DTU Compute
Department of Applied Mathematics and Computer Science

Deep Reinforcement Learning For Portfolio Optimisation

Master Thesis

Deep Reinforcement Learning For Portfolio Optimisation

Master Thesis
June, 2022

Dimosthenis Karafylias

Copyright: Reproduction of this publication in whole or in part must include the customary bibliographic citation, including author attribution, report title, etc.
Cover photo: Vibeke Hempler, 2012
Published by: DTU, Department of Applied Mathematics and Computer Science, Richard Petersens Plads, 324, DK-2800 Kgs. Lyngby, Denmark
www.compute.dtu.dk

Acknowledgements

I would like to thank my supervisor Kourosh Marjani Rasmussen for all of the insightful and constructive contributions he readily offered during the time of this project.

I am eternally grateful to my parents and family for their unconditional care and love all these years. Their truly tremendous support made this endeavour possible.

Lastly, I would be remiss in not mentioning my friends for their motivation and for being understanding.

Thank you all.

Abstract

Economic applications oftentimes involve decision making under highly uncertain conditions. As an economic application, portfolio optimisation fits this profile. Recent advancements in Machine Learning, have been of a great benefit to the Financial Engineering field and have motivated the usage of Reinforcement Learning as a method to solve decision making in complex environments. This work evaluates Reinforcement Learning in portfolio optimisation tasks and examines its application both in the cryptocurrency and the stock market. Reinforcement Learning is leveraged as a solution to the problem of optimal redistribution of wealth in a selected subset of assets, so to maximise a long-term return. A daily reallocation is considered as a convenient time frame, which also capitalises on frequent changes of prices. Being highly volatile, the cryptocurrency market can be exploited for gains on those windows. Out of a possible universe of assets, the choices are limited to a well diversified subset by employing the MST (Minimum Spanning Tree) as a method that searches for minimal dependencies between assets.

Learning through reinforcement is the art of learning how to perform optimal reallocations, by trial and error; that is by trying good and bad actions and by exploring different re-allocations that may potentially lead to larger returns. As an added advantage of convenience, in a model free approach, there is no need of a financial market model. To this end, Twin Delayed Deep Deterministic Policy Gradient (TD3) is adapted and modified to fit an portfolio optimisation task. Certain alterations in TD3 allow for a more stable performance, overall, than the previous Deterministic Policy Gradient variants. Being able to discover market dynamics and learn the structure of a trading environment, model-free reinforcement learning is deemed successful as a trading agent. Its out-of-sample performance is demonstrated in comparison with the Standard and Poor's 500 and Dow Jones Industrial Average indexes. The analysis will characterise the performance by the achieved profit and the Sharpe Ratio.

Glossary of Terms

AI Artificial Intelligence. 2, 24

ANN Artificial Neural Network. 40

API Application Programming Interface. 6

CAPM Capital Asset Pricing Model. 18

CML Capital Market Line. 18

DSR Differential Sharpe ratio. 44

ETF Exchange-Traded Fund. 5, 7, 9, 12, 48, 50

EWP Equally weighted Portfolio. 12

MDP Markov Decision Process. 26–28, 32, 42

MPT Modern Portfolio Theory. 15, 43

MST Minimum Spanning Tree. 20, 21, 51

RRL Recurrent Reinforcement Learning. 41

TD3 Twin Delayed Deep Deterministic policy gradient algorithm. 35, 40, 42, 46, 47, 51–53

USD United States Dollar. 6

Notation

Financial Terms and Statistics

σ_x	Standard deviation of a variable $\mathcal{X}$
σ_x^2	Variance of a variable $\mathcal{X}$
m^l	l_{th} statistical moment of a variable $\mathcal{X}$
$R_{i,t}$	Gross return of the i_{th} asset at time t
$r_{i,t}$	Log return of the i_{th} asset at time t
$R_{p,t}$	Gross return of a portfolio p at time t
$r_{p,t}$	Log return of a portfolio p at time t
$w_{i,t}$	Portfolio weight of the i_{th} asset at time t

Reinforcement Learning

γ	Discount factor, $0 < \gamma \leq 1$
$\mu_\theta(s)$	Deterministic policy under network parameters θ
$\pi_\theta(a\|s)$	Stochastic policy under network parameters θ, given a state s and action a
$a \in \mathcal{A}$	Actions
G_t	Return, $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
$P(s, r\|s', a)$	Transition probability of getting to the next state s' from the current state s with action a and reward r
$Q(s, a), q$	Action-value function; how good an action is under a given state
$r \in \mathcal{R}$	Rewards
$s \in \mathcal{S}$	States
S_t, A_t, R_t	State, action, and reward at time step t of one trajectory
$V(s), u$	State-value function; measures the expected return of a state s

Contents

Chapter 1

Introduction

"A good portfolio is not only a good selection of stocks and bonds, but it also is a balanced whole that provides the investor with protections and opportunities with respect to a wide range of contingencies" [Markowitz, 1968]. Modern financial portfolio optimisation emerges as an extension of Harry M. Markowitz's work on diversification of investments. While heavily borrowing principles from his early work, state-of-the-art methods channel this financial knowledge into more technology exposed domains.

1.1 Research Question

This report investigates the hypothesis that reinforcement learning, as a completely agnostic solution, that learns by trial and error, is able to model sequential decision making in portfolio optimisation. The task involves daily reallocation of assets' weights, in a fixed portfolio composition, while under a fixed budget, so to maximise financial returns. The portfolio constituents will be selected in a precedent node reduction step by the graph-inspired Minimum Spanning Tree method. The performance will be subsequently tested in an out-of-sample forecast performance manner, on a portfolio composition that includes cryptocurrencies, stocks and ETFs.

1.2 Motivation

Reinforcement learning, a main paradigm of Machine Learning was popularised by recently witnessed breakthroughs such as with AlphaGo defeating the European Go champion [Silver, A. Huang et al., 2016], Deep Q-Networks that surpass human expert performance on Atari games [Mnih et al., 2015] and DeepStack beating, with statistical significance, professional poker players in heads-up no-limit Texas hold'em [Moravcík et al., 2017]. Unlike other Machine Learning paradigms, reinforcement learning holds the notion of own-learning, from scratch, by trial and error actions. Reinforcement Learning has recently found potential playgrounds in Robotics and autonomous driving vehicles, Computer Vision and image recognition, in Healthcare and personalised medicine, in Financial Engineering and portfolio optimisation among others.

Of all the forms of machine learning, reinforcement learning is the closest to the kind of learning that humans and other animals do. Many of the core algorithms of reinforcement learning

were originally inspired by biological learning systems. Humans consciously attempt to reflect on own learning and cognitive processes. The motivation behind this study is further bolstered from various researches arguing that reinforcement primarily dominates the late phase of learning and is responsible for long-term retention.

As Artificial intelligence (AI) is gradually permeating the field of Finance and Economics, many quantitative approaches emerge both in Finance and Machine Learning published literature. Considering that quantitative financial approaches share optimisation routines with AI and psychology and neuroscience interconnects behavioural Finance and AI, more applications of AI in Economics and Finance will be witnessed [Y. Li, 2018]. Reinforcement learning can help in almost all schemes of sequential decision making tasks [Y. Li, 2018]. Under the examined context, it discovers optimal actions by trial and error and while it does not model the market, it is flexible to its episodic trends. By dint of the discounted rewards, it constitutes a non-myopic approach that optimises for longer-term benefits.

1.3 Structure

This report is organised in 3 parts. Part I introduces term definitions and sets a financial background necessary to follow the rest of the report. Part II is comprised of a more technical review of the method employed and delves into complexities necessary to understand the implementation of Reinforcement Learning in portfolio optimisation. Part III reviews part of the literature on trading agents and summarises the experiments' main findings and results. Each chapter was designed to address a separate issue, while maintaining as much as possible a naturally progressing flow that eventually leads to a more clear understanding of the experiments. The reader is advised to follow the structure of the report as outlined below. However, if familiar with any of the discussed notions, the corresponding chapter may be skimmed or skipped. Following, is a concise chapter list with a brief description for each chapter.

Part I - Financial Background

Chapter II: Financial Concepts. This chapter lists necessary financial definitions.

Chapter III: Portfolio optimisation Theory. This chapter explains the main objective of portfolio optimisation, its conception, the way it is used in the contemporary literature and puts the following chapters in the context of portfolio optimisation.

Part II - Research, Design & Methods

Chapter IV: Minimum Spanning Tree. This chapter analyses the Minimum Spanning Tree method as a potential approach to form an uncorrelated portfolio.

Chapter V: Reinforcement Learning. This chapter introduces both notation and theory on reinforcement learning and the employed TD3 algorithm.

Part III - Research Summary

Chapter VI: Trading Agents. This chapter sets the trading agents' objectives, reviews seminal works and outlines the technical details of this implementation.

Chapter VII: Experiments. This chapter evaluates the performance of Reinforcement Learning as a trading architecture, when using real market-data in a simulated environment.

Chapter VIII: Conclusion. This chapter raises limitations and puts related research directions forward, for consideration.

Part I

Financial Background

Chapter 2

Financial Concepts

This chapter is part of the background series that introduces important theories and concepts as fundamental building blocks of this work. It initially focuses on defining several financial terms, which in turn are an integral part of the the chapter's main design; Financial time series. In case the reader is already familiar with the most common financial concepts, this chapter may be skimmed or skipped.

2.1 Definitions

This section includes the most typical, yet important definitions of financial terms that will be used throughout the report.

2.1.1 Asset

An asset is a resource controlled by an individual or any other economic entity, such as business. A liquid asset is an asset that can be easily converted into cash in a short amount of time. Cash and other marketable securities fall in the category of liquid assets [Investopedia, 2021a]. Consequently, a financial asset is a liquid asset that derives its value from contractual rights or ownership claim [Investopedia, 2021b]. Cash, bonds, stocks, mutual funds and bank deposits are often used as typical examples of financial assets and in contrast to land, commodities or any other tangible asset, they may not have an inherent value in them or even a physical form.

Security
A security is a tradeable financial asset of any kind. There are several types of securities, but their legal definition may vary according to nation-wide jurisdictions. In the United States, securities can be loosely categorised into debt securities (e.g. bonds), equity securities (e.g. stocks) and derivatives (e.g. futures contracts, options, swaps).

2.1.2 Liquidity

Liquidity describes how easily an asset or security can be converted into cash without affecting the price it can be bought or sold in the market [Investopedia, 2021c]. The most liquid asset is the cash itself. Conversely, land and real estate can be considered as non-liquid assets as

receiving cash from a sale may take a considerable amount of time. The market price is always governed by the forces of supply and demand. In general terms, the resulting market price, where supply and demand balance, is referred to as the equilibrium price and represents an agreement between producers and consumers of the good [GALE, 1955], or buyers and sellers of the financial asset, for that matter. "*A market is often said to be liquid when the prevailing structure of transactions provides a prompt and secure link between the demand and supply of assets, thus delivering low transaction costs*" [Gabrielsen, Marzo and Zagaglia, 2011]. Massimb and Phelps, 1994 define liquidity as the ability to immediately execute small market orders without the occurrence of large changes in price.

2.1.3 Trading costs

Market Impact
Recall, the Law of Supply and Demand referenced in section 2.1.2. Transactions always determine the resulting market price. Thus, the market impact of a transaction can be defined as the difference between the current market price and the speculated market price in the absence of the transaction [Torre, Ph. and Ferrari, 2000]. Market price is a hidden cost. Since, there cannot be both occurrence and non-occurrence of a transaction, the market price in the absence of the transaction, can only be estimated.

Slippage
Slippage refers to the difference between the expected price of a trade, upon placing an order, and the price at which the trade is actually executed by the system. It can occur at any time, but is most prevalent during periods of higher volatility [Investopedia, 2021d].

Transaction Costs
Transaction costs are expenses incurring each time upon buying or selling an asset. They typically include brokers' commissions for their services to customers. As a rule of thumb, transaction costs, for the majority of the exchange platforms, lie in a range between 0.1% and 2% for every transaction [Investopedia, 2021e]. The cryptocurrencies' market fees typically fall in the lower end.

2.1.4 Exchange-Traded Funds (ETFs) and Mutual Funds

An ETF is, in short, a pooled investment that usually tracks an index[1] [Investopedia, 2022a]. A typical ETF example is the SPDR S&P 500 ETF (SPY) which tracks the S&P 500 Index (contains 500 large-cap and mid-cap stocks from U.S. companies). Usually, ETFs are more appealing to investors than mutual funds, because the former can be traded as regular stocks at any point throughout the day; they often carry lower fees and are more liquid than mutual funds. Additionally, ETFs can be shorted and used to hedge or leverage a position.

2.1.5 Cryptocurrencies

Blockchain is defined as a public ledger or similarly, a distributed database of records of all transactions or digital events that have been executed and shared among participating parties. Each transaction in the public ledger is verified by consensus of a majority of the participants in the system, and once entered, the information can never be erased [Crosby et al., 2016]. Blockchain technology has uses in other domains except Finance.

[1] Indexes (or indices) reflect the performance of a basket of securities intended to replicate a certain area of the market.

Bitcoin, the most widely used and accepted decentralised digital currency to date, uses an underlying blockchain implementation to record its transactions. It was conceptualised by Satoshi Nakamoto in "Bitcoin: A Peer-to-Peer Electronic Cash System" as a purely peer-to-peer version of electronic cash that would allow online payments to be sent directly from one party to another without going through a financial institution [Nakamoto, 2008]. Their proposition introduced a novel proof-of-work consensus mechanism that made changes in the transactions history computationally impractical and burdensome for attackers. Without delving into much unnecessary technicalities, this mechanism would prevent double-spending; i.e. signing-over the same coin to two different users or the ability for an adversary to use some of the coins for two or more payments [Karame, Androulaki and Capkun, 2012]. The double-spend attack is still one of the major security issues in most blockchain systems. There is however room for improvement and ways to mitigate those risks. Zhang and Lee, 2019 argued for extending the number of block confirmations. Karame, Androulaki and Capkun, 2012 reason that there is an increasing reliance on Bitcoin among businesses, and its usage in fast payment scenarios (i.e. exchange time between the currency and goods is short; e.g. ≤ 30 seconds) is only expected to increase.

While the best known example of cryptocurrency might be Bitcoin, there exist rouglhy 20,000 registered cryptocurrencies according to CoinMarketCap, excluding the ones too small to be featured in exchanges. The total cryptocurrency market capitalisation is estimated to be approximately 932,38 billion USD, of which roughly 42,8 % attributed to Bitcoin[2]. To put this number into perspective, Amazon.com, Inc. and Tesla, Inc. have estimated market valuations of 1,14 trillion and 730,61 billion USD, respectively[3].

Many of them are effectively just tokens, but there are also real, practical infrastructure and technology benefits. Cryptocurrencies differ from the traditional financial assets because of their decentralised and open nature [Jiang, D. Xu and Liang, 2017]. There is a variety of smaller in trading volume currencies, that require smaller investments to observe fluctuations in their price movements. This direct consequence enables trading software to capture and learn from the market the impacts of their own trading actions. Due to their openness, markets are more accessible. The majority of cryptocurrency exchanges are open round-the-clock and they even provide application programming interfaces (API) for fetching market data and carrying out algorithmic trading actions [Jiang, D. Xu and Liang, 2017]. This is a major advantage and as such, sparks a motivation for considering cryptocurrencies as an investment option in this report.

It is also worth noting, that there is a growing research activity in cryptocurrency trading and especially from 2018 onwards. A recent survey [Fang et al., 2022], pointed out that 85 % of papers have appeared since 2018, demonstrating the emergence of cryptocurrency trading as a new research area in financial trading. According to the same survey, there was also a substantial increase of approximately 200 % from 2017 to 2018, followed by a 100 % rise from 2018 to 2019. Published activity kept steadily increasing thereafter.

Cryptocurrencies are gradually receiving acceptance across the trading communities and they have also begun to be integrated in hedge funds', institutional investors' and individuals' portfolios and trading strategies [Fang et al., 2022]. Regulation is recently a main topic of discussion across many circles and there are opposing powers pulling in different directions. Ultimately, the outcome will likely impact the cryptocurrency market and its trading landscape. The way, however, remains to be seen.

[2]CoinMarketCap •*coinmarketcap.com*, Accessed on 24-June-2022
[3]Yahoo Finance •*finance.yahoo.com*, Accessed on 24-June-2022

2.2 Financial Time Series

Financial Time Series are central in portfolio optimisation and as such they are vigorously studied. They provide a broad toolkit to approach portfolio optimisation under a richness of objectives and complexities. In Finance, a time series tracks the movement of the chosen data points, such as a security's price, over a specified period of time with data points recorded at regular intervals. Financial time series inherit all the features of time series. There is however a key difference that distinguishes the former from other time series and that is their uncertainty. As a result, financial time series analysis, employs different statistical methods [Tsay, 2005]. This section will formalise time series in the context of Finance.

2.2.1 Portfolio

As an extension of the previous definitions in section 2.1, in the context of Finance, the term portfolio is used more often to describe a collection of liquid financial assets, although it could also include non-liquid ones. As mentioned in section 2.1.1, bonds, stocks, ETFs and cryptocurrencies are examples of liquid financial assets. A portfolio of assets is constructed according to an objective (e.g. to maximise expected returns or minimise risk), a time-frame and a specified risk-tolerance. At this point, it would be useful to mathematically formalise the term and keep a consistent tone in the parts that follow.

A portfolio of m assets and a total monetary value t would satisfy:

$$\sum_{i=1}^{m} w_{i,t} = 1, \tag{2.1}$$

where $w_{i,t}$ is the ratio of the i_{th} asset's value to the total amount t invested; also referred to as *portfolio weight*, or proportion of the total portfolio value invested in the i_{th} asset.

2.2.2 Gross Returns

Denoting an asset's price at a time index t as P_t, the asset's **gross** or **simple return** R_t from time $t-1$ to t is formulated as:

$$\begin{aligned} 1 + R_t = \frac{P_t}{P_{t-1}} \Rightarrow P_t = P_{t-1}(1 + R_t) \\ \Rightarrow R_t = \frac{P_t}{P_{t-1}} - 1 = \frac{P_t - P_{t-1}}{P_{t-1}}. \end{aligned} \tag{2.2}$$

2.2.3 Continuously Compounding Returns

As an extension of gross return R_t in eq. (2.2), **continuously compounding** or **log returns** r_t are defined as:

$$r_t = \log(1 + R_t) = \log \frac{P_t}{P_{t-1}}, \tag{2.3}$$

where **log** here represents the natural logarithm as seen most mathematical textbooks; also seen as **ln** in other contexts and applications.

2.2.4 Portfolio Returns

These notions can be extended to include the entirety of assets m that a portfolio contains. Using the right tools, portfolio returns provide indications on the investment's performance and risk. *Gross* or *simple returns* of a portfolio p that contains m assets at time t:

$$R_{p,t} = \sum_{i=1}^{m} w_i R_{i,t} = w_1 R_{1,t} + w_2 R_{2,t} + \dots \tag{2.4}$$

Recall from section 2.2.1 that $w_{i,t}$ is the weight of the i_{th} asset at time t. While the portfolio gross returns aggregate across assets, the log returns aggregate across time. For the log returns it holds:

$$\begin{aligned} r_{p,t} &= \log\left[1 + \sum_{i=1}^{m} w_{i,t} R_{i,t}\right] = \log\left[1 + w_{1,t} R_{1,t} + w_{2,t} R_{2,t} + \dots\right] \\ &= \log\left[\sum_{i=1}^{m} \frac{P_{i,t}}{P_{i,t-1}}\right] \end{aligned} \tag{2.5}$$

Unfortunately, due to their nonlinear property, log returns cannot be written as a sum of log returns of the portfolio's constituents as in eq. (2.4). They can only be approximated as:

$$r_{p,t} \approx \sum_{i=1}^{m} w_{i,t} r_{i,t} \approx \sum_{i=1}^{m} w_{i,t} R_{i,t}, \tag{2.6}$$

by using the $log(1 + x) \approx x$ approximation [Miskolczi, 2017].

2.2.5 Statistical Moments

The l_{th} moment of a continuous random variable $\mathcal{X}$ is defined as follows:

$$m'_l = \mathbb{E}(\mathcal{X}^l) = \int_{-\infty}^{\infty} x^l f(x)\, dx, \tag{2.7}$$

where $\mathbb{E}$ symbolises the expected value of the variable $\mathcal{X}$ and $f(x)$ is the probability density function of $\mathcal{X}$. A probability density function, also abbreviated as pdf, describes how likely a variable is to be sampled from a given distribution.

Mean
The mean or first moment of $\mathcal{X}$ is the expected value of $\mathcal{X}$ and measures the central location of the distribution [Tsay, 2005]. It is denoted by μ_x. Consequently, every following statistical moment, also referred to as central moment, is defined as:

$$m'_l = \mathbb{E}\left[(\mathcal{X} - \mu_x)^l\right] = \int_{-\infty}^{\infty} (x - \mu_x)^l f(x)\, dx, \tag{2.8}$$

Variance
Variance reflects how disperse a set of numbers is from their average value (mean). The positive square root of variance, is called *standard deviation* (*SD*) of $\boldsymbol{\mathcal{X}}$ and is denoted by σ_x. As a second central moment, variance is the expectation of the squared deviation of a random variable $\boldsymbol{\mathcal{X}}$ from its mean. Using eq. (2.8) variance is written as:

$$\boldsymbol{Var(\mathcal{X})} = \mathbb{E}[(\boldsymbol{\mathcal{X}} - \mu_x)^2] = \sigma_x^2$$

.

Skewness
The third central moment, also known as skewness is a measure of symmetry of $\boldsymbol{\mathcal{X}}$ around its mean [Tsay, 2005]. For a unimodal distribution *negative skew* indicates that the left tail is longer and the mass of the distribution is concentrated to the right of the figure, whereas in a *positive skew* case, the right tail is longer and the distribution appears as a left leaning curve. Skewness is formulated as:

$$\boldsymbol{Skew(\mathcal{X})} = \mathbb{E}\left[\frac{(\boldsymbol{\mathcal{X}} - \mu_x)^3}{\sigma_x^3}\right]$$

.

Kurtosis
The forth central moment, kurtosis, is a measure of the shape of a distribution. A distribution with a *positive excess kurtosis* (larger than that of a normal distribution) puts more mass in its tails (measures outliers) and appears as more peaked. Conversely, in a distribution with a *negative excess kurtosis* there is less extremity of deviations; the distribution has "thinner tails" and appears as less peaked than a normal distribution. Kurtosis is written as:

$$\boldsymbol{Kurt(\mathcal{X})} = \mathbb{E}\left[\frac{(\boldsymbol{\mathcal{X}} - \mu_x)^4}{\sigma_x^4}\right]$$

.

2.3 Evaluation Criteria

Having introduced some statistical background, it now becomes more intuitive to put some context around portfolio construction and management. In a brief glimpse of section 2.2.5, statistical moments may possess the necessary properties to be able to quantify a portfolio's performance. A portfolio[4] provides an investor with opportunities to grow the returns and minimise the risk of their collective investment, over individual assets. In light of this, the next parts to follow, describe such evaluation criteria that equip an investor with advantageous tools to build and manage a portfolio of investments.

2.3.1 Statistical Moments

Statistical moments are often used as *evaluation criteria* and may characterise a distribution of returns, either with respect to single assets to measure individual risk, or to quantify the performance of a particular strategy based on the combined portfolio returns. An example plot of returns for the SPDR S&P 500 ETF Trust (SPY) ETF and their distribution is seen in fig. 2.1 and fig. 2.2, respectively. The returns' distribution has a mean of $\boldsymbol{\mu = 0}$. A slightly negative skew is

[4]portfolio: a collection of financial assets (section 2.2.1).

an indication of a left longer tail and distributed mass to the right of the figure. A largely positive excess kurtosis signifies "heavier tails" and extremity of deviations. Recall that a normal (Gaussian) distribution (drawn in red colour) has a skewness of 0 and a kurtosis of 3 and often serves as a basis for comparisons.

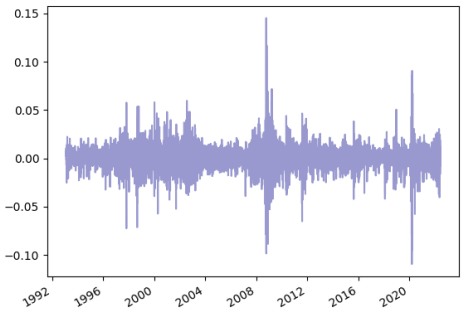

Figure 2.1: SPY Returns

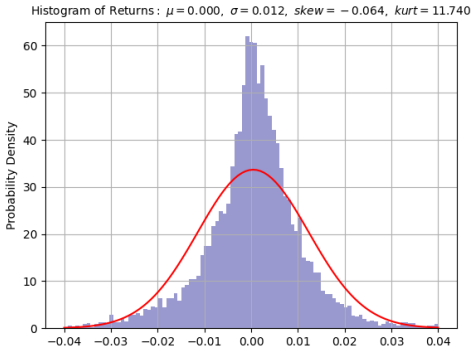

Figure 2.2: Distribution of SPY Returns

Remark 1: Due to its property of maintaining the units of measurement, the more frequent practice is to regard *standard deviation* σ_x as a measure of risk (see usage in section 2.3.3). A large standard deviation means that the returns fluctuate a lot so that the investor might be lucky and get a large return, but might also be unlucky and lose a lot [Rice, 2006].

Remark 2: For portfolio returns that are positively skewed, the probability of getting positive returns is higher than that of negative returns. Many studies contend that risk-averse investors should prefer positively skewed returns [Harvey et al., 2010; Naqvi et al., 2017].

Remark 3: Presence of excess kurtosis indicates a higher probability of extreme events also known as fat-tail risk [Naqvi et al., 2017; Khan et al., 2020].

2.3.2 Covariance and Correlation

Covariance
While variance measures the variability of a random variable, covariance measures the joint variability or degree of association of two random variables [Rice, 2006]. For two random variables $\mathcal{X}$ and $\mathcal{Y}$ with respective means μ_x and μ_y, covariance is defined as:

$$Cov(\mathcal{X}, \mathcal{Y}) = \mathbb{E}\left[(\mathcal{X} - \mu_x)(\mathcal{Y} - \mu_y)\right]. \tag{2.9}$$

This formulation can be interpreted as if $\mathcal{X}$ is larger than its mean and $\mathcal{Y}$ tends to be larger than its mean as well, then the association is positive leading to a positive covariance. In the antithetical paradigm, if $\mathcal{X}$ is lower than its mean, yet $\mathcal{Y}$ does not follow the same pattern, then the covariance is negative [Rice, 2006]. The sign of covariance shows the tendency in the relationship between the two variables. On the flip side, the magnitude of the covariance is not easy to interpret as it is not a normalised quantity. Covariance is particularly useful when evaluating financial assets in terms of their returns. The risk is actually lower when the returns are independent, rather than when they are positively correlated.

Remark 4: It may be a better decision to invest in uncorrelated, or weakly related for that matter, securities rather than placing investments in assets that belong in the same sector[5] [Rice, 2006].

Correlation
Using eq. (2.9) the correlation coefficient $\rho_{\mathcal{X},\mathcal{Y}}$ takes the form:

$$\rho_{\mathcal{X},\mathcal{Y}} = \frac{Cov(\mathcal{X},\mathcal{Y})}{\sqrt{Var(\mathcal{X})Var(\mathcal{Y})}} = \frac{\mathbb{E}\left[(\mathcal{X}-\mu_x)(\mathcal{Y}-\mu_y)\right]}{\sqrt{Var(\mathcal{X})Var(\mathcal{Y})}} = \frac{\mathbb{E}\left[(\mathcal{X}-\mu_x)(\mathcal{Y}-\mu_y)\right]}{\sigma_x\sigma_y} \in [-\mathbf{1},\mathbf{1}]. \tag{2.10}$$

Correlation arises as a dimensionless quantity and as a normalised version of the covariance contains values in the range $[-\mathbf{1},\mathbf{1}]$ [Rice, 2006]. A value of $+\mathbf{1}$ would imply a perfectly positive and linear correlation ($\mathcal{X}$ increases as $\mathcal{Y}$ increases), while $-\mathbf{1}$ would imply a perfectly negative and linear correlation ($\mathcal{X}$ increases as $\mathcal{Y}$ decreases).

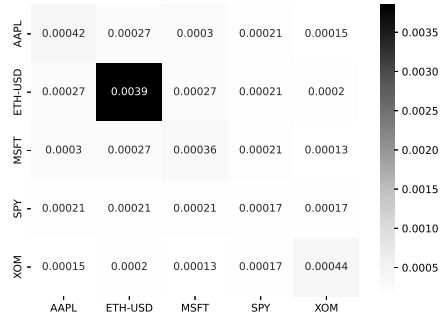

Figure 2.3: Covariance matrix of returns

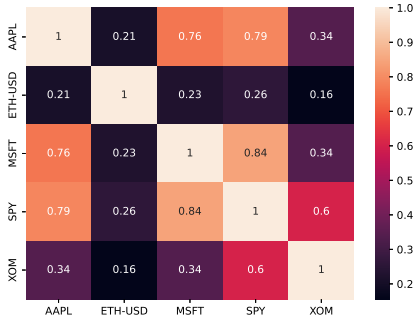

Figure 2.4: Correlation matrix of returns

By means of the fig. 2.3 and fig. 2.4 above, Ether (ETH), also known as the native cryptocurrency of Ethereum, appears as rather uncorrelated with Exxon Mobil Corporation (XOM) and Apple Inc. (AAPL) with respect to their gross returns. To the contrary, as one might postulate, Microsoft Corporation (MSFT) and Apple Inc., both specialising in the technology and consumer electronics domain, have a strong positive correlation in their returns.

2.3.3 Sharpe Ratio

The Sharpe Ratio is defined as the ratio of the excess expected return of an investment to its return volatility [Lo, 2002]. Here, standard deviation is employed as a measure of the investment's volatility. Thus, Sharpe Ratio (SR) is formulated as:

$$SR = \frac{\mathbb{E}\left[R_p - R_f\right]}{\sigma_p} = \frac{\mathbb{E}\left[R_p - R_f\right]}{\sqrt{Var\left[R_p - R_f\right]}}, \tag{2.11}$$

where returns R_p refer to the gross returns of a portfolio p. R_f is the risk-free return; usually the return of a U.S. Treasury security (e.g. Treasury Bond).

[5] assuming this would imply a strong correlation.

Remark 5: Sharpe ratio can be very useful for decision-making. A rising return or a falling standard deviation, lead to a rise in the Sharpe ratio. Hence, a higher Sharpe ratio is good and between two alternatives, the Sharpe ratio criterion is to choose the one with the higher Sharpe ratio [Dowd, 2000].

	PnL[6]	$\mu_{returns}$	$\sigma_{returns}$	$Skew$	$Kurt$	SR
[MSFT, SPY]	1.001	0.0001	0.014	−0.406	0.646	0.118
[MSFT, SPY, XOM]	1.199	0.0008	0.012	−0.603	1.503	1.073

Table 2.1: Statistics for two different portfolio constructions

Table 2.1 re-uses some stocks from section 2.3.2 and employs additional statistics to characterise two uniform portfolio constructions[7]. Recall that Microsoft Corporation (MSFT) has a very strong positive correlation with the SPDR S&P 500 ETF Trust (SPY) ETF and a weak correlation with Exxon Mobil Corporation (XOM), as also depicted in fig. 2.4. On that grounds, a different portfolio is constructed with the addition of XOM. The revised portfolio has a larger "fat-tail" risk and outperforms the first one in terms of the achieved PnL and Sharpe ratio, with its increased returns and lower risk (*standard deviation:* $\sigma_{returns}$). According to the Sharpe Ratio criterion, an investor should prefer the revised portfolio.

2.3.4 Maximum Drawdown

Denoting as $r(w,t)$ the function that describes the gross returns (uncompounded) at time t and as $w = (w_1, w_2, w_3, \ldots, w_m)$ the weights of the m portfolio constituents, the *drawdown* function is defined as the difference between the *maximum* of the function $r(w,t)$ until time t and the value of $r(w,t)$ at precisely time t:

$$\mathcal{D}(w,t) = \max_{0 \leq \tau \leq t} \{r(w,\tau)\} - r(w,t). \tag{2.12}$$

On a given time interval $[0,T]$ the *maximum drawdown* is formulated as the process of maximising the drawdown function $\mathcal{D}(w,t)$:

$$\mathcal{MD}(w) = \max_{0 \leq t \leq T} \{\mathcal{D}(w,t)\}. \tag{2.13}$$

While maximum drawdown remains an attractive measure of risk and is still prevalent as a metric in many studies, it suffers from heavy statistical errors, being based on a single observation of the maximal loss. As typically seen though, it may be integrated as a supplementary feature in more comprehensive analyses [Chekhlov, Stanislav Uryasev and Zabarankin, 2005].

[6]Profit and Loss (PnL): multiplier of the initial portfolio value (value at the time of the investment).
[7]uniform portfolio; also referred to as equally weighted portfolio (EWP): a portfolio that gives the same importance (weight) to each and every asset that includes.
[7]Those portfolios are merely for illustration purposes and were not a product of optimisation.

2.3.5 Conditional Value at Risk

Conditional Value at Risk (*CVaR*), also referred to as excess loss, is defined with respect to a probability level β such that a β-*CVaR* portfolio will experience losses above an amount α with a probability equal to β. A β-*VaR* (*Value at Risk*) portfolio will not exceed α in losses with a probability β. It holds that $\boldsymbol{CVaR \geq VaR}$. The values of β typically considered are: $\mathbf{0.9, 0.95}$ and $\mathbf{0.99}$ [Rockafellar and Stanislav Uryasev, 2000].

Symbolising gross returns as a random variable $\mathcal{X}$, the cumulative distribution function (cdf) takes the form: $F_{\mathcal{X}}(x) = P\{\mathcal{X} \leq x\}$, where $\mathcal{X} = (x_1, x_2, x_3, \dots)$. For a probability β, also referred to as confidence level, with $\beta \in [0, 1]$:

$$VaR_{\beta}(\mathcal{X}) = \min\{x \mid F_{\mathcal{X}}(x) \geq \beta\}. \tag{2.14}$$

For $CVaR_{\beta}(\mathcal{X})$ it holds that $\mathcal{X} \geq VaR_{\beta}(\mathcal{X})$ [Sarykalin, Serraino and Stan Uryasev, 2008]:

$$CVaR_{\beta}(\mathcal{X}) = \int_{-\infty}^{\infty} x \, dF_{\mathcal{X}}^{\beta}(x) \text{ , where} \tag{2.15}$$

$$F_{\mathcal{X}}^{\beta}(x) = \begin{cases} 0, & \text{when } x < VaR_{\beta}(\mathcal{X}) \\ \frac{F_{\mathcal{X}}(x) - \beta}{1 - \beta}, & \text{when } x \geq VaR_{\beta}(\mathcal{X}) \end{cases}$$

Although VaR is a very popular measure of risk, it has undesirable mathematical characteristics such as a lack of subadditivity and convexity [Rockafellar and Stanislav Uryasev, 2000]. Contrarily, CVaR is a more coherent risk measure exhibiting positively homogeneous and convex properties [Pflug, 2000], albeit more widely used in the insurance rather than the finance industry. CVaR can be efficiently minimised using linear programming and nonsmooth optimisation techniques [Rockafellar and Stanislav Uryasev, 2000]. VaR and CVaR measure different parts of the distribution. Depending on what is needed, one may be preferred over the other [Sarykalin, Serraino and Stan Uryasev, 2008].

Chapter 3

Portfolio Optimisation Theory

A financial portfolio is a balanced whole, a master asset with single financial assets as constituents. It is ultimately constructed to provide an investor with a richness of opportunities and financial objectives over single assets. A good portfolio formulation enables investors to more wholly amplify the market's positive aspects, while at the same time, weakening the negative impacts of single assets. Objectives are usually decomposed into the maximisation of the expected returns or the minimisation of the financial risk.

Both economic and non-economic factors, such as political forces can extensively influence the state of a financial market and induce uncertainty as a salient feature in monetary investments. So, for an objective to remain consistent over the course of time, a portfolio should be perennially introspected and managed. Portfolio management or portfolio optimisation, as they are interchangeably used, pertain to the constant reallocation of a monetary fund within a set of selected investment products.

Portfolio optimisation, thus, resembles a sequential decision making process and can be modelled as such. By definition, the time window of the reallocating activity relates to the investor's individual objective. Inherent in portfolio optimisation strategies, are assumptions made for elements like risk and transaction costs. The methods used under those strategies may borrow already seen statistical tools, such as statistical moments, covariance and correlation.

This chapter aims to provide intuition behind the portfolio optimisation task and the several notions that coexist in both the so-called "traditional methods" and the more recent advents in the Machine Learning and Deep Learning epochs.

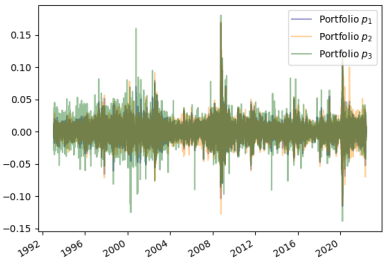

Figure 3.1: Examples of portfolio returns

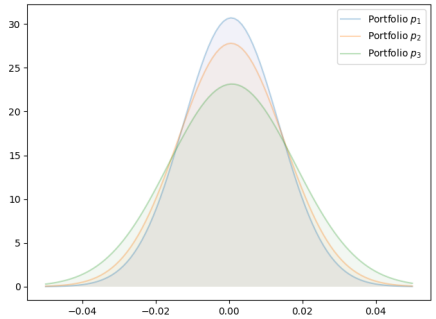

Figure 3.2: Example distributions of portfolio returns

3.1 Markowitz Model

The Markowitz model [Markowitz, 1952] emerges as a foundational part of Modern Portfolio Theory (MPT). Harry M. Markowitz formalised the noteworthy ideas of diversification and risk reduction. Both notions still remain as governing principles in most portfolio constructions. Merely comparing portfolio returns, would be a very naive approach to capture the complexity of investment decisions. Hence, financial risk is integrated as a separate criterion. Markowitz also postulated that more efficient portfolios can grow the expected portfolio return, while concurrently decreasing financial risk [Markowitz, 1952]. Thus, according to the Markowitz model, there is always an interplay between return and risk that is prominent in any portfolio construction.

Mathematically, risk is represented by the standard deviation of the portfolio returns. More formally, denoting with $w_i = (w_1, w_2, \cdots w_m)$, $i = 0, \cdots m$ the portfolio weight each of the m assets carries in the portfolio, it holds that: $\sum_{i=1}^{m} w_i = 1$. Weights can also be negative if short-selling[1]is allowed [Luenberger, 1998]. Further, let $\bar{r} = [\bar{r}_1, \bar{r}_2, \cdots \bar{r}_m]$ be the mean (expected) returns and $\sigma_{i,j}$ the covariance matrix for $i, j = 1, 2, \cdots m$. The Markowitz model assumes a given target expected return $\bar{r}_{target}$ [2]and minimises the risk as follows:

$$\begin{aligned} \underset{w}{\textbf{Minimize}} \quad & \frac{1}{2} w^T \sum_{i,j=1}^{m} w_i w_j \sigma_{i,j} \\ \text{subject to} \quad & \sum_{i}^{m} w_i \bar{r}_i = \bar{r}_{target}, \\ \text{and} \quad & \sum_{i=1}^{m} w_i = 1, \end{aligned} \tag{3.1}$$

where $\frac{1}{2}$ is a scaling and serves convenience purposes [Luenberger, 1998]. The model addresses this balance between the risk and return factors. When solved, it yields the weights for an efficient portfolio with the target expected return $\bar{r}_{target}$.

To solve, a *Lagrangian* function $\mathcal{L}$ must be formed. Using λ and μ as *Lagrange* multipliers for the first and the second constraint in eq. (3.1), the *Lagrangian* is derived as:

$$\mathcal{L} = \frac{1}{2} \sum_{i,j=1}^{m} w_i w_j \sigma_{i,j} - \lambda \left(\sum_{i=1}^{m} w_i \bar{r}_i - \bar{r}_{target} \right) - \mu \left(\sum_{i=1}^{m} w_i - 1 \right). \tag{3.2}$$

The *Lagrangian* in eq. (3.2) is differentiated with respect to each w_i and then the derivative is set to $\mathbf{0}$. Applying the first order condition of optimality[3]:

[1]Short selling refers to the case where an investor borrows a security and sells it on the market, planning to buy it back later for less money, obtaining the difference in price as a profit [Investopedia, 2022b].
[2]$\bar{r}_{target}$: the desired expected return.
[3]Stationary points of a function f (including minima, maxima, and saddle points) satisfy the first order condition $\nabla f(\mathbf{x}) = \mathbf{0}_{N \times 1}$ where x is an N-dimensional point of f.

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_i} &= 0, \quad i = 1, 2, \cdots m \\ \frac{\partial \mathcal{L}}{\partial \lambda} &= 0 \quad \text{and} \\ \frac{\partial \mathcal{L}}{\partial \mu} &= 0\end{aligned}$$

and using eq. (3.2):

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w_i} &= \sum_{j=1}^{m} w_j \sigma_{i,j} - \lambda \bar{r}_i - \mu = 0, \quad i = 1, 2, \cdots m \\ \frac{\partial \mathcal{L}}{\partial \lambda} &= -\sum_{i=1}^{m} w_i \bar{r}_i + \bar{r}_{target} = 0, \\ \frac{\partial \mathcal{L}}{\partial \mu} &= -\sum_{i=1}^{m} w_i + 1 = 0.\end{aligned}$$

The optimal vector of portfolio weights w_*, can be obtained solving the following vector-matrix formulation with linear-algebraic methods:

$$\begin{aligned}\mathbf{\Sigma} w_* &= \lambda \bar{r} + \mu \mathbf{1}, \\ w_*^T \bar{r} &= \bar{r}_{target} \quad \text{and} \\ w_*^T \mathbf{1} &= \mathbf{1},\end{aligned}$$

where $\mathbf{\Sigma}$ is the vector-form covariance matrix $\sigma_{i,j}$. The optimal weights are now be formed as:

$$w_* = \mathbf{\Sigma}^{-1} (\lambda \bar{r} + \mu \mathbf{1}) = \lambda \sigma^{-1} \bar{r} + \mu \mathbf{\Sigma}^{-1} \mathbf{1}.$$

But, first, the multipliers have to be derived:

$$\begin{aligned}\bar{r}^T \mathbf{\Sigma}^{-1} (\lambda \bar{r} + \mu \mathbf{1}) &= \lambda \bar{r}^T \mathbf{\Sigma}^{-1} \bar{r} + \mu \bar{r}^T \mathbf{\Sigma}^{-1} \mathbf{1} = \bar{r}_{target}, \\ \mathbf{1} \mathbf{\Sigma}^{-1} (\lambda \bar{r} + \mu \mathbf{1}) &= \lambda \mathbf{1}^T \mathbf{\Sigma}^{-1} \bar{r} + \mu \mathbf{1}^T \mathbf{\Sigma}^{-1} \mathbf{1} = \mathbf{1}.\end{aligned}$$

This leads to:

$$\begin{bmatrix} \bar{r}^T \mathbf{\Sigma}^{-1} \bar{r} & \bar{r}^T \mathbf{\Sigma}^{-1} \mathbf{1} \\ \mathbf{1}^T \mathbf{\Sigma}^{-1} \bar{r} & \mathbf{1}^T \mathbf{\Sigma}^{-1} \mathbf{1} \end{bmatrix} \begin{bmatrix} \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} \bar{r}_{target} \\ \mathbf{1} \end{bmatrix}$$

Setting the constraints $\alpha = \bar{r}^T \mathbf{\Sigma}^{-1} \bar{r}, b = \bar{r}^T \mathbf{\Sigma}^{-1} \mathbf{1}$ and $c = \mathbf{1}^T \mathbf{\Sigma}^{-1} \mathbf{1}$, the optimal weights are finally obtained as:

$$w_* = \mathbf{\Sigma}^{-1} (\lambda \bar{r} + \mu \mathbf{1}) = \frac{\mathbf{1}}{ac - b^2} \mathbf{\Sigma}^{-1} \Big[(c \bar{r}_{target} - b) \bar{r} + (a - b \bar{r}_{target}) \mathbf{1} \Big],$$

where $\bar{r}$ are the mean (expected) returns of the assets and $\bar{r}_{target}$ is the target expected return.

Recall that in this derivation, weights were not constrained to be only positive. That is, short-selling was allowed. If the antithetical perspective were to be reviewed, then short-selling can be prohibited by enforcing each and every weight to be positive. Is worth noting that, in short-selling, all assets are used, either borrowed or as active investments; whereas when prohibiting short-selling, some assets are not used at all [Luenberger, 1998]. The Markowitz model, under prohibition of short-selling, is formulated as:

$$\begin{aligned} \underset{w}{\textbf{Minimize}} \quad & \frac{1}{2} w^T \sum_{i,j=1}^{m} w_i w_j \sigma_{i,j} \\ \text{subject to} \quad & \sum_{i}^{m} w_i \bar{r}_i = \bar{r}_{target}, \\ \text{and} \quad & \sum_{i=1}^{m} w_i = 1, \\ \text{and} \quad & w_i \geq 0, \quad \forall i = 1, 2, \cdots m. \end{aligned} \tag{3.3}$$

This problem, however, involving an extra constraint, is no longer reducible to linear formulations, thereby requiring non-linear methods to be solved. It is usually termed as a **quadratic program** and involves increased complexity gradient computations in its solution [Perold, 1984; Luenberger, 1998]. More recently, there are programming suites solving quadratic and convex optimisation problems, using algorithms such as Sequential Least Squares Programming (SLSQP) [Kraft, 1988].

Equation (3.1) yields the *efficient* portfolios, whereby the risk is minimised for a given return target. An *Efficient Frontier* is termed as the feasible region where those optimal portfolios lie with respect to the returns-volatility[4]two-dimensional plane. This line, seen as the limit of optimality, reflects this trade-off decision between expected returns and risk. In the subfigures below, the *Efficient Frontier* for a small subset of assets, is drawn in red colour. Figure 3.3 pertains to the case where short-selling is allowed, while in fig. 3.4, short-selling is prohibited. Both consider transaction costs. When short-selling is not allowed, the Efficient Frontier exhibits a slight downward shift, including portfolios with lower returns.

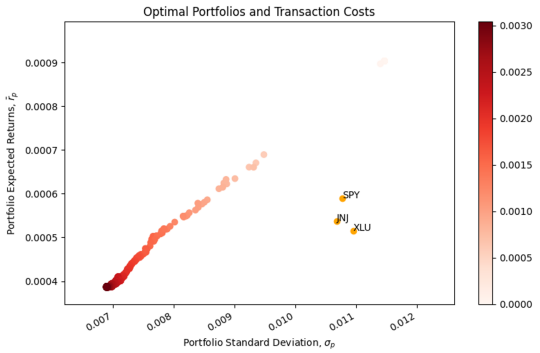

Figure 3.3: Efficient Frontier with short-selling

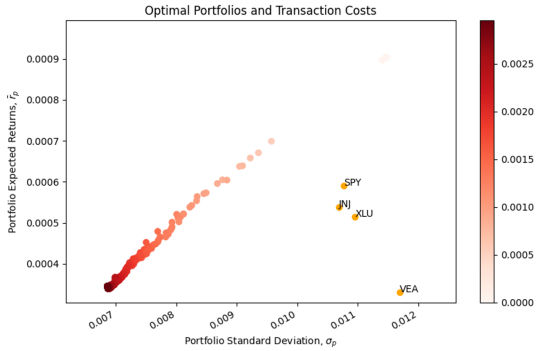

Figure 3.4: Efficient Frontier without short-selling

[4]Volatility here is represented by the standard deviation (σ).

3.2 Capital Asset Pricing Model (CAPM)

Markowitz's seminal work [Markowitz, 1952] has motivated a plethora of extension schemes, where the optimisation objective is modified. The *capital asset pricing model* (*CAPM*), mainly attributed to independent contributions from Jack Treynor (1961, 1962), William F. Sharpe (1964), John Lintner (1965) and Jan Mossin, is a theory that extends on the Markowitz mean-variance portfolio theory. Central in CAPM is the **market portfolio**, essentially a bundle, a summation of all the assets available in the investment universe [Luenberger, 1998]. An asset's presence in the market portfolio is determined by the asset's contribution to the totality of the capital market value. By nature of its absolute diversification, the market portfolio is not susceptible to unsystematic risk; only systematic[5].

What's a particularly interesting derivation is the **capital market line** or **pricing line**. The capital market line (CML) effectively reveals the positive correlation between expected returns and financial risk. In case risk is represented by the standard deviation σ, CML is a straight line [Luenberger, 1998]:

$$\textbf{CML:} \quad \bar{r} = r_f + \frac{\bar{r}_M - r_f}{\sigma_M}\sigma, \tag{3.4}$$

where M relates to the market portfolio and r_f is the risk-free rate of return. Following is that, all efficient assets should lie on this line between the risk free rate and the market portfolio. The area above the CML is infeasible, while portfolios below the CML are deemed as sub-optimal or inefficient [Luenberger, 1998].

Sharpe Ratio

Note that the factor or slope of the CML line: $\frac{\bar{r}_M - r_f}{\sigma_M}$, is the **Sharpe ratio** of the market portfolio. Similarly, the subfigures below illustrate the Efficient Frontier along with the CML, for a small universe of assets; in both, transaction costs[6]are subtracted in each optimisation step. The market portfolio is the one where the CML is tangent to the Efficient Frontier.

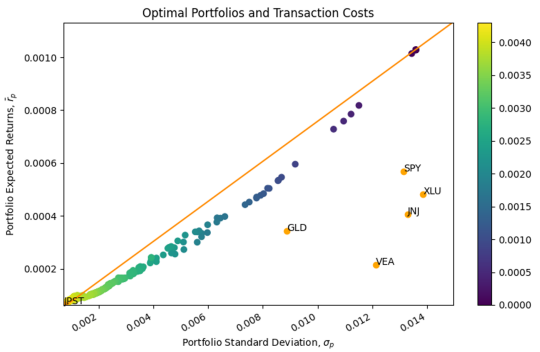

Figure 3.5: CML with short-selling

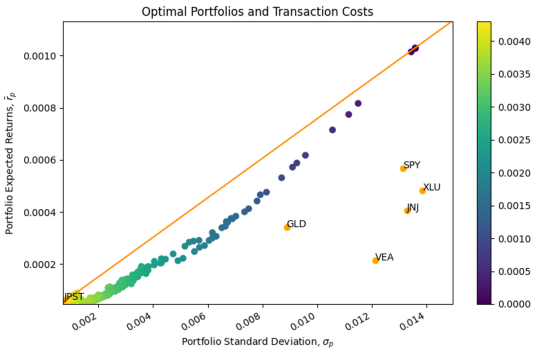

Figure 3.6: CML without short-selling

Sharpe ratio can also be considered as an objective for the mean-variance optimisation task. It merely requires the expected returns and the standard deviation and is eventually able to yield an efficient portfolio that falls in the CML.

[5]systematic risk: risk that affects the market as a whole; unsystematic risk: risk inherent to a particular asset class.
[6]Transaction costs: expenses incurring each time upon buying or selling an asset, section 2.1.3.

Part II

Research, Design & Methods

Chapter 4

Minimum Spanning Tree

This chapter stays in touch with portfolio optimisation, albeit more heavily focusing on the precursor step of asset (pre-)selection. However, this phase still deserves a distinct mention as many of the notions used, are derived from other areas of Mathematics; namely the Graph Theory. The following section will attempt to introduce the Minimum Spanning Tree (MST). A successive literature review, will explicate the MST as a method with observed implementations in the Finance domain and connect it with some of its possible applications in asset diversification.

4.1 Introduction

This chapter shifts from the portfolio optimisation task to the preceding phase of asset selection. The Finance industry may still lack a supportive measure for selecting uncorrelated assets that will later be subject to optimisation. This section describes the MST, as a graph-theory adopted method that searches for minimally correlated assets, thereby limiting a large universe of assets to a smaller uncorrelated subset. Another motivating characteristic, is the interpretability a topological structure would provide in analysing the relation between assets [Mantegna, 1998].

A MST is a weighted graph whose nodes might represent cities, whose edges might represent communication links and whose edges' weights might be the cost of transportation between the cities [Graham and Hell, 1985]. The objective of an algorithm that solves the minimum spanning tree problem is to search for the transportation links that give the least total amount of cost; in other words, to find the edges, out of many, that yield the least summation of weights, while connecting all the nodes. Note that an MST, connects all nodes together without forming any cycles. MST algorithms are able to handle efficiently thousands of nodes [Graham and Hell, 1985].

All algorithms assume the same vertices[1], yet they differ in their criteria of selecting the appropriate edge to append to the graph [Graham and Hell, 1985]. They are usually termed as greedy or "myopic" heuristics. Greedy algorithms select the best choice available at each iteration, without regard to possible future consequences. They quickly converge to a solution, although it might be a sub-optimal one [Encyclopedia of Mathematics, 2014]. Using the Kruskal's algorithm[2] [Kruskal, 1956], a minimum spanning tree is constructed as shown in alg. 1.

[1] nodes and vertices are used interchangeably; edges is the term used for the links that connect the nodes.

[2] Kruskal's alg. runs in $\mathcal{O}(\mathcal{E} \log \mathcal{E})$ polynomial time, where $\mathcal{E}$: number of edges. It shares the same principles with the Prim's alg. [Prim, 1957]. Prim's alg., being faster, runs in $\mathcal{O}(\mathcal{E} + \mathcal{V} \log \mathcal{V})$ using Fibonacci Heap, where $\mathcal{V}$: no. of vertices. Prim's alg. is more cumbersome to implement, hence often used in graphs with many edges [Rafid, 2019].

Algorithm 1: Kruskal's algorithm for constructing a MST

Input: A weighted connected graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, $\mathcal{V}$: vertices, $\mathcal{E}$:edges
Output: $\mathcal{E}_t$: The set of edges composing the minimum spanning tree $\mathcal{G}$

sort $\mathcal{E}$ in ascending order of the edge weights: $w(e_{i_1}) \leq \cdots \leq w(e_{i_{|\mathcal{E}|}})$
$\mathcal{E}_t \leftarrow \varnothing$; counter $\leftarrow 0$ `/* Initialise the set of tree edges and its size */`
$k \leftarrow 0$ `/* Initialise the number of processed edges */`

while *counter* $< |\mathcal{V}| - 1$ **do**
  $k \leftarrow k + 1$
  if $\mathcal{E}_t \bigcup e_{i_k}$ *is acyclic* **then**
    $\mathcal{E}_t \leftarrow \mathcal{E}_t \bigcup e_{i_k}$ `/* counter ← counter +1 */`
  end
end

4.2 Literature Review

Having introduced some useful notation around MST, the objective of its utility can now be restated; What if those communication links (edges) were weighted with the correlation coefficients of the assets' returns, instead of an arbitrary cost notion. Then, the MST would form paths between assets based on their correlation, revealing relationships between their returns. This exact idea was explored by Mantegna, 1998; Bonanno et al., 2004; Onnela et al., 2002; Millington and Niranjan, 2021; Birch and Soramaki, 2015 and by others on the cryptocurrency market [J. Y. Song, Chang and J. W. Song, 2019; Stošić et al., 2018]. As an aforementioned added advantage, MST algorithms can handle a large amount of nodes; in this case assets.

Mantegna, 1998, as a forerunner implementation, used daily logarithmic price differences' time series of stocks from the S&P 500 market index, among other smaller sets, in search for hierarchical structures in the portfolio. To fulfil the axioms of a distance metric[3], an edge's distance d was not directly set as the correlation coefficient, but rather as a modified function of it:

$$d(x, y) = 1 - \rho_{x,y}^2, \tag{4.1}$$

where $\rho_{x,y}$ is the *Pearson sample correlation coefficient*:

$$\rho_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}, \tag{4.2}$$

where n is the size of the sample, as in number of days of historical returns, x_i, y_i are the sampled returns and $\overline{x}, \overline{y}$ the distribution means of the asset's returns time series x and y.

The author reports that the MST's hierarchical formation unravelled connections between assets from a large and complex structure. In the obtained topology, this was manifested in clusters of companies that belong to the same industry sector.

In a similar fashion, yet introducing slight modifications, later studies justified the spanning tree as a tool to reveal community clusters out of non-trivial hierarchical market structures.

[3] axioms considered: (*i*) $d(x, y) = 0$ *if and only if* $x = y$, (*ii*) $d(x, y) = d(y, x)$, (*iii*) $d(x, y) \leq d(x, k) + d(k, y)$.

As a key insight derived from the analyses, a market's diversification potential, can be very closely resembled by the behaviour of the tree [Onnela et al., 2002].

4.2.1 Variations

- Definition of a new edge distance: $d'(x,y) = \sqrt{2(1-\rho_{x,y})}$, such that $d'(x,y) \in [0,2]$, to penalise negative correlations. Intuitively thinking, lower correlations, or negative ones for that matter, yield larger distances. So, importantly, as shown in fig. 4.1, uncorrelated and negatively correlated assets are distant from each other, while correlated ones are adjacent. Recall that correlation was defined with respect to assets' returns. Thus, this distance inclines towards grouping together the *positively* correlated assets.

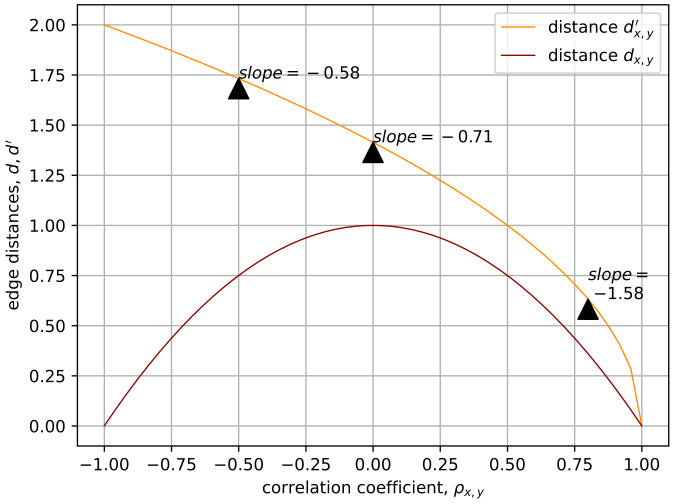

Figure 4.1: MST edge distances $d'(x,y) = \sqrt{2(1-\rho_{x,y})}$ shown in orange; and $d(x,y) = 1 - \rho^2_{x,y}$ shown in dark red.

- Millington and Niranjan, 2021 argued that the *Spearman correlation coefficient*[4] is a better performing metric and more stable with respect to the outputs of the tree, than the *Pearson correlation coefficient*. Being a non-parametric (distribution-free) measure of correlation, the *Spearman correlation coefficient* captures a monotonic relationship between the assets' returns time series, while Pearson's correlation is a measure of their linear relationship [Hauke and Kossowski, 2011; Ahlgren, Jarneving and Rousseau, 2003].

In connection with the **Mean-Variance Portfolio Theory**, Hüttner, Mai and Mineo, 2016 contended that non-central assets in a graph, form a well diversified portfolio in a Markowitz setting, albeit this behaviour was merely based on empirical evidence. Further, Onnela et al., 2002 argued that the assets of the optimal *Markowitz portfolio*, lie, almost exclusively, on the outskirts of the tree, thereby being weakly or negatively correlated[5].

[4]Spearman's correlation coefficient is defined as: $r_s = \rho_{R(x),R(y)} = \frac{cov(R(x),R(y))}{\sigma_{R(x)}\sigma_{R(y)}}$, where ρ is the Pearson's correlation but applied on the categorically ranked variables $R(x)$ and $R(y)$.

[5]Onnela et al., 2002 used $d'(x,y) = \sqrt{2(1-\rho_{x,y})}$ as the edge distance.

Chapter 5

Reinforcement Learning

Learning from interacting with the environment is a foundational idea underlying the nature of learning and enables humans to navigate life and build experiences. From toddlers to older adults, humans exercise actions and observe immediate or distant consequences from the environment. From learning motor skills, such as riding a bicycle, or more demanding cognitive skills, such as decision making and paying attention or hold to a conversation, there is always an either internal or external awareness that allows humans to adapt new behaviours or refine their current ones [Sutton and Barto, 2018]. In terms of behavioural psychology, reinforcement is a consequence which when applied, will strengthen an organism's future behaviour when perceiving a stimulus (object or event that elicits a sensory or behavioural response). The motivation behind this study is further bolstered from various researches arguing that reinforcement primarily dominates the late phase of learning and is responsible for long-term retention [Abe et al., 2011; Taylor, Krakauer and Ivry, 2014; Shmuelof et al., 2012; V. Huang et al., 2011].

From a computational approach, learning through reinforcement is coined by Sutton and Barto, 2018 as the activity of learning what to do; observing situations/perceiving the environment's response and subsequently acting properly to achieve a desired objective. Usually, the objective is to maximise a reward. As discussed above, this exact same pattern, is observed in what behavioural psychology calls reinforcement; an action's consequence that strengthens a certain future behaviour towards a consciously desired goal. In this report, reinforcement learning, is adopted as a principle for effectively solving problems of a scientific and particularly economic nature. It will be regarded as a computational approach that attempts to attain a desired objective and learns by perceiving the environment's response to interactions.

The introductory section discusses certain notions encountered in the course of the learning process and defines the key elements of reinforcement learning. Sections that follow, will gradually delve into the most essential mathematical and formal components and eventually analyse the approach implemented in this report.

5.1 Introduction

This section is concerned with major principles from the reinforcement learning literature and brings forth important notions, defined in a theoretical manner and later used to mathematically formulate the decision process.

5.1.1 Main Distinction

As presented in the Artificial Intelligence (AI) literature, reinforcement learning, a main paradigm of Machine Learning, differs from the other approaches since it is goal-oriented. A key distinction is the employment of an agent that interacts with its environment [Sutton and Barto, 2018]. "An agent is just something that acts (agent comes from the Latin agere: to act). Of course, all computer programs do something, but computer agents are expected to do more: operate autonomously, perceive their environment, persist over a prolonged time period, adapt to change, and create and pursue goals." [Russell and Norvig, 2021].

Reinforcement learning is different from supervised learning, as it does not require labelled input to generalise to unseen instances. Instead, the agent often interacts with highly uncertain and dynamic environments. As this may lead to confusions, reinforcement learning is also distinct from other unsupervised learning approaches. Unsupervised learning, for the most part, tries to uncover hidden structures in unlabelled inputs and then, through mimicry, is able to perform tasks such as video generation, creative writing or speech synthesis. While this would be a certainly useful inclusion in a reinforcement learning architecture, it does not independently solve for achieving a desired objective and maximising a reward [Sutton and Barto, 2018].

5.1.2 The Reinforcement Learning Model

Reinforcement learning settings are concerned with sequential decision making. An agent is connected to an environment with means of its environment perception (observation of the environment state) and action. On every interaction step, the agent observes the current environment's state and then chooses which action to perform. The action has the ability to change the environment state and lead to a new one. This state transition is communicated to the agent through a numerical reward; also referred to as the reward signal in literature [Kaelbling, Littman and Moore, 1996]. A reward can be either positive or negative.

Interaction is a recurrent process. That is, upon completion of a step, the agent has to observe the environment state again and potentially refine its behaviour. The exact decision mechanisms that are responsible, are subject to discussion in the next sections of this chapter.

5.1.3 Key Elements

Policy
A policy can be viewed as the link between the states perceived from the environment, and the actions that will be taken in response to those states. It could also be described as what psychology calls a stimulus-response rule. It is the most invaluable and instrumental part of the reinforcement learning architecture, as policy, alone, is sufficient to determine the agent's behaviour [Sutton and Barto, 2018].

Reward Signal
A reward signal is associated with the desired objective. In most settings, an agent tries to maximise the total reward received over the course of time, when interacting with the environment. As a consequence, the policy has to be frequently updated, to favour those behaviours that ultimately lead to better long-term rewards. In behavioural psychology terms, a certain behaviour is strengthened (reinforced), by the actor observing the consequences of its actions. Reward can also be modelled as a function that takes into account multiple environment variables.

Value
A state value estimates the total reward expected to be accumulated over time, starting from a specified environment state. It is focused more on a long-term strategy and provides an assessment of the current and expected environment states, whereas reward holds a more of an instantaneous benefit character. For instance, a particular state in the environment, may yield a low reward, but the state value can be still high, if succeeding states frequently yield high rewards. As a paradigm, a reward could be analogised to either the pain (lower rewards) or the pleasure (higher rewards) that humans feel, while a value to a pleasing or displeasing feeling regarding the present situation. A value is always linked to a value function which essentially estimates the quality of an action in a given state. On a final note, values are far more complex to compute than rewards, which are essentially an outcome of the environment. Values have to be constantly re-estimated for as long as the agent observes and remains active on the environment [Sutton and Barto, 2018].

5.1.4 Exploration - Exploitation Dilemma

Exploration and exploitation are two different cornerstones of problem-solving by state search. Every algorithm that explores an environment, needs to address the balance between exploration and exploitation [Črepinšek, Liu and Mernik, 2013]. Reinforcement learning encounters the same dilemma. An agent has to exploit situations already experienced and are likely to yield a large reward. But, at the same time, to avoid being trapped in local minima through fast convergence, it also has to explore different, seemingly "sub-optimal", actions, but which, nonetheless, may uncloak improved solutions.

A well known formulation, called the multi-armed bandit problem, exemplifies this quandary. In Las Vegas, an one-armed bandit is a slot machine. A gambler can insert a coin and has to decide whether to pull the lever that has paid off best, or one that has not been tried yet, in pursue of better long-term rewards. In practice, algorithms explore by (***1***) modifying the reward (e.g. added bonus for exploration) [Brafman and Tennenholtz, 2001], (***2***) performing modelled actions, such as by using statistical tools to come up with confidence bounds that regulate the uncertainty of the actions [Auer, 2003], (***3***) injecting noise to actions (e.g. noise sampled from a Gaussian distribution) [Fortunato et al., 2017].

5.2 Formalising The Problem

This section formalises reinforcement learning as a problem, introduces notation and concretises the so far theoretically explained, key elements of reinforcement learning. Many of these formalisations are central to pave the way for the methods considered in the course of the report.

5.2.1 Markov Decision Processes

From an antithetical perspective, reinforcement learning is the problem faced by a learning agent that must learn a behaviour through **trial-and-error** interactions with a dynamic environment [Kaelbling, Littman and Moore, 1996; Szepesvari, 2010]. Trial and error is a noteworthy dynamic as the learner has to discover which actions yield the most reward by trying them. There exist no prior information on the preferred behaviours and environment states. The environment is usually adopted as an abstract representation of the agent's surroundings; things

that an agent interacts with. The agent must be able to perceive the state of its environment, to some extent, and must take actions that consequently affect the state. In addition, it should also satisfy the core reinforcement learning philosophy to pursue long-term objectives relating to the state of the environment.

Markov Decision Processes (MDP), an idea adopted from Dynamical Systems Theory, provide a broad framework to model decision making. MDPs have been extensively used in theoretical studies because the framework is rich enough to model problems involving sequential choices made over time and under uncertainty (e.g. economic ones) [Rust, 1994]. The family of sequential decision problems require that the agent has to perform a sequence of decisions. Under this basis, the agent has to be able to learn from delayed rewards and potentially trade off immediate for delayed rewards [Sutton and Barto, 2018]. This special property of reinforcement learning agents, is very attractive for financial applications, where investment horizons range from few days and weeks to years or decades. As briefly referenced in section 5.1.3, shortsighted agents can perform very poorly, while giving weight to "over the long run" - rewards rather than just to immediate ones, is hugely important in order to succeed [Sutton and Barto, 2018].

In a MDP, the agent interacts with the environment in steps. Assuming a discrete space for simplicity, at each time step t, where $t = 1, 2, 3, \ldots$, the agent receives a representation of the environment's state, $S_t \in \mathcal{S}$, and then selects an action, $A_t \in \mathcal{A}(s)$. In the next time step and as a consequence of its action, the agent receives a numerical reward, $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$, and finds itself in a new state, S_{t+1}.

Thus, the agent - environment interaction results in the following sequence:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots.$$

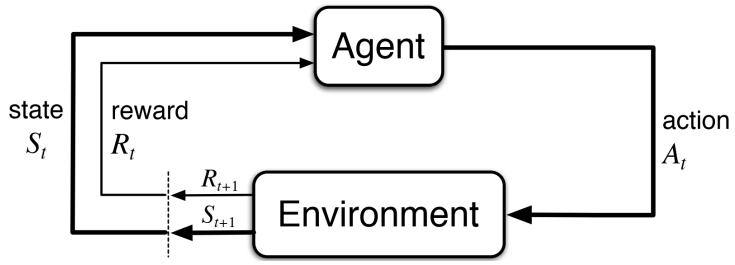

Figure 5.1: Agent - Environment setting in a MDP, source: Sutton and Barto, 2018

In a **finite** MDP, states, actions, and rewards ($\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$) all have a finite number of elements. Thus, the variables R_t and S_t are considered to be random and have discrete probability distributions conditional to the preceding state and action. In other words, particular values $s' \in \mathcal{S}$ and $r \in \mathcal{R}$ will have a certain probability to occur at a time step t, which is only dependent to the previous state and action. The function p describes the MDP dynamics:

$$p(s', r \mid s, a) = Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}, \tag{5.1}$$

for all s', $s \in \mathcal{S}$, $r \in \mathcal{R}$ and $a \in \mathcal{A}(s)$. The '|' notation denotes conditional probability. As a probability distribution function, p has to satisfy:

$$\sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r \mid s, a) = 1, \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}(s). \tag{5.2}$$

As mentioned, the probability of each possible S_t and R_t depends only on the preceding state and action, S_{t-1} and A_{t-1}. The state, however, must include information about all aspects of the past agent - environment interaction that matter for the future. In that case, the state is said to have the **Markov property** [Sutton and Barto, 2018].

5.2.2 Policy & Value Functions

Value functions were firstly alluded to in section 5.1.3 as ways to estimate how good is to be in a given state, or how good is to perform a certain action in a given state. These estimations will be expressed in relation to the future rewards, or more formally in connection with the expected return G_t for a given time step t [Sutton and Barto, 2018]. Value functions are always formulated according to a policy that is responsible for selecting actions. Recalling the definition of policy in section 5.1.3, a policy π is the underlying decision mechanism that determines a probability to perform a certain action in a given state. A policy can be deterministic or stochastic. Certain methods to estimate policies will be explained in subsequent sections.

Episodes
In most reinforcement learning settings, the agent-environment interaction is naturally broken into episodes [Sutton and Barto, 2018]. An episodic task refers to any task that solves a problem of repeated interaction. That is, every task ends in the so-called terminal state and subsequently the next episode starts from the exact same initial state, independently of how the previous episode ended. For example, playing an entire game can be considered as one episode, with the terminal state being reached when one player loses or wins. By contrast, there are also continuing tasks that span a longer time frame and solve different types of control problems.

Expected Return
The criterion of **discounted rewards** is the most widely used as well as easier to deal with mathematically [Szepesvari, 2010; Kaelbling, Littman and Moore, 1996]. The notion of **discounting** appears as a mathematical trick to bound an infinite sum. Rewards far in the future worth exponentially less than the reward received at the first state [Szepesvari, 2010]. While following a discounting behaviour, the agent selects actions that maximise the sum of discounted rewards received over time. Thus, the expected return G_t, over an infinite horizon, is formulated as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{5.3}$$

while in relation to the next state:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \\ &= R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots \right) \\ &= R_{t+1} + \gamma G_{t+1}, \end{aligned} \tag{5.4}$$

where $\gamma \in [0, 1]$ is called the **discount factor** or **discount rate** and $k = 0, 1, 2, 3, \dots$ tracks the time step's increment. Note that time step $t = 1, 2, 3, \dots$ has a different count. From a different perspective, a reward received k time steps in the future, will worth γ^{k-1} times the same value if it were to be received in the immediate next state.

For a value of γ very close to zero, the agent is "myopic" as in maximising only the immediate rewards. If $\gamma = 0$, eq. (5.3) yields just the reward of the next state. For values of γ close to 1, the agent is more farsighted, taking into account future rewards. If $\gamma = 1$, the MDP is called undiscounted and the reward of each and every state will carry the same importance.

Ultimately, in a more formal fashion, the agent's goal is to come up with a good way of choosing the actions, same as learning a suitable policy, so as to maximise the expected total discounted reward, or expected return G_t [Szepesvari, 2010; Lillicrap et al., 2015].

State-value Function
In an MDP space, a state-value function u_π under a policy π, can be defined by:

$$u_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s\right], \text{ for all } s \in \mathcal{S}, \tag{5.5}$$

where $\mathbb{E}_\pi$ symbolises the expected value of a variable (given a policy π); in this case the expected return given a state in the environment. A state-value function u_π characterises the quality of a given state; **how good is to be in a given state**.

Action-value Function
In a similar manner, under policy π and action a performed in state s, an action-value function can be defined as:

$$q_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right], \text{ for all } s \in \mathcal{S}. \tag{5.6}$$

An action-value function q expresses the value of taking an action a (under state s and policy π) as the **expected return of action** a [Sutton and Barto, 2018].

5.2.3 Bellman Equation

Maintaining a similar outlook as seen with expected discounted rewards from future states, a state-value function can hold the same property. Borrowing some principles from dynamic programming, a complicated problem can be simplified when broken down into simpler subproblems in a recursive manner. That is, computing value functions for each state, hence breaking the problem into smaller ones. For a policy π and a state s, a state-value function u_π can be formulated as:

$$\begin{aligned} u_\pi(s) &= \mathbb{E}_\pi\left[G_t \mid S_t = s\right] \\ &\overset{\text{eq. (5.4)}}{=} \mathbb{E}_\pi\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\ &= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) \Big[r + \gamma \mathbb{E}_\pi\left[G_{t+1} \mid S_{t+1} = s'\right]\Big] \\ &= \sum_a \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a)\left[r + \gamma u_\pi(s')\right], \text{ for all } s \in \mathcal{S}. \end{aligned} \tag{5.7}$$

Equation (5.7) is known as the **Bellman equation**, named after Richard Bellman. It expresses a relationship between the value of a state s and the values of its successor states s'. It states that the value of the start state must equal the discounted value of the expected next state, plus the reward expected along the way [Sutton and Barto, 2018].

5.2.4 Optimal Value Functions

A policy π is better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states. By way of explanation, $\pi \geq \pi'$ if and only if $u_\pi(s) \geq u_{\pi'}(s)$ for all $s \in \mathcal{S}$. A policy satisfying this property is called *optimal* and is denoted with π_*. There might be more than one optimal policies.

Optimal state-value function
An *optimal state-value function* u_* holds true for all optimal policies π_* and is formulated as:

$$u_*(s) = \max_\pi u_\pi(s), \text{ for all } s \in \mathcal{S}. \tag{5.8}$$

Optimal action-value function
The *optimal action-value function* q_* for all optimal policies π_* is formulated as:

$$q_*(s,a) = \max_\pi q_\pi(s,a), \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}. \tag{5.9}$$

Then, the optimal policy can be derived from: $\pi_* = \textbf{arg max}_a\, q_*(s,a)$.

The optimal action-value function yields the expected return for taking action a in state s and subsequently following the optimal policy π_*. In terms of u_*, the optimal action-value function q_* can be expressed as:

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma u_*(S_{t+1}) | S_t = s, A_t = a\right]. \tag{5.10}$$

5.2.5 Bellman optimality equations

The Bellman equation can be extended to include optimal policies. The value of a state, considering an optimal policy π_*, equals the expected return for the best action from that state [Sutton and Barto, 2018]. An *optimal state-value function* u_* is given by:

$$\begin{aligned}
u_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}\left[G_t | S_t = s, A_t = a\right] \\
&\overset{\text{eq. (5.4)}}{=} \max_a \mathbb{E}_{\pi_*}\left[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a\right] \\
&= \max_a \mathbb{E}_{\pi_*}\left[R_{t+1} + \gamma u_*(S_{t+1}) | S_t = s, A_t = a\right] \\
&= \max_a \sum_{s',r} p(s', r \mid s, a)\Big[r + \gamma u_*(s')\Big].
\end{aligned} \tag{5.11}$$

Similarly, the Bellman optimality equation for an *optimal action-value function* q_* is defined as:

$$\begin{aligned}
q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \,\middle|\, S_t = s, A_t = a\right] \\
&= \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \max_{a'} q_*(s', a')\right].
\end{aligned} \tag{5.12}$$

5.3 Q-Learning

Finding an optimal action-value function is a complex task owing to the optimal policy being initially unknown. On that account, instead of estimating the action-value function directly, an action-value function Q is learned through a recursive process of coincident policy and value function improvement [Watkins, 1989]. *Q-learning*, an early breakthrough in reinforcement learning, is an algorithm that approximates the optimal action-value function q_*, independently of the policy being followed. Coined by Christopher Watkins [Watkins, 1989], Q-learning updates are given by:

$$\underbrace{Q^{new}(S_t, A_t)}_{\text{updated}} \leftarrow \underbrace{Q(S_t, A_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \Bigg[\underbrace{\underbrace{R_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \overbrace{\max_a Q(S_{t+1}, a)}^{\text{estimate of optimal future value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q(S_t, A_t)}_{\text{old value}} \Bigg]}_{\text{temporal difference}}, \tag{5.13}$$

where $\alpha \in (0, 1]$ is a constant step-size parameter, also referenced as *learning rate* and $\gamma \in [0, 1]$ is the *discount factor* as seen in eqs. (5.3) and (5.4). The learned action-value function, Q, directly approximates q_*, the optimal action-value function, and has been shown to converge to q_* with probability 1 [Sutton and Barto, 2018].

Algorithm 2: Q-Learning for estimating q_*

Input: Algorithm parameters: learning rate $\alpha \in (0, 1]$

Output: Optimal action-value function q_*

Initialise $Q(s, a)$ $\forall s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$
for $1, \ldots, N$ *episodes* **do**
  Initialise S
  for $1, \ldots, t$ *episode time steps* **do**
    Choose action A from state S using the policy π derived from Q
    Take action A
    Observe reward R and next state S'
    Update Q: `/* using eq. (5.13) */`
    $Q^{new}(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$
    $S \leftarrow S'$ `/* state is updated */`
  end
  until S is terminal
end

Before learning, Q is initialised as a table containing arbitrary values (see alg. 2). A tabular[1] estimate Q is constructed with actions A_t as columns and with states S_t as rows. Then, at each

[1] tabular: information arranged in rows and columns, in the form of a table.

time step t, the agent performs a chosen action A_t, observes a reward R_t and then enters a new state S_{t+1}. Q is subsequently updated according to eq. (5.13) and the tabular estimate Q now contains updated elements $Q^{new}(S_t, A_t)$.

The core of the algorithm is a Bellman optimality equation for the action-value function (cf. eq. (5.12)) taking the form of a simple value iteration update. The learning rate α works as a weighted average of the new information (the temporal difference term) adding to the old value $Q(S_t, A_t)$. A learning rate of $\mathbf{0}$ would imply that the agent only retains prior information, whereas in case of $\alpha = \mathbf{1}$, the agent completely ignores prior knowledge to explore possibilities. The latter setting, however, would be optimal only in fully deterministic environments, where the next state in known.

Q-learning is a *model-free* reinforcement learning approach. Model-free methods do not need any kind of environment model at all. *"They cannot think about how their environments will change in response to a single action"* [Sutton and Barto, 2018]. Model-free architectures primarily focus on learning and thus can be regarded as *"trial and error - learners"*, as opposed to the model-based methods that rely on planning. Those settings, have certain advantages over model-based ones, not requiring sufficiently accurate models of complex environments.

5.3.1 Deep Q-Learning

Q was previously defined as a tabular estimate of the optimal action-value function q_*, being recursively updated at each and every time step t. While the popular Q-learning algorithm is a foundational idea in model-free reinforcement learning, recent advancements have shown that state spaces in recent applications, being more complex, are too large to learn all action values in all states separately [van Hasselt, Guez and Silver, 2015]. So, instead, a **differentiable and parameterised** Q-Learning variant $Q(s, a; \theta_t)$ is learned (cf. eq. (5.13)). This is the extension of Q-Learning to function approximation [Szepesvari, 2010].

$$\theta_{t+1} = \theta_t + \alpha(\underbrace{R_t + \gamma \max_a Q(S_{t+1}, a; \theta_t)}_{\text{target } y_t^Q} - \underbrace{Q(S_t, A_t; \theta_t)}_{\text{current value}}) \nabla_{\theta_t} Q(S_t, A_t; \theta_t), \tag{5.14}$$

where $y_t^Q = R_t + \gamma \max_\alpha Q(S_{t+1}, a; \theta_t)$ is referenced as the target Q value. That is, the current value $Q(S_t, A_t; \theta_t)$ is updated towards the target value y_t^Q, as with the temporal difference target value in eq. (5.13).

Remark 1: Under this deep learning framework, a shift in notation is observed, so that the action-value function q is sometimes symbolised as Q and the optimal action-value function q_* is also denoted as Q_*.

Remark 2: Recall, that the objective is to converge to optimal action value functions and then by extracting optimal policies, achieve greater expected returns. An optimal policy can be retrieved from: $\pi_* = \arg\max_a Q_*(S_t, A_t)$.

This is the point where *Deep Learning* steps in. Function approximation (modelling) involves training *Artificial Neural Networks* (*ANN*) on input–output data so as to approximate the underlying rules relating the inputs to the outputs [Basheer and Hajmeer, 2000]. It has been widely argued and generally accepted that reinforcement learning is benefited from a combination with powerful function approximators, so to scale to more complex tasks. Environments evolved to be too complex and large to be searched exhaustively, thereby demanding the generalisation

properties provided by artificial neural networks [Thrun and Schwartz, 1993]. That is, $Q(S_t, A_t; \theta_t)$, as an approximation, is represented using neural networks, a particularly useful method of building progressively more abstract representations of the data. In this instance, θ_t are the network parameters (weights).

Literature Review
Gerald Tesauro reported a remarkable performance on the game of backgammon with an ensemble of Q-Learning and an artificial neural network. The network learned from scratch to play the entire game and was ultimately able to reach human-expert level capabilities, beating **50%** of the time other networks trained on large expert data sets [Tesauro, 1991; Tesauro, 1990].

Deep Q-Network (DQN)
A *Deep Q-Network* (*DQN*) is a neural network with multiple layers and for a given state s it outputs a vector of action values in the form $Q(s, \cdot; \theta_t)$. It takes in information about the state of the environment and outputs the estimate of the action-value function for each possible action [Hernandez-Garcia and Sutton, 2019]. This network can be trained by adjusting the network weights θ_t, to reduce the mean-squared error in the Bellman equation between the approximate target values $y_t^Q = R_t + \gamma \max_a Q(s, a; \theta_t^-)$ and the optimal target values $y_t^Q = R_t + \gamma \max_a Q_*(s, a)$, where $a \in \mathcal{A}(s)$ and $s \in \mathcal{S}$. By way of the error minimisation, Q is accurately estimated so that $Q(S_t, A_t; \theta_t) \approx Q_*(S_t, A_t)$.

Note that this setting uses two sets of parameters; namely θ_t, which are learned every time step, and θ_t^-, as parameters of the target network, that are updated periodically. The update frequency of θ_t^- is a hyper-parameter also referenced as target network's update frequency [Hernandez-Garcia and Sutton, 2019]. On a further note, targets y_t^Q constitute a separate neural network which is not trained and remains frozen as an estimated target, yet is used to make more accurate predictions of Q values. *DQN*, introduced by Mnih et al. (2015), is a popular architecture to approximate the action-value function [Mnih et al., 2015].

For large state spaces, reinforcement learning is known to be unstable or even diverge when approximating the optimal action value function with a nonlinear function approximator such as a neural network. Mnih et al., 2015 use their DQN - Reinforcement Learning composite with two important ingredients. Experience replay, as a biologically inspired mechanism, randomises over the data, thereby removing correlations in the observations and smoothing over changes in the data distribution. Secondly, Q-function updates towards target values y_t^Q, are periodically met, so to reduce correlations with the target. Both propositions notably improved the algorithm's performance [Mnih et al., 2015].

5.4 Double Q-Learning

Although, traditional *Q-Learning* can optimally solve MDPs, in a way that Q converges to the optimal action-value function q_*, there are instances in some stochastic reinforcement learning environments, where Q-Learning performs poorly [van Hasselt, 2010]. Performance shortage is mainly exhibited in the form of overestimations of the action value functions [van Hasselt, 2010]. Thrun and Schwartz, 1993 further contend that the max operator in Q-learning (eq. (5.13)), always picks the largest value, making approximations highly sensitive to overestimations.

Positive biases arise in several contexts such as economics, decision making or bidding, thereby influencing beliefs, choice of actions and mechanistically cause overestimation of control and overconfidence [Steen, 2004; Capen, Clapp and Campbell, 1971; Thaler, 1988].

Hado van Hasselt, in mitigation of overestimation, proposed an alternative double estimator to find a more accurate estimate for the maximum value [van Hasselt, 2010]. *Double Q-Learning*, as a double estimator, stores two Q functions: Q^A and Q^B. Each of the Q functions is updated with a value from the other Q function for the next state. Note that in this instance, instead of using the value $Q^A(S, a^*) = \max_a Q^A(S, a)$ to update Q^A, as in Q-learning, Q^A is updated based on $Q^B(S, a^*)$. As Q^B was updated on the same problem, yet with a different set of experience samples, it can be considered a less biased estimate. In a similar fashion, Q^B is updated based on $Q^A(S, b^*)$. While, it is vital that both Q functions learn from separate sets of experiences, an action can be chosen from either Q^A or Q^B.

Algorithm 3: Double Q-Learning for estimating $Q_1 \approx Q_2 \approx q_*$

Input: Algorithm parameters: learning rate $\alpha \in (0, 1]$

Output: Optimal action-value function q_*

Initialise $Q(s, a)\ \forall s \in \mathcal{S}^+, a \in \mathcal{A}(s)$
for $1, \ldots, N$ *episodes* **do**
  Initialise S
  for $1, \ldots, t$ *episode time steps* **do**
    Choose action A from state S based on Q^A or Q^B
    Take action A
    Observe reward R and next state S'
    Update either Q^A or Q^B `/* could be a random selection */`
    if Q^A **then**

$$a^* = \arg\max_a Q^A(S', a)$$

$$Q^A(S, A) \leftarrow Q^A(S, A) + \alpha \left[R + \gamma Q^B \left[S', \overbrace{\arg\max_a Q^A(S', a)}^{\text{maximising action } a^*} \right] - Q^A(S, A) \right]$$

    else if Q^B **then**

$$b^* = \arg\max_a Q^B(S', a)$$

$$Q^B(S, A) \leftarrow Q^B(S, A) + \alpha \left[R + \gamma Q^A \left[S', \overbrace{\arg\max_a Q^B(S', a)}^{\text{maximising action } b^*} \right] - Q^B(S, A) \right]$$

    $S \leftarrow S'$ `/* state is updated */`
  end
  until S is terminal
end

The algorithm is shown to be convergent to the optimal action-value function q_*. *Double Q-learning* may underestimate the maximum expected action values, but certainly does not suffer from the overestimation bias to the extent Q-learning does [van Hasselt, 2010].

5.5 Actor-Critic Methods

So far, the approaches considered in sections 5.3 and 5.4 were value-based or *action-value methods* as often referenced. Their actions were selected on the basis of action-value functions. Policies were subsequently derived from the corresponding action-value estimates. There are added benefits, however, when parameterising for optimal policies, rather than action-value functions. Approximating a policy π is often a simpler and more computationally expedient approach to converge to optimal policies [Simsek, Algorta and Kothiyal, 2016].

Policy Gradient Methods is a family of algorithms that learn a parameterised policy whereby actions are selected without consulting an action-value function. All methods that follow this general scheme of policy-learning are termed as policy gradient methods, whether or not they also learn an approximate value function [Sutton and Barto, 2018].

An *Actor-Critic* architecture is a paradigm of a method that learns approximations to both policy and value functions. The *actor* constituent is a reference to the learned policy, while the *critic* component refers to the learned value function, which is usually an action-value function. Actor-Critic implementations are seeking to combine the strong points of actor-only and critic-only methods [Konda and Tsitsiklis, 1999].

Actor methods work with parameterising policies and map states to actions in a probabilistic manner. *Critic methods* rely exclusively on value function approximation and map states to expected cumulative future rewards. Simply put, the actor is concerned with control, while the critic addresses the problem of prediction. Those problems are distinct, but are solved concurrently to find an optimal policy [Bhatnagar et al., 2009]. In practice, actor and critic are implemented as deep neural networks and they approximate an optimal policy and a state-value function, respectively. Many works feature actor-critic frameworks to solve real-world robotic tasks [Haarnoja, A. Zhou, Hartikainen et al., 2018; Haarnoja, A. Zhou, Abbeel et al., 2018].

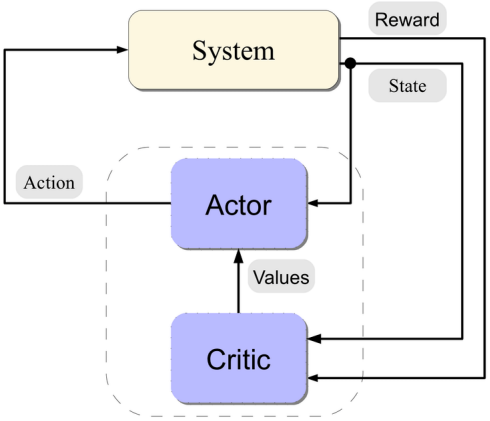

Figure 5.2: The Actor-Critic framework[2], source: Szepesvari, 2010

[2]System or Environment

5.6 Twin Delayed Deep Deterministic policy gradient algorithm (TD3)

5.6.1 Introduction

Value-based methods, such as Q-Learning may exhibit consistent overestimation biases, thereby leading to suboptimal policies and divergent behaviours [Thrun and Schwartz, 1993; Sutton and Barto, 2018]. Fujimoto, Hoof and Meger, 2018 contend that this behaviour persists in actor-critic settings. To address this concern, they introduce novel mechanisms to minimise overestimation biases and further enhance the performance of actor-critic settings. Their proposition, namely the **Twin Delayed Deep Deterministic policy gradient algorithm (TD3)**, builds on the Deep Deterministic Policy Gradient algorithm (DDPG) [Lillicrap et al., 2015] and naturally, by reduction, to its predecessor: Deterministic Policy Gradient algorithm (DPG) [Silver, Lever et al., 2014], as actor-critic methods that still suffer from overestimation biases that lead to policy breaking and brittle performance with respect to hyper-parameters. Note that under the mentioned algorithms the policy is **deterministic**.

The authors set forth their work as an attempt to solve overestimation bias and variance reduction in both value and policy updates function approximations, in actor-critic settings. Upon evaluation against other state of the art algorithms, such as DDPG [Lillicrap et al., 2015], PPO [Schulman, Wolski et al., 2017], ACKTR [Y. Wu et al., 2017], TRPO [Schulman, Levine et al., 2015] and SAC [Haarnoja, A. Zhou, Abbeel et al., 2018], authors report that *"TD3 matches or outperforms all other algorithms in both final performance and learning speed across all tasks"*. The algorithms were evaluated on seven continuous control domains from OpenAI gym [Brockman et al., 2016].

5.6.2 Proposed Methodology

Twin Delayed Deep Deterministic policy gradient algorithm (TD3) is a model-free, actor-critic algorithm suitable for environments with continuous[3] action spaces. Its superiority against other state of the art methods owes to the addition of three key ingredients: the ***Clipped Double Q-Learning***, the ***Delayed Policy Updates*** and the ***Target Policy Smoothing***.

Clipped Double Q-Learning
The authors introduce a novel clipped variant of Double Q-Learning [van Hasselt, 2010] to replace the critic. TD3 learns two Q-functions instead of one, hence "twin". Here, Q-functions are action-value functions. Each of the Q estimates is used to update the other as seen in alg. 3 and section 5.4 (Double Q-Learning). The value estimates, being independent, can be used to make unbiased predictions of the actions selected using the opposite value estimate. Using Double Q-Learning's original formulation, for a pair of actors $(\pi_{\phi_1}, \pi_{\phi_2})$, and a pair of critics $(Q_{\theta_1}, Q_{\theta_2})$, where π_{ϕ_1} is optimised with respect to Q_{θ_1} and π_{ϕ_2} with respect to Q_{θ_2}:

$$\begin{aligned} y_1 &= r + \gamma Q_{\theta_2'}\left(s', \pi_{\phi_1}(s')\right), \\ y_2 &= r + \gamma Q_{\theta_1'}\left(s', \pi_{\phi_2}(s')\right), \end{aligned} \tag{5.15}$$

where $y = r + \gamma Q_{\theta'}\left(s', \pi_{\phi}(s')\right)$ is the learning target, r is the reward, s' is the next state, γ is the discount factor, ϕ are the policy network parameters and θ are the value network parameters. Recall that in actor-critic settings the actor is effectively the policy π_ϕ, while the critic

[3] In a continuous action space actions are infinite, as opposed to a discrete action space whereby the agent chooses a distinct action from a finite action set [Masson and Konidaris, 2015].

corresponds to the value function Q_θ. Both policy and value function are treated as deep neural network representations and are not direct calculations.

Remark: For an optimal policy π_ϕ, the **objective function** of reinforcement learning is to maximise expected return: $\mathcal{J}(\phi) = \mathbb{E}_{\pi_\phi}\left[\sum_{t=0}^{T} \gamma^t r(s,a)\right]$.

To limit overestimation bias, the less biased value estimate Q_{θ_2} is upper-bounded by the biased estimate Q_{θ_1}. Thus, by *taking the minimum* between the two estimates, the *single* target of the Clipped Double Q-learning algorithm is defined as:

$$y_1 = r + \gamma \min_{i=1,2} Q_{\theta_i'}\left(s', \pi_{\phi_1}(s')\right). \tag{5.16}$$

Clipped Double Q-learning is proven that will converge to the optimal action-value function Q_*, as defined by the Bellman optimality equation, with a probability of 1 [Fujimoto, Hoof and Meger, 2018]. The value target does not introduce any additional overestimation. Note that, this update rule might still build an underestimation bias, but this is far preferable to overestimation biases, as the value of underestimated actions will not be propagated through the policy update [Fujimoto, Hoof and Meger, 2018].

Delayed Policy Updates
Target networks are widely leveraged as a tool for stability. By providing a stable objective in the learning procedure, those networks optimise for a greater coverage of the training data [Fujimoto, Hoof and Meger, 2018]. On a further note, those targets should remain fixed otherwise residual errors[4] will begin to accumulate. That is, to avoid overly divergent values, the target policy (actor) should remain frozen so that the error over multiple updates is reduced.

Results additionally show that the update rates and the reciprocity between actor and critic should be appropriately timed to circumvent divergent behaviours. The proposition concludes that this is best achieved when the policy target network is updated after a fixed number d of critic updates. In other words, the policy should be updated less frequently than the Q functions [OpenAI, 2018]. In principle, this should result in higher quality policy updates. Following the deterministic policy gradient algorithm (Silver, Lever et al., 2014, see appendix A.1 for the theorem's proof), the policy is updated with respect to Q_{θ_1} every $t \mod d$ time step iterations. For a deterministic target policy $\mu_{\phi_{target}}$, the policy is learned by maximising Q_{θ_1}:

$$\max_{\phi} \mathbb{E}_{s\sim\mathcal{B}}\left[Q_{\theta_1}\left(s, \mu_{\phi_{target}}(s)\right)\right], \tag{5.17}$$

where $s \sim \mathcal{B}$ denotes that states are sampled from an experience replay buffer[5] $\mathcal{B}$ as the one used in [Mnih et al., 2015].

Target Policy Smoothing
Learning a deterministic policy[6] tends to exploit regions whereby available data are insufficient to train the model, leading to catastrophic failures. This could be also regarded as overfitting or model bias [Kurutach et al., 2018]. When updating Q values, a deterministic policy may induce a high variance in the target Q value.

[4] residual error: difference between the observed and the estimated value.

[5] Replay buffers are used to store trajectories of experience when executing a policy in an environment. During training, replay buffers are queried for a subset of the transition between states (either a sequential subset or a sample) to "replay" the agent's experience [Eysenbach, Salakhutdinov and Levine, 2019; TensorFlow, 2018].

[6] deterministic policy: A policy $\pi : \mathcal{S} \to \mathcal{A}$ that represents a clear mapping from a given state s to an action a, as opposed to a stochastic policy whereby there is a probability of performing an action a under a given state s.

Regularisation is a strategy that addresses high variance. The proposed approach, target policy smoothing, intuitively suggests that similar actions should have similar values. A direct modification is made in the training process, so that the target value is enforced to fit around an expected target value with a very small deviation ϵ:

$$y = r + \mathbb{E}_{\epsilon}\left[Q_{\theta'}\left(s', \pi_{\phi'}(s') + \epsilon\right)\right], \tag{5.18}$$

where effectively a form of bootstrapping[7]is used to apply a smoothing effect. Adding a small amount of Gaussian noise ϵ to the policy in order to avoid overfitting, the target is updated as:

$$y = r + \gamma Q_{\theta'}\left(s', \pi_{\phi'}(s') + \epsilon\right), \epsilon \sim \text{ clip } (\mathcal{N}(0, \sigma), -c, c), \tag{5.19}$$

where ϵ is sampled from $\mathcal{N}(0, \sigma)$ and clipped within $(-c, c)$ to ensure that the target is close to the original action. A target action for a target policy $\mu_{\phi_{target}}$ can be defined as:

$$a'(s') = \text{ clip } \left(\mu_{\phi_{target}}(s') + \text{ clip } (\epsilon, -c, c), a_{Low}, a_{High}\right), \ \epsilon \sim \mathcal{N}(0, \sigma). \tag{5.20}$$

Without any change, if the Q-function approximator develops incorrect sharp peaks for some actions, the policy will quickly exploit them triggering brittle or faulty behaviours. Target policy smoothing attenuates those behaviours and keeps the target close to the original action [OpenAI, 2018]. This favours values for actions that are more robust to interference and noise.

Algorithm 4: Twin Delayed Deep Deterministic policy gradient algorithm (TD3)

Initialise critic networks $Q_{\theta_1}, Q_{\theta_2}$ and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialise target networks $\theta_1' \leftarrow \theta_1, \theta_2' \leftarrow \theta_2, \phi' \leftarrow \phi$
Initialise replay buffer $\mathcal{B}$

for $t = 1, \ldots, T$ *time steps in each episode* **do**
  Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon, \ \epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'
  Store transition tuple (s, a, r, s') in replay buffer $\mathcal{B}$
  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \ \epsilon \sim \text{ clip } (\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \tilde{a})$
  Update critics $\theta_i \leftarrow \arg\min_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
  if $t \mod d$ **then**
    Update ϕ by the deterministic policy gradient: `% Silver, Lever et al., 2014`
    $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a) \mid_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    Update target networks:
    $\theta_i' \leftarrow \tau\theta_i + (1 - \tau)\theta_i'$
    $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
  end
end

[7]In bootstrapping, an estimated value is used to update the same kind of estimated value, such as when using a Q value to update another Q value in Q-Learning.

5.6.3 Summary

Averting overestimation bias can considerably boost the performance of state-of-the-art implementations. Provided the stability the delayed target policy updates offer and the better estimated actions when explicitly smoothing target policies, TD3 supersedes other trending approaches in distinct evaluating tasks. By dint of optimising for more simple modifications, this proposition is rather versatile, generalising its usage to both online and offline reinforcement learning settings.

Part III

Research Summary

Chapter 6

Trading Agents

This chapter provides a brief introductory literature review around trading agents and thereupon delves deeper into the usage of Deep Reinforcement Learning and particularly of the Twin Delayed Deep Deterministic policy gradient algorithm (TD3) as a model-free, sequential decision-maker, that has no financial market knowledge, yet is able to learn through trial and error interactions and effectively navigate a trading landscape. Its performance is reported in chapter 7.

6.1 Literature review

A portfolio is constructed with the objective of maximising the expected returns given a certain risk level and a time horizon [Neuneier, 1995]. The Markowitz model, seen in chapter 3, was the first attempt to formalise mean-variance analysis as a constraint optimisation task and to explicitly propound that efficient portfolios conveniently address the interplay between risk and expected return. This idea remained as a pillar of Quantitative Finance and effectively shaped investments into a mathematical and scientific art [R. Zhou and Palomar, 2020].

More recently, the limitations of the vanilla Markowitz and its subsequent more sophisticated derivations such as the minimum Conditional Value-at-Risk portfolio [Rockafellar and Stanislav Uryasev, 2000], the minimum variance portfolio [Chopra and Ziemba, 1993], the risk parity portfolio [Roncalli and Weisang, 2016] and the maximum decorrelation portfolio [Christoffersen et al., 2012], have been rigorously studied, motivating the usage of more agnostic optimisation routines, that do not assume variance as the sole quantifier of risk.

A reinforcement learning agent learns to trade through trial and error interaction with the environment and by executing different strategies [Moody and Saffell, 2001]. Note that, this is distinct from other predictive models that are trained on historical data to forecast the financial product's price and subsequently buy or sell the asset. The latter methods, a somewhat controversial discussion, are still popular, yet have been often criticised as sub-optimal and heavily reliant on historical returns; an ill-based indicator for future performance [Moody and L. Wu, 1997; Sato, 2019]. Atsalakis and Valavanis, 2009 were the first to propose artificial neural networks (ANNs) as a medium for predicting the stock market behaviour. However, an ANN is sensitive to hyperparameter tuning and its network architecture, performance-wise [Soleymani and Paquet, 2020]. Alternatives, using Support Vector Machines (SVM) and Random Forests (RF) to improve accuracy [Patel et al., 2015], didn't report significant improvements.

This chapter sees the casting of reinforcement learning in portfolio allocation, being a multi-step and dynamically evolving optimisation solver that does not imply any inherent risk assumptions [Benhamou et al., 2020]. Contrary to other traditional methods, reinforcement learning is context-agnostic, yet is able to continuously stay in touch with the objective of maximising a reward, throughout the interaction with the environment. Reinforcement learning methods can be successfully applied to large state spaces without necessitating appropriate models. Neuneier, 1995 and Moody and L. Wu, 1997 were the first to successfully implement a critic and actor, respectively, method according to Fischer, 2018, inspiring further research.

While an umbrella literature review of Deep Reinforcement Learning in portfolio optimisation is not feasible within few pages, in the interest of conciseness, only seminal works will be briefly discussed as a medium to provide context for upcoming sections. The first application, of a critic-only approach was seen by Neuneier, 1995. Daily prices history of the DAX index were used along with other variables (such as interest rates, stock indices). With the reward function being modelled as the immediate (log) return, the agent executed discrete buy and hold actions. Dempster and Romahi, 2002; Cumming, 2015 demonstrated the applicability of reinforcement learning in high-frequency historical FX datasets. Both employed cumulative returns over the trading period, as a reward signal. Dempster and Romahi, 2002 traded with buy and sell signals, while Cumming, 2015 with open long, open short, close and idle state, using a novel method of transformed candle sticks.

With respect to actor-only approaches, Moody and Saffell, 2001 trained a Recurrent Reinforcement Learning (RRL) agent both to trade half-hourly U.S. Dollar/British Pound foreign exchange (FX) rate data and to learn optimal allocation strategies between the S&P 500 and the three-month Treasury Bills. As a reward function they use the differential Sharpe ratio [Moody and L. Wu, 1997], an objective function based on exponential moving averages, that is structured for on-line trading strategies. Jiang, D. Xu and Liang, 2017 applied Deep Recurrent Reinforcement Learning (Deep RRL) to manage a portfolio constituted by the eleven most traded cryptocurrencies by their volume of transactions, and Bitcoin as cash. Using high, low and close prices and log returns as the reward function, they reported 4-fold returns in 50 days.

Fewer implementations of actor-critic based traders are known. H. Li, Dagli and Enke, 2007 compared an actor-only to an actor-critic approach to predict stocks' short term movements. The analysis concluded in two different findings. First, the hybrid actor-critic architecture was found to be superior to its actor-only alternative. Secondly, the authors contend that an investor's psychology is to some extent predictable and thus can be partially imitated by reinforcement learning. Bekiros, 2010 mapped an agent's belief to fuzzy inference principles, in a fuzzy actor-critic reinforcement learning architecture that had a significantly higher predictive ability and profitability than a Recurrent Neural Network, a Markov-switching model and a Buy-and-Hold strategy. Performance was consistently superior when tested on the NASDAQ Composite, FTSE 100 and and Japan's Nikkei 225 indices.

The plethora of financial applications motivate potential playgrounds for reinforcement learning. For a more comprehensive view of the applications of reinforcement learning in Finance, such as in options pricing or robo-advising, a reader can consult Hambly, R. Xu and Huining Yang, 2022.

6.2 Deep Reinforcement Learning For Portfolio Optimisation

The portfolio optimisation problem is reformulated as a Markov Decision Process (MDP), where the environment is a portfolio of multiple financial products and the control is the fractions of the capital allocation [Sato, 2019]. In a real trading setting, the future returns and the state transition probabilities are not known. Thus, uncertainty in financial markets states makes the portfolio optimisation problem a stochastic optimal control problem in continuous state and action spaces – a problem which could be solved by model-free reinforcement learning.

In a standard reinforcement learning setting, the current knowledge of the market is formalised by the state s_t, where t is the time step. An action a_t is the decision with respect to the current allocation (portfolio weights of each asset) [Benhamou et al., 2020]. Execution of the action, yields a reward signal r_t from the environment and provides intuition on whether the actions are good or bad [Moody and Saffell, 2001]. A reward signal can either be the cumulative portfolio returns over a given time horizon, or other quantities such as the Sharpe ratio. Recall that an MDP models sequential choices made over time and under uncertainty [Rust, 1994]. That is, the agent is required to continuously re-observe the environment and perform a sequence of decisions thereafter. If the state has the **Markov property**[1], then this interaction is summarised as $\langle s_t, a_t, r_t, s_{t+1} \rangle$.

The desired objective is to maximise the reward signal [Sutton and Barto, 2018]; in this case cumulative returns. Recall from section 5.1.3 that the policy has to be frequently updated, to favour actions that ultimately lead to better long-term rewards.

The usage of deep learning is to represent the policy and the action-value function. In actor-critic settings, the actor is effectively the policy π_ϕ, while the critic corresponds to the value function Q_θ. Twin Delayed Deep Deterministic policy gradient algorithm (TD3) learns two Q-functions instead of one. TD3 falls under the umbrella of actor-critic algorithms that use deterministic policies and has seen very few implementations on portfolio management.

6.2.1 Modelling trading

Having introduced Deep Reinforcement Learning as a potential decision-maker for the portfolio optimisation problem, this part connects the higher-level intuitions with the formal notions of the experiment setting. This part summarises technical implementations, so that the experiment description in chapter 7, will be focused more on results and comparisons.

Prices
The experiments make use of business calendar days' adjusted closing prices[2] for a universe of handpicked assets. The price vector, for each asset is obtained as: $p_t = [p_1, p_2, p_3, \cdots]^T$ for a total of T collected prices. To capture a relative progression of prices between assets, since they have a significant variation in their absolute price values, the prices are normalised according to the following principle, such that a normalised price P_t is between 0 and 1:

$$P_t = \frac{p_t - p_{min}}{p_{max} - p_{min}}, \quad P_t \in [0, 1] \tag{6.1}$$

where p_{min} and p_{max} are the lowest and highest prices, of the same asset, observed during the entire period of T days. This is the input that the model is trained on.

[1] If a state contains information about all aspects of the past agent–environment interaction that matter for the future, the state is said to have the Markov Property (section 5.2.1).
[2] Adjusted closing prices account for stock splits and for other factors in corporate actions (e.g. dividends).

Returns
The immediate simple returns are computed as:

$$r_t = \frac{p_t}{p_{t-1}} - 1, \tag{6.2}$$

where p_t corresponds to the price of an asset at a particular day and p_{t-1} is the sampled price from the previous business day. As a vector, for each asset:

$$r_t = \left(0, \frac{p_{t=2}}{p_{t=1}}, \cdots \frac{p_T}{p_{T-1}}\right)^T, \tag{6.3}$$

where T symbolises the number of trading days.

Transaction costs
Following Ormos and Urbán, 2013; Betancourt and Chen, 2021; Hongyang Yang et al., 2020, transaction costs are set to be equal to a typical trading fee rate: $c_t = 0.1\%$. The trading fee remains constant and is applied to every transaction of every asset.

Portfolio wealth
The final portfolio wealth is derived as a product:

$$\mathbb{W} = \prod_{t=1}^{T} \Big[(1 - c_t) r_t \cdot w_{t-1} + 1 \Big], \tag{6.4}$$

where c_t is the transaction cost and w_{t-1} denotes the asset's weight in the portfolio.

Rewards
A simple reward signal might consider the profits made. At each time step t and for a transaction cost c_t, the reward is given, in relation to profit, as:

$$\rho_t = \sum_{i=1}^{m} (1 - c_t)\, r_{i,t} \cdot w_{i,t-1}, \quad i = 1, \cdots m \text{ assets.} \tag{6.5}$$

Differential Sharpe Ratio
Conforming with Modern Portfolio Theory (MPT), a fund manager should seek to maximise returns while considering risk, instead of exclusively optimising for profits. While Sharpe ratio [Sharpe, 1994] is the most widely used risk-adjusted metric, it does not facilitate on-line learning. Following Moody and Saffell, 1998; Moody and L. Wu, 1997, a more suitable reward signal, called *Differential Sharpe ratio*, is considered.

The differential Sharpe ratio is based on exponential moving averages of returns and standard deviations of returns (risk) and is particularly appealing to many trading strategies through expediting convergence and its ability to be used in live trading[3]; a very attractive feature in a reinforcement learning setting.

The first and second moments of returns distributions are defined as:

$$A_n = \frac{1}{n} R_n + \frac{n-1}{n} A_{n-1} \text{ and } B_n = \frac{1}{n} R_n^2 + \frac{n-1}{n} B_{n-1}. \tag{6.6}$$

[3]on-line (real-time) update of the parameters.

Then the *running Sharpe ratio* for a decay rate[4] η:

$$S_\eta(t) = \frac{A_\eta(t)}{K_\eta\big(B_\eta(t) - A_\eta^2(t)\big)^{\frac{1}{2}}}, \text{ with } K_\eta = \left(\frac{1-\frac{\eta}{2}}{1-\eta}\right)^2 \text{ and } A_\eta(0) = B_\eta(0) = 0. \tag{6.7}$$

Finally, differential Sharpe ratio (DSR) is derived as:

$$\begin{aligned} D_\eta(t) &\equiv \frac{\partial S_t}{\partial \eta} = \frac{B_\eta(t-1)\Delta A_\eta(t) - \frac{1}{2}A_\eta(t-1)\Delta B_\eta(t)}{(B_\eta(t-1) - A_\eta(t-1)^2)^{\frac{3}{2}}}, \quad \text{where} \\ S_\eta(t) &\approx S_\eta(t-1) + \eta \frac{\partial S_\eta(t)}{d\eta}\bigg|_{\eta=0} + O(\eta^2). \end{aligned} \tag{6.8}$$

The exponential moving estimates $A_\eta(t)$ and $B_\eta(t)$ are updated as:

$$\begin{aligned} A_\eta(t) &= A_\eta(t-1) + \eta\Delta A_\eta(t) = A_\eta(t-1) + \eta\big(R_t - A_\eta(t-1)\big), \\ B_\eta(t) &= B_\eta(t-1) + \eta\Delta B_\eta(t) = B_\eta(t-1) + \eta\big(R_t^2 - B_\eta(t-1)\big) \end{aligned} \tag{6.9}$$

In a simpler form, DSR is formulated as [Moody and L. Wu, 1997]:

$$\frac{\partial D_\eta(t)}{\partial R_t} = \frac{B_{t-1} - A_{t-1}R_t}{\big(B_{t-1} - A_{t-1}^2\big)^{\frac{3}{2}}} \tag{6.10}$$

Note that the Differential Sharpe ratio uses first and second order moments of returns. Therefore it was preferred over profits' maximisation, as a more complete reward signal able to consider risk and lead to more consistent strategies.

Discounted rewards
For a discount factor γ (was set equal to $\gamma = 0.99$) [5], the discounted rewards G_t in relation to the next state:

$$\begin{aligned} G_t &= \rho_{t+1} + \gamma\rho_{t+2} + \gamma^2\rho_{t+3} + \gamma^3\rho_{t+4} + \dots \\ &= \rho_{t+1} + \gamma\left(\rho_{t+2} + \gamma\rho_{t+3} + \gamma^2\rho_{t+4} + \dots\right) \\ &= \rho_{t+1} + \gamma G_{t+1} \end{aligned} \tag{6.11}$$

Actions
Following a policy π, the agent executes an action a_t at each time step t. The time step corresponds to the business day that the price was collected. That is, the agent observes the daily prices of m assets, one day at a time, and decides on an action, knowing previous prices, previous actions and rewards received from the environment in the form of returns. No knowledge of future states is available.

Actions are eventually yielded as a column vector $w_t = [w_1, w_2, \cdots w_m]$, for $i = 1, \cdots m$ assets in the portfolio, whereby the following conditions should hold:

[4] also referenced as adaptation rate [Moody and Saffell, 2001].
[5] The discount factor γ is conventionally set to 0.99. Although, François-Lavet, Fonteneau and Ernst, 2015 argue that an increasing discount factor improves performance and speeds-up the convergence.

$$\sum_{i=1}^{m} w_t = 1, \quad w_i \in [0, 1] \tag{6.12}$$

Network architecture
An action is selected from a target actor network $\pi_{\phi'}$ [6], where ϕ' is the updated actor network parameter ϕ. An action-value function, also referenced as critic, $Q(s, a) = \mathbb{E}_\pi [r_t \mid s, a]$ is the expected return of performing action a_t in state s and following policy π thereafter. Both actor and critic networks are constructed as two layer feedforward neural networks (FNN) with 256 hidden nodes, rectified linear units (ReLU) [Nair and Hinton, 2010] as activation functions between the layers and a final output layer. In the actor network, the output layer features as many dimensions as the number of assets.

A feedforward neural network (FNN) is a non-cyclical network with a distinct set of input units. A learning algorithm adjusts the weights between units so that the output gives some desired non-linear function of the input [Sanger, 1989]. For an encyclopedic review of deep neural networks, a reader is advised to follow the historical survey by Schmidhuber, 2014.

Activation functions
A rectified linear unit is the positive part of its argument; similar to a ramp function:

$$\textbf{ReLU:} \quad f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

The actor network includes a softmax ouptut layer [Fukushima, 2004; Bridle, 1989], which is responsible for producing actions (a_t). This activation function is particularly useful in this instance since the output class is very conveniently modelled as a probability distribution, thereby exponentially scaling the output between 0 and 1:

$$\textbf{softmax:} \quad \sigma(\vec{x})_i = \frac{e^{x_i}}{\sum_{i=1}^{m} e^{x_i}} \Rightarrow \sum_{i=1}^{m} a_i = 1$$

where $\vec{x}$ is the input vector (output tensor of the second hidden layer) and m is the number of the output dimensions; in this case the number of assets, so each asset has a separate weight allocation in each time step t. Actions a_i and weights w_i are used interchangeably here.

Hyperparameters
Deep neural networks exhibit strong performance on many machine learning tasks, but they are particularly sensitive to the setting of their hyperparameters [Domhan, Springenberg and Hutter, 2015]. Hence, experimenting with different **learning rates**, an actor learning rate of $\mathbf{3 \cdot 10^{-4}}$ and a critic learning rate of $\mathbf{5 \cdot 10^{-4}}$ for the stochastic gradient descent optimiser Adam [Kingma and Ba, 2014], were found to be beneficial performance-wise. Greater learning rates of the span near $\mathbf{10^{-3}}$ lead to overly unstable, divergent performances and volatility in returns, whereas lower ones that spread around $\mathbf{10^{-5}}$ were found to be severely underperforming even after many training epochs, potentially due to the structure of the objective function[7] that made optimisation very slow. Adam was also preferred in comparison with the Stochastic gradient descent (SGD) optimisation algorithm.

[6] The target policy π_ϕ may also be seen as μ_ϕ. Typically, μ_ϕ is more often used to describe deterministic policies.

[7] $\mathcal{J}(\phi) = \mathbb{E}_{\pi_\phi}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right]$.

Introducing **noise** to the target value estimate y [8], favours actions of higher value, close to the target estimates and not susceptible to perturbations. In terms of the policy noise and policy noise clipping, it was noticed that clipping for overall larger policy noises, could induce instability issues, albeit higher financial returns were also observed in some training experiments. Nonetheless, a higher randomisation is undesired when it comes to financial profit. Thus, they were set to a balanced average between overfitting prevention and stability. Table 6.1 summarises the choices of hyperparameters.

	hyperparameters	value
TD3 hyperparameters	optimiser	Adam [Kingma and Ba, 2014]
	actor learning rate	3×10^{-4}
	critic learning rate	5×10^{-4}
	batch size	64
	discount factor (γ)	0.99
	target network update rate (τ)	5×10^{-3}
	policy noise[9]	0.2
	policy noise clipping (c)[10]	$(-0.4, 0.4)$
	policy update frequency[11]	2
Architecture	critic hidden dim	256
	critic hidden layer(s)	2
	critic activation function	ReLU [Fukushima, 2004; Nair and Hinton, 2010]
	actor hidden dim	256
	actor hidden layer(s)	2
	actor activation function	softmax [Bridle, 1989]

Table 6.1: Hyperparameters and architecture in TD3

6.2.2 Back-testing

A back-test refers to a setting where the trading agent starts from a selected time point in the market history, without any knowledge of the future and paper trades (simulation of a trade without real money) from then onward [Jiang, D. Xu and Liang, 2017]. In the event of back-testing, two conventional hypotheses are often assumed. In a real trading environment, if a trading market is sufficiently liquid, the following two assumptions are close to realistic:

1. ***Zero slippage:*** Recalling the definition of *slippage* in section 2.1.3, the market liquidity for each asset is high enough, so that every trade can be executed, immediately, at the price the order was placed [Jiang, D. Xu and Liang, 2017].
2. ***Zero market impact:*** Recalling *market impact* from section 2.1.3, the capital invested is insignificant and thus, has no influence on the market [Jiang, D. Xu and Liang, 2017].

[8] $y = r + \gamma Q_{\theta'}\left(s', \pi_{\phi'}(s') + \epsilon\right), \epsilon \sim \text{clip}\left(\mathcal{N}(0, \sigma), -c, c\right)$.

[9] policy noise: noise added to target policy during critic update.

[10] noise clip: range to clip target policy noise.

[11] update frequency: frequency of delayed policy updates with respect to value function updates.

Chapter 7

Experiments

The previous chapter already described the aptness of deep reinforcement learning, although briefly, in a portfolio optimisation task. So far, it is described as a complex system that can sequentially transform time series into a set of allocation weights, while handling several biases and overestimations in the process. The deep learning community has provided an ample toolkit of functionalities and versatile functions that are channelled to the end of portfolio optimisation. Reinforcement learning knows no financial ground truth, yet it is able to discover structure in real financial price time series. This chapter will present empirical results as follows:

- Portfolio construction using the Minimum Spanning Tree, as a notion inspired from graph theory. It will be discovered how a larger universe of assets is able to be reduced to a smaller uncorrelated subset.
- Portfolio optimisation using Twin Delayed Deep Deterministic policy gradient (TD3). This section provides empirical results, through illustrations and different economic metrics. The constructed portfolio will be compared against the known Standard & Poor's 500 and Dow Jones Industrial Average indexes.

7.1 Cryptocurrencies, Stocks and ETFs Portfolio

A number of assets were initially handpicked to comprise a larger universe of assets. In a real trading environment, this process is an essential preliminary step and would involve fundamental research[1] or even technical analysis[2]. The products are selected in a way that the resulting universe would be inclusive of multiple asset classes and potentially of assets with varied return patterns. The universe of handpicked assets is shown in table 7.1.

[1] fundamental analysis: measuring the intrinsic value of a security and estimate whether it is overvalued or undervalued [Investopedia, 2021f]

[2] technical analysis: evaluate investments by analysing statistics from trading activities, such as price and volume [Investopedia, 2022c]

	Name	Ticker
Stocks	Microsoft Corporation	MSFT
	Exxon Mobil Corporation	XOM
	NVIDIA Corporation	NVDA
	Apple Inc.	AAPL
	Tesla, Inc.	TSLA
	Alphabet Inc.	GOOG
	Berkshire Hathaway Inc. Class B	BRK-B
	Meta Platforms, Inc.	FB
	JPMorgan Chase & Co.	JPM
	Johnson & Johnson	JNJ
	Amazon.com Inc.	AMZN
	Goldman Sachs Group Inc.	GS
	Boeing Co.	BA
	UnitedHealth Group Inc.	UNH
	Lockheed Martin Corporation	LMT
	Moderna, Inc.	MRNA
	The Coca-Cola Company	KO
	The Procter & Gamble Company	PG
	Advanced Micro Devices, Inc.	AMD
	Pfizer Inc.	PFE
Cryptocurrencies	Polkadot	DOT-USD
	Ethereum	ETH-USD
	Avalanche	AVAX-USD
	Solana	SOL-USD
	Cosmos	ATOM-USD
	Bitcoin	BTC-USD
	Cardano	ADA-USD
	Polygon	MATIC-USD
ETFs	Invesco QQQ Trust	QQQ
	Utilities Select Sector SPDR Fund	XLU
	iShares 0-5 Year TIPS Bond ETF	STIP
	iShares 20+ Year Treasury Bond ETF	TLT
	SPDR Portfolio S&P 500 High Dividend ETF	SPYD
	Vanguard Developed Markets Index Fund	VEA
	SPDR S&P 500 ETF Trust	SPY
	SPDR Dow Jones Industrial Average ETF Trust	DIA
Commodities	United States Oil Fund, LP	USO
	SPDR Gold Shares	GLD

Table 7.1: Universe of assets

The selected universe includes companies that specialise in technology, capital markets, healthcare, aerospace as well as cryptocurrencies, commodities and several ETFs; a few of them with assets invested in U.S. debt securities. Following Mantegna, 1998; Bonanno et al., 2004 and others, the universe is topologically decomposed with an aim of revealing community clusters that will be later further lessened to obtain a smaller uncorrelated subset. A minimum spanning tree construction with the assets from the universe is shown in fig. 7.1. There is an observable clustering of cryptocurrencies (upper-left corner), technology companies (bottom), healthcare services companies (mid-left), financial services companies (mid-top) and others.

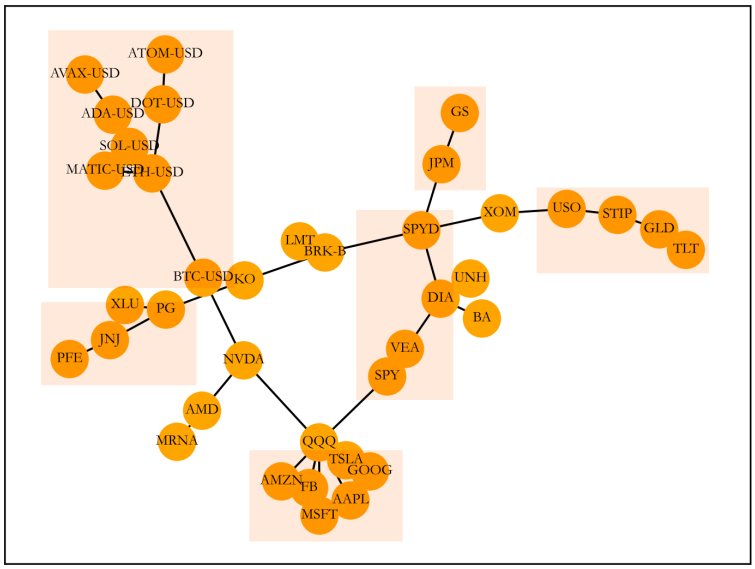

Figure 7.1: Minimum Spanning Tree representation of the universe

Using *Spearman's correlation*, the tree is able to cluster together companies of the same sector, as they are correlated in their returns' time series. Recalling section 4.2.1 and fig. 4.1, the distance $d'(x,y) = \sqrt{2(1-\rho_{x,y})}$ penalises weak and negative correlations, thereby putting together assets with strong correlations between their returns. An implicit advantage is that the least correlated an asset is, the larger its distance is from the others. Intuitively, the least correlated assets will lie on the outskirts of the tree. This graduality in distance provides an attractive feature, useful to further reduce the number of assets and perhaps the correlation between them. That is, by removing assets that do not belong to a tree edge, ensures that the remainder ones have the biggest distance from the centre and by implication, they are weakly correlated. Figure 7.2 shows a newly formed tree after removing assets from the first construction in fig. 7.1. Figure 7.3 depicts the outcome of a second reduction, being the third tree in succession and the one that eventually contains the final subset of assets.

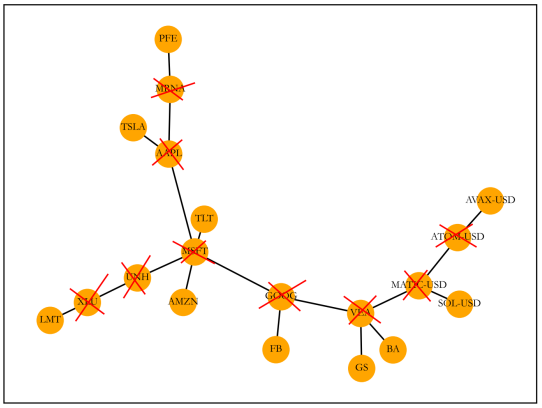

Figure 7.2: Node reduction (Tree no. 2)

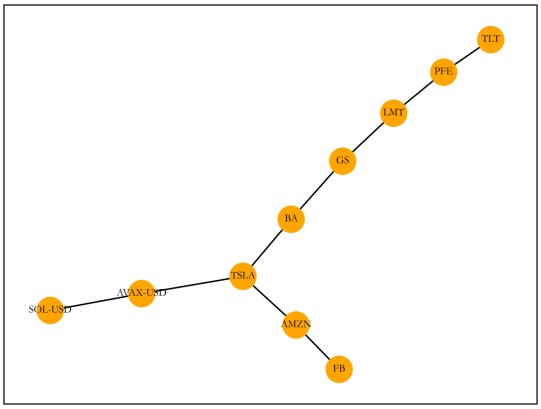

Figure 7.3: Final subset (Tree no. 3)

7.1.1 Portfolio Inspection

Seeing how the final universe of assets is filtered out of a larger universe, this part provides further insight on the obtained portfolio. The selected subset is summarised in table 7.2. Figure 7.4 shows the price time series of each one of the assets in the constructed portfolio.

	Name	Ticker	Description
Stocks	Tesla, Inc.	TSLA	Auto Manufacturer
	Meta Platforms, Inc.	FB	Internet Content & Information
	Amazon.com Inc.	AMZN	Internet Retail
	Goldman Sachs Group Inc.	GS	Capital Markets
	Boeing Co.	BA	Aerospace & Defence
	Lockheed Martin Corporation	LMT	Aerospace & Defence
	Pfizer Inc.	PFE	Drug Manufacturers – General
Cryptocurrencies	Avalanche	AVAX-USD	Blockchain with smart contract functionality
	Solana	SOL-USD	Blockchain with smart contract functionality
ETFs	iShares 20+ Year Treasury Bond ETF	TLT	$\geq$ 95% of assets in U.S. government bonds

Table 7.2: Portfolio constituents

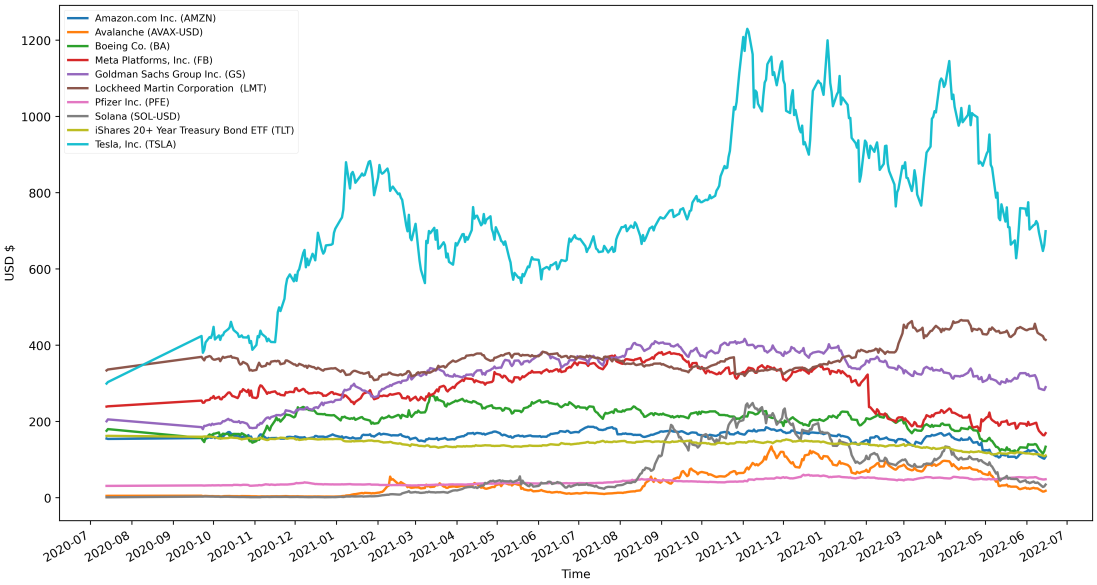

Figure 7.4: Asset prices USD (**$**)

Figure 7.5 shows the distribution of daily returns, as a percentage, for all assets. The variance of returns is included in the legend. As someone might postulate, a long-term treasury bond (TLT) exhibits less variable returns in comparison to other assets. On the other extreme, cryptocurrencies have far more variance in their returns' distribution, which almost appears as uniform.

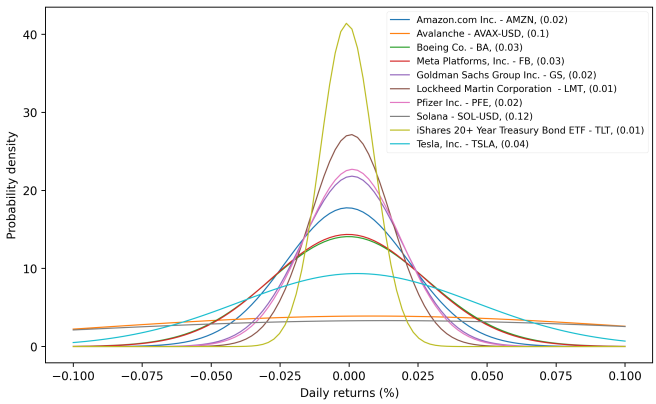

Figure 7.5: Distribution of daily returns

In the maps below, both correlation coefficients are tested. The subset, seems to be carefully filtered by the MST, as indicated by the weak correlations between the returns. There are a few pairs such as (FB – AMZN) and (BA – GS) that show a moderately positive correlation in both maps. There are some instances of slightly negative correlations, yet, the majority of pairs have almost no relationship to each other.

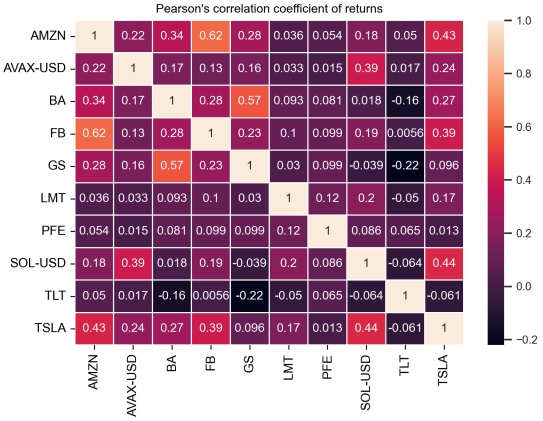

Figure 7.6: Pearson's correlation of daily returns

Figure 7.7: Spearman's correlation of daily returns

7.1.2 Evaluation

Upon selecting a diversified and uncorrelated subset as a product of the Minimum Spanning Tree, this portfolio is optimised and subsequently tested against known benchmarks. In order to ensure availability of prices for all of the assets, the obtained closing prices were sampled with business days as the default frequency and excess, previously recorded historical prices, were discarded, so for all of the price time series to have the same starting date. The Twin Delayed Deep Deterministic policy gradient algorithm (TD3) was trained for 400 episodes[3] as a suitable time window that allowed a stable convergence. Results are summarised in fig. 7.8 and table 7.4.

[3]One episode corresponds to one iteration over the whole training set until the terminal training date is reached.

	Start date	End date
Training	13-July-2020	15-June-2021
Back-test	16-June-2021	16-June-2022

Table 7.3: Training and back-test sets

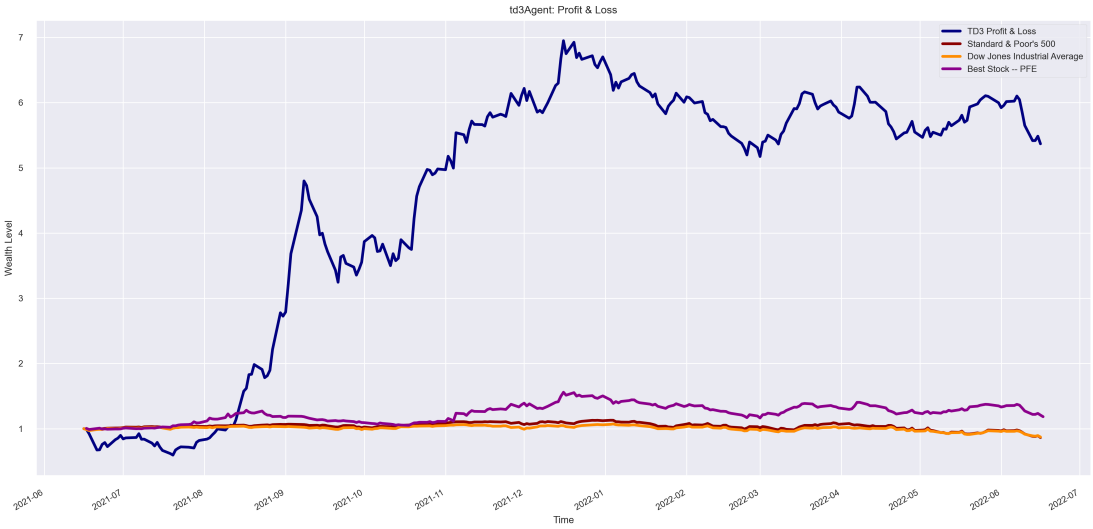

Figure 7.8: Return on investment (PnL) against Standard & Poor's 500, Dow Jones Industrial Average and the best stock 'PFE' (w.r.t. cumulative returns) over a period of one year.

	TD3	S&P500	DJIA	Best stock - PFE
Cumulative Returns	4.368	-0.137	-0.128	0.184
Profit and Loss (PnL)	5.368	0.863	0.873	1.184
Mean return	0.008	-0.001	-0.001	0.001
Standard deviation of returns	0.055	0.012	0.010	0.020
Skewness of returns	1.721	-0.510	-0.439	0.893
Kurtosis of returns	13.106	0.876	0.757	3.542
Sharpe ratio	2.352	-0.661	-0.743	0.698
Maximum drawdown	-1.221	-2.353	-2.267	-1.474
Conditional Value at Risk - 99%	-0.151	-0.039	-0.032	-0.044

Table 7.4: Summary of statistics

In the illustrated Profit and Loss (PnL) plot (fig. 7.8), the TD3 reinforcement learning agent outperforms both indices (Standard & Poor's 500, Dow Jones Industrial Average) and the best stock selected based on the highest PnL achieved at the end of the testing period. A positive excess kurtosis signifies extremity of deviations in returns as a result of the agent investing in fewer assets than the available ones. A possible explanation might feature transaction costs as the main resistance in frequent exchanges of wealth between assets. Positively skewed returns should be preferred according to the skewness criterion in section 2.3.1. The Sharpe ratio verifies a balanced and risk-adjusted performance of returns.

Note that due to the nature of discount, rewards far in the future worth exponentially less, yet this agent is far from a greedy approach that would seek immediate returns. Overall, the design of the reward function (Differential Sharpe ratio, see section 6.2.1) and the notion of discounted rewards, pose the TD3 trading agent as a rather suitable approach for a longer-term portfolio optimisation.

Chapter 8

Conclusion

An ensemble method of Deep Reinforcement Learning and Minimum Spanning Tree was examined as a complete method of constructing and optimising a diversified portfolio. During this analysis concepts from Machine Learning, graph theory and portfolio optimisation were discussed. The effectiveness of the Minimum Spanning Tree as a versatile and visually interpretable aid in portfolio decorrelation and the ability of deep reinforcement learning to transcend known benchmarks under the current market recession without supervised training, pose themselves as promising outcomes. The present chapter raises limitations alluded to during the experiment outline and puts potential research questions, forward, for consideration.

8.1 Limitations and Future Work

Deep neural networks are very complex in their interpretation, hence notoriously known as "black boxes". Their hidden layers exhibit many complex relationships. Oftentimes more than necessary. Nonetheless, financial applications favour transparency and interpretability to a level that a model's action can be explained by financial theories, as there is significant risk involved. Much of the state-of-the-art methods find benefits in biologically inspired heuristics to infer meaningful representations.

Overly complex architectures potentially lead to overfitting and sub-optimal policies. As observed in the experiments, a large number of hyper-parameters, not only makes model-free reinforcement learning susceptible to overfitting and lowers its degree of generalisation ability, but also makes tuning an arduous process. This may potentially hinder a wider adoption of reinforcement learning in portfolio optimisation.

The design of features and rewards could be closely aided by financial domain knowledge. Much of the literature calls for an integration of machine learning schemes with fundamentals, sentiment analysis and technical indicators, that would channel accumulated expert financial knowledge into refined reward signals [Sato, 2019; Hambly, R. Xu and Huining Yang, 2022; Filos, 2019].

Other possible directions would include, behaviour cloning; a process of acquiring knowledge through self-supervised observation of an expert performing a particular task. Unlike standard reinforcement learning regimes, learning through imitation heavily relies on the quality of the observation. Ultimately, an agent would be able to infer optimal policies when observing experts execute profitable strategies in speculative markets.

Last but not least, there is still a wide range of domains inaccessible to reinforcement learning due to the high cost and danger of interacting with the environment. Offline reinforcement learning breaks the assumption that the agent can interact with the environment. It is a paradigm that uses previously collected data without requiring additional online data collection [Fujimoto and Gu, 2021]. This is a particularly handy property as oftentimes data collection, in real-world financial applications, can be limited or expensive. Others characterise offline reinforcement learning as a tremendously promising method for making it possible to turn large data sets into powerful decision making engines [Levine et al., 2020].

Bibliography

Markowitz, Harry M. (1968). *Portfolio Selection: Efficient Diversification of Investments*. Yale university press.

Silver, David, Aja Huang et al. (2016). 'Mastering the game of Go with deep neural networks and tree search'. In: *nature* 529.7587, pp. 484–489.

Mnih, Volodymyr et al. (Feb. 2015). 'Human-level control through deep reinforcement learning'. In: *Nature* 518, pp. 529–33. DOI: 10.1038/nature14236.

Moravcík, Matej et al. (2017). 'DeepStack: Expert-Level Artificial Intelligence in No-Limit Poker'. In: *CoRR* abs/1701.01724. arXiv: 1701.01724. URL: http://arxiv.org/abs/1701.01724.

Li, Yuxi (2018). *Deep Reinforcement Learning*. DOI: 10.48550/ARXIV.1810.06339. URL: https://arxiv.org/abs/1810.06339.

Investopedia (2021a). *Investopedia – Liquid Asset Definition*. URL: https://www.investopedia.com/terms/l/liquidasset.asp.

– (2021b). *Investopedia – Financial Asset Definition*. URL: https://www.investopedia.com/terms/f/financialasset.asp.

– (2021c). *Investopedia – Liquidity Definition*. URL: https://www.investopedia.com/terms/l/liquidity.asp.

GALE, DAVID (1955). *THE LAW OF SUPPLY AND DEMAND*. Vol. 3. 1. Mathematica Scandinavica, pp. 155–169. URL: http://www.jstor.org/stable/24490348.

Gabrielsen, Alexandros, Massimiliano Marzo and Paolo Zagaglia (2011). *Measuring market liquidity: An introductory survey*. arXiv: 1112.6169 [q-fin.TR].

Massimb, Marcel N. and Bruce D. Phelps (1994). 'Electronic Trading, Market Structure and Liquidity'. In: *Financial Analysts Journal* 50.1, pp. 39–50. DOI: 10.2469/faj.v50.n1.39. eprint: https://doi.org/10.2469/faj.v50.n1.39. URL: https://doi.org/10.2469/faj.v50.n1.39.

Torre, Nicolo G., D. Ph. and Mark J. Ferrari (2000). *The Market Impact Model TM*.

Investopedia (2021d). *Investopedia – Slippage Definition*. URL: https://www.investopedia.com/terms/s/slippage.asp.

– (2021e). *Investopedia – Transaction Costs*. URL: https://www.investopedia.com/terms/t/transactioncosts.asp.

– (2022a). *Investopedia – ETF Definition*. URL: https://www.investopedia.com/terms/e/etf.asp.

Crosby, Michael et al. (2016). *BlockChain technology: beyond bitcoin.* Applied Innovation.

Nakamoto, Satoshi (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*.

Karame, Ghassan O., Elli Androulaki and Srdjan Capkun (2012). 'Double-Spending Fast Payments in Bitcoin'. In: *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. CCS '12. Raleigh, North Carolina, USA: Association for Computing Machinery, pp. 906–917. ISBN: 9781450316514. DOI: 10.1145/2382196.2382292. URL: https://doi.org/10.1145/2382196.2382292.

Zhang, Shijie and Jong-Hyouk Lee (2019). 'Double-Spending With a Sybil Attack in the Bitcoin Decentralized Network'. In: *IEEE Transactions on Industrial Informatics* 15.10, pp. 5715–5722. DOI: 10.1109/TII.2019.2921566.

Jiang, Zhengyao, Dixing Xu and Jinjun Liang (2017). *A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem*. arXiv: 1706.10059 [q-fin.CP].

Fang, Fan et al. (2022). *Cryptocurrency Trading: A Comprehensive Survey*. Financial Innovation.

Tsay, Ruey S. (2005). *Analysis of Financial Time Series: Second Edition*. John Wiley & Sons.

Miskolczi, Panna (June 2017). 'Note on simple and logarithmic return'. In: *Applied Studies in Agribusiness and Commerce* 11.1-2, pp. 127–136. DOI: 10.19041/APSTRACT/2017/1-2/16. URL: https://ojs.lib.unideb.hu/apstract/article/view/6976.

Rice, John A. (2006). *Mathematical Statistics and Data Analysis.* Third. Belmont, CA: Duxbury Press.

Harvey, Campbell R. et al. (2010). 'Portfolio selection with higher moments'. In: *Quantitative Finance* 10.5, pp. 469–485. DOI: 10.1080/14697681003756877. eprint: https://doi.org/10.1080/14697681003756877. URL: https://doi.org/10.1080/14697681003756877.

Naqvi, Bushra et al. (2017). 'Portfolio optimisation with higher moments of risk at the Pakistan Stock Exchange'. In: *Economic Research-Ekonomska Istraživanja* 30.1, pp. 1594–1610. DOI: 10.1080/1331677X.2017.1340182. eprint: https://doi.org/10.1080/1331677X.2017.1340182. URL: https://doi.org/10.1080/1331677X.2017.1340182.

Khan, Kanwal Iqbal et al. (2020). 'Sustainable Portfolio Optimization with Higher-Order Moments of Risk'. In: *Sustainability* 12.5. ISSN: 2071-1050. DOI: 10.3390/su12052006. URL: https://www.mdpi.com/2071-1050/12/5/2006.

Lo, Andrew W (2002). 'The statistics of Sharpe ratios'. In: *Financial analysts journal* 58.4, pp. 36–52.

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

Chekhlov, Alexei, Stanislav Uryasev and Michael Zabarankin (2005). 'Drawdown measure in portfolio optimization'. In: *International Journal of Theoretical and Applied Finance* 8.01, pp. 13–58.

Rockafellar, R. Tyrrell and Stanislav Uryasev (2000). 'Optimization of Conditional Value-at-Risk'. In: *Journal of Risk* 2, pp. 21–41.

Sarykalin, Sergey, Gaia Serraino and Stan Uryasev (Sept. 2008). 'Value- at-Risk vs Conditional Value-at-Risk in Risk Management and Optimization'. In: ISBN: 978-1-877640-23-0. DOI: 10.1287/educ.1080.0052.

Pflug, Georg Ch. (Sept. 2000). 'Some Remarks On The Value-At-Risk And The Conditional Value-At-Risk'. In: DOI: 10.1007/978-1-4757-3150-7_15.

Markowitz, Harry M. (1952). 'Portfolio Selection'. In: *The Journal of Finance* 7.1, pp. 77–91. ISSN: 00221082, 15406261. URL: http://www.jstor.org/stable/2975974 (visited on 06/06/2022).

Luenberger, David G. (1998). *Investment Science*. Oxford University Press.

Investopedia (2022b). *Investopedia – Short Selling*. URL: https://www.investopedia.com/terms/s/shortselling.asp.

Perold, Andre F (1984). 'Large-scale portfolio optimization'. In: *Management science* 30.10, pp. 1143–1160.

Kraft, D. (1988). *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR. URL: https://books.google.dk/books?id=4rKaGwAACAAJ.

Mantegna, Rosario (Feb. 1998). 'Hierarchical Structure in Financial Markets'. In: *arXiv.org, Quantitative Finance Papers* 11. DOI: 10.1007/s100510050929.

Graham, Ronald L and Pavol Hell (1985). 'On the history of the minimum spanning tree problem'. In: *Annals of the History of Computing* 7.1, pp. 43–57.

Encyclopedia of Mathematics (2014). *Encyclopedia of Mathematics – Greedy algorithm*. URL: http://encyclopediaofmath.org/index.php?title=Greedy_algorithm&oldid=34629.

Kruskal, Joseph B. (1956). 'On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem'. In: *Proceedings of the American Mathematical Society* 7.1, pp. 48–50. ISSN: 00029939, 10886826. URL: http://www.jstor.org/stable/2033241 (visited on 09/06/2022).

Prim, R. C. (1957). 'Shortest connection networks and some generalizations'. In: *The Bell System Technical Journal* 36.6, pp. 1389–1401. DOI: 10.1002/j.1538-7305.1957.tb01515.x.

Rafid, Im (May 2019). *Performance evaluation for Kruskal's and Prim's Algorithm in Minimum Spanning Tree using Networkx Package and Matplotlib to visualizing the MST Result*.

Bonanno, G. et al. (Jan. 2004). 'Networks of equities in financial markets'. In: *The European Physical Journal B - Condensed Matter and Complex Systems* 38, pp. 363–371. DOI: 10.1140/epjb/e2004-00129-6.

Onnela, J-P et al. (2002). 'Dynamic asset trees and portfolio analysis'. In: *The European Physical Journal B-Condensed Matter and Complex Systems* 30.3, pp. 285–288.

Millington, Tristan and Mahesan Niranjan (2021). 'Construction of minimum spanning trees from financial returns using rank correlation'. In: *Physica A: Statistical Mechanics and its Applications* 566, p. 125605. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2020.125605. URL: https://www.sciencedirect.com/science/article/pii/S0378437120309031.

Birch, Jenna and Kimmo Soramaki (Jan. 2015). 'Analysis of Correlation Based Networks Representing DAX 30 Stock Price Returns'. In: *Computational Economics* 47. DOI: 10.1007/s10614-015-9481-z.

Song, Jung Yoon, Woojin Chang and Jae Wook Song (2019). 'Cluster analysis on the structure of the cryptocurrency market via Bitcoin–Ethereum filtering'. In: *Physica A: Statistical Mechanics and its Applications* 527, p. 121339. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2019.121339. URL: https://www.sciencedirect.com/science/article/pii/S0378437119304893.

Stošić, Darko et al. (Oct. 2018). 'Collective behavior of cryptocurrency price changes'. In: *Physica A: Statistical Mechanics and its Applications* 507, pp. 499–509. DOI: 10.1016/j.physa.2018.05.050.

Hauke, Jan and Tomasz Kossowski (2011). 'Comparison of values of Pearson's and Spearman's correlation coefficient on the same sets of data'. In:

Ahlgren, Per, Bo Jarneving and Ronald Rousseau (2003). 'Requirements for a cocitation similarity measure, with special reference to Pearson's correlation coefficient'. In: *Journal of the American Society for Information Science and Technology* 54.6, pp. 550–560.

Hüttner, Amelie, Jan-Frederik Mai and Stefano Mineo (May 2016). 'Portfolio selection based on graphs: Does it align with Markowitz-optimal portfolios?' In: *Dependence Modeling* 6, pp. 63–87. DOI: 10.1515/demo-2018-0004.

Sutton, Richard S. and Andrew G. Barto (2018). *Reinforcement Learning: An Introduction*. Second. Cambridge, MA : The MIT Press.

Abe, Mitsunari et al. (2011). 'Reward Improves Long-Term Retention of a Motor Memory through Induction of Offline Memory Gains'. In: *Current Biology* 21.7, pp. 557–562. ISSN: 0960-9822. DOI: https://doi.org/10.1016/j.cub.2011.02.030. URL: https://www.sciencedirect.com/science/article/pii/S0960982211002211.

Taylor, Jordan A., John W. Krakauer and Richard B. Ivry (2014). 'Explicit and implicit contributions to learning in a sensorimotor adaptation task.' In: *The Journal of neuroscience : the official journal of the Society for Neuroscience vol. 34,8 (2014): 3023-32*. DOI: 10.1523/JNEUROSCI.3619-13.2014.

Shmuelof, Lior et al. (2012). 'Overcoming Motor "Forgetting" Through Reinforcement Of Learned Actions'. In: vol. 32. 42. Society for Neuroscience, 14617–14621a. DOI: 10.1523/JNEUROSCI.2184-12.2012. eprint: https://www.jneurosci.org/content/32/42/14617.full.pdf. URL: https://www.jneurosci.org/content/32/42/14617.

Huang, Vincent et al. (May 2011). 'Rethinking Motor Learning and Savings in Adaptation - Paradigms: Model-Free Memory for Successful Actions Combines with Internal Models'. In: vol. 70, pp. 787–801. DOI: 10.1016/j.neuron.2011.04.012.

Russell, Stuart J. and Peter Norvig (2021). Fourth. Pearson Education, Inc.

Kaelbling, L. P., M. L. Littman and A. W. Moore (1996). 'Reinforcement Learning: A Survey'. In: DOI: 10.48550/ARXIV.CS/9605103. URL: https://arxiv.org/abs/cs/9605103.

Črepinšek, Matej, Shih-Hsi Liu and Marjan Mernik (2013). 'Exploration and Exploitation in Evolutionary Algorithms: A Survey'. In: *ACM Comput. Surv.* 45.3. ISSN: 0360-0300. DOI: 10.1145/2480741.2480752. URL: https://doi.org/10.1145/2480741.2480752.

Brafman, Ronen and Moshe Tennenholtz (Jan. 2001). 'R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning.' In: vol. 3, pp. 953–958. DOI: 10.1162/153244303765208377.

Auer, Peter (2003). *Using Confidence Bounds for Exploitation-Exploration Trade-Offs*. JMLR.org, pp. 397–422.

Fortunato, Meire et al. (2017). *Noisy Networks for Exploration*. DOI: 10.48550/ARXIV.1706.10295. URL: https://arxiv.org/abs/1706.10295.

Szepesvari, Csaba (2010). *Algorithms for Reinforcement Learning*. Morgan and Claypool Publishers. ISBN: 1608454924.

Rust, John (1994). 'Chapter 51 Structural estimation of markov decision processes'. In: vol. 4. Handbook of Econometrics. Elsevier, pp. 3081–3143. DOI: https://doi.org/10.1016/S1573-4412(05)80020-0. URL: https://www.sciencedirect.com/science/article/pii/S1573441205800200.

Lillicrap, Timothy P. et al. (2015). *Continuous control with deep reinforcement learning*. DOI: 10.48550/ARXIV.1509.02971. URL: https://arxiv.org/abs/1509.02971.

Watkins, Christopher (Jan. 1989). 'Learning From Delayed Rewards'. PhD thesis.

van Hasselt, Hado, Arthur Guez and David Silver (2015). 'Deep Reinforcement Learning with Double Q-learning'. In: *CoRR* abs/1509.06461. arXiv: 1509.06461. URL: http://arxiv.org/abs/1509.06461.

Basheer, I.A and M Hajmeer (2000). 'Artificial neural networks: fundamentals, computing, design, and application'. In: *Journal of Microbiological Methods* 43.1. Neural Computting in Micrbiology, pp. 3–31. ISSN: 0167-7012. DOI: https://doi.org/10.1016/S0167-7012(00)00201-3. URL: https://www.sciencedirect.com/science/article/pii/S0167701200002013.

Thrun, Sebastian and Anton Schwartz (June 1993). 'Issues in Using Function Approximation for Reinforcement Learning'. In: *Proceedings of 4th Connectionist Models Summer School*. Erlbaum Associates.

Tesauro, Gerald (1991). 'Practical Issues in Temporal Difference Learning'. In: *Advances in Neural Information Processing Systems*. Ed. by J. Moody, S. Hanson and R.P. Lippmann. Vol. 4. Morgan-Kaufmann. URL: https://proceedings.neurips.cc/paper/1991/file/68ce199ec2c5517597ce0a4d89620f55-Paper.pdf.

– (1990). 'Neurogammon: a neural-network backgammon program'. In: *1990 IJCNN International Joint Conference on Neural Networks*, 33–39 vol.3. DOI: 10.1109/IJCNN.1990.137821.

Hernandez-Garcia, J Fernando and Richard S. Sutton (2019). 'Understanding multi-step deep reinforcement learning: a systematic study of the DQN target'. In: *arXiv preprint arXiv:1901.07510*.

van Hasselt, Hado (Jan. 2010). 'Double Q-learning.' In: *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010*, pp. 2613–2621.

Steen, Eric (Sept. 2004). 'Rational Overoptimism (and Other Biases)'. In: *American Economic Review* 94, pp. 1141–1151. DOI: 10.1257/0002828042002697.

Capen, E. C., R. V. Clapp and W. M. Campbell (1971). 'Competitive Bidding in High-Risk Situations'. In: *Journal of Petroleum Technology* 23, pp. 641–653.

Thaler, Richard H. (Mar. 1988). 'Anomalies: The Winner's Curse'. In: *Journal of Economic Perspectives* 2.1, pp. 191–202. DOI: 10.1257/jep.2.1.191. URL: https://www.aeaweb.org/articles?id=10.1257/jep.2.1.191.

Simsek, Özgür, Simón Algorta and Amit Kothiyal (2016). 'Why Most Decisions Are Easy in Tetris - And Perhaps in Other Sequential Decision Problems, As Well'. In: *ICML*.

Konda, Vijay and John Tsitsiklis (1999). 'Actor-Critic Algorithms'. In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen and K. Müller. Vol. 12. MIT Press. URL: https://proceedings.neurips.cc/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

Bhatnagar, Shalabh et al. (2009). 'Natural actor–critic algorithms'. In: *Automatica* 45.11, pp. 2471–2482.

Haarnoja, Tuomas, Aurick Zhou, Kristian Hartikainen et al. (2018). 'Soft actor-critic algorithms and applications'. In: *arXiv preprint arXiv:1812.05905*.

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel et al. (2018). 'Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor'. In: *CoRR* abs/1801.01290. arXiv: 1801.01290. URL: http://arxiv.org/abs/1801.01290.

Fujimoto, Scott, Herke van Hoof and David Meger (2018). *Addressing Function Approximation Error in Actor-Critic Methods*. DOI: 10.48550/ARXIV.1802.09477. URL: https://github.com/sfujim/TD3.

Silver, David, Guy Lever et al. (2014). 'Deterministic Policy Gradient Algorithms'. In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML'14. Beijing, China: JMLR.org, pp. I–387–I–395.

Schulman, John, Filip Wolski et al. (2017). 'Proximal Policy Optimization Algorithms'. In: *CoRR* abs/1707.06347. arXiv: 1707.06347. URL: http://arxiv.org/abs/1707.06347.

Wu, Yuhuai et al. (2017). 'Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation'. In: *CoRR* abs/1708.05144. arXiv: 1708.05144. URL: http://arxiv.org/abs/1708.05144.

Schulman, John, Sergey Levine et al. (2015). 'Trust Region Policy Optimization'. In: *CoRR* abs/1502.05477. arXiv: 1502.05477. URL: http://arxiv.org/abs/1502.05477.

Brockman, Greg et al. (2016). 'OpenAI Gym'. In: *CoRR* abs/1606.01540. arXiv: 1606.01540. URL: http://arxiv.org/abs/1606.01540.

Masson, Warwick and George Dimitri Konidaris (2015). 'Reinforcement Learning with Parameterized Actions'. In: *CoRR* abs/1509.01644. arXiv: 1509.01644. URL: http://arxiv.org/abs/1509.01644.

OpenAI (2018). *OpenAI – Twin Delayed DDPG*. URL: https://spinningup.openai.com/en/latest/algorithms/td3.html#documentation.

Eysenbach, Benjamin, Ruslan Salakhutdinov and Sergey Levine (2019). 'Search on the Replay Buffer: Bridging Planning and Reinforcement Learning'. In: *CoRR* abs/1906.05253. arXiv: 1906.05253. URL: http://arxiv.org/abs/1906.05253.

TensorFlow (2018). *Replay Buffers*. URL: https://www.tensorflow.org/agents/tutorials/5_replay_buffers_tutorial.

Kurutach, Thanard et al. (2018). 'Model-ensemble trust-region policy optimization'. In: *arXiv preprint arXiv:1802.10592*.

Neuneier, Ralph (1995). 'Optimal Asset Allocation using Adaptive Dynamic Programming'. In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky, M.C. Mozer and M. Hasselmo. Vol. 8. MIT Press. URL: https://proceedings.neurips.cc/paper/1995/file/3a15c7d0bbe60300a39f76f8a5ba6896-Paper.pdf.

Zhou, Rui and Daniel P. Palomar (2020). 'Understanding the Quintile Portfolio'. In: *IEEE Transactions on Signal Processing* 68, pp. 4030–4040. DOI: 10.1109/TSP.2020.3006761.

Chopra, Vijay Kumar and William T. Ziemba (1993). 'The Effect of Errors in Means, Variances, and Covariances on Optimal Portfolio Choice'. In:

Roncalli, T. and G. Weisang (2016). 'Risk parity portfolios with risk factors'. In: *Quantitative Finance* 16.3, pp. 377–388. DOI: 10.1080/14697688.2015.1046907. eprint: https://doi.org/10.1080/14697688.2015.1046907. URL: https://doi.org/10.1080/14697688.2015.1046907.

Christoffersen, Peter et al. (May 2012). 'Is the Potential for International Diversification Disappearing? A Dynamic Copula Approach'. In: *Review of Financial Studies* 25. DOI: 10.2139/ssrn.2066076.

Moody, John and Matthew Saffell (July 2001). 'Learning to trade via direct reinforcement'. In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 12, pp. 875–89. DOI: 10.1109/72.935097.

Moody, John and Lizhong Wu (1997). 'Optimization of trading systems and portfolios'. In: *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*, pp. 300–307. DOI: 10.1109/CIFER.1997.618952.

Sato, Yoshiharu (2019). 'Model-Free Reinforcement Learning for Financial Portfolios: A Brief Survey'. In: *ArXiv* abs/1904.04973.

Atsalakis, George S. and Kimon P. Valavanis (2009). 'Surveying stock market forecasting techniques – Part II: Soft computing methods'. In: *Expert Systems with Applications* 36.3, Part 2, pp. 5932–5941. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2008.07.006. URL: https://www.sciencedirect.com/science/article/pii/S0957417408004417.

Soleymani, Farzan and Eric Paquet (2020). 'Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder—DeepBreath'. In: *Expert Systems with Applications* 156, p. 113456. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.113456. URL: https://www.sciencedirect.com/science/article/pii/S0957417420302803.

Patel, Jigar et al. (2015). 'Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques'. In: *Expert Systems with Applications* 42.1, pp. 259–268. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2014.07.040. URL: https://www.sciencedirect.com/science/article/pii/S0957417414004473.

Benhamou, Eric et al. (2020). 'Bridging the gap between Markowitz planning and deep reinforcement learning'. In: *CoRR* abs/2010.09108. arXiv: 2010.09108. URL: https://arxiv.org/abs/2010.09108.

Fischer, Thomas G. (2018). *Reinforcement learning in financial markets - a survey*. eng. FAU Discussion Papers in Economics 12/2018. Nürnberg. URL: http://hdl.handle.net/10419/183139.

Dempster, Michael and YS Romahi (Nov. 2002). 'Intraday FX Trading: An Evolutionary Reinforcement Learning Approach'. In: pp. 697–708. ISBN: 978-3-540-44025-3. DOI: 10.1007/3-540-45675-9_52.

Cumming, James (2015). *An Investigation into the Use of Reinforcement Learning Techniques within the Algorithmic Trading Domain*.

Li, Hailin, Cihan H. Dagli and David Enke (2007). 'Short-term Stock Market Timing Prediction under Reinforcement Learning Schemes'. In: *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pp. 233–240. DOI: 10.1109/ADPRL.2007.368193.

Bekiros, Stelios (June 2010). 'Heterogeneous trading strategies with adaptive fuzzy Actor-Critic reinforcement learning: A behavioral approach'. In: *Journal of Economic Dynamics and Control* 34, pp. 1153–1170. DOI: 10.1016/j.jedc.2010.01.015.

Hambly, Ben M., Renyuan Xu and Huining Yang (Feb. 2022). 'Recent Advances in Reinforcement Learning in Finance'. In: URL: https://people.maths.ox.ac.uk/hambly/PDF/Papers/RL-finance.pdf.

Ormos, Mihály and András Urbán (2013). 'Performance analysis of log-optimal portfolio strategies with transaction costs'. In: *Quantitative Finance* 13.10, pp. 1587–1597. DOI: 10.1080/14697688.2011.570368. eprint: https://doi.org/10.1080/14697688.2011.570368. URL: https://doi.org/10.1080/14697688.2011.570368.

Betancourt, Carlos and Wen-Hui Chen (2021). 'Deep reinforcement learning for portfolio management of markets with a dynamic number of assets'. In: *Expert Systems with Applications*

164, p. 114002. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.114002. URL: https://www.sciencedirect.com/science/article/pii/S0957417420307776.

Yang, Hongyang et al. (Jan. 2020). 'Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy'. In: *SSRN Electronic Journal*. DOI: 10.2139/ssrn.3690996.

Sharpe, William F. (1994). 'The Sharpe Ratio'. In: *The Journal of Portfolio Management* 21.1, pp. 49–58. ISSN: 0095-4918. DOI: 10.3905/jpm.1994.409501. eprint: https://jpm.pm-research.com/content/21/1/49.full.pdf. URL: https://jpm.pm-research.com/content/21/1/49.

Moody, John and Matthew Saffell (1998). 'Reinforcement Learning for Trading'. In: *Advances in Neural Information Processing Systems*. Ed. by M. Kearns, S. Solla and D. Cohn. Vol. 11. MIT Press. URL: https://proceedings.neurips.cc/paper/1998/file/4e6cd95227cb0c280e99a195be5f6615-Paper.pdf.

François-Lavet, Vincent, Raphael Fonteneau and Damien Ernst (2015). 'How to discount deep reinforcement learning: Towards new dynamic strategies'. In: *arXiv preprint arXiv:1512.02011*.

Nair, Vinod and Geoffrey Hinton (June 2010). 'Rectified Linear Units Improve Restricted Boltzmann Machines'. In: vol. 27, pp. 807–814.

Sanger, Terence D (1989). 'Optimal unsupervised learning in a single-layer linear feedforward neural network'. In: *Neural networks* 2.6, pp. 459–473.

Schmidhuber, Jürgen (2014). 'Deep Learning in Neural Networks: An Overview'. In: *CoRR* abs/1404.7828. arXiv: 1404.7828. URL: http://arxiv.org/abs/1404.7828.

Fukushima, Kunihiko (2004). 'Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position'. In: *Biological Cybernetics* 36, pp. 193–202.

Bridle, John S. (1989). 'Training Stochastic Model Recognition Algorithms as Networks Can Lead to Maximum Mutual Information Estimation of Parameters'. In: *Proceedings of the 2nd International Conference on Neural Information Processing Systems*. NIPS'89. Cambridge, MA, USA: MIT Press, pp. 211–217.

Domhan, Tobias, Jost Tobias Springenberg and Frank Hutter (2015). 'Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves'. In: *Twenty-fourth international joint conference on artificial intelligence*.

Kingma, Diederik P. and Jimmy Ba (2014). 'Adam: A Method for Stochastic Optimization'. In: DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

Investopedia (2021f). *Investopedia – Fundamental Analysis*. URL: https://www.investopedia.com/terms/f/fundamentalanalysis.asp.

– (2022c). *Investopedia – Technical Analysis*. URL: https://www.investopedia.com/terms/t/technicalanalysis.asp.

Filos, Angelos (2019). *Reinforcement Learning for Portfolio Management*. DOI: 10.48550/ARXIV.1909.09571. URL: https://arxiv.org/abs/1909.09571.

Fujimoto, Scott and Shixiang Shane Gu (2021). 'A Minimalist Approach to Offline Reinforcement Learning'. In: *CoRR* abs/2106.06860. arXiv: 2106.06860. URL: https://arxiv.org/abs/2106.06860.

Levine, Sergey et al. (2020). 'Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems'. In: *CoRR* abs/2005.01643. arXiv: 2005.01643. URL: https://arxiv.org/abs/2005.01643.

Appendix A

Policy Gradient

A.1 Deterministic Policy Gradient Theorem

A.1.1 Theorem 1

The following proof from Silver, Lever et al., 2014 is presented here for completeness.

We consider a deterministic policy $\mu_\theta : \mathcal{S} \rightarrow \mathcal{A}$ with a parameter vector $\theta \in \mathbb{R}^n$. We define a performance objective $J(\mu_\theta) = \mathbb{E}\left[r_1^\gamma \mid \mu\right]$, and define the probability distribution $p(s \rightarrow s', t, \mu)$ and the discounted state distribution $\rho^\mu(s)$. This lets us to write the performance objective as an expectation,

$$\begin{aligned} J(\mu_\theta) &= \int_{\mathcal{S}} \rho^\mu(s) r(s, \mu_\theta(s)) \, \mathrm{d}s \\ &= \mathbb{E}_{s \sim \rho^\mu}\left[r(s, \mu_\theta(s))\right]. \end{aligned}$$

Regularity conditions A.1: $p(s' \mid s, a), \nabla_a p(s' \mid s, a), \mu_\theta(s), \nabla_\theta \mu_\theta(s), r(s, a), \nabla_a r(s, a), p_1(s)$ are continuous in all parameters and variables s, a, s' and x.

Regularity conditions A.2: There exists a b and L such that $\sup_s p_1(s) < b, \sup_{a,s,s'} p(s' \mid s, a) < b, \sup_{a,s} r(s, a) < b$, $\sup_{a,s,s'} \|\nabla_a p(s' \mid s, a)\| < L$, and $\sup_{a,s} \|\nabla_a r(s, a)\| < L$.

Theorem 1 **(*Deterministic Policy Gradient Theorem*).** Suppose that the MDP satisfies conditions A.l These imply that $\nabla_\theta \mu_\theta(s)$ and $\nabla_a Q^\mu(s, a)$ exist and that the deterministic policy gradient exists. Then,

$$\begin{aligned} \nabla_\theta J(\mu_\theta) &= \int_{\mathcal{S}} \rho^\mu(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a) \Big|_{a=\mu_\theta(s)} \mathrm{d}s \\ &= \mathbb{E}_{s \sim \rho^\mu}\left[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a)|_{a=\mu_\theta(s)}\right]. \end{aligned}$$

A.1.2 Proof of Theorem 1

Note that the regularity conditions A.1 imply that $V^{\mu_\theta}(s)$ and $\nabla_\theta V^{\mu_\theta}(s)$ are continuous functions of θ and s and the compactness of $\mathcal{S}$ further implies that for any θ, $\|\nabla_\theta V^{\mu_\theta}(s)\|$, $\left\|\nabla_a Q^{\mu_\theta}(s,a)|_{a=\mu_\theta(s)}\right\|$ and $\|\nabla_\theta \mu_\theta(s)\|$ are bounded functions of s. These conditions will be necessary to exchange derivatives and integrals, and the order of integration whenever necessary in the following proof. We have,

$$
\begin{aligned}
\nabla_\theta V^{\mu_\theta}(s) =& \nabla_\theta Q^{\mu_\theta}(s,\mu_\theta(s)) \\
=& \nabla_\theta \left(r(s,\mu_\theta(s)) + \int_{\mathcal{S}} \gamma p(s' \mid s,\mu_\theta(s)) V^{\mu_\theta}(s')\, \mathrm{d}s' \right) \\
=& \nabla_\theta \mu_\theta(s) \nabla_a r(s,a)|_{a=\mu_\theta(s)} + \nabla_\theta \int_{\mathcal{S}} \gamma p(s' \mid s,\mu_\theta(s)) V^{\mu_\theta}(s')\, \mathrm{d}s' \\
=& \nabla_\theta \mu_\theta(s) \nabla_a r(s,a)|_{a=\mu_\theta(s)} \\
&+ \int_{\mathcal{S}} \gamma \left(p(s' \mid s,\mu_\theta(s)) \nabla_\theta V^{\mu_\theta}(s') + \nabla_\theta \mu_\theta(s) \nabla_a p(s' \mid s,a)|_{a=\mu_\theta(s)} V^{\mu_\theta}(s') \right) \mathrm{d}s' \qquad (1) \\
=& \nabla_\theta \mu_\theta(s) \nabla_a \left(r(s,a) + \int_{\mathcal{S}} \gamma p(s' \mid s,a) V^{\mu_\theta}(s')\, \mathrm{d}s' \right)\Bigg|_{a=\mu_\theta(s)} \\
&+ \int_{\mathcal{S}} \gamma p(s' \mid s,\mu_\theta(s)) \nabla_\theta V^{\mu_\theta}(s')\, \mathrm{d}s' \\
=& \nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu_\theta}(s,a)|_{a=\mu_\theta(s)} + \int_{\mathcal{S}} \gamma p(s \to s', 1, \mu_\theta) \nabla_\theta V^{\mu_\theta}(s')\, \mathrm{d}s'.
\end{aligned}
$$

Where in (1) we used the Leibniz integral rule to exchange order of derivative and integration, requiring the regularity conditions, specifically continuity of $p(s' \mid s,a), \mu_\theta(s), V^{\mu_\theta}(s)$ and their derivatives w.r.t. θ. And now iterating this formula we have,

$$
\begin{aligned}
=& \nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu_\theta}(s,a)|_{a=\mu_\theta(s)} \\
&+ \int_{\mathcal{S}} \gamma p(s \to s', 1, \mu_\theta) \nabla_\theta \mu_\theta(s') \nabla_a Q^{\mu_\theta}(s',a)\Big|_{a=\mu_\theta(s')} \mathrm{d}s' \\
&+ \int_{\mathcal{S}} \gamma p(s \to s', 1, \mu_\theta) \int_{\mathcal{S}} \gamma p(s' \to s'', 1, \mu_\theta) \nabla_\theta V^{\mu_\theta}(s'')\, \mathrm{d}s'' \mathrm{d}s' \\
=& \nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu_\theta}(s,a)|_{a=\mu_\theta(s)} \\
&+ \int_{\mathcal{S}} \gamma p(s \to s', 1, \mu_\theta) \nabla_\theta \mu_\theta(s') \nabla_a Q^{\mu_\theta}(s',a)\Big|_{a=\mu_\theta(s')} \mathrm{d}s' \\
&+ \int_{\mathcal{S}} \gamma^2 p(s \to s', 2, \mu_\theta) \nabla_\theta V^{\mu_\theta}(s')\, \mathrm{d}s' \qquad (2) \\
&\vdots \\
& \int_{\mathcal{S}} \sum_{t=0}^{\infty} \gamma^t p(s \to s', t, \mu_\theta) \nabla_\theta \mu_\theta(s') \nabla_a Q^{\mu_\theta}(s',a)\Big|_{a=\mu_\theta(s')} \mathrm{d}s'.
\end{aligned}
$$

Where in (2) we have used Fubini's theorem to exchange the order of integration, requiring the regularity conditions so that $\|\nabla_\theta V^{\mu_\theta}(s)\|$ is bounded.

Now taking the expectation over S_1 we have,

$$\begin{aligned}
\nabla_\theta J(\mu_\theta) &= \nabla_\theta \int_{\mathcal{S}} p_1(s) V^{\mu_\theta}(s) \mathrm{d}s \\
&= \int_{\mathcal{S}} p_1(s) \nabla_\theta V^{\mu_\theta}(s) \mathrm{d}s \qquad (3) \\
&= \int_{\mathcal{S}} \int_{\mathcal{S}} \sum_{t=0}^{\infty} \gamma^t p_1(s) p\left(s \to s', t, \mu_\theta\right) \nabla_\theta \mu_\theta\left(s'\right) \nabla_a Q^{\mu_\theta}\left(s', a\right) \Big|_{a=\mu_\theta(s')} \mathrm{d}s' \mathrm{d}s \\
&= \int_{\mathcal{S}} \rho^{\mu_\theta}(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu_\theta}(s, a) \Big|_{a=\mu_\theta(s)} \mathrm{d}s,
\end{aligned}$$

where in (3) we used the Leibniz integral rule to exchange derivative and integral, requiring the regularity conditions, specifically so that $p_1(s)$ and $V^{\mu_\theta}(s)$ and derivatives w.r.t. θ are continuous. In the final line we again used Fubini's theorem to exchange the order of integration, requiring the boundedness of the integral as implied by the regularity conditions.

Technical
University of
Denmark

Kgs. Lyngby,
Denmark