The Application of Reinforcement Learning and Signal Processing in Dynamic Investment Management

Patrick Mthisi

Supervisor(s):

Dr Y Seetharam

A research report submitted in partial fulfillment of the requirements for the degree of Master of Science in the field of e-Science

in the

School of Computer Science and Applied Mathematics

University of the Witwatersrand, Johannesburg

4 October 2021

Declaration

I, Patrick Mthisi, declare that this research report is my own, unaided work. It is being submitted for the degree of Master of Science in the field of e-Science at the University of the Witwatersrand, Johannesburg. It has not been submitted for any degree or examination at any other university.

[illegible]

Patrick Mthisi

4 October 2021

"Our theory is thoroughly static. A dynamic theory would unquestionably be more complete and, therefore, preferable. But there is ample evidence from other branches of science that it is futile to try to build one as long as the static side is not thoroughly understood...

Our static theory specifies equilibra... A dynamic theory, when one is found- will probably describe the change in terms of simpler concepts. "

J. von Neumann

Abstract

An innovative approach is adopted to develop a rigorous active portfolio management system that explicitly makes investment decisions and processes financial market information. This approach addresses two unique challenges in portfolio management: the ability to effectuate market-sensitive asset allocations and alleviate the effects of financial market uncertainty. These challenges are resolved by utilising Recurrent Reinforcement Learning (RRL) as a sequential decision-making tool. Additionally, signal processing is employed to enhance performance stability. The study proposes the Augmented Recurrent Reinforcement Learning (ARRL), a hybrid portfolio management system that integrates the RRL and signal processing modules. Using shares from nine of South Africa's primary economic sectors, the ARRL system is used to perform dynamic asset allocation, thereby taking advantage of the changes in the market opportunity set. The performance of the system is compared to standard passive portfolio management strategies. ARRL-based strategies outperform standard passive strategies by a wide margin, according to the findings.

Keywords: Portfolio management, Dynamic asset allocation, Sequential decision making, Recurrent Reinforcement Learning, Signal processing.

Acknowledgements

I sincerely express my utmost gratitude to my supervisor Dr Y Seetharam for the invaluable guidance in this research work. His experience in the field of finance and willingness to assist made this work possible. A special thanks also go to Joel Lejaha and Mahendra Ramlucken for granting me access to the data used for this work. I would also like to extend my thanks to the DST-CSIR National e-Science Postgraduate Teaching and Training Platform (NEPTTP) for making it possible for me to pursue this degree.

Contents

List of Figures

List of Tables

List of Abbreviations

ADF	**A**ugmented **D**ickey **F**uller
ANN	**A**rtificial **N**eural **N**etwork
AUM	**A**ssets **U**nder **M**anagement
ARRL	**A**ugmented **R**ecurrent **R**einforcement **L**earning
BPTT	**B**ack **P**ropagation **T**hrough **T**ime
CARA	**C**onstant **A**bsolute **R**isk **A**version
CDAE	**C**onvolutional **D**enoising **A**uto **E**ncoder
CNN	**C**onvolutional **N**eural **N**etwork
CRRA	**C**onstant **R**elative **R**isk **A**version
CML	**C**apital **M**arket **L**ine
DAE	**D**enoising **A**utoencoders
EFL	**E**fficient **F**rontier **L**ine
ELU	**E**xpontial **L**inear **U**nit
EPS	**E**arnings **P**er **S**hare
ETF	**E**xchange **T**raded **F**und
EW	**E**qually **W**eighted
HJB	**H**amilton **J**acobi **B**ellman
HFT	**H**igh **F**requency **T**rading
IR	**I**nformation **R**atio
ICA	**I**ndependent **C**omponent **A**nalysis
JSE	**J**ohannesburg **S**tock **E**xchange
KLD	**K**ullback **L**eibler **D**ivergence
LSTM	**L**ong **S**hort **T**erm **M**emory
MDD	**M**aximum **D**rawdown
MDP	**M**arkov **D**ecision **P**rocess
ML	**M**achine **L**earning
MSSA	**M**ultichannel **S**ingular **S**pectrum **A**nalysis
MPT	**M**odern **P**ortfolio **T**heory
OIS	**O**vernight **I**ndex **S**wap
PDE	**P**artial **D**ifferential **E**quation
PPO	**P**roximal **P**olicy **O**ptimization
QL	**Q**-**L**earning
RDAE	**R**ecurrent **D**enoising **A**uto **E**ncoder
REIT	**R**eal **E**state **I**nvestment **T**rust
RL	**R**einforcement **L**earning
RRL	**R**ecurrent **R**einforcement **L**earning
SDAE	**S**tacked **D**enoising **A**utoencoders
SPG	**S**tochastic **P**olicy **G**radient
SR	**S**harpe **R**atio
SSA	**S**ingular **S**pectrum **A**nalysis

STR	**S**ortino **R**atio
SVD	**S**ingular **V**alue **D**ecomposition
TPV	**T**erminal **P**ortfolio **V**alue
VW	**V**alue **W**eighted

List of Symbols

$\mathcal{A}$	class of control policies in Control Theory
$a(t)$	action at time t such that $a(t) \in \mathcal{A}$
δ	explicit transaction costs
$D(\Pi, T)$	bequest or scrap function
$\mathbb{E}\vert$	conditional expectation operator
$\mathcal{F}(t)$	filtration up to time t
$G(t)$	discounted Cumulative Rewards
γ	discount factor in Reinforcement Learning
$\mathbf{h}(t)$	vector of portfolio holdings at time t
$\mathcal{H}$	hessian or second-order condition
M	number of assets in a financial portfolio
μ_Π	expected reward of the portfolio
∇	gradient or first-order condition
Ω	parameter space
$\mathcal{P}$	state transition probabilities
$\pi_i(t)$	decision function for asset i at time t
Π_θ	set of parameterised policies
$\mathbb{P}$	physical measure for stochastic processes
q_i	market order for asset i
μ_i	stock's daily drift
$\mathbb{1}$	unit vector
r	risk-free rate
$\mathcal{R}$	reward distribution
ρ_{ij}	correlation between the i^{th} and the j^{th} risky assets
$S_i(t)$	price of a financial security i at time t
$\mathcal{S}$	set of possible states
σ_i	volatility of risk asset i
σ_Π	portfolio volatility
Σ	variance-covariance matrix
$s(t)$	environment's state at time t
Θ	regression parameter space
θ	regression coefficient, $\theta \in \Theta$
U_T	terminal value of the objective function
v	average daily number of traded shares

Chapter 1

Introduction

Portfolio managers are responsible for developing investment strategies aimed at boosting the financial growth of their clients and the same time, achieve other investment targets within a finite time frame called an investment horizon. The central question that confronts portfolio managers across the investment industry is how to allocate Assets Under Management[1] (AUM) across multiple risky financial securities. Given a plethora of asset classes such as stocks, bonds, commodities, and so forth, a portfolio manager has to decide how best to distribute AUM across an array of these asset classes to optimise an investment objective subject to different constraints such as limited capital resources and minimum consumption required for survival [20]. The investment objective typically involves maximising wealth, economic utility, or risk-adjusted returns. For portfolio managers to achieve this objective, they are guided by a theory known as Markowitz's Modern Portfolio Theory (MPT), an approach that allocates assets according to their expected returns, standard deviations, and correlations to ensure that the optimal portfolio is located on the top part of the efficient frontier of feasible portfolio combinations [89].

The main criticism of MPT is that the asset allocations are static, meaning asset allocations that are implemented at the beginning of the period cannot be revisited until the period ends. Such an approach is quite problematic as it results in asset allocations that are not reflective of the latest relative price changes [106]. Financial markets are highly dynamic, so it is probably a lot better to allow some leeway to recalibrate initial investment decisions. Furthermore, the tools which portfolio managers have at their disposal are often manual, leaving room for both error and delays in execution. The instantaneous flow of asset information presents a portfolio manager with Big Data challenges, and even more so in High-Frequency trading environments. In such situations, the quicker a portfolio manager can execute decisions in the best interest of the portfolio in the face of troves of information or high volumes of data, the more likely it is to achieve investment objectives. In this study, an alternative to static or one-period MPT in the form of a multi-period dynamic asset allocation is proposed. The dynamic portfolio management process in this study utilises sequential decision making, an approach that offers a time-adaptive technique for dealing with rapidly changing financial markets to achieve an overall goal [23, 106]. Reinforcement Learning (RL)

[1]"Asset Under Management is the overall market value of the assets managed by the fund" [6]

provides a vast toolkit of methods that tackle sequential decision-making problems, and it also presents an opportunity to automate active decision making in asset allocation.

1.1 Background

People make decisions every day, and these decisions have both immediate and long-term effects [53, 118]. Decision-making must not be conducted in isolation. Essentially, to accomplish good overall success, the interdependence between current and future actions must be taken into consideration [23]. More precisely, the right course of action is not inherently the one that optimises immediate utility, but rather the one that optimises overall utility, and in doing so depends on actions that could be taken in the future [53]. Such a decision-making paradigm is referred to as sequential decision-making. Sequential decision tasks are characterised by a temporarily ordered set of choices or actions whose anticipated mutual outcomes can be conveyed in a single utility function [53]. RL can be applied to solve sequential decision problems using Bellman's principle of optimality discussed in Chapter 3. In essence, RL agents learn to orientate their activities in alignment with the environment. Additionally, RL agents make a sequence of decisions, basing decision $d+1$ on what was learned from decision d and its consequence [37]. The linchpin of RL entails determining a policy or plan of action that maximises the expected value of cumulative rewards [123, 137].

1.1.1 Research Area

The study focuses primarily on applying Machine Learning (ML) and Big Data paradigms to active investing. Active investing takes a hands-on approach to portfolio management and requires a portfolio manager to earn returns above the average market returns by exploiting short-term asset price fluctuations. A dynamic portfolio management system is designed using ML algorithms and Big Data solutions. The system not only automates portfolio rebalancing by automatically pivoting in and out of stocks but also handles high information flow.

1.1.2 Research Problem

The study detailed in this report looks at mitigating problems associated with the one-period MPT; primarily, transitioning from passive investing to active investing that constitutes dynamic asset allocation. A setting for dynamic asset allocation typically involves a portfolio manager exploiting the prevailing trend of financial assets to achieve returns that exceed a targeted benchmark. Different sectors of the economy behave differently, and dynamic asset allocation ensures that AUM is allocated to top-performing stocks, thereby guaranteeing that portfolios have the highest exposure to stock price inertia. As a result, if the trend sustains, the strategy will generate profits by increasing position to top-performing stocks and vice-versa for worst-performing stocks while remaining diversified across several economic sectors. Unlike classical one-period MP punctuated by infrequent reviews, dynamic asset allocation continuously re-balances portfolio allocations over multiple periods. The result is a sequence of investment decisions made at specific time points in response to financial markets' dynamics, and the resulting profit and losses are path-dependent.

Dynamic investing in this study is configured as a sequential decision task, and RL is used to find an optimal investment strategy by maximising a specified utility function. This study proposes a hybrid RL and signal processing dynamic asset-allocation system. We call it the Augmented Recurrent Reinforcement Learning (ARRL) system. This dual-purpose system automates asset allocation while also performing feature learning to improve the quality of inputs.

1.1.3 Prior Literature

Single period theories, starting with Markowitz's work and studies by Sharpe [131], and Ross [124], share one common thread: there is a single time horizon, and an investment strategy is based on return and risk estimates across that time horizon. One of the inherent hazards of investing in financial markets is exposure to changes in the market opportunity set, which can be gradual or precipitate as a sudden large shock [78]. The primary disadvantage of portfolio strategies under single-period theories is that they are only reviewed at the end of the investment horizon, thus fail to safeguard the portfolio against changes in the market opportunity set. The multi-period approach, particularly dynamic investment strategies, entails an investment strategy or policy that changes over time, allowing for risk management throughout the investment horizon [57]. The dynamic investment problem has remained a major topic in financial theory, dating back to Merton [91, 90] in its contemporary form. Dynamic asset allocation is motivated by the need to assist long-term investors or fund managers in their risk management efforts by providing means to implement periodic portfolio reviews. In addition, dynamic asset allocation exploits market opportunities whenever they arise and adjust the asset allocations accordingly. RL offers a way to implement unsupervised optimal multi-period asset allocations. Its advantage lies in the fact that it is not bounded by restrictive assumptions such as those under Merton's studies.

Neuneier was one of the leading proponents of applying RL to portfolio management in his 1996 paper titled "Optimal Asset Allocation using Dynamic Programming" [104]. In the said paper, the asset allocation problem is presented as a Markov Decision Process (MDP) [2]. More specifically, an RL-based algorithm called Q-learning [3] is used to solve portfolio optimisation. A Q-learning-based portfolio gave an outstanding performance in the testing period compared to a passive investing strategy. Furthermore, if there was no noticeable trend to follow, and also if there was too much uncertainty in the stock market, the portfolio strategy kept a neutral position in the market.

Subsequently, Moody et al [99, 100] pioneered the Recurrent Reinforcement Learning (RRL) approach, an adaptive policy search algorithm that finds approximates solutions to the field of portfolio management. The term "Recurrent" means that previous outputs form part of the inputs [99]. More notably, studies by Moody et al and Du et al [36] compared Q-learning to RRL and noted that Q-learning-based portfolios did not perform as well as RRL-based portfolios. That is because Q-learning suffers from Bellman's curse of dimensionality from large state and action spaces [130]. Additionally, Q-learning is more sensitive to value function selection, causing it to have unstable

[2]A mathematical framework for decision-making [137].

[3]Q-learning refers to an algorithm used to estimate reinforcement functions [96, 137].

performance, further exacerbated by non-ergodicity in financial markets. Furthermore, the RRL framework allows a simple representation of asset allocation problems than search algorithms based on value functions and also avoids Bellman's curse of dimensionality [36].

The groundbreaking idea from the study by Moody et al is the use of Sharpe Ratio and its differential as objective functions [99]. RRL is different from the Temporal Difference learning (TD-learning) approaches such as Q-learning that estimate a value function of the control problem. RRL provides more flexibility in the choice of objective functions. This novel idea propelled further interest in RRL and motivated other researchers to try-out a myriad of objective functions. A study by Almahdi et al [4] looks at alternative performance measures such as the Sterling ratio, Calmar ratio, and so forth, as well as implementing dynamic stop-loss as a risk management add-on. The resulting portfolios delivered superior returns that surpassed most hedge fund indices.

Studies that are discussed up to this point do not guide how RRL-based portfolios could be configured to handle portfolios consisting of two or more risky assets. Jiang et al [58] pioneered a topology called the Ensemble of Identical Independent Evaluator (EIIE). The EIIE topology evaluates one risky asset at a time for multi-asset portfolios and outputs a scalar or weight to denote its inclination to invest in that risky asset. Portfolio weights are computed by applying the softmax function to the scalars [58, 99]. EIIE topology is recurrent by design because portfolio weights from previous trading periods form part of the input; thus EIIE topology is a natural extension to RRL for multi-asset portfolios. This topology has some crucial benefits that include portfolio scalability, data-use efficiency, and asset collection plasticity. Liang et al [73] use the EIIE topology for adversarial training that involves adding random noise to market prices of five randomly chosen Chinese stocks to dampen the signal-to-noise ratio [4] and assess the performance of the portfolio constructed using that data. An important take away from this study is performance instability due to a low signal-to-noise ratio [73, 74].

Several methods are implemented to improve the performance stability of RRL-based portfolios. Maringer et al [88] augment the existing RRL with regime switching properties to account for non-linearities in financial data. The regime-switching model explicitly defines different regimes and assumes that the dynamics of economic variables are regime dependent. Gold [40] and Lu [81] introduce hierarchical feature learning using Deep Learning (DL) to enable RL agents to create their information and learn it directly from raw inputs. Deep Artificial Neural Networks (DNNs) are combined with RRL to perform hierarchical feature learning. Deng et al [30] incorporate DL into a typical RRL framework, thereby creating a deep RRL (DRRL) architecture. Also, fuzzy-learning is incorporated into the DRRL model to reduce instability in the initial time series. Combined with fuzzy-learning, the DRRL method is very stable under various market conditions and produces reliable profits during experimentation.

Other works include using value-based approaches such as Deep Q-Network (DQN) by El-Saawy et al [59] that make use of DNNs to estimate state-action value functions used to infer an optimal

[4]Signal-to-noise ratio is the ratio between the desired information versus background noise [67].

investment strategy. Alternatively, policy gradient-based methods such as Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimization (PPO) are used by [61, 73, 130].

1.1.4 Problematising

The most glaring problem with prior literature on portfolio management using RL is lack of clarity concerning the type and quality of information used as inputs. In most literature, inputs are limited only to either k most recent risky asset returns [99, 100] or to stock prices and volume traded [58]. As part of the study, our input space is expanded and includes other features from technical analysis.

Another contentious issue is that there is an overreliance on feed-forward DNNs for feature learning and value-function approximation or a combination of both. Using feed-forward DNNs potentially impair the efficacy of a trading or portfolio management system. DNNs often require long computational time and cause out-of-memory issues during training. Also, feed-forward DNNs pose more challenges during deployment as they are viewed as black-box algorithms due to their opaqueness. DL algorithms sift through millions of data points to find trends and connections that are often overlooked by human experts. However, the decisions they make based on these results often confound even the engineers who develop them. Thus mistakes could go undetected for long periods and compound losses, leading to serious financial repercussions. In this study, relatively simple and more computational efficient signal processing methods are employed, namely:

1. Multichannel Singular Spectrum Analysis (MSSA): is a model-free approach for the recognition and identification of time series structures [41].
2. Independent Component Analysis (ICA): this is an efficient statistical signal processing technique that is used to identify independent components from the observed data [80].
3. Denoising Autoencoders (DAE): these are unsupervised artificial neural networks (ANNs) that learn dense representations of perturbed input data called latent representations or codings [69, 105, 155].

1.2 Problem Statement

In Finance, MPT is the most prominent and widely adopted portfolio management technique despite its shortcomings [84, 89]. This study entails using RRL to dynamically inform asset allocation decisions to maximise a given investment objective. RRL agents do not only identify a locus of points that signals to buy, hold or sell a stock on a given stock price trajectory but also provide prescriptive portfolio allocations across multiple periods [157]. Thus RRL can essentially be used to actively and automatically allocate scarce capital resources to achieve an investment goal, and in doing so, automates portfolio management.

Data obtained from financial markets is limited only to stock prices as well as volume traded. In this study, additional features from technical analysis [102] are used to expand the input domain to improve the learning process of RRL. Also, signal-processing techniques mentioned in Section 1.1.4 are employed in our study due to their simplicity and transparency.

1.3 Research Aims and Objectives

1.3.1 Purpose

The primary goal of this study is to develop an integrated RRL and signal-processing portfolio management system. The system is applied to solve dynamic decision-making tasks pertinent to dynamic asset allocation. The benefits associated with this approach over theories such as Markowitz's MPT include:

1. There are cost-saving implications of delegating investment decision making to RRL agents while freeing up human attention to focus on higher-end tasks such as due diligence, company analysis, and investment research.
2. The integrated portfolio management system in this study can learn and distill useful information from the financial markets using novel signal processing methods.
3. Using RRL to automate asset allocation dynamically unlocks efficiency gains due to the lightning speed execution of investment decisions without much need for human intervention.

Given the above-mentioned benefits of our integrated approach, the likelihood of achieving investment objectives is higher. Furthermore, the portfolio's composition and a reactive portfolio management process can safeguard investors' wealth quicker against adverse market movements.

1.3.2 Research Question

This study covers two specific research questions, namely:

1. Which signal-processing approaches can be utilised to enhance the signal-to-noise ratio in financial data?
   Financial markets are quite versatile and non-stationary. A good signal processor ensures that only intrinsic information is utilised, thereby ensuring robustness against environment uncertainty. The expectation is an improvement in the portfolio's performance since only relevant information pertinent to the stock's dynamics will be used to inform asset allocation decisions.
2. How good is the performance and risk profile of the proposed ARRL system as an active investing tool relative to passive investing strategies?
   Active investing is costly because constant buying and selling of stocks attract high transaction costs that lower profit margins. Furthermore, active fund managers have more discretion to take positions in any investment they believe would yield high returns. However, this freedom adds an extra risk dimension called active risk whose consequences are dire should fund managers get their investment decisions wrong. Wrong investment decisions lead to suboptimal asset allocations that put a fund in an untenable position.

1.3.3 Research Aims

This study addresses the following research aims:

1. Probe the efficacy of the proposed ARRL system for a multi-asset portfolio.
2. Improve performance and robustness of the proposed ARRL dynamic asset-allocation system.
3. Achieve sustained performance and fulfill the active investing mandate of generating returns that exceed the benchmark.

1.3.4 Objectives

The following are carried out to achieve our research aims:

1. Assess efficacy of the ARRL system for a multi-asset portfolio of stocks from nine major economic sectors in South African.
2. Use signal-processing methods to improve the quality of inputs presented to the proposed portfolio management system [5].
3. Expand the input space beyond the generic and readily available financial market data variables.

1.4 Assumptions and Definitions

The experiments that are carried out as part of our study are performed under the following assumptions:

1. The fund managed by the portfolio manager is small to medium-sized: the amount of AUM falls within reasonable bounds that do not trigger significant market impact (MI). MI is defined as a change in stock price relative to a reference price caused by a transaction [65]. MI is calculated as follows:
$$MI = \pm\kappa\sigma\left(\frac{q}{v}\right)^{\lambda},$$
where σ is stock price's daily volatility, q denotes volume of the executed transaction, v is average daily number of traded shares of stock, while κ and λ are numerical constants that can be estimated from a sample of historical transactions [65]. For small to medium-sized funds, the magnitude of q is minute compared to v, thus the quotient $\frac{q}{v} \approx 0$. This assumption eliminates any tendency of potential herding behaviour which can artificially increase or decrease stock prices.
2. The financial market has high liquidity: trades are immediately executed at the latest bid and ask[6] prices, thereby eliminating slippage. Let S^{bid} and S^{ask} be the stock's bid price and ask price, respectively, then the bid-ask spread at time t is calculated as:
$$s^{(ba)}(t) = S^{bid}(t) - S^{ask}(t).$$

[5]This addresses the garbage in, garbage out (GIGO) principle, a colloquial recognition of poor quality inputs leading to unreliable output.

[6]A bid price is a maximum price a buyer of an asset is prepared to pay, and the minimum price that a seller is prepared to accept is called ask price [34].

Slippage occurs when $s^{(ba)}(t)$ varies between a time t when a market order is initiated and time $t+\eta$ when the order is completed. Frequent buying and selling of financial securities in a highly liquid market ensure that η is minimal, thus slippage is negligible in such cases.

3. Transaction costs, denoted by c, are a substantial component of realistic models of stock market's microstructure [65]. Sources of transaction costs include commissions (and similar explicit transaction charges - denoted by δ), MI and $s^{(ba)}$. Following a definition by [21, 65], transaction costs can be calculated as:

$$c = \delta + \frac{s^{(ba)}}{2} + MI = \delta + \frac{s^{(ba)}}{2} + \frac{1}{1+\lambda}\sigma\left(\frac{q}{v}\right)^{\lambda}.$$

The amount of money paid by a buyer is given by $Sq\,(1+c)$, and the amount received by a stock seller is calculated as $Sq\,(1-c)$ where S is stock price [65]. Notice that transaction costs are a portion of the market order that is paid by buyers of financial assets but not received by sellers. By the earlier assumptions, we can set slippage to be approximately equal to 0 and the quotient $\frac{q}{v} \approx 0$ thereby making transaction costs $\approx \delta$. It is important to note that δ denote explicit costs of trading and are not sources of financial risk. Any indirect or implicit costs associated with loss of capital during the investment period are not included in δ.

1.5 Limitations

1. Inputs are limited to the stock price, the volume traded, and technical analysis variables. Limiting inputs to said features assume that all the available information is reflected in these variables and that portfolio managers act only based on that information. In practice, portfolio managers would explore other data sources such as data from sentiment analysis and fundamental data.
2. Another setback comes from the assumption concerning fund size. Funds are assumed to range from small to medium-sized. The assumption is ideal for retail investors but not for corporate or institutional investors who can influence the market price. The assumption limits the scope of our study. Successful portfolio managers tend to acquire significant AUM that causes their market orders to have a highly significant market impact. To allow for market impact, further adjustments that are beyond the scope of this study would have to be employed.
3. The ARRL system is tested using nine tickers. The JSE floats 350 shares, of which the top 100 by market capitalisation are liquid enough to be considered investable. Limiting the portfolio to only nine stocks hinders the ability to move across shares within an available investable universe, thus exposing the portfolio to concentration risk.
4. Portfolio stocks are selected based on historical performance. As a result, only the winners are taken into account, whilst the losers are not considered, thus placing the winners at the forefront as a representative and comprehensive sample. This is known as survivorship bias, and it has the effect of skewing the fund's average results upwards.

Despite these limitations, the experimentation carried out in our study is still viable and provides a template for further work that could incorporate the limitations stated above.

1.6 Contribution

1. This study can be applied to other domains that call for innovative methods of learning decision-making policies. Such domains include autonomous driving [110, 117] as well as in the field of Medicine such as learning correlative and adverse interactions between medicines [146].
2. The signal processors used in our study are relatively easier to deploy in production and produce highly tractable outputs. They can be applied to the analysis of seasonality for inventory sales or consumer credit behaviour since they are capable of extracting the underlying factors in time series data.
3. The combination of signal-processing and RRL results in a hybrid dual-purpose module. This module capable of siphoning out relevant information and making critical decisions. The approach can be applied to many disciplines where automation is a requirement. The result is cost-savings and operational efficiency.

1.7 Outline

This work has the following outline:

- Chapter 2: Reviews portfolio management literature and provides a formulation of dynamic asset allocation as a stochastic optimal control problem.
- Chapter 3: Provides a discussion of RL approaches and RRL.
- Chapter 4: Outlines and discusses signal processors used in the study.
- Chapter 5: Describes the methodology, data preparation, and analytical metrics used.
- Chapter 6: Presents results and summary.
- Chapter 7: Concludes the study, and gives thoughts and ideas for future work.

Chapter 2

Portfolio Management

Fund managers buy and sell financial assets with the intent to generate returns in excess of a benchmark rate of return, such as the risk-free return or the bank's rate of return for cash deposits [152]. Portfolio management is concerned with optimising portfolio selection to get the best value for money. This section provides a concise portfolio management formulation that extends MPT to dynamic asset allocation using the groundbreaking work by Merton [90, 91, 92] and Ziemba [158]. Markowitz's MPT is also referred to as one-period, static, or do-nothing portfolio management theory because portfolio allocations are carried out at the beginning of the investment period and remain static until the end of that period [27, 89, 120]. Regardless of what happens to relative asset values, portfolio allocations are not rebalanced during the investment tenor [115]. In practice, fund managers recalibrate their initial investment decision according to the way the future unveils itself [93, 152]. That is precisely what dynamic asset allocation entails.

2.1 Conceptual Framework

2.1.1 Portfolio Reward and Risk

This section relates the parameters of constituent portfolio assets to the expected rewards and volatility of a portfolio. Consider a portfolio consisting of $M \geq 1$ risky assets. The values today of the i^{th} risky asset is denoted by S_i, and its random return is given by r_i over a finite investment horizon T. The random return r_i is assumed be elliptically distributed [52, 89, 109]. The Gaussian distribution is the widely accepted distribution for r_i:

$$r_i \sim N(\mu_i T, \sigma_i^2 T).$$

The correlation between the returns on the i^{th} and j^{th} risky assets is given by ρ_{ij}. The parameters μ_i, σ_i and ρ_{ij} correspond to the drift, volatility and correlation, respectively [52]. Let W_i be the number of shares or amount of risky asset i in the portfolio. The value of a portfolio comprised of $M \geq 1$ risky assets is given by:

$$\Pi = \sum_{i=1}^{M} W_i S_i.$$

The terminal portfolio value after horizon T is obtained by:

$$\Pi + \Delta\Pi = \sum_{i=1}^{M} W_i S_i (1 + r_i).$$

The holding period return is given by:

$$\frac{\Delta\Pi}{\Pi} = \sum_{i=1}^{M} \left(\frac{W_i S_i}{\sum_{i=1}^{M} W_i S_i} \right) r_i = \sum_{i=1}^{M} h_i r_i, \tag{2.1}$$

where h_i is the fraction of the fund's initial AUM allocated to risky asset i. The set $\{h_i, \ldots, h_M\}$ uniquely determines the probability distribution of the terminal portfolio value [93]. The expected reward on a portfolio is a function of individual asset's drift, and it is calculated as:

$$\mu_\Pi = \frac{1}{T}\mathbb{E}\left[\frac{\Delta\Pi}{\Pi}\right] = \sum_{i=1}^{M} h_i \mu_i = \mathbf{h}^\top \mu, \tag{2.2}$$

where $\mathbf{h} = (h_1, \ldots, h_M)$ and $\mu = (\mu_1, \ldots, \mu_M)$ are vectors of portfolio allocations and their drifts, respectively. The portfolio's standard deviation is also a function of asset allocations and their variance-covariance, and it is given by:

$$\sigma_\Pi = \frac{1}{\sqrt{T}}\sqrt{\text{var}\left(\frac{\delta\Pi}{\Pi}\right)} = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{M} h_i h_j \rho_{ij}\sigma_i\sigma_j} = \left\{ (h_1, \ldots, h_M)^\top \begin{bmatrix} \sigma_1^2 & \cdots & \sigma_{1M} \\ \vdots & \ddots & \vdots \\ \sigma_{M1} & \cdots & \sigma_M^2 \end{bmatrix} (h_1, \ldots, h_M) \right\}^{\frac{1}{2}} = \left\{\mathbf{h}^\top \Sigma \mathbf{h}\right\}^{\frac{1}{2}},$$

where Σ is the variance-covariance matrix given by:

$$\Sigma = (\sigma_{ij})_{i,j=1}^{M,M} \in \Re^{M\times M}.$$

The covariance between the ith and jth risky assets is given by $\sigma_{ij} = \rho_{ij}\sigma_i\sigma_j$, while the diagonal entries of Σ are individual volatilities of the risky assets [89, 120]. Given that the risky asset's random return r_i has a Gaussian distribution, the portfolio's return which is a linear combination of risky assets' returns (as shown by Equation 2.1) has a Gaussian distribution given by:

$$\frac{\Delta\Pi}{\Pi} \sim N\left(\mathbf{h}^\top \mu, \mathbf{h}^\top \Sigma \mathbf{h}\right).$$

2.2 Modern Portfolio Theory

In 1952, Nobel Laureate Markowitz [89] pioneered the Modern Portfolio Theory (MPT), a practical guide that defines efficient portfolios. An efficient portfolio is a portfolio with the highest reward for a given level of risk, or the lowest risk for a given reward [84, 120]. Efficient portfolios are situated along a risk-reward semi-parabola called the efficient frontier line (EFL) [89]. Successful portfolio optimisation using the reward versus risk paradigm places a portfolio along EFL [120, 158]. In practice, not only do portfolio managers buy and short risky assets in the financial market, but they can borrow or lend money in the money or cash market, as noted by Tobin's separation theorem [142]. Tobin's separation theorem, which is a special case of the n-fund mutual fund theorem, asserts that an optimal portfolio is made up of a risk-free asset and an efficient portfolio comprised entirely of risky assets [56, 125, 142]. Short-term securities sold in the money market (referred to as risk-free assets) serve as investment vehicles for low-cost capital for a short time. These risk-free assets are highly liquid and have maturities ranging from one day to one-year [71, 85]. Examples of short-term securities include short-term instruments such as the 91-day South African Reserve Bank treasury bill (T-bill), overnight repurchase agreement, commercial paper, swaps, or certificates of deposits. These securities are guaranteed to earn a risk-free rate of return denoted by r_f [83]. The risk-free rate of return could either be the overnight index swap rate (OIS), repo rate, or the money market Rate. The inclusion of a risk-free asset into a portfolio leads to a new efficient frontier called the capital market line (CML) [125, 142]. The point at which CML is tangent to EFL is called the market portfolio, or tangency portfolio [120].

Let h_0 be the weight of a risk-free asset which is defined as the weight of a portfolio that remains once allocation to risky assets is complete [125, 142]. The inclusion of a risk-free asset results in a budget equation given by:

$$h_0 + \mathbf{h}^\top \mathbb{1} = 1\ ;\ h_0 \equiv 1 - \mathbf{h}^\top \mathbb{1}. \tag{2.3}$$

Using Tobin's separation theorem, the expected return on a portfolio in Equation 2.2 can be re-written to include a risk-free asset as:

$$\mu_\Pi = h_0 r_f + \mathbf{h}^\top \mu = \left(1 - \mathbf{h}^\top \mathbb{1}\right) r_f + \mathbf{h}^\top \mu = r_f + \mathbf{h}^\top \left(\mu - r_f \mathbb{I}\right).$$

2.2.1 Maximise Risk-adjusted Return

Mean-variance optimisation criterion, first proposed by Markowitz, corresponds to finding asset allocation $\mathbf{h}$ that maximises expected return subject to a penalty related to the variance of portfolio returns [84, 89, 120], and it is given by:

$$\max_{\mathbf{h}} \quad J(\mathbf{h}) = r_f + \mathbf{h}^\top \left(\mu - r_f \mathbb{1}\right) - \frac{\lambda}{2}\left(\mathbf{h}^\top \Sigma \mathbf{h}\right), \tag{2.4}$$

where $\lambda > 0$ is a penalty term that portrays a level of risk aversion. The optimization is quadratic with respect to the decision variable $\mathbf{h}$, and there are no constraints [89, 152]. The candidate solution is derived from the gradient or the first-order condition given by:

$$\nabla J(\mathbf{h}) = \frac{\partial V}{\partial \mathbf{h}}(\mathbf{h}) = \left(\mu - r_f \mathbb{1}\right) - \lambda \Sigma \mathbf{h} = 0,$$

which can be solved analytically to give a candidate solution:

$$\mathbf{h}^* = \frac{1}{\lambda}\Sigma^{-1}\left(\mu - r_f \mathbb{1}\right). \tag{2.5}$$

Calculating the Hessian or second-order condition yields the candidate solution, which is a unique maximiser. The second-order condition of $J(\mathbf{h})$ is given by:

$$\mathcal{H}J(\mathbf{h}) = \frac{\partial^2 J}{\partial \mathbf{h}^2}(\mathbf{h}) = -\lambda \Sigma < 0,$$

thus, confirming that $J(\mathbf{h})$ attains its maximum value at $\mathbf{h}^*$.

2.2.2 Risk Minimisation

The portfolio selection problem can be formulated as a risk minimisation optimisation problem [94]. The constraint is typically a portfolio return set to equal a prespecified value denoted by m. Portfolio risk is easier to control than portfolio return, thus portfolio risk minimisation is more extensively applied than expected portfolio return maximisation [152]. The formulation of MPT as portfolio risk minimisation is given by:

$$\begin{aligned} \min_{\mathbf{h}} \quad & \frac{1}{2}\mathbf{h}^\top \Sigma \mathbf{h}, \\ \text{subject to:} \quad & r_f + \mathbf{h}^\top \left(\mu - r_f \mathbb{1}\right) = m. \end{aligned} \tag{2.6}$$

The optimisation problem in Equation 2.6 has equality constraints, therefore by applying the Lagrange method, the unconstrained minimisation is obtained by using the Lagrange function:

$$\min_{\mathbf{h}} \quad L(\mathbf{h}, \lambda) = \frac{1}{2}\mathbf{h}^\top \Sigma \mathbf{h} + \lambda \left(m - r_f - \mathbf{h}^\top (\mu - r_f \mathbb{1})\right), \tag{2.7}$$

where λ is a Lagrange multiplier. Equation 2.7 is unconstrained, thus we can proceed to calculate $\nabla L(\mathbf{h}, \lambda)$ to obtain the first-order condition:

$$\nabla L(\mathbf{h}, \lambda) = \frac{\partial L}{\partial \mathbf{h}}(\mathbf{h}, \lambda) = \Sigma \mathbf{h} - \lambda(\mu - r_f \mathbb{1}) = 0,$$

and $\mathcal{H}L(\mathbf{h}, \lambda)$ to obtain the second-order condition:

$$\mathcal{H}L(\mathbf{h}, \lambda) = \frac{\partial^2 L}{\partial h^2}(\mathbf{h}, \lambda) = \Sigma > 0.$$

The second-order condition is equal to the variance-covariance matrix, which is positive definite. The optimal weight is computed analytically as:

$$\mathbf{h}^* = \lambda \Sigma^{-1}\left(\mu - r_f \mathbb{1}\right)$$

By substituting $\mathbf{h}^*$ into the constraint in Equation 2.6, λ is given by:

$$\lambda = \frac{m - r_f}{(\mu - r_f \mathbb{1})^\top \Sigma^{-1} (\mu - r_f \mathbb{1})}$$

Finally, the optimal asset allocation $\mathbf{h}^*$ is calculated as:

$$\mathbf{h}^* = \frac{(m - r_f)\Sigma^{-1}(\mu - r_f \mathbb{1})}{(\mu - r_f \mathbb{1})^\top \Sigma^{-1} (\mu - r_f \mathbb{1})} \tag{2.8}$$

2.2.3 Markowitz in Practice: Pros and Cons

Optimal portfolio allocations in Equation 2.5 and Equation 2.8 are equivalent in that they sweep out the same efficient frontier of portfolios that are mean-variance efficient. In a single period, this works well, and the joint normality assumption discussed in Section 2.1 implies that the risk/return trade-offs give identical frontier portfolios. The Markowitz MPT, because it is a single period approach, is based on the assumption that fund managers cannot change their minds later, therefore asset allocations, once set, are not re-visited until at the end investment horizon. Essentially, regardless of the relative asset values, portfolio rebalancing is not performed [115]. The single-period method is ideal for fund managers that focus on the short term. The strategy will only evolve when another investment review takes. The result is a portfolio management strategy that does not protect from adverse price fluctuations or adjust to changes in the market opportunity set.

2.3 Dynamic Asset Allocation

Capital markets are dynamic; they are affected by constant shifts of supply and demand. For instance, the resignation of well-famed personnel, technological advancement, or social perceptions may generate shocks or disequilibria in supply and demand, altering the relative values of financial instruments. Stock portfolios in these markets often require constant monitoring and rebalancing to capture fluctuations in value caused by transient forces that misalign supply and demand from their equilibrium levels [120]. Dynamic asset allocation is an active investing strategy that entails multi-period deployment of AUM among a mix of risky assets in an optimal way in order to maximise the fund's value, or utility [29, 157]. The portfolio asset mix is constantly adjusted to align to the changes of the market opportunity set, and it typically involves the selling of assets that are in decline and buying assets that are increasing in value in order to capture positive returns wherever and whenever they exist in the financial market [115]. Once implemented, dynamic asset allocation uses scientific methods to inform portfolio rebalancing based on empirical observations of the objective data. Unlike passive or one-period investing, dynamic asset allocation provides an opportunity to recalibrate initial investment decisions across multi-periods [106, 152]. The result is a sequence of actions that determine a portfolio's asset mix, and ultimately, its performance. The portfolio manager is mandated to devise a portfolio rebalancing plan to suit prevailing economic conditions and investors' goals.

One of the most critical considerations a fund manager must make is which strategy to employ. In Section 2.2.3, we remarked that if the short-term dominates, MPT works quite well, and the fund manager typically considers assets that perform well in the short-term such as T-bills and bonds. Staunton et al [31] demonstrate that in the long run, equity investments have grown at a faster rate than T-bill, bonds, and inflation. Thus, the fund manager can no longer rely on a short-term approach for equity investments because the impact of changes in the market opportunity set and market risk becomes more significant the longer the investment horizon becomes [57]. A prudent fund manager should rebalance asset allocations optimally and periodically to align investor's risk tolerance, time horizon, and investment objectives with market dynamics. Rather than having a single efficient frontier, as the single period approach suggests, a multi-period framework establishes efficient frontiers for various rebalancing frequencies. Furthermore, in a multi-period setting, the joint-normality assumption that results in a single efficient frontier no longer holds. In a multi-period context, joint-lognormality is the obvious choice. In a continuous-time dimension, this results in a multivariate Brownian motion with constant drift and covariance (see Section 2.3.1) [57].

In perspective, dynamic asset allocation encapsulates theories from discrete or continuous-time financial models and optimal multi-period portfolio rebalancing. In this section, we layout theoretical groundwork for dynamic asset allocation using novel ideas by Davis et al [29], Merton [91, 93] and Ziemba [158]. A review of the commonly used dynamic asset allocation approaches is also included.

2.3.1 The Wealth Process

Adopting the notation already introduced in Section 2.1, the portfolio is assumed to comprise of $M \geq 1$ risky assets and a risk-free asset. The risky asset $i = 1, \dots, M$ is presumed to follow a Geometric Brownian Motion (GBM) satisfying a stochastic differential equation (SDE) of the form:

$$\begin{aligned} \frac{dS_i(t)}{S_i(t)} &= \mu_i dt + \sum_{j=1}^{M} \sigma_{ij} dW_j(t) \\ &= \mu_i dt + \sum_{j=1}^{M} \sigma_{ij} \Phi \sqrt{dt} \ ; \ \Phi \sim N(0,1) \ ; \ S_i(0) = s_i(0), \end{aligned} \tag{2.9}$$

where $W_i(t) \sim N(0,t)$ is a Brownian motion process on a probability space given by $\{\Omega, \mathcal{F}(t), \mathbb{P}\}$ [52, 93]. The underlying filtration given by $\mathcal{F}(t)$ coincides with the sigma-field given by $\sigma\left(W(s) : 0 \leq s \leq t\right)$ [103]. Similarly, a money market instrument given by B has a price evolution process satisfying:

$$\frac{dB(t)}{B(t)} = r_f dt \ ; \ B(0) = 1.$$

Let $\sum_{i=1}^{M} h_i(t)$ be the total allocation to risky assets at time t. The allocation to the money market instrument B is given by $h_0(t) = 1 - \sum_{i=1}^{M} h_i(t)$. Thus, at any given time $t \leq T$, the budget equation is given by:

$$h_0(t) + \sum_{i=1}^{M} h_i(t) = \sum_{i=0}^{M} h_i(t) \underset{t}{\equiv} 1. \tag{2.10}$$

Unlike the budget equation given by Equation 2.3 in which asset allocations are static, dynamic asset allocation is a multi-period portfolio management process where portfolio rebalancing is performed throughout the investment horizon.

Using the guideline from Merton [90, 92], we now derive the wealth process of a portfolio at time t. Let $\Pi(t)$ be the fund value or wealth at time t. Considering a time interval of length η, define $C(t)$ to be consumption at time t where $t = t_0 + \eta$. The total wealth at time t is given by:

$$\Pi(t) = \left[h_0(t_0) \frac{B(t)}{B(t_0)} + \sum_{i=1}^{M} h_i(t_0) \frac{S_i(t)}{S_i(t_0)} \right] \left[\Pi(t_0) - C(t_0)\eta \right]. \tag{2.11}$$

By subtracting $\Pi(t_0)$ on both sides of Equation 2.11 and using the budget equation given by Equation 2.10, the discrete time intertemporal wealth process is given by:

$$\begin{aligned}
\Delta\Pi(t_0,t) &= \Delta\Pi(t_0,t) = \Pi(t) - \Pi(t_0) \\
&= \left[h_0(t_0)\frac{\Delta B(t_0,t)}{B(t_0)} + \sum_{i=1}^{M} h_i(t_0)\frac{\Delta S_i(t_0,t)}{S_i(t_0)} \right] [\Pi(t_0) - C(t_0)\eta] - C(t_0)\eta \\
&= \left[h_0(t_0)\left(e^{\eta r_f} - 1\right) + \sum_{i=1}^{M} h_i(t_0)\left(e^{r_i(t_0)\eta} - 1\right) \right] [\Pi(t_0) - C(t_0)\eta] - C(t_0)\eta \\
&= \left[h_0(t_0)e^{\eta r_f} + \sum_{i=1}^{M} h_i(t_0)e^{r_i(t_0)\eta} - \left(h_0(t_0) + \sum_{i=1}^{M} h_i(t_0) \right) \right] [\Pi(t_0) - C(t_0)\eta] - C(t_0)\eta \\
&= \left[h_0(t_0)e^{\eta r_f} + \sum_{i=1}^{M} h_i(t_0)e^{r_i(t_0)\eta} - 1 \right] [\Pi(t_0) - C(t_0)\eta] - C(t_0)\eta \\
&= \left[\left(1 - \sum_{i=1}^{M} h_i(t_0)\right) e^{\eta r_f} + \sum_{i=1}^{M} h_i(t_0)e^{r_i(t_0)\eta} - 1 \right] [\Pi(t_0) - C(t_0)\eta] - C(t_0)\eta \\
&= \left[e^{\eta r_f} + \sum_{i=1}^{M} h_i(t_0)\left(e^{r_i(t_0)\eta} - e^{\eta r_f}\right) - 1 \right] [\Pi(t_0) - C(t_0)\eta] - C(t_0)\eta,
\end{aligned}$$

where $r_i(t_0)\eta = \log\frac{S_i(t)}{S_i(t_0)} = \left(\mu_i - \frac{1}{2}\sigma_i^2\right)\eta + \sum_{j=1}^{M}\sigma_{ij}\Phi\sqrt{\eta}$ [1] is the rate of return for risky asset $i = 1,\ldots,M$ for the period η [91, 92, 93]. By applying a limit to $\Delta\Pi(t_0,t)$ as $\eta \to 0$ and the Maclaurin series of an exponential function, Merton et al [93] demonstrate that the continuous-time portfolio wealth process, $d\Pi(t)$, on the probability space $\{\Omega, \mathcal{F}(t), \mathbb{P}\}$ is a limiting case of the discrete-time intertemporal wealth process:

$$\begin{aligned}
d\Pi(t) &= \left[\left(r_f + \sum_{i=1}^{M} h_i(t)(\mu_i - r_f) \right) \Pi(t) - C(t) \right] dt + \Pi(t)\sum_{i=1}^{M}\sum_{j=1}^{M}\sigma_{ij}h_i(t)\Phi\sqrt{dt} \\
&= \left[\left(r_f + \mathbf{h}(t)^\top(\mu - r_f\mathbb{1}) \right) \Pi(t) - C(t) \right] dt + \Pi(t)\mathbf{h}(t)^\top\Sigma\mathbf{h}(t)dW(t) \\
&= \Pi(t)\mu\left(C(t), h(t,\Pi(t))\right) + \Pi(t)\sigma\left(h(t,\Pi(t))\right)dW(t).
\end{aligned}$$

2.3.2 Maximising Expected Utility

The MPT framework relies on variance as a measure of an investor's risk preference. In the short period, it is adequate since outcomes are assumed to be normally distributed. However, dynamic asset allocation often results in option-like payoffs [57]. As a result, using the variance in a multi-period setting may distort results. Alternatively, overall portfolio value or wealth from a particular strategy could be taken into account. However, standard economic theory states that wealth is generally not a good indicator of investors' satisfaction since it assumes that investors are insensitive to risk [29, 99, 120]. Instead of considering wealth, which ignores risk or variance that does not account for asymmetrical outcomes, a utility function that incorporates a degree of satisfaction investors derive from their invested wealth often is utilised.

[1] This rate of return is called logarithmic rate of return, and it is distributed as $r_i(t_0)\eta \sim N\left(\left(\mu_i - \frac{1}{2}\sigma_i^2\right)\eta, \sigma_i^2\eta\right)$ [52]. The risky asset value is Log-Normally distributed.

Utility functions offer a way to rank possible sets of outcomes, such as values that arise from multi-period portfolio rebalancing [57, 78]. A common approach is to consider expected utility of wealth, denoted by $\mathbb{E}[U(\Pi_t)]$. Therefore, in a dynamic setting, portfolio maximisation is given by:

$$\max \quad \mathbb{E}[U(\Pi_t)]. \tag{2.12}$$

The maximisation in Equation 2.12 is very basic. Alternatively, the utility could be combined with risk measures such as volatility or tail losses subject to a cap on the variance of outcomes or tail losses, resulting in a function that focuses on the desired outcomes and tolerance for risk. In Section 2.3.3, we will give a brief discussion of common types of utility functions.

2.3.3 Stochastic Control and Bellman Equations

In 1967, Merton [91] proposed a formulation for multi-period asset allocation in continuous time as a stochastic optimal control problem where the principal goal is to maximise the utility of wealth. Essentially, the problem of choosing an optimal asset allocation and consumption rules boils down to choosing $\mathbf{h}(t)$ and $C(t)$ optimally [152]. One way of doing this is to maximise expected utility given by:

$$\begin{aligned} \max_{C,\mathbf{h}} \quad & \mathbb{E}|_0 \left[\int_0^T e^{-\rho t} U(C(\tau)) d\tau + D\left(\Pi, T\right) \right] \\ \text{subject to:} \quad & C(t) \geq 0,\ \Pi(t) > 0,\ \Pi(0) = \Pi_0 > 0, \end{aligned} \tag{2.13}$$

where $\mathbb{E}|_0$ is a conditional expectation operator given that $\Pi(0) = \Pi_0$ is known at inception. The expectation in Equation 2.13 is in two parts. The first part denotes the utility of consumption function $U(C(t))$ over the investment horizon. The utility function is assumed to be strictly concave[2] [93]. Widely used utility functions include the Constant Relative Risk Aversion (CRRA) [89, 130] or the Constant Absolute Risk Aversion (CARA) [14, 77]. Both of these utility functions are based on the assumption of time homogeneity [78]. This means that maximising expected utility over adjacent periods separately should have the same result as maximizing over the entire period, at least if the two subperiods are independent and identically distributed [57]. The utility function is discounted at a rate of ρ which measures investors' patience and choice for current consumption versus future consumption [152]. The second part given by $D(\Pi, T)$ is called the bequest valuation function or scrap function, and it measures satisfaction derived from having wealth leftover at time T [93]. The maximisation formulation given by Equation 2.13 improves portfolio management in the sense that portfolio managers take into consideration investors' consumption needs in the portfolio selection process. In growth models where the objective is to maximise long-term growth, there is no consumption, therefore $C(t) = 0\ \forall t = 0, \ldots, T$. All investment proceeds are reinvested back into the portfolio [152].

Bellman's dynamic programming principle [17] is still at the heart of control theory [13]; it is used to derive Bellman equations in discrete time [32] and Hamilton-Jacobi-Bellman partial differential equations (HJB PDE) in continuous time [112, 113]. In order to derive optimality equations, Equation

[2]Strictly concave functions ensure that $U^{'}(C) > 0; U^{''}(C) < 0$ [152].

2.13 is restated in a dynamic programming format so that the Bellman principle of optimality can be applied [17, 93, 158]. Let the discrete-time value function be:

$$J(\Pi, t) = \max_{C,\mathbf{h}} \mathbb{E}|_t \left[\int_t^T e^{-\rho t} U(C(\tau)) d\tau + D\left(\Pi, T\right) \right], \tag{2.14}$$

over control $\mathbf{h} \in \mathcal{A}$ and $C \in \mathcal{C}$. By considering time $t^{'} = t + \eta$, where η is the length of time interval between periods and an unknown portfolio value $\Pi^{'}$ at $t^{'}$, the discrete-time value function $J(\Pi, t)$ can be re-written as recursive equation given by:

$$\begin{aligned} J(\Pi, t) &= \max_{C,\mathbf{h}} \mathbb{E}|_t \left[\int_t^{t^{'}} e^{-\rho t} U(C(\tau)) d\tau + \int_{t^{'}}^T e^{-\rho t} U(C(\tau)) d\tau + D\left(\Pi, T\right) \right] \\ &= \max_{C,\mathbf{h}} \mathbb{E}|_t \left[\int_t^{t^{'}} e^{-\rho t} U(C(\tau)) d\tau + J(\Pi^{'}, t^{'}) \right]. \end{aligned} \tag{2.15}$$

The value $J(\Pi^{'}, t^{'})$ depends on the state at time $t^{'}$. The recursive approach then implies the decision problem at time t amounts to finding a policy that maximises the expected value of $J(\Pi^{'}, t^{'})$ [57]. Note that decisions are taken one at time, and the aim is to maximise some objective prospectively. Taking the limit as $\eta \to 0$ and applying some stochastic calculus, Merton et al [93] and Davis et al [29] show that the continuous-time value function is given by:

$$\begin{aligned} J(\Pi, t) &= \max_{C,\mathbf{h}} \mathbb{E}|_t \left[\int_t^{t+dt} e^{-\rho t} U(C(\tau)) d\tau + J(\Pi + d\Pi, t + dt) \right] \\ &= \max_{C,\mathbf{h}} \mathbb{E}|_t \left[e^{-\rho t} U(C(t)) dt + J(\Pi, t) + \frac{\partial J}{\partial t} dt + \frac{\partial J}{\partial \Pi} d\Pi + \frac{1}{2} \sum_{i=1}^M \sum_{j=1}^M \sigma_{ij} h_i h_j \Pi^2 \frac{\partial^2 J}{\partial \Pi^2} dt \right]. \end{aligned}$$

Finally, after subtracting $J(\Pi, t)$ from both sides, taking expectation and dividing by dt, we obtain continuous-time HJB PDE given by:

$$\begin{aligned} 0 &= \frac{\partial J}{\partial t} + \max_{C,\mathbf{h}} \left[e^{-\rho t} U(C) + \left\{ \left(r_f + \sum_{i=1}^M (\mu_i - r_f) h_i \right) \Pi - C \right\} \frac{\partial J}{\partial \Pi} + \frac{1}{2} \sum_{i=1}^M \sum_{j=1}^M \sigma_{ij} h_i h_j \Pi^2 \frac{\partial^2 J}{\partial \Pi^2} \right] \\ &= \frac{\partial J}{\partial t} + \max_{C,\mathbf{h}} \left[e^{-\rho t} U(C) + \left\{ \left(r_f + \mathbf{h}^\top (\mu - r_f \mathbb{1}) \right) \Pi - C \right\} \frac{\partial J}{\partial \Pi} + \frac{1}{2} \mathbf{h}^\top \Sigma \mathbf{h} \Pi^2 \frac{\partial^2 J}{\partial \Pi^2} \right]. \end{aligned}$$

Optimising over consumption at time t yields:

$$e^{-\rho t} U^{'}(C) = \frac{\partial J}{\partial \Pi}.$$

Optimising over $\mathbf{h}$ at time t yields the optimal asset allocation given by [3]:

$$\mathbf{h}^* = -\frac{\frac{\partial J}{\partial \Pi}}{\Pi \frac{\partial^2 J}{\partial \Pi^2}} \Sigma^{-1} \left(\mu - r_f \mathbb{1} \right).$$

In summary, portfolio asset allocation $\mathbf{h}$ is defined by a M-dimensional stochastic process, where the i^{th} component of $\mathbf{h}$, given by $h_i(t)$, is equal to a fraction of AUM invested in the i^{th} risky asset at time $t \leq T, \forall i = 1, \dots, M$ [29].

[3]We have conveniently dropped time indexing in all the equations to make the equation neater, that is, $h_i \equiv h_i(t)$.

2.3.4 Cox-Huang Approach

One of the daunting tasks in Merton's solution is determining the value of $J(\Pi, t)$ analytically. The Cox-Huang technique, commonly known as the martingale strategy, was developed by Cox et al [28] and Karatzas et al [62]. Instead of concentrating on each decision, the focus is on the end result, which is a collection of consumption amounts $\{c(t)\}_{t=0}^{T-1}$ and the final wealth $\Pi(T)$. This method draws heavily from option pricing theory, in which the value of these "payoffs" at $t = 0$ equals the current wealth. Subject to this constraint, payoffs that maximise the expected payoff in Equation 2.13 can be sought. As a result, the task boils down to finding a strategy $\{h_i(t)\}_{t=0}^{T-1}$ that, when combined with consumption policies, generates these payoffs. The discounted value of the payoff is a martingale from a probabilistic standpoint. The martingale representation theorem ensures the existence of a strategy that supports this value. From an economic perspective, this strategy delta hedges the discounted value at each point in time. In this respect, Bellman's value function, $J(\Pi, t)$, can be replaced by the value of the future portion of the optimal payoff [57].

2.3.5 Numerical Approach

The recursive form of Bellman's equation (see Equation 2.15) makes it well-suited for numerical approximation and optimisation. At each time step, an estimate of the value function can be made, allowing a solution to evolve by going backward in time while taking portfolio constraints into account. An approximation methodology is required for generating function estimations. Tree and lattices are the most prevalent variations in finance.

2.3.5.1 Tree Approaches

The tree encodes a simplified set of future paths of asset prices, starting with the current prices. The tree can be as basic as a recombining binomial tree where asset prices, after a one-time step, either go up or down [28, 57]. After n time steps, there are 2^n possible paths for asset prices. The tree can be non-recombining or even allow for three possible states in trinomial trees. Standard arbitrage arguments can then be used to price any stochastic payoff at time $n\Delta t$, that is, price is given by working backwards, one step at a time. Provided that the price at the final nodes is known, the price one step back is given by constructing an appropriate hedge portfolio for the payoff at each note and then valuing the hedge portfolio. Iterating this procedure produces price at the initial node, that is, at time $t = 0$. According to the central limit theorem, the binomial tree approaches the continuous-time model as the step size approaches zero.

2.3.5.2 Lattice Approach

An SDE for asset classes, such as the one illustrated in Equation 2.9, can be written down as a partial differential equation (PDE) that determines the price evolution over time. The generic form of the PDE is given by:

$$\frac{\partial S}{\partial t}(x, t) = a + b\frac{\partial S}{\partial x}(x, t) + c\frac{\partial^2 S}{\partial x^2}(x, t),$$

where a, b, and c are functions. The purpose of the lattice approach is to find an approximation to the solution of this PDE, rather than building a tree and arguing that the tree is a reasonable approximation to the continuous limit [57]. The full specifics of this approach, and other approaches such as the combined lattice/tree approach, are outside the scope of our study.

The highlighted strategies have one thing in common: dynamic optimisation may be broken down into individual decision components. Note that the goal is to optimise a function of a strategy that consists of asset allocations at each possible future path. The curse of dimensionality is a major drawback of these techniques, casting doubt on their scalability [57]. In other words, as the number of assets in the portfolio grows, so does the number of states and parameters that must be estimated, resulting in unacceptably long computer run times. To overcome this problem, several approaches have been offered, including employing the Fokker-Planck equation. However, restrictive assumptions, such as constant drifts or a constant variance-covariance matrix, are set, which may not be practical in realistic circumstances.

2.4 Summary

In this chapter, we demystified the theory behind one-period optimal asset allocation guided by Markowitz's MPT. In addition, we also looked at dynamic asset allocation and why it is necessary as the investment horizon increase, primarily focusing on why the normality assumption pertinent to one-period investing no longer applies in multiperiod portfolio rebalancing settings. We also explained why symmetrical risk measurements like variance are inadequate in dynamic asset allocation, thus paving the way to introduce utility functions. A review of the common types of dynamic asset allocation strategies was provided in the chapter. Dynamic asset allocation was presented as a stochastic optimal control problem that can be solved to yield an optimal control policy based on the innovative work of Davis et al [29], Merton et al [91, 90, 92, 93], and Ziemba [158]. In portfolio management, the stochastic optimal control problem aims to identify a portfolio allocation strategy, $\{h_i(t)\}_{t=0}^{T-1}$, that maximises a specified value function. The estimation of the value function is the obvious flaw in this approach. The Cox-Huang approach proposed by Cox et al [28] and Karatzas et al [62] was presented, and it offers a way to approximate value functions. Finally, numerical techniques that use trees and lattices were reviewed as well as the Fokker-Planck equation to handle the curse of dimensionality. Dynamic asset allocation is, without a doubt, a significant improvement over Markowitz's one-period MPT for the reasons mentioned in the text. However, the common approaches presented have their own set of issues: the curse of dimensionality plagues these approaches. In addition, assumptions of constant drifts and variance-covariance matrices might not generalise quite well in real-world situations. Furthermore, while covariance-variance matrices are relatively stable over time, asset drifts are difficult to estimate. In reality, financial markets are complex adaptive systems where market dynamics can be challenging to model, making RL a viable option to explore.

Chapter 3

Reinforcement Learning

RL is a biology-inspired paradigm that lends itself as a learning problem as well as an ML subfield [61]. As a learning task, it simply means learning to assign states of a system to actions to maximise numerical rewards [81, 137]. The learner has to decide which actions offer the highest pay-off. The learner learns from experience since information about an optimal course of action to take is not availed to the learner [137]. The primary elements of RL are the agent and the environment it interacts with [137]. In this chapter, a brief theoretical framework of RL is discussed as well as a review of subclasses RL algorithms fall into. Finally, Recurrent Reinforce Learning (RRL) pioneered by Moody et al [99, 100] is discussed and its application to solve multi-period portfolio allocation.

3.1 Reinforcement Learning Process

An RL agent engages with the environment on a constant basis. More specifically, the agent is presented with a situation that it has to act on and then the environment responds to that action by sending out a reward signal, and at the same time presents the agent with new situations in subsequent phases. A reward signal is a special numerical value that the agent tries to optimize by choosing the right action(s). Figure 3.1 illustrates how the agents interacts with the environment:

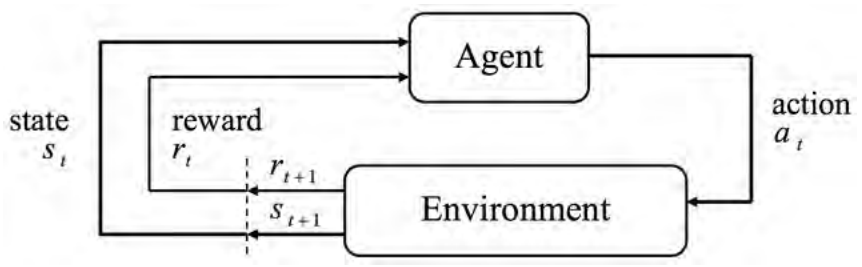


FIGURE 3.1: A schematic presentation of RL Process [137]

Let $\mathcal{A}$ and $\mathcal{S}$ denote a set of possible actions and a set of possible states, respectively. Figure 3.1 shows that at each discrete time step t, the agent receives some presentation of a state of the environment $s(t) \in \mathcal{S}$. The agent decides on an action $a(t) \in \mathcal{A}$ to take for which it receives an immediate reward signal $R(t) \in \mathcal{R} \subset \Re$. The environment then presents a new state $s(t+1)$ at time step $t+1$. The agent receives a reward of $R(t+1)$ at that subsequent time step as consequence of its actions. The process continues until the environment gives a terminal state which ends the episode.

An RL problem is typically formulated as a Markov Decision Process (MDP), a basic mathematical paradigm for the task of learning to accomplish a goal by engaging with the environment [137]. To act optimally, RL agents learn the parameters of the MDP using observations from the environment. RL agents face a trade-off between exploration and exploitation. Exploration entails gathering more experimental data about the consequences of the actions, while exploitation involves acting consistently with past observation to maximise the rewards [8].

3.2 Markov Decision Processes

3.2.1 Definition

An MDP is a discrete-time stochastic control process that provides a basis for decision making in circumstances where outcomes are partly unpredictable and partly under the influence of a decision-maker [137].

MDPs are ideally characterised as a five-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$ where:

- $\mathcal{S}$: denotes a collection of possible states $s(t)$ endowed with σ-field $\mathcal{S}$.
- $\mathcal{A}$: denotes a set of possible actions $a(t)$ endowed with σ-field $\mathcal{A}$.
- $\mathcal{R}$: is a reward distribution for state-action pairs $(s(t), a(t)) \in \mathcal{D} \subset \mathcal{S} \times \mathcal{A}$ where $\mathcal{D}$ is measurable with respect to σ-field $(\mathcal{S} \cup \mathcal{A})$.
- $\mathcal{P}$: denotes state transition probabilities denoted by:
$$p(s(t+1) = s^{'} | s(t) = s, a(t) = a) = p(s^{'} | s, a).$$
- $\gamma \in (0,1]$: denotes a discount factor that measures relative value of deferred versus imminent rewards. This is parallel to how ρ is defined in Section 2.3.2.

An MDP is a way of describing the environment's dynamics. Collectively, $\mathcal{R}$ and $\mathcal{P}$ define a model of the environment.

3.2.2 Markov Property

State transition probabilities $\mathcal{P}$ satisfy the Markov property [137, 138]. A stochastic process $\{s(t)\}_{t\geq 1}$ has a Markovian property if $\forall (k+1)$-tuples $(t_1, t_2, \ldots, t_{k+1})$ with $t_1 < t_2 < \cdots < t_{k+1}$:

$$p(s(t_{k+1}) \in \mathcal{S} | \mathcal{F}(t_k)) = p(s(t_{k+1}) \in \mathcal{S} | s(t_k) \in \mathcal{S}),$$

where $\mathcal{F}(t_k) = \{s(t_i) \in \mathcal{S}\}_{i=1}^{k}$ is filtration up to t_k of a stochastic process $\{s(t)\}_{t\geq 1}$ [15, 139]. Markov property is often interchangeably referred to as a memoryless property in Financial models in the sense only a current state is relevant in determining statistics about the future [52]. To that effect, MDPs, which by definition, satisfy a Markov property, are viewed as extensions of Markov chains. The main difference from generic Markov chains is that MDPs include choices and their corresponding rewards [18, 137].

3.2.3 Markov Decision Process Algorithm

An MDP provides a setting for a decision-making problem and a solution to the problem is called a policy. The primary task of a decision-maker is to learn a policy denoted by a mapping given by:

$$\pi : \mathcal{S} \rightarrow \mathcal{A}.$$

In essence, given a current state $s(t) \in \mathcal{S}$ at time t, a policy maps $s(t)$ to an action $a(t) \in \mathcal{A}$ [96]. A policy fully defines the behaviour of an agent. Policy functions can be either be deterministic where given a current state of the environment, a policy function outputs an action as given by $\pi(s(t)) = a(t)$ or stochastic, where for a probability distribution over actions, denoted by p_π, the policy function becomes $\pi(a(t) = a|s(t) = s) = p_\pi(a|s)$ which allows for exploration in the state space [1, 96, 137]. RL is used to search for optimal policies that solve MDP-styled problems. An RL agent has some latitude to sift through different choices to come out with an optimal policy that results in maximum cumulative discounted rewards given by:

$$G(t) = R(t) + \gamma R(t+1) + \gamma^2 R(t+2) + \cdots = \sum_{k\geq 0} \gamma^k R(t+k). \tag{3.1}$$

In essence, an optimal policy is a policy that, if pursued, would cause an agent to receive the highest amount of long-term rewards. Algorithm 1 shows how an MDP operates:

Algorithm 1 Markov Decision Process

```
1: $s(0) \sim p(s(0))$,                                      ▷ Environment samples an initial state.
2: while $t < T$ do
3:     $s(t) \leftarrow s(t+1) \sim p(\cdot|s(t), a(t))$,          ▷ Environment samples next state $s(t+1)$.
4:     $a(t) \leftarrow a(t+1) \in \mathcal{A}$,                      ▷ Agent selects action $a(t+1)$ given $s(t+1)$.
5:     $R(t) \leftarrow R(t+1) \sim \mathcal{R}(\cdot|s(t), a(t))$,    ▷ Reward signal $R(t+1), \forall(s(t), a(t)) \in \mathcal{D}$.
6: return $\tau = \{s(t), a(t), R(t)\}_{t\geq 0}^{T}$.              ▷ Trajectory $\tau$ until end of MDP episode.
```

Due to randomness in the initial state and transition probability as shown in Algorithm 1, the environment is non-deterministic [96]. As a result, an optimal policy, denoted by π^*, is obtained by maximising the expected sum of long-run rewards which averages out the randomness:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[G(t)|\pi\right] \; ; \; s(0) \sim p(s(0)), a(t) \sim \pi(\cdot|s(t)), s(t+1) \sim p(\cdot|s(t), a(t)).$$

3.3 A Conceptual Framework

3.3.1 On-Policy State-Value Function

In Algorithm 1, following a policy produces a sample of trajectories given by:

$$\tau = \{s(t), a(t), R(t)\}_{t\geq 0}.$$

An on-policy state-value function is defined as the expected accumulated rewards when starting in $s \in \mathcal{S}$ and following π, and it is given by:

$$V^\pi(s) := \mathbb{E}_\pi\left[G(t)|s(t) = s\right] = \mathbb{E}_\pi\left[\sum_{k\geq 0} \gamma^k R(t+k)|s(t) = s\right].$$

An RL agent evaluates all $s \in \mathcal{S}$ and a state-value function determines how good a state s is [96]. The optimal on-policy state-value function is the maximum state-value function given by:

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

3.3.2 On-Policy State-Action Value Function

Q-values are used to evaluate the quality of state-action pairs $(a_t, s_t) \in \mathcal{D}$. An on-policy Q-value is defined as the expected cumulative rewards from taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ and then following policy π thereafter [61, 96], and it is computed as:

$$Q^{\pi}(s,a) := \mathbb{E}_{\pi}\left[G(t)|s(t) = s, a(t) = a\right] = \mathbb{E}_{\pi}\left[\sum_{k\geq 0} \gamma^k R(t+k)|s(t) = s, a(t) = a\right]. \tag{3.2}$$

Among all policies, an optimal Q-value function is the maximum state-action value function calculated as:

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a).$$

3.3.3 Bellman Equations

Bellman equations (BE) refer to recursive equations that decompose value functions into a sum of immediate rewards and discounted future values [96]. According to Sutton et al [137], for any policy π and any state s, a consistency condition holds between the value of s and values of the possible successor states $s^{'}$, and it is given by:

$$V^{\pi}(s) = \sum_{a\in\mathcal{A}} \pi(a|s)\left[R(s,a) + \gamma \sum_{s^{'}\in\mathcal{S}} p(s^{'}|s,a)V^{\pi}(s^{'})\right] = \mathbb{E}_{a\sim\pi, s^{'}\sim\mathcal{P}}\left[R(s,a) + \gamma V^{\pi}(s^{'})\right].$$

A corresponding BE for an on-policy Q-value function is similarly obtained by:

$$Q^{\pi}(s,a) = \sum_{s^{'}\in\mathcal{S}} p(s^{'}|s,a)\left[R(s,a) + \gamma \sum_{a^{'}\in\mathcal{A}} \pi(a^{'}|s^{'})Q^{\pi}(s^{'},a^{'})\right] = \mathbb{E}_{s^{'}\sim\mathcal{P}}\left[R(s,a) + \gamma\mathbb{E}_{a^{'}\sim\pi}[Q^{\pi}(s^{'},a^{'})]\right].$$

3.3.4 Bellman Optimality Equations

According to Mitchell et al [96] and Sutton et al [137], a value function linked to an optimal policy for an MDP satisfies Bellman optimality equation given by:

$$V^*(s) = \max_{a\in\mathcal{A}} \mathbb{E}_{s^{'}\sim\mathcal{P}}\left[R(s,a) + \gamma V^*(s^{'})\right]. \tag{3.3}$$

A corresponding Bellman-style optimality condition for a Q-value is given by:

$$Q^*(s,a) = \mathbb{E}_{s^{'}\sim\mathcal{P}}\left[R(s,a) + \gamma \max_{a^{'}\in\mathcal{A}} Q^*(s^{'},a^{'})\right]. \tag{3.4}$$

The intuition behind Bellman's optimality equations is that if an optimal value function is known, then an optimal strategy is inferred from the optimal value function. The presence of "max" over the actions as shown in Equation 3.3 and Equation 3.4 conveys the idea that an agent selects an action that leads to the highest value in any situation when it has to choose an action [61].

3.4 Reinforcement Learning Algorithms

RL algorithms based on asynchronous dynamic programming and stochastic approximation were developed to solve MDP problems [22]. RL algorithms are categorised depending on whether a model of the environment is available to an RL agent.

3.4.1 Model-based Reinforcement Learning Algorithms

A model of the environment is specified explicitly in model-based MDPs. Specification of a model of the environment allows agents to have complete knowledge, thereby enabling them to plan prospectively using perfect information, observe the impact for a range of possible options, and explicitly decide an action plan. The results of prospective planning can then be distilled into a learned policy. The specified reward function and state transitions are fully exploited to retrieve an optimal policy using dynamic programming algorithms [137]. Bellman optimality equations can be solved using the recursive backward value iteration method of dynamic programming. The biggest drawback is that the environment's ground-truth model is generally not readily accessible to an RL agent [61].

3.4.2 Model-free Reinforcement Learning algorithms

In model-free RL algorithms, an RL agent does not attempt to understand the environment. These algorithms overlook the model of the environment, thus operate as trial-and-error algorithms. The agent has incomplete information [137]. In this section, a more detailed discussion of widely used model-free RL algorithms is provided.

3.4.2.1 Q-Learning

An optimal policy can be solved by applying the Q-learning algorithm, an off-policy temporal difference (TD) control algorithm that uses BE to iteratively update the state-action value function [137, 147]. The iterative update process is given by:

$$Q_{t+1}(s,a) = \mathbb{E}\left[R(s,a) + \gamma \max_{a'} Q_t(s',a')|s,a\right].$$

Q-learning refines the policy greedily with respect to action values by the "max" operator as:

Algorithm 2 Q-learning Algorithm [137]:

```
1: Initialize Q-value arbitrarily, with the learning rate given by $\eta$,
2: for each episode do
3:     initialize state $s$,
4:     for each step of episode, state $s$ is not terminal do
5:         $a \leftarrow$ action for $s$ derived by Q, receive $r, s'$,
6:         $Q(s,a) \leftarrow Q(s,a) + \eta \left[r + \max_{a'} Q(s',a') - Q(s,a))\right]$,
7:     end
8: end
```

Q-learning given by Algorithm 2 relies on TD-error and also on the fact that $\lim_{t\to\infty} Q_t = Q^*$. Problems start to be more apparent as the state and action spaces become large or continuous, leading to scalability issues. Q-learning which requires the computation of $Q(s,a)$ for every state-action pair becomes more computationally demanding, and in some cases, impractical. The phenomenon is referred to as Bellman's curse of dimensionality [73, 100]. However, a function approximator given by $Q(s,a;\theta)$ can be used to estimate the Q-value function as:

$$Q(s,a;\Theta) \approx Q^*(s,a), \tag{3.5}$$

where $\theta \in \Theta$ are weight parameters [148].

3.4.2.2 Deep Q-learning

Large applications of RL require the use of function approximators such as DNNs, decision trees, or instance-based methods [138]. A function approximation given by Equation 3.5 aims to generalise from examples of a function to construct an approximation of an entire function [72]. Figure 3.2 shows a Deep Q-Network (DQN) which combines Q-learning and a DNN, with the latter used as a function approximator [59, 72, 73].

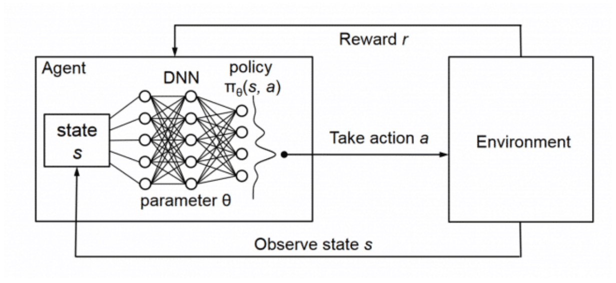


FIGURE 3.2: Deep Q-Network [87]

An algorithm for DQN is given by:

Algorithm 3 Deep Q-learning adapted from [72]

```
1: Learning rate: $\eta$,
2: Initialize $\theta = (\theta_1, \dots, \theta_{i-1}, \theta_i, \theta_{i+1}, \dots) \in \Theta$,   ▷ Initialize parameters.
3: $y_i := \mathbb{E}_{s' \sim \varepsilon}\left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a)\right]$,
4: $L_i(\theta_i) := \mathbb{E}_{s,a\sim\rho(\cdot)}\left[(y_i - Q(s,a;\theta_i))^2\right]$,   ▷ Loss function in a forward pass.
5: $\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{,a\sim\rho(\cdot);s'\sim\varepsilon}\left[(y_i - Q(s,a:\theta_i))\nabla_{\theta_i} Q(s,a;\theta_i)\right]$,   ▷ Gradient.
6: while $\theta$ not converged do
7:     $i \leftarrow i + 1$   ▷ Get gradients at step $i$,
8:     $g_i \leftarrow \nabla_{\theta_i} L_i(\theta_i)$,
9:     $\theta_i \leftarrow \theta_{i-1} - \eta \cdot g_i$,   ▷ Update parameters using Gradient Descent.
10: return $\theta^* \in \Theta$.
```

Parameters $\theta \in \Theta$ are weights in the DNN. For both Q-learning and DQN, an optimal policy is inferred from Q-values as:

$$\pi^* = \arg\max_{\pi} Q^*(s,a).$$

Algorithm 3 has its setbacks: the first problem is that the learning process is inefficient since samples are correlated. Furthermore, current parameters determine subsequent training examples, leading to feedback loops. These problems can be solved by incorporating experience replays that have the primary function of stabilising the training process. During Q-learning updates, samples are drawn at random from replay memory [1]. In other words, experience replays ensure that more important experience transitions can be replayed more frequently to learn more efficiently [72].

3.4.2.3 Policy Gradient

The RL approaches in Section 3.4.2.1 and 3.4.2.2 fall under critic or action-value methods. They entail estimating action-values then selecting actions based on the learned values of actions. Policy gradient (PG) methods are a subclass of policy-based algorithms that learn a parameterised policy that can select actions without consulting a value function [138].

Define class of parameterised policies as:

$$\Pi_\Theta = \{\pi(a|s,\theta) : \theta \in \Theta \subset \Re^k\},$$

where $\pi(a|s,\theta) = Pr\{a(t) = a|s(t) = s, \theta(t) = \theta\}$. The policy is never deterministic since $\pi_\theta(a|s) \in (0,1)\ \forall s, a, \theta$ thereby ensuring exploration [137]. The gradient of a scalar performance metric $J(\theta)$ is used during the policy parameters' learning process. The objective is to determine an optimal policy $\pi(a|s,\theta^*)$ where $\theta^* = \arg\max\limits_\theta J(\theta)$. According to Williams [151], parameters $\theta^* \in \Theta$ are estimated through stochastic gradient ascent:

$$\Delta\theta_t = \eta \nabla_\theta J(\theta_t), \tag{3.6}$$

where η is positive-definite step size [137]. Assuming every episode starts at state s_0, and for ease of exposition, define the performance measure in the episodic case as $J(\theta) = V^{\pi_\theta}(s_0)$. Noting that:

$$\nabla V^\pi(s) = \sum_{s' \in \mathcal{S}} \sum_{k=0}^{\infty} Pr(s \to s', k, \pi) \sum_a \nabla \pi(a|s') Q^\pi(s', a),$$

where $Pr(s \to s', k, \pi)$ denotes the probability of transitioning from state s to state s' in k steps under policy π, an analytic expression for $\nabla_\theta J(\theta)$ is given as a proportionality expression by the Policy Gradient Theorem (PGT) as:

$$\begin{aligned}
\nabla_\theta J(\theta) = \nabla V^\pi(s_0) &= \sum_s \left(\sum_{k=0}^{\infty} Pr(s \to s', k, \pi) \right) \sum_a \nabla \pi(a|s',\theta) Q^\pi(s', a) \\
&= \sum_s \phi(s) \sum_a \nabla \pi(a|s',\theta) Q^\pi(s', a) \\
&= \sum_{s'} \phi(s') \sum_s \frac{\phi(s)}{\sum_{s'} \phi(s')} \sum_a \nabla \pi(a|s',\theta) Q^\pi(s', a) \\
&= \sum_{s'} \phi(s') \sum_s \alpha(s) \sum_a \nabla \pi(a|s',\theta) Q^\pi(s', a) \\
&\propto \sum_s \alpha(s) \sum_a \nabla \pi(a|s',\theta) Q^\pi(s', a),
\end{aligned} \tag{3.7}$$

where $\alpha(s)$ is the on-policy distribution under π [130, 137]. Equation 3.7 does not depend on state transition probabilities or the derivative of the state distribution [137]. PG approaches have many benefits, including the ability to learn precise probabilities for actions to be taken [138]. Moreover, they can learn sufficient exploration levels and asymptotically approach deterministic policies [130]. Furthermore, they can handle spaces of continuous motion [73, 137].

3.4.2.4 REINFORCE: Monte Carlo PG

For the weight update in Equation 3.6 to be correct, sampling must be carried out in a manner that ensures that the sample gradient's expectation is proportional to the actual gradient of $J(\theta)$ [138]. Using PGT in Equation 3.7, following a policy π ensures that:

$$\nabla_\theta J(\theta) \propto \sum_s \alpha(s) \sum_a \nabla \pi(a|s,\theta) Q^\pi(s,a) = \mathbb{E}_\pi \left[\sum_a \nabla \pi(a|s(t),\theta) Q^\pi(s,a) \right]. \tag{3.8}$$

REINFORCE algorithm, put forward by Williams [151], is classical a PG method whose update at time t involves $a(t)$ and sampling the expectation. A weighting is introduced without changing the equality in Equation 3.8; a is replaced by the sample $a(t) \sim \pi$ and using Equation 3.2:

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_\pi \left[\sum_a \pi(a|s(t),\theta) Q^\pi(s(t),a) \frac{\nabla \pi(a|s(t),\theta)}{\pi(a|s(t),\theta)} \right] \\ &= \mathbb{E}_\pi \left[Q^\pi(s(t),a(t)) \frac{\nabla \pi(a|s(t),\theta)}{\pi(a|s(t),\theta)} \right] \\ &= \mathbb{E}_\pi \left[G(t) \frac{\nabla \pi(a|s(t),\theta)}{\pi(a|s(t),\theta)} \right] \\ &= \mathbb{E}_\pi \left[G(t) \nabla \log\left(\pi(a|s(t),\theta)\right) \right], \end{aligned} \tag{3.9}$$

where $G(t)$ is given by Equation 3.1 [138, 151]. The final expression in Equation 3.9 is a quantity that is sampled at each time step, and its expectation is equal to $\nabla_\theta J(\theta)$ [137], thus giving the REINFORCE update:

$$\Delta\theta_t = \eta G(t) \nabla \log\left(\pi(a(t)|s(t),\theta_t)\right).$$

Beginning at time t, all possible rewards for the whole episode are included, so in this sense, REINFORCE is a Monte Carlo algorithm [137]. REINFORCE algorithm works by pushing up the probabilities of actions seen when $G(t)$ is high and does the opposite when $G(t)$ is low [151].

3.4.2.5 REINFORCE with Baseline

Being a Monte Carlo technique, REINFORCE usually has high variance that delays convergence, thus slowing the learning process [137]. One of the variance reduction methods involves introducing an arbitrary baseline function, denoted by $b(s)$, that depends on the state [138]. In that case, PGT becomes:

$$\nabla_\theta J(\theta) \propto \sum_s \alpha(s) \sum_a \left(Q^\pi(s,a) - b(s)\right) \nabla \pi(a|s,\theta). \tag{3.10}$$

Any arbitrary function that does not vary with a can be used as a baseline [137]. Note that PGT's validity still holds since the quantity subtracted is zero:

$$\sum_a b(s) \nabla \pi(a|s,\theta) = b(s) \nabla 1 = 0.$$

The update rule for REINFORCE that includes a general baseline is:

$$\Delta\theta_t = \eta \left(G(t) - b(s(t))\right) \nabla \log \left(\pi(a(t)|s(t), \theta_t)\right).$$

3.4.2.6 Actor-Critic Methods

Intuitively, action $a(t)$ in a state $s(t)$ is highly desirable if the advantage function, given by:

$$A^{\pi}\left(s(t), a(t)\right) = Q^{\pi}\left(s(t), a(t)\right) - V^{\pi}(s(t)),$$

is large. For Actor-Critic methods, PGT becomes:

$$\begin{aligned}\nabla_\theta J(\theta) &\propto \sum_s \alpha(s) \sum_a \left(Q^{\pi}(s,a) - V^{\pi}(s(t))\right) \nabla \pi(a|s,\theta) \\ &= \sum_s \alpha(s) \sum_a \left(A^{\pi}\left(s(t), a(t)\right)\right) \nabla \pi(a|s,\theta).\end{aligned} \tag{3.11}$$

The difference between Equation 3.10 and Equation 3.11 is quite subtle; the value-function is only used as a baseline to stabilise the learning process by the former, while in the latter, it acts as a critic [137, 138]. Actor-Critic methods combine PG and TD-learning by training both an actor (the policy) and a critic (the Q-function) [66]. Effectively, the actor decides which action to take, and the critic tells the actor how good its action is and how it should adjust [137]. The bias introduced by TD-learning and dependence on state representations is advantageous because variance is minimised and learning is accelerated [137]. REINFORCE with the baseline is unbiased and converges to a local minimum, but it tends to learn slowly and is inconvenient to apply online since it is a Monte Carlo method [137, 138].

3.4.2.7 Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) is an off-policy method that chains together value-based DQN and Deterministic Policy Gradient (DPG) for large continuous domains [73]. PG methods discussed up to this point are referred to as Stochastic Policy Gradient (SPG) methods. As shown by the PGT in Equation 3.7, the summation (or integration) is over both the state and the action spaces. On the contrary, DPG integrates only over the state space. DDPG addresses continuous control problems by using the DPG method that entails utilising an actor to continuously output continuous actions [76]. A critic in the form of the Q-value would then evaluate and improve the policy [3, 73]. The algorithm trades off variance reduced by DPG with bias introduced from Q-learning.

3.5 Recurrent Reinforcement Learning

Trading systems form part of the active portfolio management process. Placing trading orders is a systematic task that takes into account several practical factors [30]. An active portfolio's performance is dependent on progressions of interconnected investment decisions and is thus path-dependent [99]. Historic actions and the corresponding positions need to be explicitly modelled in the policy learning component [30]. Furthermore, active portfolio managers typically set up optimal portfolios that satisfy immediate targets. Portfolio rebalancing is performed at subsequent periods as the future unveils itself [4]. Multi-period rebalancing inevitably incurs transaction costs associated with the buying and selling of stocks, market impact, and taxes, thus knowledge of internal state information is a prerequisite [30, 40, 99]. An active portfolio management system must be recurrent, that is, the prior output needs to be incorporated as input to preserve internal state information.

Moody et al [99, 100] put forward the Recurrent Reinforce Learning (RRL) approach to find approximate solutions in the field of portfolio management. RRL is an adaptive policy search algorithm that uses previous output as input for sequential decision making in portfolio optimisation [36, 81, 100]. RRL is guided by the RL algorithms discussed in Section 3.4.2, more particularly, leaning towards the PG approach. For ease of exposition, assume a portfolio consisting of a single risky asset and a money market instrument. Define a decision function for a risky asset by:

$$\pi(t) = \phi(\theta; \pi(t-1), I(t)) = \phi\left(b + u\pi(t-1) + \theta I(t)\right), \tag{3.12}$$

where $I(t)$ is the information set and $\theta \in \Theta = \{b, u, \theta\}$ denotes system parameters of the regression [99, 100]. Equation 3.12 is recurrent because the output of the prior decision is part of the input for the current decision [30]. Incorporating $\pi_i(t-1)$ into the regression deters changing trading positions too frequently, thus avoid incurring heavy transaction costs [30, 59, 81, 99]. Similarly, let the decision function for a money market instrument be $\pi_0(t)$. For a portfolio that allows long, neutral, or short positions, the decision space follows a stochastic process given by:

$$(\pi(t), \pi_0(t)) \in [-1, 1] \times \Re. \tag{3.13}$$

Note that $\pi(t)$ and $\pi_0(t)$ in Equation 3.13 are allocations to a risky asset and risk-free asset, respectively, for the simplified portfolio [36, 98]. The reward signal realised at the end of time interval $(t-1, t]$ is given by:

$$R(t) = \pi(t-1)r(t) - \delta|\pi(t) - \pi(t-1)|, \tag{3.14}$$

where $r(t)$ is the risky asset return at time t and δ are mandatory transaction costs incurred when flipping positions due to rebalancing [4, 30]. Position $\pi(t)$ is established at time t for the interval $(t-1, t]$, and its re-evaluation occurs at $t+1$ in the subsequent period $(t, t+1]$ [100]. Transactions costs δ penalise excessive portfolio rebalancing. A reward signal $R(t)$ that scores the action at time t is received from the environment by the control system. The overall portfolio performance can be expressed as a function of a sequence of portfolio rewards throughout the whole training period, denoted by T, as:

$$U(T|\theta) = U(R(1), \ldots, R(T)|\theta). \tag{3.15}$$

RRL is set up as a PG method, with $U(T|\theta)$ being the performance function [4]. In a manner similar to Section 3.4.2.3, let $\Pi_\Theta = \{\pi : \theta \in \Re\}$ be a class of parameterised policies. The optimal policy given by π^*_θ is obtained when $\theta^* = \arg\max_\theta U(T|\theta)$. Optimal parameters $\theta^* \in \theta$ are estimated by stochastic gradient ascent given by:

$$\Delta\theta_t = \eta \nabla_\theta U(T|\theta).$$

The maximisation for an entire sequence of T trades is performed via the batch-version of RRL [99]. Using the chain rule of differentiation, gradient $\nabla_\theta U(T|\theta)$ is obtained by:

$$\nabla_\theta U(T|\theta) = \sum_{t=1}^{T} \frac{dU(T|\theta)}{dR(t)} \left\{ \frac{dR(t)}{d\pi(t)} \frac{d\pi(t)}{d\theta} + \frac{dR(t)}{d\pi(t-1)} \frac{d\pi(t-1)}{d\theta} \right\}. \tag{3.16}$$

RRL is generally stable due to its recurrent nature and the fact that it is not a Monte Carlo-based approach, a baseline function is typically not considered. Note that $\frac{d\pi(t)}{d\theta}$ is a total derivative that depends on the entire sequence of previous trades due to the rebalancing action [99]. The gradient $\nabla_\theta U(T|\theta)$ can be computed using Backpropagation Through Time (BPTT) [46, 149] or dynamic backpropagation [75] to account for temporal dependencies [30, 99]. Moody et al [99, 100] uses the Sharpe Ratio and its differential as performance functions for the batch version of RRL and the online version, respectively. Almahdi et al [4] uses other performance measures such as the Calmar Ratio [2], while Deng et al [30] uses cumulative profits. Ratios such as the Sharpe Ratio and the Calmar Ratio are called safety-first ratios, and are typically used as risk-sensitive performance measures, while total or cumulative profit is a risk insensitive performance measure [4, 30].

The online RRL version considers only the terms that depend on the most recently realised portfolio return during a forward pass through the data as given by:

$$\frac{dU(t|\theta)}{d\theta} = \frac{dU(t|\theta)}{dR(t)} \left\{ \frac{dR(t)}{d\pi(t)} \frac{d\pi(t)}{d\theta} + \frac{dR(t)}{d\pi(t-1)} \frac{d\pi(t-1)}{d\theta} \right\}. \tag{3.17}$$

The terms in the curly brackets in Equation 3.16 and Equation 3.17 are computed as:

$$\begin{aligned} \frac{d\pi(t)}{d\theta} &= \frac{d}{d\theta}\left(\phi(b + \theta I(t) + u\pi(t-1)\right) = \phi' \left(I(t) + u\frac{d\pi(t-1)}{d\theta} \right), \\ \frac{dR(t)}{d\pi(t)} &= -\delta \operatorname{sign}\left(\pi(t) - \pi(t-1)\right), \\ \frac{dR(t)}{d\pi(t-1)} &= r(t) - \delta \operatorname{sign}\left(\pi(t) - \pi(t-1)\right), \end{aligned}$$

where ϕ' is a first derivative of the function ϕ in Equation 3.12 [4, 98].

3.6 Summary

The universe of RL algorithms is quite vast, and it is still under intense study. This chapter provides a discussion of MDP-based RL algorithms. MDP-based RL algorithms fall into two broad classes, namely, model-based versus model-free. Model-free RL algorithms also fall into two classes, namely, value-based and policy-based. As highlighted, the former suffers from Bellman's curse of dimensionality, making their application to portfolio management less popular. The chapter concludes with a discussion of RRL, a policy-based technique pioneered by Moody et al [99] to approximate solutions in the field of portfolio management. Note that the decision function in Equation 3.12 includes an information set, denoted by I, containing state representations of the environment. Another imperative of this study is to improve decision-making by engineering the quality of the information contained in the information set I. Chapter 4 looks at some of those engineering or signal processing methods proposed in the study.

Chapter 4

Signal Processing and Feature Extraction Methods

Temporary disequilibrium in supply and demand and the resulting corrections introduce noise that has the effect of obscuring genuine underlying patterns in financial markets. Moreover, financial markets are highly susceptible to unanticipated shocks and market contagion that manifest from to time, causing stock prices to deviate from their fundamental values [11, 16]. Noise in financial markets is attributed to the level of liquidity as noted by Hu et al [51], financial and business cycles as discussed by the Financial Instability Hypothesis by Minsky [95], or perceptions regarding earnings which shift investor preference as discussed by Taylor et al [140]. The objective of our study is to attenuate the impact of market disturbance and noise on equity portfolios.

This chapter discusses candidate methods used to improve the quality of information used to inform decision-making in portfolio management. These methods cover a broad spectrum; they include algorithms that isolate trends, harmonic patterns, and noise to algorithms that extract latent features from financial time series data. The objective of these methods is to increase the signal-to-noise ratio, thereby impacting decision-making in asset allocations. Improving the signal-to-noise ratio mitigates the risk of making erroneous decisions that lead to suboptimal asset allocations. The consequences of suboptimal asset allocations are dire and directly affect profit margins, increase drawdowns, and ultimately lead to the failure of a fund.

4.1 Singular Spectrum Analysis

Singular Spectrum Analysis (SSA) is a model-free algorithm that involves decomposing time series data into components that can be analysed by economic reasoning [141]. It is a non-parametric time series analysis technique that combines classical analysis, dynamical systems, signal processing, and multivariate geometry [48].

4.1.1 General Review of SSA and Usefulness

The SSA algorithm decomposes a given univariate time series data $\mathbf{s}$ into m additive, independent and interpretable components as:

$$\mathbf{s} = \sum_{i=1}^{m} \tilde{s}_i.$$

Examples of such components include slowly-varying trend, harmonic components, and structureless noise [41, 47, 48]. SSA solves numerous problems that include time series decomposition, trend extraction, periodicity detection and extraction, the revelation of data structure, signal extraction, denoising, filtering, forecasting, missing data imputation, change point detection and so forth [42, 47]. SSA has proven to be very powerful and has already become a predominant approach for climatic, geophysical, and meteorological time series data analysis [41].

4.1.2 Details of the SSA algorithm

This section gives a detailed summary of the main steps of the SSA algorithm. For a much more extensive exposition, the reader is referred to [41] by Golyandina et al or to [47, 48] by Hassani et al. SSA algorithm is summarised by two complementary stages [48, 141]:

1. Decomposition: two step process involving Embedding and Singular Value Decomposition (SVD).
2. Reconstruction: incorporates Grouping and Hankelization.

This section details four steps of SSA in chronological order for univariate time series data. The multivariate variate version proposed in the study is discussed later in the chapter.

4.1.2.1 Embedding

Consider univariate time series data of length T given by $\mathbf{s} = (s_0, \ldots, s_T)$. Define an embedding function $\mathcal{T}$ that maps $\mathbf{s}$ to a sequence of multidimensional L-lagged vectors denoted by $\mathbf{E}_1, \ldots, \mathbf{E}_K$ where $\mathbf{E}_i = (s_i, \ldots, s_{i+L-1})^\top$ and $K = T - L + 1$ for a window length L such that $2 < L < T$. A trajectory matrix defined by $E = [\mathbf{E}_1 : \ldots : \mathbf{E}_K]$ is given by:

$$E = \mathcal{T}(\mathbf{s}) = (e_{ij})_{i,j=1}^{L,K} = \begin{bmatrix} s_1 & s_2 & \cdots & s_K \\ s_2 & s_3 & \cdots & s_{K+1} \\ s_3 & s_4 & \cdots & s_{K+2} \\ \vdots & \vdots & \ddots & \vdots \\ s_L & s_{L+1} & \cdots & s_T \end{bmatrix}, \tag{4.1}$$

where $e_{ij} = s_{i+j-1}$. The mapping function $\mathcal{T}$ is such that $\mathcal{T} : \mathbf{s} \mapsto E \in \mathcal{M}_{L,K}^{(H)}$ where $\mathcal{M}_{L,K}^{(H)}$ is a space of structured Hankel-like matrices, that is, E has equal entries in the diagonal, thus $i + j = c$, a constant [41, 47]. In embedding, the only parameter is window length, denoted by L, which reflects the resolution of the method. The larger the window, the more detailed the decomposition is [133].

4.1.2.2 Singular Value Decomposition

This stage decomposes trajectory matrix the E in Equation 4.1 using SVD into a sum of unit rank bi-orthogonal elementary matrices [47]. Let $W = EE^\top$, then the eigenvalues of W in decreasing order of magnitude are $\lambda_1 \geq \ldots \geq \lambda_L \geq 0$. Similarly, eigenvectors corresponding to the eigenvalues are denoted by an orthonormal system $\{\mathbf{U}_i\}_{i=1}^{L}$. Let $\mathbf{V}_i = E^\top \frac{\mathbf{U}_i}{\sqrt{\lambda_i}}$ $\forall i = 1, \ldots, r$, then the SVD of trajectory matrix E is given by:

$$E = \sum_{i=1}^{r} E_i = \sum_{i=1}^{r} \sqrt{\lambda_i}\mathbf{U}_i\mathbf{V}_i^\top, \tag{4.2}$$

where $r = \max(i : \lambda_i > 0) = \operatorname{rank} E$ [41]. The matrices E_i have unit rank, therefore they are elementary matrices. The collection $\left(\sqrt{\lambda_i}, \mathbf{U}_i, \mathbf{V}_i\right)$ is called an i^{th} eigentriple of the SVD [41]. A set $\{\sqrt{\lambda_i}\}_{i=1}^{r}$ which consists of singular values of matrix E is an orthonormal basis of columns of E, and it is referred to as the spectrum of matrix E [47, 48, 133]. A final point to note is that the SVD that is performed is optimal, meaning that $\forall\ E^{(l)}$ of rank $l < r$, matrix $E^{(l)} = \sum_{i=1}^{l} E_i$ provides the best approximations to the trajectory matrix E, that is, $\|E - E^{(l)}\|$ is minimum [47].

4.1.2.3 Grouping

This step involves partitioning elementary matrices $\{E_i\}_{i=1}^{r}$ into several groups and summing matrices within each group [47]. The set of indices from the expansion in Equation 4.2 are partitioned in to m disjoint subsets, that is, $\{1, \ldots, r\} = \bigsqcup_{i=1}^{m} I_i$ where grouping with $I_i = \{i\}$ is called elementary [133]. Let $I = \{i_1, \ldots, i_p\}$ be a group of indices within $i_1, \ldots, i_p$, then the resulting matrix E_I corresponding to group I is defined as $E_I = \sum_{j=1}^{p} E_{i_j}$ [41]. These matrices are computed for $I = I_1, \ldots, I_m$ and the expansion from Equation 4.2 leads to a decomposition given by:

$$E = \sum_{i=1}^{m} E_{I_i}. \tag{4.3}$$

A procedure of choosing sets $I_1, \ldots, I_m$ is called eigentriple grouping [41, 47].

4.1.2.4 Reconstruction

This last step of SSA transforms each matrix of the grouped decomposition in Equation 4.3 into a new series of length T. More specifically, the initial time series $\mathbf{s}$ becomes a sum of m time series data of length T given by:

$$\mathbf{s} = \sum_{i=1}^{m} \tilde{s}_i \ ; \ \tilde{s}_i = \mathcal{T}^{-1} H(E_{I_i}),$$

where H is an operator called diagonal averaging or Hankelization [41, 48, 133].

4.2 Multichannel Singular Spectrum Analysis

For multivariate time series financial data, the decomposition and reconstruction procedures discussed in section 4.1.2 are applied simultaneously to a collection of time series time data. This method is called Multichannel SSA (MSSA) [48, 135]. Let $\left\{\mathbf{s}^{(k)} = (s_j^{(k)})_{j=1}^{T_k}, k = 1, \ldots, m\right\}$ be a collection of m time series data of length T_k [42]. In order to cater for multivariate time series data, the range for the window length is modified to $2 < L < \min\left(T_K\,, k = 1, \ldots, m\right)$ [42, 135]. The rest of the algorithm is similar to the univariate time series case in Section 4.1.2. In this context, the study explores the applicability and robustness of MSSA in finding structure in financial market data and enhancing the signal-to-noise ratio via data denoising.

4.3 Independent Component Analysis

Independent Component Analysis (ICA) is a novel statistical signal processing tool designed to find independent latent source signals from observed mixture signals without prior knowledge of the mixing mechanism [80]. ICA assumes a generative model by which the observed signals are instantaneous linear mixtures of independent source signals or underlying factors [55, 107]. The goal of ICA is to extract independent sources signals, as well as the mixing process [9]. Doing so helps to reveal the driving mechanism that otherwise remains hidden [107].

4.3.1 Details of ICA

Consider an observed m-dimensional original signal denoted by s which is a collection of $\left\{\mathbf{s}^{(k)} = (s_j^{(k)})_{j=1}^T\,, k = 1, \ldots, m\right\}$. Assuming the multivariate signal s is linearly transformed to an n-dimensional signal v such that $m > n$ as:

$$v = \mathcal{L}s, \tag{4.4}$$

where $\mathcal{L}$ is a compression linear transform such as Karhunen-Loeve Transform [107]. Principal Component Analysis (PCA) utilises such a transform to extract latent variables using second-order statistics [25, 80]. More specifically, rows of $\mathcal{L}$ denoted by ℓ_j for $j = 1, \ldots, n$ are orthonormal eigenvectors of the covariance matrix $\Sigma = \mathbb{E}\left[ss^\top\right]$. The multivariate time series s is assumed to be stationary and $\mathbb{E}[s] = 0$. The latent variables, denoted by $\{v_j\}_{j=1}^n$ called Principal Components, are uncorrelated, and the generative model in Equation 4.4 uses second-order statistics of the original multivariate time series s [10, 54, 55]. In general, PCA projects data to an orthogonal space, and the projections point towards a direction that gives maximum variance [9].

Back et al [9, 10] notes that ICA uses higher-order statistics to reveal high-level transients. These high-level transients reveal more useful information or hidden data structures that are not immediately observable when using PCA, thus making ICA more appropriate for multivariate time series analysis [25]. The observed multidimensional signal is assumed to be driven by the generative model:

$$s = Av = \sum a_j v_j, \tag{4.5}$$

where A is mixing matrix which is unknown, and $v = \{v_j\}_{j=1}^n$ are independent source signals [9, 10, 25]. A key assumption is that an observed signal s reflects a reaction of a system to independent source signals, and these source signals are not immediately observable since they are buried within the observed signal as well as noise [9]. The goal of ICA is to determine a linear transform by which the $\{v_j\}_{j=1}^n$ are statistically independent [9, 25]. A solution is sought of the form:

$$\hat{v} = y = Ws, \tag{4.6}$$

where W is the separating or un-mixing matrix that causes elements of y to be statistically independent [107]. Latent variables given by $y = \{y_j\}_{j=1}^n$ are an approximation of independent source signals $v = \{v_j\}_{j=1}^n$, and are referred to as Independent Components [9, 107]. From Equations 4.5 and 4.6, perfect separation is achieved when $A = W^{-1}$ [9].

4.3.2 ICA Methods

ICA methods are set up as optimization problems. A measure of the independence of latent variables $\{y_j\}_{j=1}^n$ is defined and used as an objective function. The optimisation techniques involve solving for the separating matrix W that maximises a given measure of independence [80]. Despite there being several functions in literature used to measure independence, the resulting separating matrices do not differ significantly [49, 70].

4.3.2.1 Minimising Mutual Information

Oja et al [107] formulates entropy $H(y)$ as:

$$H(y) = -\int f(y)\log f(y)dy = -\mathbb{E}[\log f(y)],$$

where $f(y)$ is a probability density function of y. The Kullback-Leibler Divergence or Mutual Information between joint density function $f(y)$ and a product of its marginal densities $f(y_j)\,, j = 1, \ldots, n$ is given by:

$$I(y_1, \ldots, y_n) = H(y) - \sum_{i=1}^{n} H(y_i),$$

where $\{H(y_i)\}_{i=1}^n$ are marginal entropies of the outputs. Mutual Information is interpreted as a measure of reduction in uncertainty [70]. It is strictly positive and will equal zero when $\{y_j\}_{j=1}^n$ are independent [55, 107].

Amari et al [5] proposes the INFOMAX algorithm, an iterative procedure that computes the separating matrix W for which Mutual Information is zero. The details of the algorithm are:

Algorithm 4 INFOMAX algorithm [70]

```
1: initialize $W(0)$,
2: **while** $W(t)$ not converged **do**
3:     $t \leftarrow t + 1$,                                   ▷ time step for approximation.
4:     $W(t) = W(t-1) + \eta(t-1)\left(\mathbb{1} - f(y)y^\top\right)W(t-1)$,
5: **return** $W(t)$.
```

where $\eta(t)$ is a function that determines magnitude of the steps for the separating matrix updates and $f(y)$ is a nonlinear function [70]. In summary, the INFOMAX algorithm finds a separating matrix that minimizes Mutual Information, which in a sense, is equivalent to searching for latent variables that are maximally independent [70].

4.3.2.2 Maximising Non-Gaussianity

Another way to estimate independent components is by focusing on quantifying non-Gaussianity [70, 107]. Independent components with a non-Gaussian distribution imply statistical independence [80]. According to Hyvärinen et al [54], negentropy, denoted by $N(y)$, is calculated as:

$$N(y) = H(y_G) - H(y), \tag{4.7}$$

where y_G is a random vector from a Gaussian distribution that has the same covariance as that of y. The negentropy in Equation 4.7 is effectively measuring the degree of non-Gaussianity of y. Gaussian distributions have the maximum entropy for any given covariance matrix, thus negentropy, $N(y)$, is strictly positive and will equal to zero when y has a Gaussian distribution [80]. The objective of an ICA algorithm is to estimate W such that $y = Ws$ maximises negentropy, that is, finding a y that is as much different as possible from y_G [55, 107]. In most cases, the density function $f(y)$ is not known, thereby making negentropy difficult to determine [54, 80, 107]. A classical method of estimating negentropy involves using higher-order moments such as kurtosis as given by:

$$H(y) \approx \frac{1}{12}\mathbb{E}[y^3]^2 + \frac{1}{48}\text{kurt}(y)^2,$$

where is y is a random variable having a mean of zero and a variance of one [60, 107]. However, such an approximation suffers from non-robustness associated with kurtosis. To avoid this issue, an approximation for negentropy, given by:

$$N(y) \approx \{\mathbb{E}[\phi(y)] - \mathbb{E}[\phi(\Phi)]\}^2,$$

is used, where Φ is standardized Gaussian variable and $\phi(.)$ is a non-quadratic function such as $\phi_1(y) = a^{-1}\log\cosh(ay)$ where $a \in [1,2]$ or $\phi_2(y) = e^{-\frac{y^2}{2}}$ [55, 70, 80].

The FastICA algorithm developed by Hyvärinen et al [54, 55, 107] relies on a fixed point iteration scheme for finding an optimal y as:

$$y^* = \arg\max_y \{\mathbb{E}[\phi(y)] - \mathbb{E}[\phi(\Phi)]\}^2.$$

Algorithm 5 summarises the FastICA algorithm:

Algorithm 5 FastICA algorithm [54]

```
1: initialize $w_i$,
2: $w_i^+ = \mathbb{E}\left[\phi'(w_i^\top s)w_i\right] - \mathbb{E}\left[s\phi(w_i^\top s)\right]$,
3: $w_i = \frac{w_i^+}{\|w_i^+\|}$,                      ▷ $w_i$ is a column-vector of the separating matrix $W$.
4: while $w_i$ not converged do
5:     $i \leftarrow i + 1$,
6:     $w_i^+ = w_i - \sum_{j=1}^{i-1} w_i^\top w_j w_J$,
7:     $w_i = \frac{w_i^+}{\|w_i^+\|}$,
8: return $w_i$.
```

where w_i^+ is a temporary variable used to calculate w_i and $\phi'(.)$ is the derivative of $\phi(.)$ [54, 70, 107]. The study proposes the FASTICA algorithm because of its possible parallel implementation as well as its quick convergence [128].

4.4 Autoencoders

An Autoencoder (AE) is an ANN capable of learning dense representations of input data, called latent features or codings, without any supervision [39]. Given an unlabeled training example set $s \in \Re^m$, an AE tries to learn an approximation to an identity function using backpropagation by setting target values to equal inputs as:

$$\hat{s} = f_W(s) \approx s, \tag{4.8}$$

where $f_W(s)$ is the approximation to an identity function parameterised by W so that the output $\hat{s}$ is similar to input s [43]. Learning an identity function seems trivial. However, by placing constraints on the network, a crucial structure in the data is unveiled [39, 105]. Codings typically have much lower dimensionality than the input data [39]; hence they can be interpreted as a compressed representation of the original input [114]. Moreover, codings supposedly capture stable structures in the form of dependency and regularity characteristics of the unknown distribution of the observed input data [144, 145].

4.4.1 Details of AE

A vanilla AE is a three-layer symmetrical feed-forward ANN that constraints its output to equal to its input as shown by Equation 4.8 [50]. An AE learns features by first encoding its input then reconstructing or decoding them [35]. Given an input vector $s \in \Re^m$, an AE maps s onto a latent representation $z \in \Re^n$ with $m > n$ through a nonlinear mapping given by:

$$z = g_{W_1}(s) = g_{W_1}(\Theta_1 s + b_1),$$

where Θ_1 is a parameter vector to be learned, and b_1 is a bias vector that adjusts the activation of neural units. The coding vector z is then mapped back to a reconstructed vector $\hat{s}$ via backward mapping as:

$$\hat{s} = f_{W_2}(z) = f_{W_2}(\Theta_2 z + b_2).$$

The parameter matrix Θ_2 of the reverse mapping may optimally be constrained by $\Theta_2 = \Theta_1^T$ in the case of tied weights [39, 144]. Reconstructions, denoted by $\hat{s} \in \Re^m$, are constrained such that $\hat{s} \approx s$. Each training example $s^{(i)}$ is mapped to a corresponding $z^{(i)}$ and a reconstruction $\hat{s}^{(i)}$ [145]. A reconstruction error, denoted by $\mathcal{L}_{W_1,W_2}(s, \hat{s})$, is computed as:

$$\mathcal{L}_{W_1,W_2}(s, \hat{s}) = \|s - \hat{s}\|^2,$$

AE parameters $W_1 = \{\Theta_1, b_1\}$ and $W_1 = \{\Theta_2, b_2\}$ are optimised by minimising average the reconstruction error as given by:

$$W_1^*, W_2^* = \arg \min_{W_1,W_2} \frac{1}{T} \sum_{i=1}^{T} \mathcal{L}\left(s^{(i)}, \hat{s}^{(i)}\right) = \arg \min_{W_1,W_2} \frac{1}{T} \sum_{i=1}^{T} \mathcal{L}\left(s^{(i)}, f_{W_2}(g_{W_1}(s^{(i)}))\right).$$

The optimization is done via stochastic gradient descent (SGD) which iteratively updates parameters W_1 and W_2 until the average reconstruction error reaches a global minimum. The parameter update is given by:

$$\theta_i^{(t+1)} \leftarrow \theta_i^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial \theta_i} \ ; \ \theta_i \in W_i = \{\Theta_i, b_i\} \ ; \ i = 1, 2,$$

where η denotes the learning rate [39, 145]. The reminder of this section discusses different types of AEs that are implemented in the study.

4.4.2 Stacked Denoising Autoencoder

A Stacked Denoising AE (SDAE) stacks shallow AEs, partially corrupts or add perturbations to the input, and then attempts to reconstruct the original input from its partially corrupted version [12, 38, 155]. The corrupting process makes SDAEs more robust to noise or large variation, thereby encouraging them to capture statistical dependencies between inputs [35, 119]. Also, stacking shallow AEs creates a deep neural network that helps SDAEs learn hierarchical features from complicated input [43]. The input s is perturbed using a stochastic mapping given by:

$$\tilde{s} \sim \phi(\tilde{s}|s),$$

where ϕ is a distribution determined by the original distribution of s and the random noise added to it [35]. The random noise could either be pure Gaussian noise or randomly switched-off inputs using Dropout [39, 145]. For instance, an SDAE applying Dropout takes input vector s and partially occludes it with a certain probability, denoted by ρ, to create $\tilde{s}$ using a stochastic mapping given by:

$$\tilde{s} \sim \phi_{\mathcal{D}}(\tilde{s}|s, \rho) \tag{4.9}$$

SDAEs are forced to predict missing or corrupted values from randomly selected subsets of missing patterns, thereby making SDAEs robust to partial destruction of the input vector [144]. After the corruption process, SDAEs operate like conventional AEs, with the usual mapping of $\tilde{s}$ to a latent vector representation z using a deterministic function g as given by:

$$z = g_{\Theta,\beta}(\tilde{s}) = g(\Theta\tilde{s} + \beta).$$

4.4.3 Convolutional Denoising Autoencoder

Convolutional Neural Networks (CNNs) emerged from studies centered on determining how a brain perceives images [39]. CNNs are used primarily in computer vision problems due to their ability to extract high-level features using local receptive fields or input patches [26]. However, CNNs are not limited to visual perception; they are also applicable to sequential data such as time-series data [39]. For time-series data, time itself is treated as a spatial dimensional and receptive fields are characterised by localised time patches or subsequences. These types of CNNs are called 1D-CNNs [26, 79]. They offer data efficiency benefits in that once a pattern is learned at certain positions in a sequence, it can later be recognized at a different location, thereby making the learning process translation-invariant [26]. Moreover, small low-level features in the first hidden layer are concentrated and assembled to higher-level features in the next hidden layer, and so forth, thereby enhancing representation modularity [26, 39]. Furthermore, convolutional layers typically have far fewer parameters than dense layers due to sparse communication and weight sharing, thereby reducing the chance of overfitting [79].

Convolutional Denoising Autoencoders (CDAE) are based on the standard AE architecture but with convolutional encoding and decoding layers [43]. The input vector is perturbed as shown in Equation 4.9 but the encoder is now a stacked regular 1D-CNN composed of convolutional and pooling operations [35, 39]. The convolutional layers consist of a set of filters that extract features from local patches in the input space as given by:

$$o_i^{l+1}(j) = W_i^l \,.\, x^l(j) + b_i^l,$$

where W_i^l and b_i^l are weights and bias of the i^{th} kernel in the l^{th} layer, respectively, while $x^l(j)$ represents the j^{th} local region of layer l and $o_i^{l+1}(j)$ denotes the output value of convolution operation [19, 79]. Nonlinearity is imposed to feature maps via activation functions such as the Rectified Linear Unit (ReLU) given by:

$$a_i^{l+1} = \phi(o_i^{l+1}(j)) = \max\{0, o_i^{l+1}(j)\},$$

where a_i^{l+1} represents the activation of $o_i^{l+1}(j)$. Local spatiality is preserved via weight sharing among all input locations [43]. Pooling operation down-samples the latent representation by a constant factor, taking either the maximum value computed as:

$$p_i^{l+1} = \max_{(i-1)\omega+1<k<j\omega} \{v_i^l(k)\},$$

or the average value within a certain scope with a reduced dimension as given by

$$p_i^{l+1} = \operatorname*{avg}_{(i-1)\omega+1<k<j\omega} \{v_i^l(k)\},$$

where $v_i^l(k)$ is a value of the k^{th} neuron in the i^{th} feature map of the l^{th} layer; ω denotes width of the pooling filter, and p_i^{l+1} corresponds to the value of the $l+1$ output from the pooling operation [26, 39, 44, 79]. The decoder performs the reverse by combining up-sampling layers with convolutional layers [44]. Compared to SDAEs, CDAEs supposedly preserve more structural information [35, 79].

4.4.4 Recurrent Denoising Autoencoder

Recurrent Neural Networks (RNNs) are ANNs that preserve the temporal dimension of sequential data [38]. Essentially, an RNN adds a loop that connects a neuron to itself, thereby forming a directed cycle that can retain and leverage past information [26]. By delaying the recurrent signal, each node can store past information [143]. However, basic RNNs are constrained to only memorising information that is few steps back due to vanishing and exploding gradient problems [39]. This hinders their ability to detect long-term patterns in sequential or time-series data [12, 143]. By modifying a memory cell state using forget and update gates, the Long Short Term Memory (LSTM) cell can process and capture long-term patterns in sequential data, thus providing a solution to the aforementioned problem [143].

An LSTM DAE is just an extension of previous discussions. Its appeal lies in the fact that RNNs are more reliable for modeling time series sequences and can incorporate past information, effectively improving the learning process [39, 127]. The operation of LSTM DAEs is similar to what was discussed in sections above for other AEs except that LSTM DAEs use an LSTM encoder as a bottleneck layer to learn a compressed latent representation, and an LTSM decoder reconstructs the sequence as before [39].

4.5 Summary

This chapter provides a discussion of signal processing and feature extraction methods used in this study. MSSA is a model-free algorithm that decomposes financial time series data into easily interpretable components. It has been used extensively in finance and economics, quite notably, Škare et al [135] use MSSA for analysis of financial cycles, Golyandina et al [41] use it for modeling daily realised futures volatility. ICA is a natural extension to the popular PCA that reveals high-level transients using high-order statistics of input as shown by Back et al [9, 10] and Lu et al [80] for financial time-series data. It is used extensively at the signal processing stage in Electroencephalography [121, 126]. Finally, this chapter gave a brief discussion of AEs and their variants. Du et al [35] use CDAEs for feature representation. Bao [12] and Sagheer et al [127] use LSTM-based AE for multivariate time series forecasting.

Chapter 5

Research Methodology

This chapter provides a discussion of the methodology used in our study. The goal is to design a portfolio management system that carries out dynamic asset allocation and generates sustainable performance. The said system, by design, is reactive to stock price fluctuations and can perform optimal portfolio rebalancing to suit prevailing market conditions. In Chapter 2 we derived a closed-form solution for multi-period asset allocation using HJB PDE and several other ideas from Merton et al [92, 93] and Ziemba et al [158]. The asset price dynamics were assumed to follow the GBM process. However, due to parameter estimation complexities and the fact that financial markets are complex systems where price dynamics are arduous to model, we consider exploring model-free RL as an alternative. Chapter 3 discusses MDP-based RL algorithms and concludes with a review of the RRL framework put forward by Moody et al [99, 100] specifically tailored to approximate solutions in the field of portfolio management. In our study, we adopt the RRL framework for the following reasons:

1. The RRL framework avoids Bellman's curse of dimensionality and is a policy-based method applicable to large continuous domains.
2. RRL creates a simple and elegant representation for multi-period optimal portfolio allocation.
3. An RRL's optimal policy is equivalent to optimal real-valued portfolio weights and offers more flexibility in choosing an objective function.
4. The framework is easily be adapted online by considering only recently realised portfolio returns.
5. RRL has a more stable performance compared to Q-learning when exposed to noisy datasets.

5.1 Research design

5.1.1 Overview

This section provides a formulation of the designed ARRL PM system as a portfolio management system. The system performs feature-learning and multi-period portfolio rebalancing in concert with stock price fluctuations in order to generate alpha.

5.1.2 State Space

The state space is denoted by the information set I shown in the decision function in Equation 3.12. At time t, the information set is given by:

$$I(t) = \{r(t), r(t-1), \ldots, r(t-k+1); y(t), y(t-1), y(t-2), \ldots\}, \tag{5.1}$$

where $\{r(t-i)\}_{i=0}^{k+1}$ is a set of k lagged risky asset log-returns and $\{y(t-i)\}_{i\geq 0}$ is a set which comprises an arbitrary number of external features. In our study, external features are technical indicators, and they are a function of stock price and the volume traded. Due to the recurrent nature of RRL, previous decisions have an influence of the current decision through the dependency of $\pi(t)$ and $R(t)$ on $\pi(t-1)$ as shown in Equations 3.12 and 3.14, respectively [88, 99, 100]. This influence is catered for by considering $\pi(t-1)$ to be part of the environment [58]. As a result, the state space is given by:

$$\mathbf{s}(t) = (I(t), \pi(t-1)) \in \mathcal{S}. \tag{5.2}$$

5.1.3 Action Space

In a portfolio composed of M risky assets, the action space, $\mathcal{A}$, is driven by a stochastic process:

$$(\pi(t)) \in [-1,1]^M \times \Re \; ; \; \pi(t) = (\pi_1(t), \ldots, \pi_M(t)).$$

The M-dimensional stochastic process $\pi(t)$ governs multi-period asset allocation as well as portfolio rebalancing as discussed in Chapter 3. In our study, the fund manager can buy, maintain neutral positions, or sell stocks. In order to cater for these action choices, a hyperbolic tangent function is used as ϕ in Equation 3.12 [98, 99]. Therefore, $\forall i = 1, \ldots, M$ and the parameter space Θ, the decision function is given by:

$$\pi_i(t) = \phi(\mathbf{s}(t)|\Theta) = \tanh\left(b + u\pi_i(t-1) + \theta I(t)\right) \in [-1,1] \times \Re. \tag{5.3}$$

The choice of training parameters $(\theta, b, u) \in \Theta$ with ϕ given by the hyperbolic tangent function result in the actual decision to buy, do-nothing, or sell a particular stock at time t:

$$a_i(t) = \text{sign}\left(\pi_i(t)\right) \in \{-1, +1\} \; \forall i = 1, \ldots, M,$$

where -1 and +1 indicates selling and buying of stock i at time t, respectively [36, 40, 81]. The scalar value of $\pi_i(t)$ denotes the preference or decision score that quantifies the decision or preference to invest in stock i at time t [58]. According to Almahdi et al [4], for a portfolio consisting of M stocks, the effective portfolio weights, $h_i(t)$ $\forall i = 1, \ldots, M$, are obtained by applying an exponential softmax function to the preference score as follows:

$$h_i(t) = \left(\sum_{j=1}^{M} e^{(\pi_j(t))}\right)^{-1} e^{(\pi_i(t))} \in [0,1].$$

The action with the highest preference score in each state $\mathbf{s}(t) \in \mathcal{S}$ gets assigned given the highest portfolio weight. For the case where the preference score is minute for asset i, the softmax function will distribute the weight among other assets, thereby effectively shifting weights between assets as we move forward with decisions [4]. Using the softmax function ensures that the budget constraint is satisfied:

$$\sum_{i=1}^{M} h_i(t) = \mathbf{h}^\top \cdot \mathbb{1} = 1.$$

The softmax function is recommended for a multi-asset portfolio as discussed by Moody et al [99, 100] and has been used in subsequent studies by Almahdi et al [4] and Jiang et al [58].

5.1.4 Reward Function

The reward signal, $R_i(t) \in \mathcal{R}$, is given by Equation 3.14. Transaction costs, given by δ, are assumed to be fixed at a value of 25 basis points. As discussed in Section 1.4, δ incorporates fixed commissions or similar explicit transaction charges associated with the buying or selling of stocks. The assumed value of δ is guided by annual equity brokerage fee publications from Nedbank Wealth[1]. The portfolio return at time t is given by:

$$R_p(t) = \sum_{i=1}^{M} h_i(t) R_i(t).$$

5.1.5 Feature Learning

In Chapter 4, we discussed several signal processing and feature extraction methods. For each asset $i = 1, \ldots, M$, the state space $\mathbf{s}_i(t) \in \mathcal{S}$ is given by Equation 5.2. The information set $I_i(t) \in \mathbf{s}_i(t)$ given by Equation 5.1 contains financial market features. Feature learning in our context is focused on enhancing the signal-to-noise ratio by distilling an asset's intrinsic information. Improving the signal-to-noise ratio protects the portfolio from suboptimal investment decisions. A feature learning algorithm ν is such that:

$$\nu : I_i \mapsto \tilde{I}_i \,, \quad \forall i = 1, \ldots, M, \tag{5.4}$$

where $\tilde{I}_i$ is the noise-free, compressed, or more informative version of I_i. By incorporating feature learning methods discussed in Chapter 4, the state space becomes:

$$\mathbf{s}(t) = (\nu(I(t)), \pi(t-1)) = \left(\tilde{I}(t), \pi(t-1)\right) \in \mathcal{S}. \tag{5.5}$$

5.1.6 ARRL System: Optimal Rebalancing

The performance or reinforcement function $U_i(T|\theta)$ is a function of a sequence of rewards shown as shown by Equation 3.15. The RRL portfolio optimisation framework is given by:

$$\begin{aligned} \max_{\theta \in \Theta} \quad & U_i\left(T|\theta\right) \\ \text{subject to:} \quad & \pi_i(t) = \tanh\left(\theta^\top \mathbf{s}_i(t)\right) \\ & R_i(t) = \pi_i(t-1) r_i(t) - \delta|\pi_i(t) - \pi_i(t-1)|. \end{aligned} \tag{5.6}$$

[1]https://www.nedbankprivatewealth.co.za/content/private-wealth-sa/south-africa/en/products-and-services/stockbroking/Ourstockbrokingfees.tmhl

In this study, the performance function $U(T|\theta)$ is given by the Sharpe Ratio (SR):

$$SR_i(T|\theta) = \frac{\frac{1}{T}\sum_{t=1}^{T} R_i(t)}{\sqrt{\frac{1}{T}\sum_{t=1}^{T} R_i^2(t) - \left(\frac{1}{T}\sum_{t=1}^{T} R_i(t)\right)^2}}.$$

SR is an ubiquitous safety-first ratio that measures the risk-adjusted return on a portfolio [2, 33, 132]. We opted for the SR as a performance measure because it is well-reputed and behaves like an adaptive quadratic utility function such as those discussed in Section 2.3.2 [2, 4, 99].

5.1.7 Policy Search Algorithm

The decision or policy function in Equation 5.3 is parameterised by $\theta \in \Theta$. Policy parameters are learned based on the gradient of the scalar performance measure given by SR. The optimal decision given by π_{θ^*} is obtained via $\theta^* = \arg\max_{\theta} SR(T|\theta)$. Parameters $\theta^* \in \Theta$ are estimated through stochastic gradient ascend:

$$\Delta\theta_i = \eta \nabla_\theta SR(T|\theta),$$

where η is the learning rate and $\nabla_\theta SR(T|\theta)$ is the differential SR [36, 81, 99].

The batch-mode RRL adapted for a multi-asset portfolio is shown in Algorithm 6 as follows:

Algorithm 6 ARRL System: Multi-asset batch-mode adapted from [30, 58, 98, 99].

```
1:  Input: Financial market information: $\tilde{I}(1),\dots,\tilde{I}(T)$          ▷ Given by Equation 5.4.
2:  for each asset i = 1 to M do
3:      Flat transaction costs: $\delta$
4:      Learning rate: $\eta$
5:      Initialise gradient: $g_i(0) = 0$
6:      Initialise policy parameters: $\theta_i \in \Theta = \{u, b, \theta\}$
7:      Initialise policy: $\pi_i(0) = 0$
8:      $\frac{d\pi_i(0)}{d\theta} = 0$
9:      $\mathbf{s}_i(0) = (\tilde{I}_i(0), \pi_i(0))$          ▷ Given by Equation 5.5.
10:     while $SR_i(T|\theta)$ not converged do
11:         while $t < T$ do          ▷ Loop through the training data.
12:             $t \leftarrow t + 1$
13:             $\mathbf{s}_i(t) \leftarrow \mathbf{s}_i(t+1)$          ▷ State $s(t) \in \mathcal{S}$.
14:             $\pi_i(t) \leftarrow \tanh(\theta_i^\top \mathbf{s}_i(t))$          ▷ Input a hyperbolic tangent policy parameterisation.
15:             $R_i(t) \leftarrow \pi_i(t-1)r_i(t) - \delta|\pi_i(t) - \pi_i(t-1)|$          ▷ Reward signal.
16:             $\frac{dR_i(t)}{d\pi_i(t)} \leftarrow -\delta \,\mathrm{sign}(\pi_i(t) - \pi_i(t-1))$
17:             $\frac{dR(t)}{d\pi(t-1)} \leftarrow r_i(t) - \delta \,\mathrm{sign}(\pi_i(t) - \pi_i(t-1))$
18:             $\frac{\pi_i(t)}{d\theta} \leftarrow \left(1 - \tanh^2(\theta_i^\top \mathbf{s}_i(t))\right)\left(\mathbf{s}_i(t) + u_t \frac{d\pi_i(t-1)}{d\theta}\right)$
19:             $\nabla_\theta SR(t|\theta) \leftarrow \frac{dR_i(t)}{d\pi_i(t)}\frac{d\pi_i(t)}{d\theta} + \frac{dR_i(t)}{d\pi_i(t-1)}\frac{d\pi_i(t-1)}{d\theta}$
20:             $g_i(t) \leftarrow g_i(t-1) + \nabla_\theta SR(t|\theta)$          ▷ Consolidate gradients.
21:         return $g_i(T) = \nabla_\theta SR_i(T|\theta)$
22:         $\Delta\theta_i \leftarrow \eta \nabla_\theta SR_i(T|\theta)$          ▷ Update $\theta$ iteratively using stochastic gradient ascent.
23:     return $\theta_i^* \in \Theta$
24: end
```

The maximisation displayed in Equation 5.6 is performed using Algorithm 6. Batch-mode RRL for training $\theta \in \Theta$ is adopted in our study to allow full exploitation. As a result, the ARRL PM system acts consistently with prior observations to maximise rewards. For testing and subsequent deployment, each asset $i = 1, \ldots, M$ flows independently into the ARRL PM system, which subsequently outputs preference scores used to calculate effective portfolio weights as discussed in Section 5.1.3.

5.2 Data

In this section, a review of data sources, features, and data preprocessing methods is provided.

5.2.1 Data Source

Top-performing stocks based on the highest trailing twelve months earnings-per-share (TTM EPS) across nine major economic sectors in South Africa are selected. The reason for using TTM EPS as an asset pre-selection criterion is that it is a proxy of a company's actual profitability[2]. Furthermore, the EPS ratio is correlated with trading volume, which implies greater market liquidity. TTM-EPS is based on actual figures rather than estimates. Thus using it as a pre-selection criterion means that EPS information just before the testing period (see Section 5.3.2) is considered in the asset selection process. This avoids passing future rankings to the test dataset, thereby mitigating the impact of survival bias on the results. This asset pre-selection technique is similar to the one implemented by Jiang et al [58] where the authors used eleven most-volumed non-cash assets for the portfolio. In their study, volume information just before the beginning of the back-tests was considered for pre-selection to avoid survival bias. Deng et al [30] pre-selected three futures contracts based on liquidity, while Almahdi et al [4] constructed a portfolio of the five most commonly traded exchange-traded funds from different asset categories. In our study, daily historical stock prices of the pre-selected equities are obtained from Bloomberg, and the data covers nine years starting from March 7th, 2011 to November 27th, 2020. Table 5.1 shows the stocks and corresponding economic sectors.

TABLE 5.1: Portfolio stocks and economic sectors

Company Name	**Symbol**	**Sector**
Anglo American Platinum	AMS	Non-Energy Mineral
Aspen Pharmacare Holdings	APN	Health Technology
Capitec Bank Holdings	CPI	Finance
Clicks Group	CLS	Retail Trade
Compagnie Fin Richemont	CFR	Consumer Durables
Exxaro Resources	EXX	Energy Minerals
Mondi	MNP	Process Industries
Naspers	NPN	Technology Services
Vodacom Group	VOD	Communications

[2]EPS is a portion of company's profits allocated to each outstanding share of stock

5.2.2 Features

Table 5.2 shows the technical indicators that form part of the information set. These features are computed for each of the stocks included in the analysis. These features have a broad coverage since they span across several groups of technical indicators. Technical indicators from the same group tend to be more or less correlated with each other. For our study, we considered conventional technical indicators from each of the predominant groups. The formulae to calculate these technical indicators are found in [7, 24, 45, 101, 150].

TABLE 5.2: Features: Financial market technical indicators

Feature Name	Description	Group
EMA	Exponential moving average	Overlap
HT_trendline	Hilbert transform-instantaneous trend-line	Overlap
KAMA	Kaufman adaptive moving average	Overlap
SAR	Parabolic SAR	Overlap
ADX	Average directional movement index	Momentum
CCI	Commodity channel index	Momentum
CMO	Chande momentum oscillator	Momentum
MACD	Moving average convergence / divergence	Momentum
MFI	Money flow index	Momentum
MOM	Momentum	Momentum
PPO	Percentage price indicator	Momentum
RSI	Relative strength index	Momentum
ADOSC	Chaikin A/D oscillator	Volume
OBV	On-balance volume	Volume
ATR	Average true range	Volatility

5.2.3 Data Pre-processing

5.2.3.1 Data Scaling

Most ML algorithms perform optimally when numerical features are scaled to a standard range, thereby avoiding dominance by certain other variables [97, 111]. For this research, we implement data standardization which centers each feature individually and divides the result by the standard deviation to shift the distribution to have a mean of zero and a unit standard deviation [111].

5.2.3.2 Data Stationarity

A time series is stationary if its essential statistical properties do not depend on time [153]. Data stationarity ensures that the overall behaviour of the time series data remains the same. Time series data is stationary if the mean, standard deviation, and autocorrelation are finite and constant [152]. Without stationarity, the interpretation of results from statistical inference becomes problematic [86]. A unit root test employing the augmented Dickey-Fuller (ADF) test is used to check for stationarity. ADF tests the null hypothesis that a time series has unit root [68, 152]. The Difference Transform, a technique that addresses temporal dependence, is used to eliminate trends and seasonal influences from time series that are non-stationary or have a unit root. The time series is examined once more to ensure that the transformation was successful. Results of the initial ADF tests performed on pre-transformed datasets are presented in Section A.1.

5.3 Methods

5.3.1 Instruments

Python programming software was used in the development of the proposed ARRL system for dynamic asset allocation. Python was chosen for a variety of reasons:

1. Python is an open-source programming language.
2. Python has a variety of built-in classes that integrates data cleaning and subsequent model training.
3. Python has a user interface that is easy to understand and allows Jupyter Notebook configuration.
4. Nodes and objects for Tensorflow are Python objects, thus making Python more suitable for deep-learning.

5.3.2 Training, Validation and Test Split

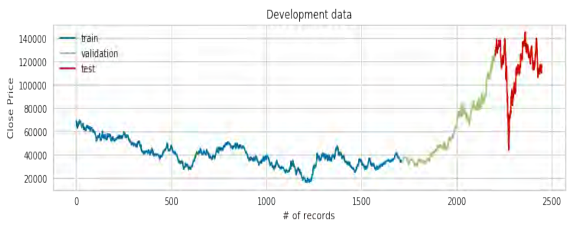
The data is partitioned into 70% training data, 20% validation data, and 10% testing. The period covered by each dataset as well as the number of records as a result of these split proportions is summarised in Table 5.3 as follows:

TABLE 5.3: Data preparation: Train, Validation and Test.

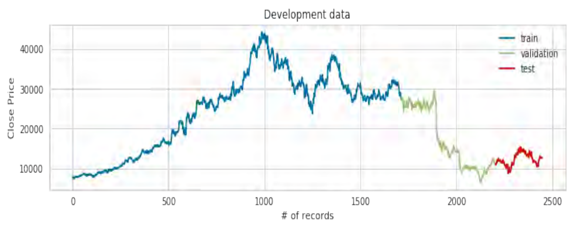
Data	Period Covered	Size	Purpose
Train	2011-03-07/2017-12-19	1,714	Model training over multiple epochs.
Validation	2017-12-20/2019-12-05	490	Validation and hyperparameter tuning.
Test	2019-12-06/2020-11-27	245	Performance evaluation.

The aforementioned data partitioning is based on standard data science techniques for evaluating model efficacy and efficiency. Deng at al [30], Gold [40] and Kanwar [61] carried out a similar data partitioning for their analysis.

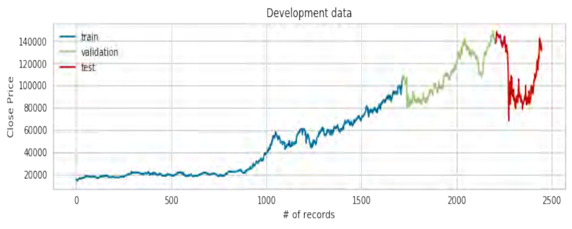
Figure 5.1 illustrates the historical price of portfolio stocks for each split period:



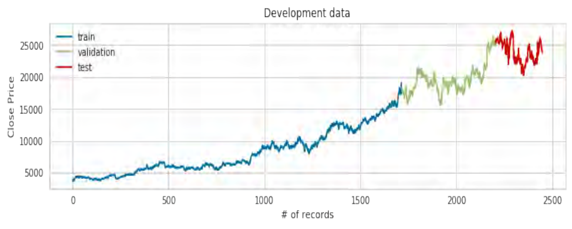
(A) Anglo American Platinum (AMS)



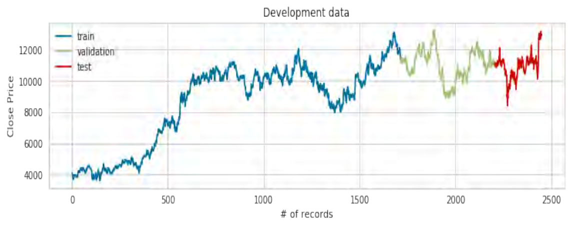
(B) Aspen Pharmacare Holdings (APN)



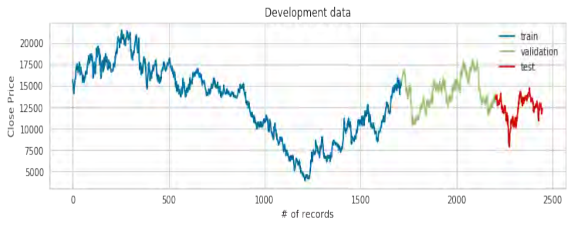
(C) Capitec Bank Holdings (CPI)



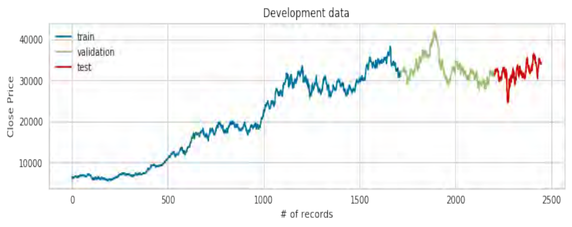
(D) Clicks Group (CLS)



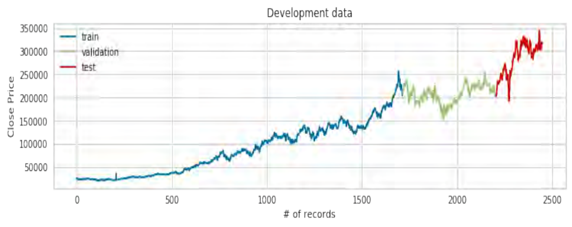
(E) Compagnie Fin Richemont (CFR)



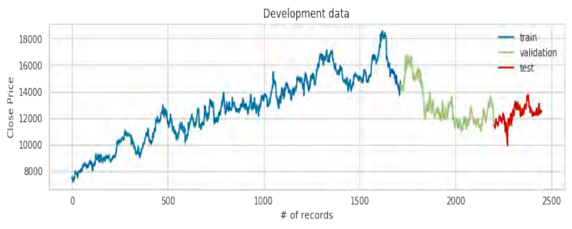
(F) Exxaro Resources (EXX)



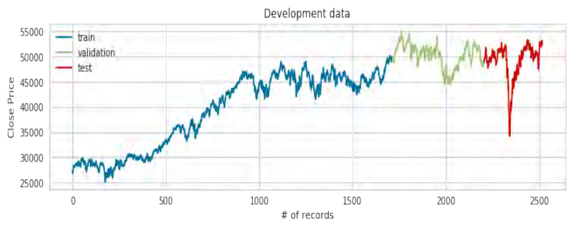
(G) Mondi (MNP)



(H) Naspers (NPN)



(I) Vodacom Group (VOD)



(J) Johannesburg Top 40 (SA40)

FIGURE 5.1: Historical time series data

5.4 Analysis

5.4.1 Portfolio Strategies

Table 5.4 lists and describes portfolio strategies considered for our analysis.

TABLE 5.4: Analysis: Portfolio Strategies.

Portolio Name	**Strategy**	**Description**
SA40	Benchmark	South African Top 40 Index.
Equally-Weighted (EW)	Passive	All equitities have equal weight positions.
Value-weighted (VW)	Passive	Market capitalisation-weighted portfolio.
ARRL	Active	Uses RRL and no feature learning.
ARRL-ica	Active	Uses ICA for feature extraction and RRL PM
ARRL-mssa	Active	Uses MSSA for signal denoising and RRL PM.
ARRL-sdae	Active	Uses SDAE with drop-out and RRL PM.
ARRL-cdae	Active	Uses CDAE for latent signal extraction and RRL PM.
ARRL-lstm-dae	Active	Uses LSTM DAE for latent signal extraction and RRL PM.

The performance benchmark for our analysis is the returns on the SA40 index. The SA40 index consists of forty of South Africa's largest market capitalisation-weighted corporations [129]. An equally-weighted (EW) portfolio employs a classic portfolio management strategy that distributes AUM equitably among portfolio equities regardless of price fluctuations [120]. For a portfolio of M equities, the assigned portfolio weight for each asset $i = 1, \ldots, M$ is given by:

$$h_i(t) = \frac{1}{M}.$$

A value-weighted (VW) portfolio uses a passive investment strategy that entails assigning weights to portfolio assets based on their relative market capitalisation [156]. Portfolio weights for constituent equities are given by:

$$h_i(t) = \frac{k_i(t)}{\sum_{j=1}^{M} k_j(t)},$$

where k_i is market capitalisation for portfolio asset i. Stocks with a larger market capitalisation are given a higher weighting in the portfolio, whereas stocks with a lower market capitalisation are given a lower weighting [116]. Active strategy portfolios listed in Table 5.4 use RRL to perform dynamic asset allocation. They only differ based on the feature learning method used, starting with ARRL that does not use any feature learning. The remaining active strategies use feature learning methods discussed in Chapter 4.

5.4.2 Evaluation Metrics

Details of the training process are given in Sections A.2 and A.3. Traditional performance measures are employed in our analysis to assess the out-of-sample performance of the designed portfolios.

5.4.2.1 Terminal Portfolio Value

The terminal portfolio value (TPV) after T periods is given by:

$$\Pi(T) = \Pi(0) \prod_{t=1}^{T} e^{R_p(t)} \; ; \; \Pi(0) = \Pi_0,$$

where Π_0 is the value of AUM at inception and $R_p(t)$ is portfolio return at time t [52, 152]. We assume $\Pi_0 = 1$ so that TPV is the portfolio value per the South African Rand (ZAR) amount invested at time $t = 0$.

5.4.2.2 Sharpe Ratio

SR is discussed in Section 5.1.6.

5.4.2.3 Sortino Ratio

Sortino Ratio (STR) is a performance measure that considers asymmetry in returns. STR uses downside risk as a measure of risk. The volatility of negative returns is used to calculate the downside risk [122, 136].

5.4.2.4 Maximum Draw-down

A maximum drawdown (MDD) is the maximum loss experienced by a portfolio from its peak to its trough before a new peak is reached. MDD is an indicator of downside risk over a specified time frame.

5.4.2.5 Information Ratio

Information ratio (IR), also known as appraisal ratio, calculates average excess return per unit of additional risk with respect to the benchmark [6]. It is a ratio of average excess returns to tracking-error (TE):

$$IR = \frac{(\mu_p - \mu_b)}{TE} \; ; \; TE = vol(R_p - R_b),$$

where μ_p and μ_b are average returns of the portfolio and its benchmark, respectively and $vol(R_p - R_b)$ is the volatility of excess returns.

Chapter 6

Results and Discussion

6.1 Results and Discussion

6.1.1 Performance Results

Table 6.1 summarises the out-of-sample final portfolio value per Rand amount invested evaluated by TPV and risk-adjusted-returns given by SR, STR, and IR.

TABLE 6.1: Out-of-sample: Risk-adjusted Return

Portfolio Name	TPV	SR	STR	IR
SA40	1.0912	0.0184	0.0211	-
EW	1.0774	0.0160	0.0196	-0.0068
VW	1.2630	0.0492	0.0573	0.0793
ARRL	1.9141	0.1314	0.1516	0.2404
ARRL-ica	**2.4997**	**0.1953**	**0.2481**	**0.3597**
ARRL-mssa	2.1270	0.1407	0.1767	0.2510
ARRL-sdae	1.5457	0.0840	0.1020	0.1344
ARRL-cdae	1.3159	0.0545	0.0731	0.0756
ARRL-lstm-dae	1.8677	0.1327	0.2013	0.2047

The highlighted values in the results Table 6.1 show the performance of the winning portfolio strategy. ARRL-ica consistently out-perform all other portfolios in terms of risk-adjusted returns across all the evaluation metrics considered. The least performing actively managed portfolio is ARRL-cdae. However, it is marginally better than the VW portfolio, the best passively-managed portfolio in our study. A majority of actively-managed portfolios appear to out-perform all passive portfolios in the study.

Figure 6.1 is the TPV profile that illustrates the performance of the portfolios over time for the out-of-sample dataset:

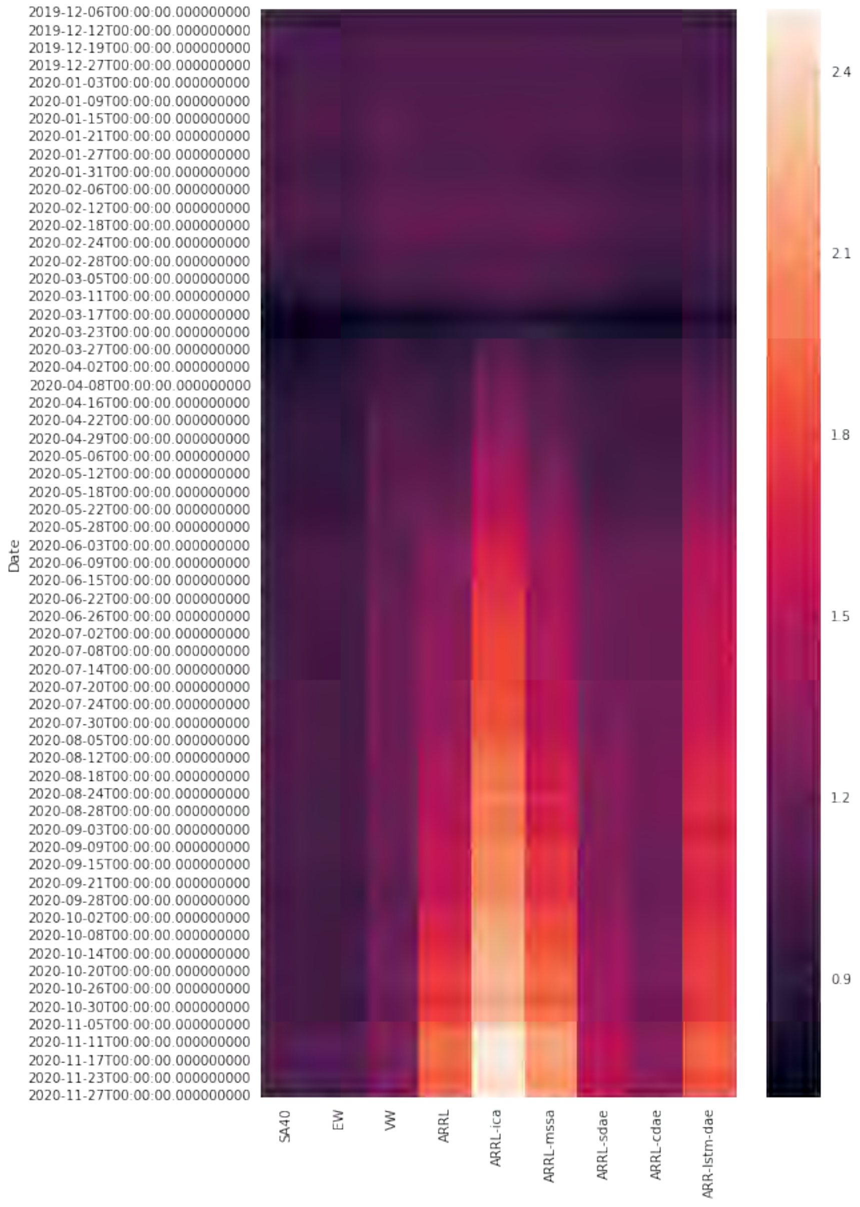


FIGURE 6.1: Out-of-sample: Portfolio performance over time

Figure 6.1 confirms that all portfolios experienced a market contraction towards the end of the first quarter (Q1) of 2020. However, as the TPV profile shows, most actively managed portfolios rallied following the 2020 Q1 market downturn.

6.1.2 Risk Assessement

In this section, we use MDD as a risk indicator to assess the impact of the 2020 Q1 market contraction on the portfolios. According to MDD, the ARRL-lstm-dae portfolio has the least downside risk, followed by ARRL-ica. These portfolios were able to protect upwards of 5% and 10% equity compared to other portfolios and the benchmark, respectively, during the 2020 Q1 market contraction, as illustrated in Table 6.2:

TABLE 6.2: Out-of-sample: Maximum Drawdown

Portolio Name	MDD
SA40	-35.0738%
EW	-32.2155%
VW	-32.4622%
ARRL	-31.0016%
ARRL-ica	**-24.2326%**
ARRL-mssa	-30.9984%
ARRL-sdae	-31.0002%
ARRL-cdae	-30.8870%
ARRL-lstm-dae	**-23.4751%**

Figure 6.2 shows the out-of-sample drawdown profiles, with severe MDDs noticeable in the earlier part of the period.

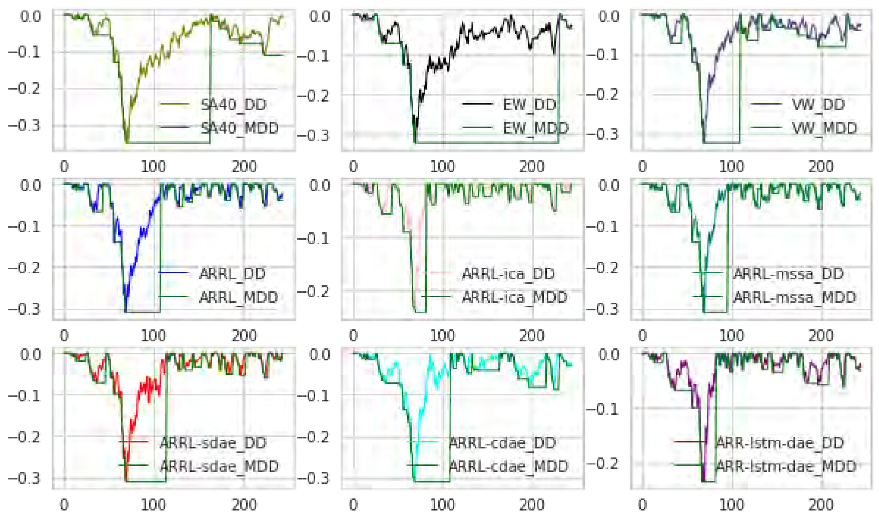


FIGURE 6.2: Out-of-sample: Portfolio drawdowns over time

6.1.3 Asset Allocation and Return Distribution

Table 6.3 shows the out-of-sample asset mix comparison. The actual portfolio re-balancing or adjustments during the out-of-sample period for actively-managed portfolios are in Section A.4. The asset allocations indicated in Table 6.3 for actively managed portfolios illustrate the average asset allocations across the out-of-sample period.

TABLE 6.3: Out-of-sample: Average Asset Allocations

Portfolio Name	AMS	APN	CPI	CLS	CFR	EXX	MNP	NPN	VOD
VW	12.20	2.05	5.06	2.14	22.58	1.20	3.38	**44.85**	6.55
ARRL	12.73	10.35	10.88	9.90	9.79	11.87	10.86	**12.96**	10.66
ARRL-ica	9.47	14.43	10.00	11.81	10.91	5.44	12.62	10.00	**15.31**
ARRL-mssa	12.75	10.02	10.90	9.92	10.07	11.83	10.72	**12.98**	10.80
ARRL-sdae	11.74	11.76	10.53	11.39	7.23	9.91	12.64	10.80	**14.00**
ARRL-cdae	11.85	10.49	9.37	12.12	11.80	**13.74**	9.92	11.40	9.30
ARRL-lstm-dae	10.87	**12.76**	8.75	12.52	8.54	12.04	9.58	12.24	12.70

Considering that actively managed portfolios permit the buying, holding, or selling of stocks in response to daily stock price fluctuations, the average asset allocations are well-spread across the economic sectors. In addition, these average allocations appear to be closer to an EW portfolio that distributes AUM equally among portfolio equities.

Table 6.4 shows the results of the Wilcoxon Signed-Rank Test performed on the null hypothesis, H_0, that portfolio returns' distribution (shown in Section A.5) is the same as that of the benchmark.

TABLE 6.4: Out-of-sample: Wilcoxon Signed-Rank Test

Portolio Name	p-value	Wilcoxon Signed Rank Test
EW	0.8173	Same distribution (fail to reject H_0)
VW	0.3543	Same distribution (fail to reject H_0)
ARRL	0.0000	Different distribution (reject H_0)
ARRL-ica	0.0000	Different distribution (reject H_0)
ARRL-mssa	0.0000	Different distribution (reject H_0)
ARRL-sdae	0.0066	Different distribution (reject H_0)
ARRL-cdae	0.3905	Same distribution (fail to reject H_0)
ARRL-lstm-dae	0.0004	Different distribution (reject H_0)

6.2 Summary

The top three portfolios according to the performance evaluations and risk are ARRL-ica, ARRL-mssa, and ARRL-lstm-dae. ARRL-mssa is the most risky of the three, and that is consistent with the observation that MSSA effectively isolates noise from the original input. Although performance improves relative to the ARRL portfolio, there is a marginal improvement in terms of risk.

In the first quarter of 2020, all portfolios experienced double-digit losses. Only a few RRL-based portfolios were able to provide considerable equity protection relative to passive portfolios. The observation is consistent with the study by Molina [98]. The study concludes that RRL-learners are unable to predict precipitous stock price decline. Our findings suggest that by applying signal processing, the severity of drawdowns can be reduced.

Overall, our analysis shows that actively managed portfolios outperform passively-managed portfolios and the benchmark, albeit marginally for ARRL-cdae. The outperformance is corroborated in most previous studies [4, 30, 36, 81, 100]. Furthermore, most actively managed portfolios have return distributions that differ significantly from the benchmark, supporting the benefits of combining RRL with signal processing to generate excess returns or alpha.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

An RRL-based portfolio management system is adopted to perform unsupervised optimal portfolio rebalancing. Various signal processing approaches such as ICA, MSSA, and denoising autoencoders are applied to the RRL module, resulting in a hybrid dynamic asset allocation system, referred to as ARRL, that extracts latent features from multi-channel market inputs. Furthermore, the system directly outputs portfolio weights and is scalable with portfolio size. The study demonstrates that the proposed ARRL system outperforms all the surveyed classical portfolio management strategies in terms of the final accumulated portfolio value and risk-adjusted returns. More precisely, the best-performing portfolio applies ICA to the RRL-model, and it attained a 2.5-fold growth during the out-of-sample period. Moreover, the test results show that two of the best top-performing portfolios achieved significant equity savings relative to the benchmark during the market contraction experienced during the first quarter of 2021.

7.2 Future Work

Further exploration includes applying other novel feature-learning methods such as the Wavelets Transformation. Wavelets Transformation is a decomposition-based method that converts complex information into elementary forms [134]. Additionally, a regime-switching framework proposed by Maringer et al [88] can be incorporated to cater for sharp stock price declines. Sentiment analysis and text-timing of a live news feed from social media and Bloomberg or Reuters offer another channel of valuable market information. This information and fundamental data potentially summarise stock's intrinsic value much better, thereby improving the learning process. Furthermore, the analysis can be broadened to include a larger pool of investable assets. Not only does this mitigate firm-specific risk, but it also gives the system more freedom to move across shares within an available investible universe. For instance, the ARRL system can be tested on the top 100 (by market capitalisation) of the 350 shares listed on the JSE. The empirical test could also be conducted using a rolling window basis that would dynamically update model parameters.

Appendix A

Learning Graphics

A.1 Augmented Dickey-Fuller Test

Table A.1 shows the p-values of the features identified as non-stationary at 5% significance level.

TABLE A.1: Features: Augmented Dickey-Fuller Unit Root Test

Feature Name	AMS	APN	CPI	CLS	CFR	EXX	MNP	NPN	VOD
Close price	0.822	0.564	0.901	0.969	0.341	0.247	0.636	0.987	0.087
EMA	0.804	0.544	0.883	0.971	0.401	0.309	0.622	0.991	0.098
HT_trendline	0.880	0.527	0.887	0.977	0.403	0.409	0.645	0.992	0.105
KAMA	0.888	0.562	0.891	0.977	0.349	0.440	0.601	0.991	0.136
SAR	0.678	0.502	0.893	0.963	0.245	0.348	0.614	0.981	0.088
OBV	0.243	0.933	-	0.442	0.561	0.231	0.536	0.413	0.960
ATR	0.361	-	0.157	0.339	-	-	0.117	0.765	-

A.2 Training Artifical Neural Networks

Hyperparameter tuning is performed using Scikit-Learn's Randomized Grid-Search. The number of layers in deep neural networks, the number of neurons in each layer for SDAEs, feature maps for CDAEs, regularisation values, dropout probability, and learning rates are some of the parameters adjusted during the training process. A learning rate schedule is applied to yield optimal performance. The activation function of choice is the Scaled Exponential Linear Unit (SELU) that results in self-normalizing neural networks to enable high-level abstract representation [64, 82]. According to Klambauer et al [64], the self-normalizing property ensures that the output of each layer preserves a mean of zero and unit standard deviation during training, which solves the vanishing or exploding gradient problem. The LeCun initializer is employed to mitigate the issue of unstable gradients [108]. The Adaptive Moment Estimation (ADAM) is chosen as the optimizer because it combines the benefits of Momentum and Root Mean Square Propagation (RMSProp). ADAM determines autonomous adaptive learning rates for various parameters by computing the first and second-moment estimates of the gradient [63, 154]. Also, an early-stopping time during training is implemented to save computational time and mitigate over-fitting.

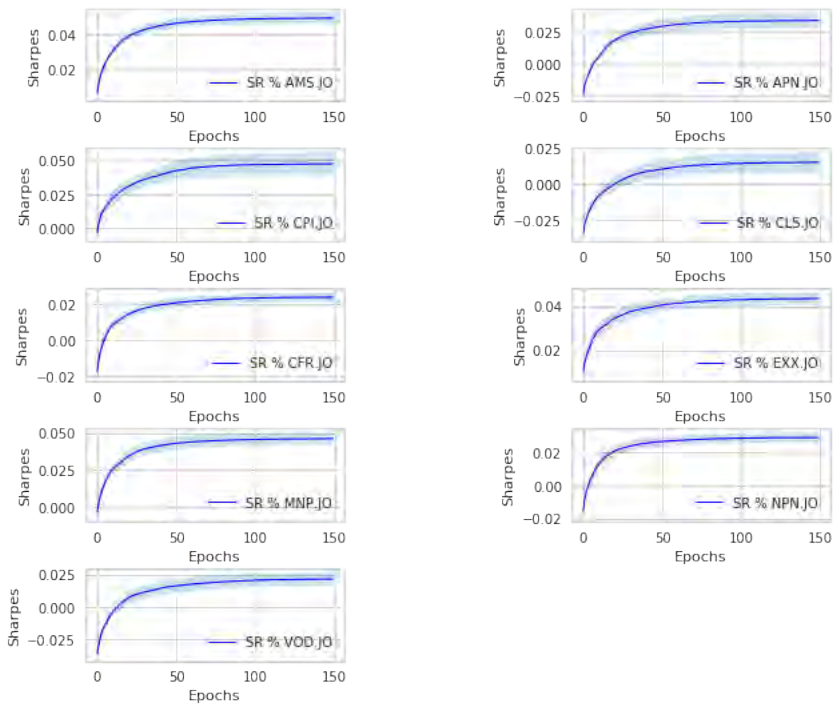
A.3 Training Results

RRL-based PM systems are trained using Algorithm 6. Model training is performed on the training dataset for 150 epochs. Parameter estimates are averaged over 30 trials. Table A.2 shows the 95% confidence interval of the training SR for each asset in the portfolio.

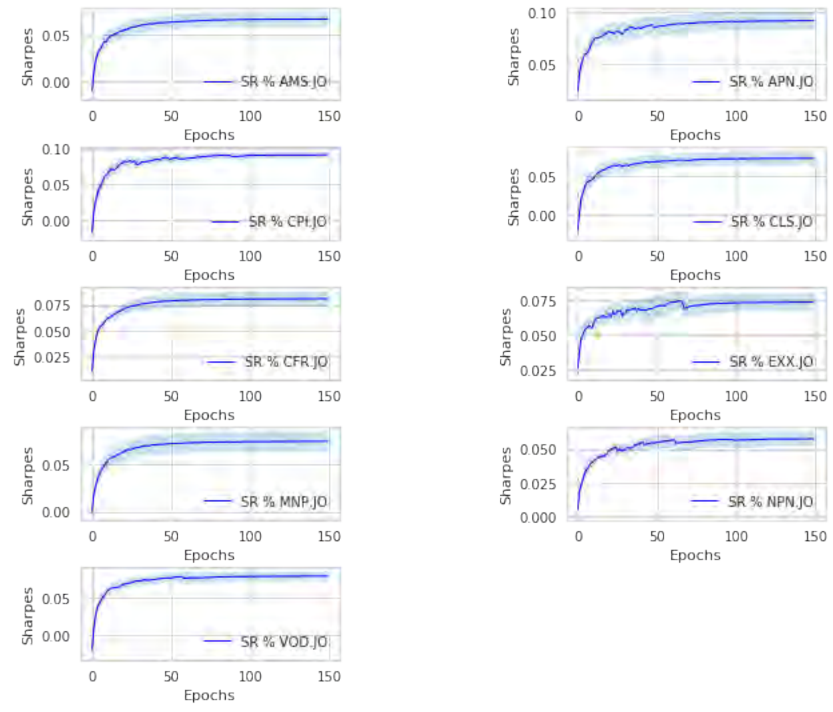
TABLE A.2: Confidence Interval of Training Sharpe Ratio (%)

Stocks	**ARRL**	**ARRL-ica**	**ARRL-mssa**	**ARRL-sdae**	**ARRL-cdae**	**ARRL-lstm-dae**
AMS	4.94 ± 0.32	6.60 ± 0.81	5.47 ± 0.42	3.63 ± 0.68	2.76 ± 0.54	4.66 ± 0.89
APN	3.37 ± 0.62	9.18 ± 0.89	3.31 ± 0.38	4.05 ± 0.82	4.58 ± 0.75	4.46 ± 0.72
CPI	4.74 ± 0.83	9.04 ± 0.48	4.72 ± 0.83	4.79 ± 0.37	4.79 ± 0.56	3.77 ± 1.25
CLS	1.52 ± 0.67	7.19 ± 0.90	1.07 ± 0.50	3.99 ± 1.00	4.70 ± 0.27	4.15 ± 1.02
CFR	2.36 ± 0.25	8.10 ± 0.73	2.41 ± 0.31	3.36 ± 0.80	4.56 ± 0.57	3.80 ± 0.64
EXX	4.35 ± 0.27	7.36 ± 0.65	4.66 ± 0.36	4.31 ± 1.04	2.81 ± 0.64	3.72 ± 0.56
MNP	4.60 ± 0.41	7.37 ± 1.07	5.14 ± 0.40	2.77 ± 1.03	3.14 ± 0.68	4.90 ± 0.83
NPN	2.89 ± 0.30	5.74 ± 0.57	3.20 ± 0.39	2.26 ± 0.48	5.36 ± 0.68	2.59 ± 0.58
VOD	2.15 ± 0.51	7.76 ± 0.66	2.48 ± 0.62	3.59 ± 0.85	4.94 ± 1.11	5.64 ± 1.10

The confidence intervals are much wide for artificial neural networks. This is due in part to the fact that deep neural networks require substantial hyperparameter tuning and that there is a trade-off between performance and simplicity. Gold [40] alludes that there is a large number of fixed parameters that need to be tuned by trial and error when training deep RRL; there is no fixed set of parameters that can reliably be deemed optimal. Overall, performance is stable, and optimal SR is obtained within 150 epochs as indicated in the graphs below:

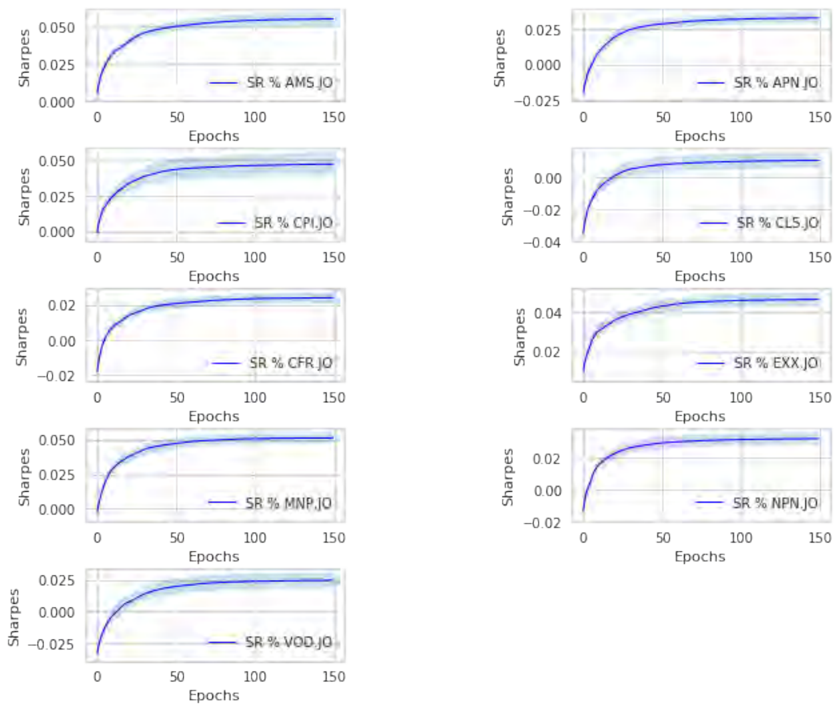


(A) ARRL training curve

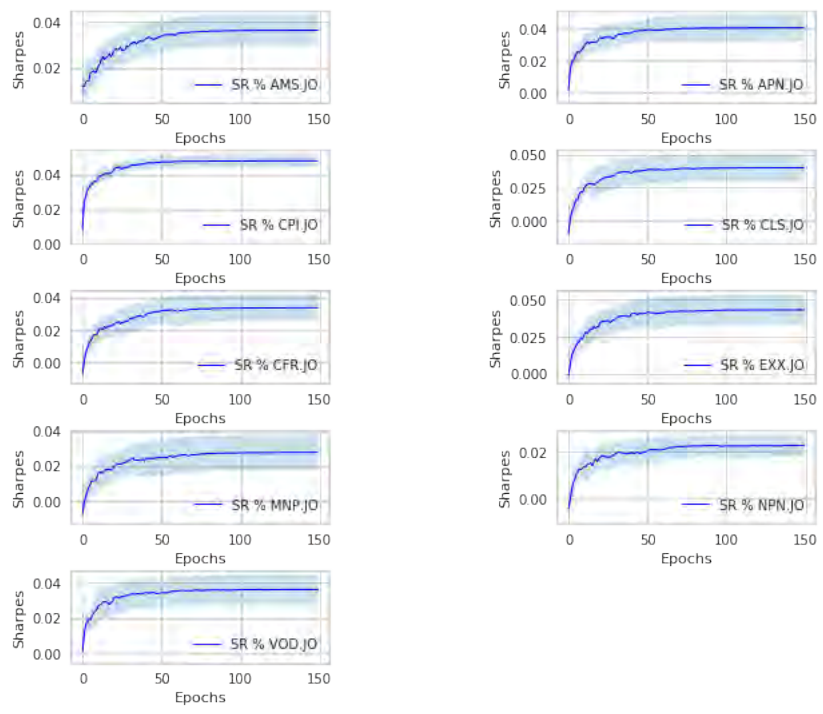


(B) ARRL-ica training curve

FIGURE A.1: Training curves for ARRL and ARRL-ica

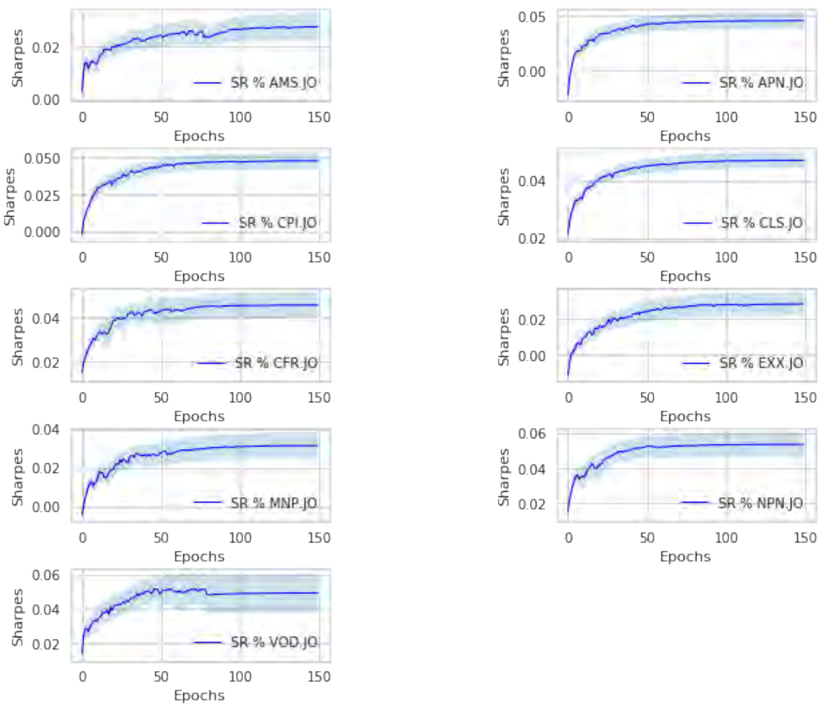


(A) ARRL-mssa training curve

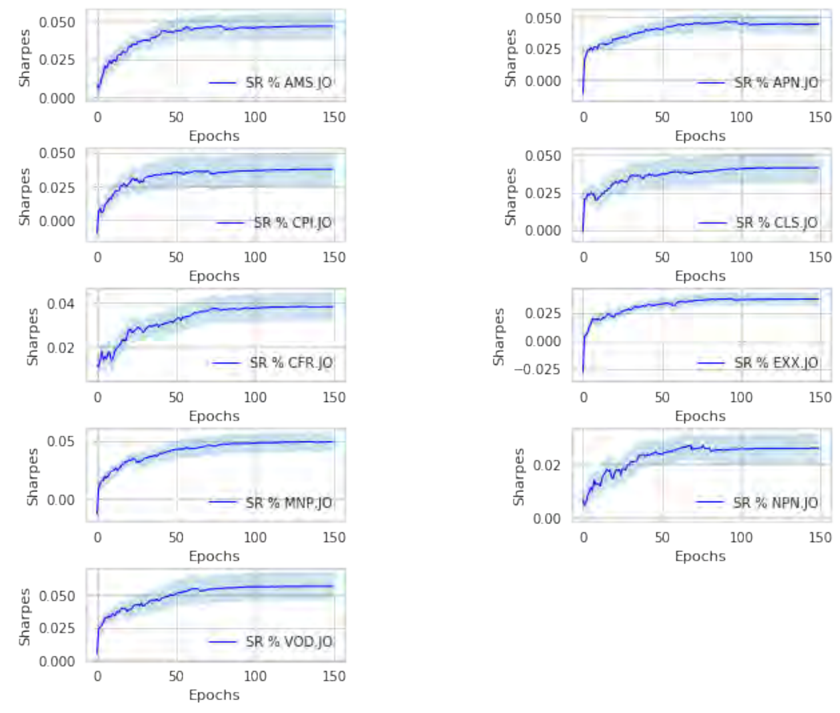


(B) ARRL-sdae training curve

FIGURE A.2: Training curves for ARRL-mssa and ARRL-sdae



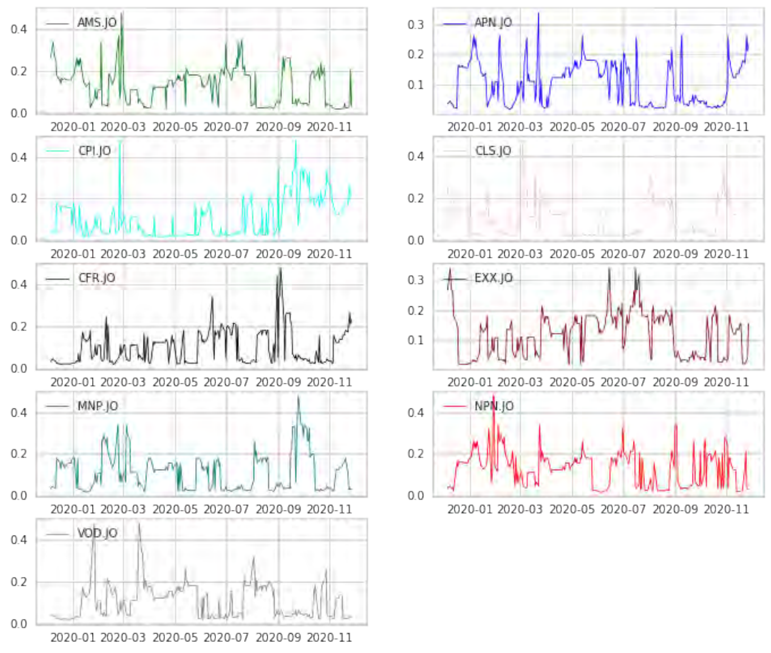
(A) ARRL-cdae training curve



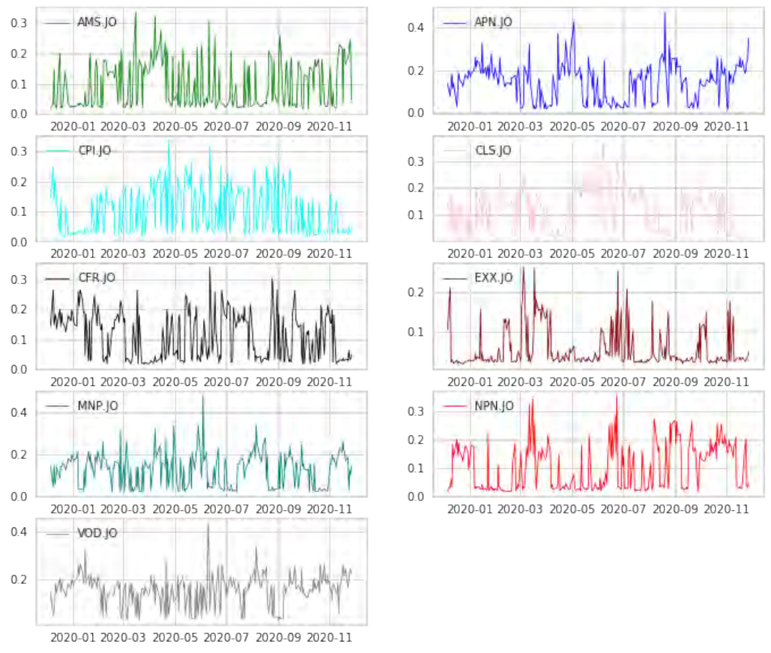
(B) ARRL-lstm-dae training curve

FIGURE A.3: Training curves for ARRL-mssa and ARRL-sdae

A.4 Out-of-sample Dynamic Asset Allocation

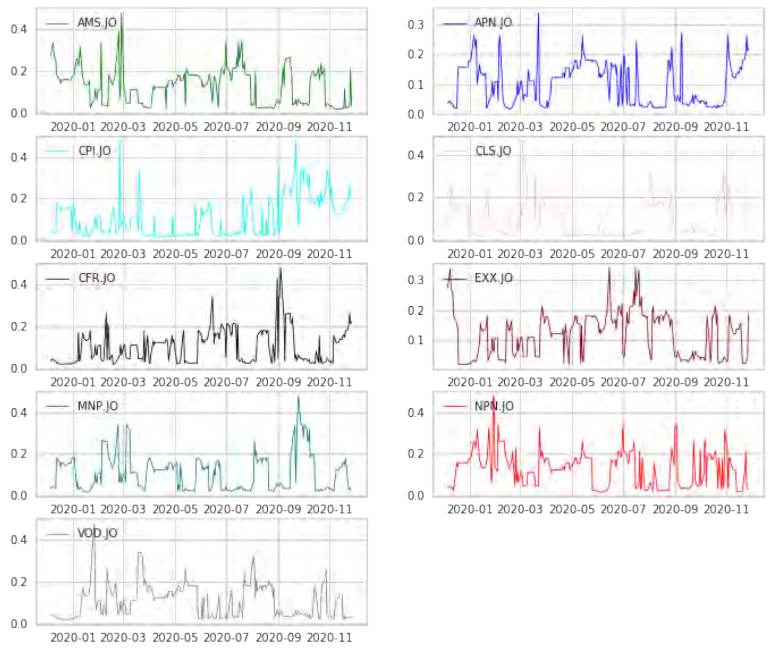


(A) Out-of-sample: ARRL asset allocations

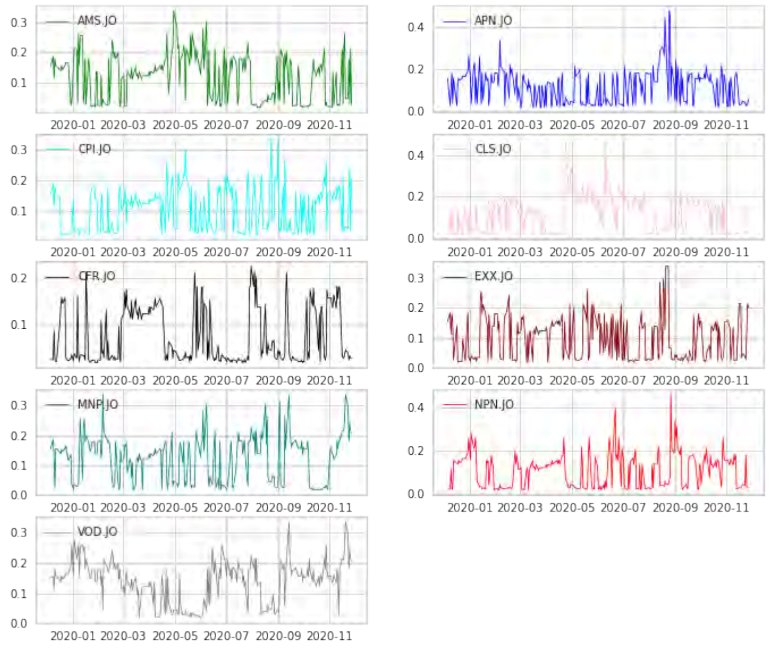


(B) Out-of-sample: ARRL-ica asset allocations

FIGURE A.4: Dynamic asset allocation for ARRL and ARRL-ica funds

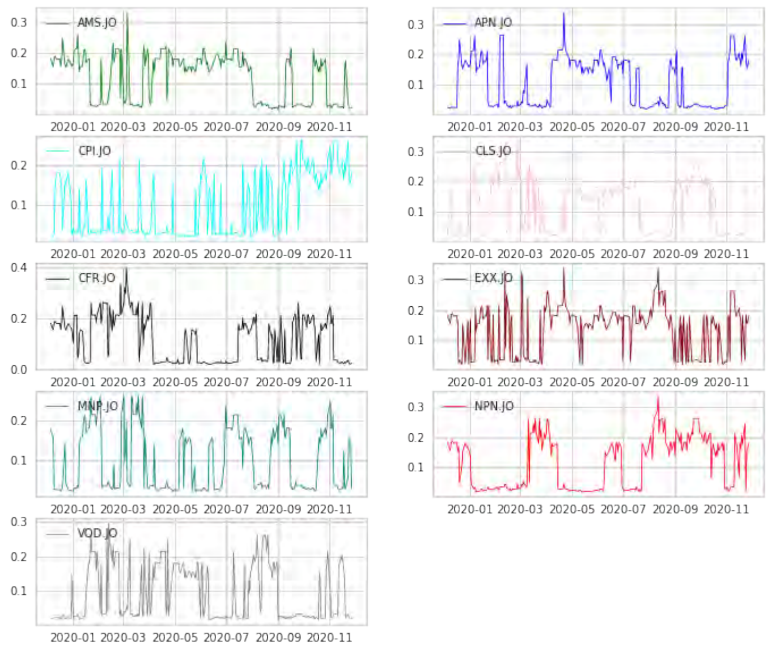


(A) Out-of-sample: ARRL-mssa asset allocations

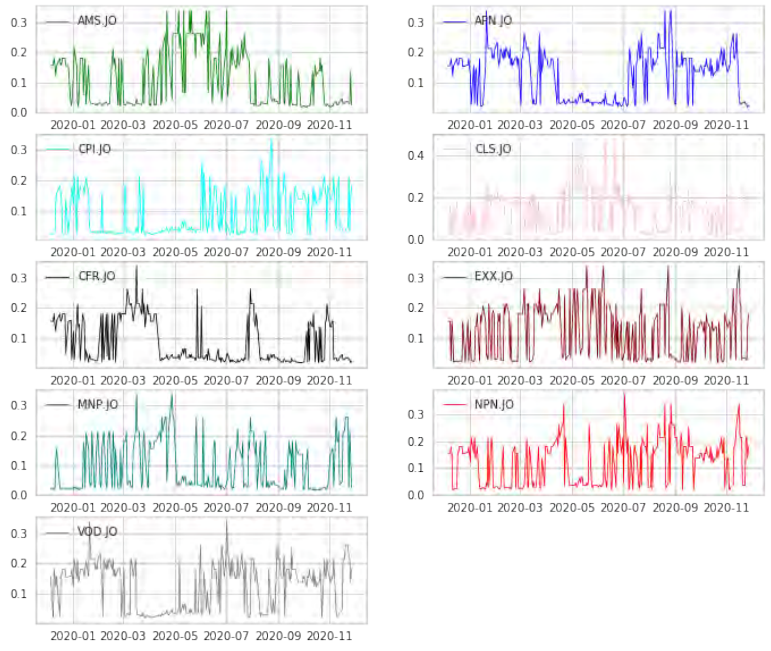


(B) Out-of-sample: ARRL-sdae asset allocations

FIGURE A.5: Dynamic asset allocation for ARRL-mssa and ARRL-sdae funds



(A) Out-of-sample: ARRL-cdae asset allocations



(B) Out-of-sample: ARRL-lstm-dae asset allocations

FIGURE A.6: Dynamic asset allocation for ARRL-cdae and ARRL-lstm-dae funds

A.5 Portfolio Return Distribution

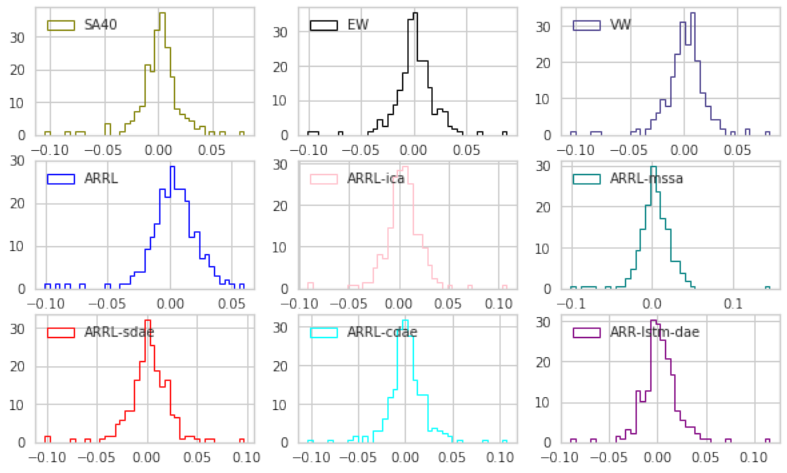


FIGURE A.7: Out-of-sample: Portfolio returns distribution

Bibliography

[1] *A long peek into reinforcement learning*. https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html.

[2] Christopher Adcock et al. "Portfolio performance persistence: does the choice of performance measure matter?" In: (2019).

[3] Tolulope Akinbulire et al. "A reinforcement learning approach to tackle illegal, unreported and unregulated fishing". In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2017, pp. 1–8.

[4] Saud Almahdi and Steve Y Yang. "An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown". In: *Expert Systems with Applications* 87 (2017), pp. 267–279.

[5] Shun-ichi Amari, Andrzej Cichocki, and Howard Hua Yang. "A new learning algorithm for blind signal separation". In: *Advances in neural information processing systems*. 1996, pp. 757–763.

[6] M.J.P Anson, D.R Chambers, and K.H Black H Kazemi. *CAIA Level 1, An Introduction to Core Topics in Alternative Investments*. 2013.

[7] Gerald Appel and W Frederick Hitschler. *Stock market trading systems*. Irwin Professional Pub, 1980.

[8] Mahsa Asadi et al. "Model-Based Reinforcement Learning Exploiting State-Action Equivalence". In: *Asian Conference on Machine Learning*. PMLR. 2019, pp. 204–219.

[9] Andrew D Back and Andreas S Weigend. "A first application of independent component analysis to extracting structure from stock returns". In: *International journal of neural systems* 8.04 (1997), pp. 473–484.

[10] Andrew D Back and Andreas S Weigend. "What Drives Stock Returns?-An Independent Component Analysis". In: *Proceedings of the IEEE/IAFE/INFORMS 1998 Conference on Computational Intelligence for Financial Engineering (CIFEr)(Cat. No. 98TH8367)*. IEEE. 1998, pp. 141–156.

[11] Taimur Baig and Ilan Goldfajn. "Financial market contagion in the Asian crisis". In: *IMF staff papers* 46.2 (1999), pp. 167–195.

[12] Wei Bao, Jun Yue, and Yulei Rao. "A deep learning framework for financial time series using stacked autoencoders and long-short term memory". In: *PloS one* 12.7 (2017), e0180944.

[13] Martino Bardi and Italo Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Springer Science & Business Media, 2008.

[14] Dirk Becherer. "Rational hedging and valuation of integrated risks under constant absolute risk aversion". In: *Insurance: Mathematics and economics* 33.1 (2003), pp. 1–28.

[15] Frank Beichelt. *Stochastic processes in science, engineering and finance*. CRC Press, 2006.

[16] Geert Bekaert et al. "The global crisis and equity market contagion". In: *The Journal of Finance* 69.6 (2014), pp. 2597–2649.

[17] Richard Bellman. "A Markovian decision process". In: *Journal of mathematics and mechanics* (1957), pp. 679–684.

[18] Richard Bellman. "A Markovian Decision Process". In: *Indiana Univ. Math. J.* 6 (4 1957), pp. 679–684. ISSN: 0022-2518.

[19] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[20] Pavlo R Blavatskyy. "Back to the St. Petersburg paradox?" In: *Management Science* 51.4 (2005), pp. 677–678.

[21] Steffen Bohn. "The slippage paradox". In: *arXiv preprint arXiv:1103.2214* (2011).

[22] Steven J Bradtke and Michael O Duff. "Reinforcement learning methods for continuous-time Markov decision problems". In: *Advances in neural information processing systems*. 1995, pp. 393–400.

[23] Jerome R Busemeyer. "Dynamic decision making". In: (1999).

[24] Tushar S Chande and Stanley Kroll. *The new technical trader: boost your profit by plugging into the latest indicators*. Vol. 44. John Wiley & Sons Incorporated, 1994.

[25] Yiu-ming Cheung and Lei Xu. "Independent component ordering in ICA time series analysis". In: *Neurocomputing* 41.1-4 (2001), pp. 145–152.

[26] Francois Chollet. *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG, 2018.

[27] Vijay K Chopra and William T Ziemba. "The effect of errors in means, variances, and covariances on optimal portfolio choice". In: *Handbook of the Fundamentals of Financial Decision Making: Part I*. World Scientific, 2013, pp. 365–373.

[28] John C Cox and Chi-fu Huang. "Optimal consumption and portfolio policies when asset prices follow a diffusion process". In: *Journal of economic theory* 49.1 (1989), pp. 33–83.

[29] Mark HA Davis and Sébastien Lleo. *Risk-sensitive investment management*. Vol. 19. World Scientific, 2014.

[30] Yue Deng et al. "Deep direct reinforcement learning for financial signal representation and trading". In: *IEEE transactions on neural networks and learning systems* 28.3 (2016), pp. 653–664.

[31] Elroy Dimson, Paul Marsh, and Mike Staunton. "Long-run global capital market returns and risk premia". In: *Available at SSRN 299335* (2002).

[32] I Capuzzo Dolcetta and M Falcone. "Discrete dynamic programming and viscosity solutions of the Bellman equation". In: *Annales de l'Institut Henri Poincare (C) Non Linear Analysis*. Vol. 6. Elsevier. 1989, pp. 161–183.

[33] Kevin Dowd. "Adjusting for risk:: An improved Sharpe ratio". In: *International review of economics & finance* 9.3 (2000), pp. 209–222.

[34] *Drawdowns*. https://www.investopedia.com/terms/d/drawdown.asp.

[35] Bo Du et al. “Stacked convolutional denoising auto-encoders for feature representation”. In: *IEEE transactions on cybernetics* 47.4 (2016), pp. 1017–1027.

[36] Xin Du, Jinjian Zhai, and Koupin Lv. “Algorithm trading using q-learning and recurrent reinforcement learning”. In: *positions* 1 (2016), p. 1.

[37] Ward Edwards. “Dynamic decision theory and probabilistic information processings”. In: *Human factors* 4.2 (1962), pp. 59–74.

[38] Aniekan Essien and Cinzia Giannetti. “A deep learning framework for univariate time series prediction using convolutional LSTM stacked autoencoders”. In: *2019 IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA)*. IEEE. 2019, pp. 1–6.

[39] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O’Reilly Media, 2019.

[40] Carl Gold. “FX trading via recurrent reinforcement learning”. In: *2003 IEEE International Conference on Computational Intelligence for Financial Engineering, 2003. Proceedings.* IEEE. 2003, pp. 363–370.

[41] Nina Golyandina, Vladimir Nekrutkin, and Anatoly A Zhigljavsky. *Analysis of time series structure: SSA and related techniques*. CRC press, 2001.

[42] Nina Golyandina et al. “Multivariate and 2D extensions of singular spectrum analysis with the Rssa package”. In: *arXiv preprint arXiv:1309.5050* (2013).

[43] Lovedeep Gondara. “Medical image denoising using convolutional denoising autoencoders”. In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2016, pp. 241–246.

[44] Emad M Grais and Mark D Plumbley. “Single channel audio source separation using convolutional denoising autoencoders”. In: *2017 IEEE global conference on signal and information processing (GlobalSIP)*. IEEE. 2017, pp. 1265–1269.

[45] Joseph Ensign Granville. *Granville’s new strategy of daily stock market timing for maximum profit*. Prentice-Hall, 1976.

[46] Jiang Guo. “Backpropagation through time”. In: *Unpubl. ms., Harbin Institute of Technology* 40 (2013), pp. 1–6.

[47] Hossein Hassani. “Singular spectrum analysis: methodology and comparison”. In: (2007).

[48] Hossein Hassani and Dimitrios Thomakos. “A review on singular spectrum analysis for economic and financial time series”. In: *Statistics and its Interface* 3.3 (2010), pp. 377–397.

[49] Simon S Haykin et al. *Neural networks and learning machines/Simon Haykin.* 2009.

[50] Geoffrey E Hinton and Richard S Zemel. “Autoencoders, minimum description length and Helmholtz free energy”. In: *Advances in neural information processing systems*. 1994, pp. 3–10.

[51] Grace Xing Hu, Jun Pan, and Jiang Wang. “Noise as information for illiquidity”. In: *The Journal of Finance* 68.6 (2013), pp. 2341–2382.

[52] John Hull et al. *Options, futures and other derivatives/John C. Hull.* Upper Saddle River, NJ: Prentice Hall, 2009.

[53] J Wesley Hutchinson and Robert J Meyer. “Dynamic decision making: Optimal policies and actual behavior in sequential choice problems”. In: *Marketing Letters* 5.4 (1994), pp. 369–382.

[54] Aapo Hyvärinen. “Survey on independent component analysis”. In: (1999).

[55] Aapo Hyvärinen and Erkki Oja. “Independent component analysis: algorithms and applications”. In: *Neural networks* 13.4-5 (2000), pp. 411–430.

[56] Jonathan E Ingersoll Jr. “Portfolio separation theorems”. In: *Theory of financial decision making* (1987).

[57] Stuart Jarvis, Adrian Lawrence, and Sheng Miao. “Dynamic asset allocation techniques”. In: *British Actuarial Journal* 15.3 (2009), pp. 573–655.

[58] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. “A deep reinforcement learning framework for the financial portfolio management problem”. In: *arXiv preprint arXiv:1706.10059* (2017).

[59] Olivier Jin and Hamza El-Saawy. “Portfolio management using reinforcement learning”. In: *Stanford University* (2016).

[60] M Chris Jones and Robin Sibson. “What is projection pursuit?” In: *Journal of the Royal Statistical Society: Series A (General)* 150.1 (1987), pp. 1–18.

[61] Nitin Kanwar et al. “Deep Reinforcement Learning-Based Portfolio Management”. PhD thesis. 2019.

[62] Ioannis Karatzas, John P Lehoczky, and Steven E Shreve. “Optimal portfolio and consumption decisions for a “small investor” on a finite horizon”. In: *SIAM journal on control and optimization* 25.6 (1987), pp. 1557–1586.

[63] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[64] Günter Klambauer et al. “Self-normalizing neural networks”. In: *Advances in neural information processing systems*. 2017, pp. 971–980.

[65] Marek Andrzej Kociński et al. “On transaction costs in stock trading”. In: *Metody Ilościowe w Badaniach Ekonomicznych* 18.1 (2017), pp. 58–67.

[66] Vijay R Konda and John N Tsitsiklis. “Actor-critic algorithms”. In: *Advances in neural information processing systems*. Citeseer. 2000, pp. 1008–1014.

[67] Mukesh Kumar et al. “Feature selection and classification of microarray data using MapReduce based ANOVA and K-nearest neighbor”. In: *Procedia Computer Science* 54 (2015), pp. 301–310.

[68] Denis Kwiatkowski et al. “Testing the null hypothesis of stationarity against the alternative of a unit root”. In: *Journal of econometrics* 54.1-3 (1992), pp. 159–178.

[69] Sascha Lange and Martin Riedmiller. “Deep auto-encoder neural networks in reinforcement learning”. In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2010, pp. 1–8.

[70] Dominic Langlois, Sylvain Chartier, and Dominique Gosselin. “An introduction to independent component analysis: InfoMax and FastICA algorithms”. In: *Tutorials in Quantitative Methods for Psychology* 6.1 (2010), pp. 31–38.

[71] Marc Levinson. “Guide to financial markets, London”. In: *The Economist* (2005).

[72] Yuxi Li. “Deep reinforcement learning: An overview”. In: *arXiv preprint arXiv:1701.07274* (2017).

[73] Zhipeng Liang et al. “Adversarial deep reinforcement learning in portfolio management”. In: *arXiv preprint arXiv:1808.09940* (2018).

[74] Zhipeng Liang et al. *Deep reinforcement learning in portfolio management*. 2018.

[75] Liang Jin and M. M. Gupta. “Stable dynamic backpropagation learning in recurrent neural networks”. In: *IEEE Transactions on Neural Networks* 10.6 (1999), pp. 1321–1334. DOI: 10.1109/72.809078.

[76] Timothy P Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).

[77] Steven A Lippman, John J McCall, and Wayne L Winston. “Constant absolute risk aversion, bankruptcy, and wealth-dependent decisions”. In: *Journal of Business* (1980), pp. 285–296.

[78] Jun Liu, Francis A Longstaff, and Jun Pan. “Dynamic asset allocation with event risk”. In: *The Journal of Finance* 58.1 (2003), pp. 231–259.

[79] Xingchen Liu et al. “Fault diagnosis of rotating machinery under noisy environment conditions based on a 1-D convolutional autoencoder and 1-D convolutional neural network”. In: *Sensors* 19.4 (2019), p. 972.

[80] Chi-Jie Lu, Tian-Shyug Lee, and Chih-Chou Chiu. “Financial time series forecasting using independent component analysis and support vector regression”. In: *Decision support systems* 47.2 (2009), pp. 115–125.

[81] David W Lu. “Agent inspired trading using recurrent reinforcement learning and lstm neural networks”. In: *arXiv preprint arXiv:1707.07338* (2017).

[82] Avinash Madasu and Vijjini Anvesh Rao. “Effectiveness of self normalizing neural networks for text classification”. In: *arXiv preprint arXiv:1905.01338* (2019).

[83] Jeff Madura. “Financial Markets and Instruments”. In: *Thomson South-Western* (2006).

[84] Myles E Mangram. “A simplified perspective of the Markowitz portfolio theory”. In: *Global journal of business research* 7.1 (2013), pp. 59–70.

[85] Steven V Mann, Frank J Fabozzi, and Moorad Choudhry. *The Global Money Markets*. 2002.

[86] Radu Manuca and Robert Savit. “Stationarity and nonstationarity in time series analysis”. In: *Physica D: Nonlinear Phenomena* 99.2-3 (1996), pp. 134–161.

[87] Hongzi Mao et al. “Resource management with deep reinforcement learning”. In: *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. 2016, pp. 50–56.

[88] Dietmar Maringer and Tikesh Ramtohul. “Regime-switching recurrent reinforcement learning for investment decision making”. In: *Computational Management Science* 9.1 (2012), pp. 89–107.

[89] Harry M Markowitz. “Investment for the long run: New evidence for an old rule”. In: *The Journal of Finance* 31.5 (1976), pp. 1273–1286.

[90] RC Merton. “Optimum consumption and portfolio-rules in a continuous-time framework”. In: *Journal of Economic Theory (December 1971)* (1971).

[91] Robert C Merton. “Lifetime portfolio selection under uncertainty: The continuous-time case”. In: *The review of Economics and Statistics* (1969), pp. 247–257.

[92] Robert C Merton. “Theory of rational option pricing”. In: *The Bell Journal of economics and management science* (1973), pp. 141–183.

[93] Robert C Merton and Paul Anthony Samuelson. “Continuous-time finance”. In: (1992).

[94] Attilio Meucci. *Risk and asset allocation*. Springer Science & Business Media, 2009.

[95] Hyman P Minsky. “The financial instability hypothesis”. In: *The Jerome Levy Economics Institute Working Paper* 74 (1992).

[96] Tom M Mitchell et al. *Machine learning*. 1997.

[97] Tom Michael Mitchell. *The discipline of machine learning*. Vol. 9. Carnegie Mellon University, School of Computer Science, Machine Learning . . ., 2006.

[98] Gabriel Molina. “Stock Trading with Recurrent Reinforcement Learning (RRL)”. In: *CS229, nd Web* 15 (2016).

[99] John Moody et al. “Performance functions and reinforcement learning for trading systems and portfolios”. In: *Journal of Forecasting* 17.5-6 (1998), pp. 441–470.

[100] John E Moody and Matthew Saffell. “Reinforcement learning for trading”. In: *Advances in Neural Information Processing Systems*. 1999, pp. 917–923.

[101] Patrick G Mulloy. “Smoothing data with less lag”. In: *Technical Analysis of Stocks and Commodities* 12.2 (1994), pp. 72–80.

[102] John J Murphy. *Intermarket technical analysis: trading strategies for the global stock, bond, commodity, and currency markets*. Vol. 6. John Wiley & Sons, 1991.

[103] Marek Musiela and Thaleia Zariphopoulou. “Portfolio choice under dynamic investment performance criteria”. In: *Quantitative Finance* 9.2 (2009), pp. 161–170.

[104] Ralph Neuneier. “Optimal asset allocation using adaptive dynamic programming”. In: *Advances in Neural Information Processing Systems*. 1996, pp. 952–958.

[105] Andrew Ng et al. “Sparse autoencoder”. In: *CS294A Lecture notes* 72.2011 (2011), pp. 1–19.

[106] Richard E Oberuc. *Dynamic portfolio theory and management: using active asset allocation to improve profits and reduce risk*. McGraw Hill Professional, 2004.

[107] Erkki Oja, Kimmo Kiviluoto, and Simona Malaroiu. “Independent component analysis for financial time series”. In: *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. IEEE. 2000, pp. 111–116.

[108] Genevieve B Orr and Klaus-Robert Müller. *Neural networks: tricks of the trade*. Springer, 2003.

[109] Joel Owen and Ramon Rabinovitch. “On the class of elliptical distributions and their applications to the theory of portfolio choice”. In: *The Journal of Finance* 38.3 (1983), pp. 745–752.

[110] Xinlei Pan et al. “Virtual to real reinforcement learning for autonomous driving”. In: *arXiv preprint arXiv:1704.03952* (2017).

[111] Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python”. In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[112] Shige Peng. “A generalized dynamic programming principle and Hamilton-Jacobi-Bellman equation”. In: *Stochastics: An International Journal of Probability and Stochastic Processes* 38.2 (1992), pp. 119–134.

[113] Shige Peng. “Stochastic hamilton–jacobi–bellman equations”. In: *SIAM Journal on Control and Optimization* 30.2 (1992), pp. 284–304.

[114] Danilo Filippo Reiszel Pereira, Natanael Nunes de Moura Junior, and Luiz Pereira Caloba. "Financial Time Series Forecasting Using Non-Linear Methods and Stacked Autoencoders". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2018, pp. 1–8.

[115] Andre F Perold and William F Sharpe. "Dynamic strategies for asset allocation". In: *Financial Analysts Journal* 44.1 (1988), pp. 16–27.

[116] Yuliya Plyakha, Raman Uppal, and Grigory Vilkov. "Why does an equal-weighted portfolio outperform value-and price-weighted portfolios?" In: *Available at SSRN 2724535* (2012).

[117] Florian Pusse and Matthias Klusch. "Hybrid online pomdp planning and deep reinforcement learning for safer self-driving cars". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 1013–1020.

[118] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[119] Xin-Yao Qian and Shan Gao. "Financial series prediction: Comparison between precision of time series models and machine learning methods". In: *arXiv preprint arXiv:1706.00948* (2017).

[120] Frank K Reilly and Keith C Brown. *Investment analysis and portfolio management*. Cengage Learning, 2011.

[121] Izabela Rejer and Pawel Górski. "Benefits of ICA in the case of a few channel EEG". In: *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2015, pp. 7434–7437.

[122] Thomas N Rollinger and Scott T Hoffman. "Sortino: a 'sharper'ratio". In: *Chicago, IL: Red Rock Capital.* (2013).

[123] Sheldon M Ross. *Introduction to stochastic dynamic programming*. Academic press, 2014.

[124] Stephen Ross. "The arbitrage pricing theory". In: *Journal of Economic Theory* 13.3 (1976), pp. 341–360.

[125] Stephen A Ross. "Mutual fund separation in financial theory—the separating distributions". In: *Theory of Valuation*. World Scientific, 2005, pp. 309–356.

[126] Srikanth Ryali et al. "Development, validation, and comparison of ICA-based gradient artifact reduction algorithms for simultaneous EEG-spiral in/out and echo-planar fMRI recordings". In: *Neuroimage* 48.2 (2009), pp. 348–361.

[127] Alaa Sagheer and Mostafa Kotb. "Unsupervised pre-training of a Deep LStM-based Stacked Autoencoder for Multivariate time Series forecasting problems". In: *Scientific Reports* 9.1 (2019), pp. 1–16.

[128] Guillermo Sahonero-Alvarez and Humberto Calderón. "A comparison of SOBI, FastICA, JADE and Infomax algorithms". In: *Proceedings of the 8th International Multi-Conference on Complexity, Informatics and Cybernetics*. 2017, pp. 17–22.

[129] *SAT40*. `https://www.bloomberg.com/quote/TOP40:IND`.

[130] Yoshiharu Sato. "Model-Free Reinforcement Learning for Financial Portfolios: A Brief Survey". In: *arXiv preprint arXiv:1904.04973* (2019).

[131] William F Sharpe. "Capital asset prices: A theory of market equilibrium under conditions of risk". In: *The journal of finance* 19.3 (1964), pp. 425–442.

[132] William F Sharpe. "The sharpe ratio". In: *Journal of portfolio management* 21.1 (1994), pp. 49–58.

[133] Alex Shlemov and Nina Golyandina. "Shaped extensions of singular spectrum analysis". In: *arXiv preprint arXiv:1401.4980* (2014).

[134] M Sifuzzaman, MR Islam, and MZ Ali. "Application of Wavelet Transform and its Advantages Compared to Fourier Transform". In: *Journal of Physical Sciences* 13 (2009), pp. 121–134.

[135] Marinko Škare and Małgorzata Porada-Rochoń. "Multi-channel singular-spectrum analysis of financial cycles in ten developed economies for 1970–2018". In: *Journal of Business Research* 112 (2020), pp. 567–575.

[136] Frank A Sortino and Lee N Price. "Performance measurement in a downside risk framework". In: *the Journal of Investing* 3.3 (1994), pp. 59–64.

[137] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[138] Richard S Sutton et al. "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in neural information processing systems*. 2000, pp. 1057–1063.

[139] Randall Swift. "Stochastic Processes in Science, Engineering and Finance by Frank Beichelt-Chapmann & Hall/CRC (2006)". In: *Journal of Statistical Theory and Practice* 1.2 (2007), pp. 285–287.

[140] Lance Taylor and Stephen A O'Connell. "A Minsky crisis". In: *The Quarterly Journal of Economics* 100.Supplement (1985), pp. 871–885.

[141] Dimitrios D Thomakos, Tao Wang, and Luc T Wille. "Modeling daily realized futures volatility with singular spectrum analysis". In: *Physica A: Statistical Mechanics and its Applications* 312.3-4 (2002), pp. 505–519.

[142] James Tobin. "Liquidity preference as behavior towards risk". In: *The review of economic studies* 25.2 (1958), pp. 65–86.

[143] Gavin Tsang, Jingjing Deng, and Xianghua Xie. "Recurrent Neural Networks for Financial Time-Series Modelling". In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE. 2018, pp. 892–897.

[144] Pascal Vincent et al. "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 1096–1103.

[145] Pascal Vincent et al. "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." In: *Journal of machine learning research* 11.12 (2010).

[146] Shanshan Wang et al. "Order-free Medicine Combination Prediction with Graph Convolutional Reinforcement Learning". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2019, pp. 1623–1632.

[147] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[148] Laurens Weijs. "Reinforcement learning in Portfolio Management and its interpretation". In: *Erasmus Universiteit Rotterdam* (2018).

[149] Paul J Werbos. “Backpropagation through time: what it does and how to do it”. In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[150] J Welles Wilder. *New concepts in technical trading systems*. Trend Research, 1978.

[151] Ronald J Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[152] Paul Wilmott. *Paul Wilmott on quantitative finance*. John Wiley & Sons, 2013.

[153] Annette Witt, Jürgen Kurths, and A Pikovsky. “Testing stationarity in time series”. In: *physical Review E* 58.2 (1998), p. 1800.

[154] Jing Yuan and Ying Tian. “An Intelligent Fault Diagnosis Method Using GRU Neural Network towards Sequential Data in Dynamic Processes”. In: *Processes* 7.3 (2019), p. 152.

[155] Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. “Online incremental feature learning with denoising autoencoders”. In: *Artificial intelligence and statistics*. 2012, pp. 1453–1461.

[156] Mengying Zhu et al. “Adaptive Portfolio by Solving Multi-armed Bandit via Thompson Sampling”. In: *arXiv preprint arXiv:1911.05309* (2019).

[157] Shu-Shang Zhu, Duan Li, and Shou-Yang Wang. “Risk control over bankruptcy in dynamic portfolio selection: A generalized mean-variance formulation”. In: *IEEE transactions on Automatic Control* 49.3 (2004), pp. 447–457.

[158] William T Ziemba et al. *The stochastic programming approach to asset, liability, and wealth management*. Citeseer, 2003.