have attempted to solve the resultant optimal control problem through dynamic program- ming Samuelson (1969) and reinforcement learning approaches such as TD learning Van Roy, Abu-Mostafa, LeBaron, Lo, and Weigend (2001) and Q learning Neuneier (1996). 2.0.1 Dynamic asset allocation using Dynamic Programming The need for multi-period consideration led Samuelson (1969) to formulate and solve for a framework that considers the lifetime consumption and investment using dynamic stochastic programming. Building on this view that previous work focused on static analysis while in 7

reality, relationship between current and future decisions should be considered, Elton and Gruber (1971) undertook a survey of the application of dynamic programming in finance. In the time since several approaches to optimizing asset allocation using dynamic pro- gramming have found application by practitioners and academia. Wallace and Høyland (2007) developed a stochastic asset allocation approach for both tactical and strategic asset allocation within a Norwegian insurance firm. Musumeci and Musumeci (1999) applied dynamic programming to retirement asset allocation problem with different investor pref- erences while Kung (2008) set up a multi period asset allocation model for long term and short term bonds. The application of dynamic programming gives a solution to the temporal problem of dynamic asset allocation. It however requires a model of the system to exist which is not always feasible. In addition, the dynamic programming approach requires the solution of the Hamillton Jacobi Bellman (HJB) Yong and Zhou (1999) equation which becomes complex when considering higher dimensional problems such as multi-asset environments. 2.0.2 Dynamic asset allocation using Reinforcement Learning From Brealey, Myers, Marcus, Wang, and Zhu (2007) we know that the investor seeking to build an optimal portfolio needs to solve an optimization for two problems. The first is the expected yields for all the available assets simultaneously and the second being portfolio construction based on their risk appetites. In multi-period models, this is further complicated by transaction costs if the investor intends to revise their decision at regular time intervals. Neuneier (1996) developed a framework for combining these two tasks, modelling the asset 8

yields and search for the optimal portfolio through combining them into a Markov Decision Process (MDP). MDP provides a model for multistage decision making in stochastic envi- ronments. Neuneier (1996) defines the MDP as a finite state set S = 1; 2:::n, a finite set of admissible control actions for each state i 2 S, a set of transition probabilities p ij , which describe the system dynamics and a return function r„i; j; u„i"" with i; j 2 S, u„i" 2 U„i". For MDPs, when the discrete state space is small and and if an accurate model of the system exists, dynamic programming can be used to obtain the solution. On the other hand for large state spaces without explicit models, reinforcement learning approaches such as Q learning can be applied to obtain the solution. Singh (1994) 2.0.2.1 Dynamic asset allocation using Q-Learning Neuneier (1996) then proceeds to apply an iterative solution on the Bellman Equation to find an optimal policy. Using an algorithm known as Q learning as the optimization strategy and neural networks as the value function approximate, he was able to obtain superior results in the German stock market. Q-Learning (QL), is a reinforcement-learning method that does not require a model of the system but optimizes the policy by sampling state-action pairs and returns while interacting with the system Barto, Sutton, and Watkins (1989). Two years later, he built on his work developing a new formulation that would only use one value function for multiple assets and allow for model free policy iteration. This new framework allowed for relaxation of one of the main assumptions from his previous work, that of friction-less markets and allows for several constraints to be considered in the asset allocation processes through the introduction of an artificial deterministic step. Neuneier (1998). The Q-learning approach therefore extends the solution of the dynamic optimal control prob- 9

lem to cases where dynamic programming encounters challenges. Its principle is founded on discounted future rewards. A key question that arises is whether it is more advantageous in the long run to focus on expected future discounted returns or to optimise for immediate returns. The Q learning approach focuses on discounted future returns whereas immediate returns approach can be achieved through Direct Recurrent Reinforcement Learning. 2.0.2.2 Dynamic asset allocation using Recurrent Reinforcement Around the same time Moody and Wu (1997) was developing a direct reinforcement ap- proach to solving the problem of asset allocation. This work compared methods optimized by a forecast of future rewards, Supervised Learning to one optimised directly through Direct Reinforcement. The framework developed takes the prior positions taken in the as- sets available as input in the current decision hence resulting in a Recurrent Reinforcement Learning (RRL) as follows F t = F„ t ; F t*1 ; I t " (2.1) where F t is the positions taken at time t, t denotes the system parameters at time t and I t denotes the information set at time t which includes the present and past values of the price series. They introduced a new measure of portfolio performance, the differential Sharpe ratio. This measure considers exponential weighted moving average of the returns and can therefore be used in determining the impact of the current returns on a trade or investment portfolio. When taken as the optimization problem, this was found to yielded better empirical results than profit optimization. The measure was shown to be more suited for online optimizing trading systems as compared to running and moving Sharpe Ratios. One key innovation of 10

this research was the use of recurrent reinforcement learning to capture path dependence of trading decisions. Moody, Wu, Liao, and Saffell (1998) then compared the RRL approach to a Q-learning approach. The key comparison was that whereas RRL optimizes for immediate rewards, Q learning approximates discounted future rewards with the following formulation of the Q learning version Barto et al. (1989) of the Bellman Equation Q „x; a" = U„x; a" + n Õ y=0 P xy „a"max b Q „y; b" (2.2) where n is the number of system states, P xy „a" is the transition probability from x to y given action a and is the discount factor. The empirical results showed that Q-learning performs better on a single asset environment but worse on multi-asset allocation problem due to excessive changing of position hence more incurred transaction costs. Given the favourable results over both supervised learning approaches and Q learning, Moody and Saffell (2001) set out to assess the performance of a recurrent reinforcement learning model in different settings. Here a new performance measure that optimizes against downside risk was introduced. This is the differential Sterling ratio also used in this research. Most of the works on recurrent reinforcement learning in finance focused on trading prob- lems. Suárez, Moody, and Saffell (2009) modified the RRL trading solution to be applied to a portfolio management setting. This was motivated by the view that Markowitz and other methods end up finding corner solutions to the portfolio optimization problem. These kind of solutions are prone to switching behaviour. The key innovation in this was to shrink the 11

current weights towards the prior portfolio weights through the following formulation: F n „ ; w" = F n + „1 * "F s n (2.3) where F n is the composition of the portfolio currently, F s n is the prior portfolio composition and is a hyper-parameter that determines the relative importance of the two positions in the final output. More recently, Maringer and Ramtohul (2010) built on the work done by Moody and Wu (1997) developing a means of capturing the effects of regimes in the market through a threshold recurrent reinforcement learning algorithm. This was aimed at addressing the concerns with the simplistic nature of solutions previously developed that were unlikely to cater for the non linearity and structural breaks observed in the market. Recurrent Reinforcement learning has also been shown to yield promising results in foreign exchange markets. Gold (2003) showed that a single layered Recurrent Reinforcement Learning outperforms a

two layered network in the foreign exchange market. Dempster and Leemans (2006) developed a three layered Recurrent Reinforcement Learning that entailed a machine learning layer, a risk management layer and a utility optimization layer. Spooner, Fearnley, Savani, and Koukorinis (2018) showed the application of reinforcement learning in market making yields promising results. Bertoluzzo and Corazza (2007) tested Recurrent Reinforcement Learning on data from 9 financial Markets around the world. Their implementation was using an adaptive neural network and the results showed positive returns in all but one of the tested applications. Almahdi and Yang (2017) employed the use of expected maximum draw down as the 12

performance measure. They further used various transaction cost consideration to validate the results of the developed recurrent reinforcement learning model. Pendharkar and Cusatis (2018) researched the application of reinforcement learning in personal retirement investment optimization. The research they considered portfolio returns or differential Sharpe ratio to measure performance. Results indicated that the high-learning frequency trading agent consistently beats both the single asset stock and bond cumulative returns by a significant margin. 13

3. RESEARCH METHODOLOGY 3.1 Structure of portfolios The portfolio developed can take varying long and short positions in an asset. This can be built up starting from a simple trading agent assumes discrete fixed size trading positions in a given asset. F t 2 »*1; 1... (3.1) where F t is the position assumed in each asset varying between -1 and 1 where negative values constitute short selling and positive values constitute long positions. In order to properly take into account the gains resultant from transitioning from previous position F t*1 to F t effect of transaction costs must be considered. The position F t is therefore a result of F t = F„ ; F t*1 ; I t " (3.2) where I t is the information at time t, F t*1 is the prior positions taken in the portfolio and is the current asset state. We adjust the amount invested in each asset through a model parameter and an activation function that takes and an asset return x as it's inputs. The activation function then gives varying quantities of f t as it's output which corresponds to a trading agent that takes varying quantities of wealth in multiple assets available to it. These parameters are initialized randomly and optimized through gradient ascent. For this analysis we use desire an activation function that gives varying long and short 14

positions in the available assets. Therefore the tanh function given as follows is selected: f t = tanh„x it " (3.3) Since we require the weights to sum up to 1 as they are proportions of the overall wealth, Figure 3-1: Plot of Tanh function this introduces a constraint in the optimization. A simple way to solve this is by using softmax Moody and Saffell, 2001 outputs outside the training hence the expression for F t becomes F t = so f tmax„ f t „ t*1 ; F t*1 I t "" (3.4) Based on these positions taken, we are able to compute returns at time T as R T = F T*1 r t * jF T*1 * F T j (3.5) where is the transaction cost, r t is the return of the portfolio at time t and F is the positions taken in the position. Here we take the transaction costs to be 0:6% Frazzini et al. (2012) We aim to develop a means to assess risk adjusted returns of different portfolio mixes at each time step. These returns are used as the basis of the investment strategy. The return 15

informs the proportion of wealth invested in each asset. 3.2 Financial performance functions 3.2.1 Profit and wealth The portfolio's ultimate aim is to maximise a measure in form of a function that can be utility functions of wealth, return or a risk adjusted return. In this analysis, we first use multiplicative profits to assess whether the strategy is profitable. This is due to the portfolio being constructed as a fixed fraction of the accumulated wealth in long and short positions. For our multi asset, regularly re-balanced portfolio, the wealth at the end of the investment period is given by W T = W 0 T Ö t=1 1 + R t = W o T Ö t=1 „ m Õ a*1 F a t*1 Z a t Z a t*1 "„1 * m Õ a=1 jF a t * F a t*1 j" (3.6) where F a t*1 is the effective portfolio weight of asset a before re-balancing 3.2.2 Risk adjusted returns: Differential Sharpe Ratio To assess the performance of our portfolio at the initial selection as well as in subsequent portfolio re-balancing, we require a performance measure. This measure should capture both the portfolio expected returns and its associated risk. Since the 1960s, Sharpe ratio Sharpe, 1994 has been used as a risk weighted measure for investment performance. S t = average„R t " * R f StandardDeviation„R t " (3.7) 16

Here we can see the main problem with this measure is that large positive returns are penalized in the performance.

deviation measure Rollinger and Hoffman, 2015. For the reason above we chose to use a risk measure that only penalizes for downside risk. One such measure is the Sterling Ratio. For simplicity of computation, we use a modified version

concerned with the downside deviation of returns. SR = AverageReturn downsidedeviation (3.8) For the purpose of this analysis we assume the absence of a risk free asset so as to assess the effects of trading in the stock market purely hence all risk measures are adjusted to assume a risk free rate of 0. The downsidedeviation

is defined as the root mean-square of the deviations of the realized return's under-performance from the target return where all returns above the target return are treated as under-performance of 0.

It is mathematically defined as downsidedeviation = r 1 N N Õ i=1 „min„0; R t * T"" 2 (3.9) where R t is the i

th return, N = total number of returns, T = target return

taken as 0 for this analysis. The aim of an online model is to able to compute the effect of the current returns on the 17

performance measure. To do this we first define an exponential weighted moving average for the return and the downside deviation Taking the first moment of returns as A given by A = 1 T T Õ 1 „R t " (3.10) the exponential moving average for returns becomes: A t = A t*1 + „R t * A t*1 " (3.11) and for squared downside we have DD 2 t = DD 2 t*1 + „min„0; R t * T" 2 * DD 2 t*1 " (3.12) taking the first order expansion of moving average returns and the squared downside devia- tion, we get: SR SR t*1 + dSR d + O„ 2 " (3.13) Since

only the first order term depends on the return R t at time t

we define this as our Differential Risk Ratio (DRR) which is given by Moody and Saffell, 2001 DRR = dSR

d =

R

t * 0:5A t*1 DD t * 1 ; R t < 0 = DD 2 t*1 „R t * 0:5A t*1 * 0:5A t*1 R 2 t " DD 3 t * 1 ; R t > 0 (3.14) 18 3.3

Recurrent Reinforcement Learning Approach 3.3.1 Maximising Immediate Return The DRR above becomes our measure of the impact of return R at time T on the performance. We aim to optimize the performance function Differential Risk Ratio through optimal choice of . The is done by computing DRR in foward passes and adjusting through gradient ascent = dDSR T „ " d (3.15) where is the learning rate selected through experimentation and the rate of change of change the performance measure with is given by: dDSR T „ " d = T Õ

t=1 dDSR T dR t „ dR t dF t dF t d + dR t dF t*1 dF t*1

d " (3.16) It should also be noted that dF t d is path dependent and therefore approximated using back propagation through time as dF t d = dF t d + dF dF t*1 dF t*1 d (3.17) As we are interested in an online learning model that is considers the effect of the current returns on performance, the equation can be written in terms of the Differential Risk ratio as = dDSR T „ " d (3.18) 19

dDSR