Can Deep Reinforcement Learning solve the portfolio allocation problem ?

Eric Benhamou

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à Université Paris-Dauphine

Can Deep Reinforcement Learning solve the portfolio allocation problem?

Soutenue par
Eric Benhamou
Le **2 Octobre 2023**

École doctorale n°ED 543
Ecole doctorale SDOSE

Spécialité
Informatique

Composition du jury :

Nicole El Karoui Professeur Emérite,UPMC	*Président*
Damien Challet Professeur, Central Supélec	*Rapporteur*
Rahul Savani Professeur, Université de Liverpool	*Rapporteur*
Clémence Alasseur Directeur de Recherche, EDF	*Examinatrice*
Evgenia Passari Maitre de Conférence, Dauphine - PSL	*Examinatrice*
Jerome Busca Professeur, ESILV	*Examinateur*
Jamal Atif Directeur de Recherche, Paris Dauphine - PSL	*Directeur de thèse*
Rida Laraki Directeur de Recherche, Dauphine - PSL & Université de Liverpool	*Directeur de thèse*

Dauphine | PSL
UNIVERSITÉ PARIS

Acknowledgement

I have many people to thank for their support during my thesis journey. My sincerest gratitude goes to my two thesis supervisors, Prof. Jamal Atif from Paris-Dauphine University, PSL, and Prof. Rida Laraki, a CNRS research director in computer science at LAMSADE (Paris Dauphine University) and an honorary professor at the University of Liverpool (computer science department). They were not only excellent researchers but also fantastic mentors who motivated and encouraged me throughout the process.

In particular, I was fortunate to learn from Jamal's unparalleled knowledge in machine learning. His expertise extends to various areas such as optimization, reinforcement learning, deep learning, and more, making him a true expert in the rapidly growing field of artificial intelligence. I am also grateful to Rida, who was instrumental in my research, offering mathematical advice, proofreading my papers, and pushing me to put in the necessary effort to finalize my thesis manuscript.

I would like to express my gratitude to Prof. Rahul Savani and Prof. Damien Challet for agreeing to serve as rapporteurs for my thesis and for their meticulous review of the manuscript. Rahul challenged me to produce a refined version of my thesis by offering thorough recommendations that improved its consistency, accuracy, and comprehensiveness. It was a demanding process, but the effort was well worth it. Damien accepted to read the manuscript twice. I am honored to have the opportunity to present my work to the distinguished members of the jury, Prof. Nicole El Karoui, Dr Clémence Alasseur, Dr Evgenia Passari and Prof Jerome Busca.

I would also like to express my gratitude to the members of the AI for Alpha team for their unwavering support throughout my thesis journey, particularly Beatrice Guez, David Saltiel, Nicolas Paris, Stéphane Fadda, and Jean-Jacques Ohana. I am thankful for the energy and inspiration brought by the members of the MILES research group at LAMSADE, including Céline Beji, Raphael Pinot, Laurent Meunier, and the PhD students. The insightful conversations with Florian Yger, Yann Chevaleyre, Benjamin Negrevergne, and Clément Royer also made a significant impact on my thesis. Lastly, I would like to acknowledge the supportive environment created by Prof. Daniela Grigori, head of LAMSADE, Université Paris-Dauphine, PSL Université de recherche, which fostered reflection and discussion.

My sincerest thanks go out to all my co-authors, including David Saltiel, Jean-Jacques Ohana, Sandrine Ungari, Abhishek Mukhopadhyay, Serge Tabachnick, Sui Kai Wong, François Chareyron, Béatrice Guez, and Nicolas Paris. I would like to specifically recognize the significant contributions of David Saltiel and Jean-Jacques Ohana in co-editing, reviewing, and providing valuable feedback on my research.

I am eternally grateful to my family, including my wife Béatrice and my two children Noemie and Raphael, for their unwavering support and patience. They put up with my countless hours in front of a computer, day and night, and their encouragement and understanding gave me the strength and motivation to complete this thesis. I am incredibly fortunate to have them by my side. Thank you from the bottom of my heart.

Résumé

Les modèles développés en théorie du portefeuille reposent principalement sur des principes statistiques et économiques. A la base, il y a un modèle. De celui-ci, en supposant les marchés financiers rationnels et sans arbitrage, on en déduit des relations. Ainsi, si on part du principe de représentation du risque par ratio de Sharpe, on aboutit au portefeuille de Markowitz. Ces principes reposent sur des biais cognitifs en termes de risque (risque réduit à la variance) et sur des limitations en termes d'optimisation (optimisation quadratique). Si on souhaite s'en affranchir et appliquer des principes d'apprentissage automatique aux marchés financiers, on n'a plus besoin de faire de choix de modèle. On cherche simplement à trouver des relations entre les données sans à priori. Dans cette thèse, nous examinons la question centrale de savoir si l'apprentissage par renforcement profond (DRL) peut fournir de nouvelles méthodes d'allocation de portefeuille. Nous soutenons que le DRL offre de nouvelles méthodes reliant directement états et actions et est donc capable de s'adapter dynamiquement à un environnement changeant plus rapidement.

Nous commençons par rappeler les fondements du DRL avant de revoir la question de l'allocation de portefeuille. Ceci nous permet de formuler les questions posées et adressées par cette thèse, à savoir comment utiliser des informations supplémentaires par rapport aux deux premiers moments des actifs du portefeuille et comment valider que cette approche se généralise en dehors de la période d'apprentissage. Ces travaux adressent la question de l'utilisation pratique du DRL sur données non stationnaires et fortement bruitées. Nous appliquons le DRL au cas d'allocation multi-actifs afin de cerner les points clefs de l'approche. Nous montrons empiriquement que le DRL permet de dépasser l'état de l'art des méthodes d'allocation de portefeuille et de mieux s'adapter aux conditions de marché. Le choix de l'architecture par réseaux de convolutions capture mieux la dépendance entre les données de marché et s'adapte à des changements de situation comme la crise du Covid. Nous étendons l'usage du DRL à un problème de sélection de modèles quantitatifs de ciblage de volatilité, développant ainsi une approche dite à base de modèles.

Nous expliquons ensuite pourquoi l'approche DRL généralise les approches quantitatives classiques de théorie du portefeuille en étendant le problème d'optimisation à un problème de contrôle optimal multi périodes. Nous montrons que les méthodes classiques peuvent etre reformulées comme un problème simple de RL et que le cadre général du deep RL va bien au dela des méthodes traditionnelles de portefeuille en finance. Nous montrons ensuite que les méthodes DRL réalisent des réductions de variance et analysons le cas particulier de la méthode acteur critique en l'interprétant comme la résolution d'un problème de simulation de Monte Carlo par variable de contrôle optimal. Nous étudions aussi les similitudes entre l'apprentissage par renforcement et l'apprentissage supervisé. Nous exhibons notamment que l'apprentissage par renforcement par descente de gradient est en fait un apprentissage supervisé avec une fonction de perte d'entropie croisée et des labels égaux aux récompenses optimales. Ce résultat bien que théorique en raison de l'impossibilité de connaître à l'avance les récompenses optimales établit un lien profond entre les deux méthodes d'apprentissage.

Nous concluons cette thèse en résumant nos contributions et présentons des développements futurs, que ce soit par des prolongements naturels ou des questions nouvelles que cette thèse suscite.

Abstract

The portfolio theory models are built on statistical and economic principles. The underlying idea is to have a model from which relationships and mathematical equations between assets can be derived, given the absence of arbitrage and frictionless markets. The initial model, which represents risk using the Sharpe ratio, leads to the Markowitz portfolio. However, these principles are limited by cognitive biases towards risk (limited to variance) and optimization restrictions (quadratic optimization). To overcome these limitations and apply automatic learning principles to financial markets, a model-agnostic approach is taken, where relationships between the data are sought without prior assumptions. This thesis explores the potential of deep reinforcement learning (DRL) in providing innovative solutions to the portfolio allocation problem.

In this thesis, we start by exploring the potential of DRL in addressing the portfolio allocation problem. We begin by revisiting the fundamentals of DRL to provide the framework to incorporate additional information besides the first two moments of the portfolio assets and to validate that this approach generalizes outside the learning period. Our work specifically focuses on the practical application of DRL on non-stationary and highly noisy data, by examining the case of multi-asset allocation among various hedging strategies. Our empirical findings show that DRL has the potential to outperform traditional portfolio allocation methods and to better adapt to changing market conditions. We highlight the advantages of using a convolutional network architecture in capturing the dependence between market data and adapting to changing situations. We also showcase the potential of DRL in model selection for volatility-targeting models. Our proposed methodology, termed as "selecting the experts with DRL", involves leveraging DRL to identify and pick the most optimal models from the pool of volatility-targeting models available. Our implementation of this approach serves a dual purpose. On one hand, it showcases the efficacy of DRL in the realm of model selection, demonstrating its usefulness in this context. On the other hand, it also highlights how DRL can be integrated as a building block in more conventional financial methods, thus expanding its scope of application in the financial domain.

Following the practical demonstrations of the effectiveness of Deep Reinforcement Learning (DRL), we turn our attention to a more theoretical aspect. Specifically, we highlight how DRL introduces a new dimension to traditional portfolio theory methods by transforming the initial optimization problem into a multi-period optimal control problem. This expands the scope of portfolio optimization beyond the traditional methods. Compared to traditional portfolio methods, the DRL approach can incorporate a larger number of financial variables to determine portfolio allocation, which makes it an innovative and appealing concept. Additionally, it considers not only immediate rewards but also future rewards, making it a much richer model that can capture regime changes. Ultimately, instead of relying solely on simple portfolio weights, the DRL approach seeks to discover a function, resulting in a more comprehensive model. We then demonstrate that actor-critic methods in DRL lead to a reduction in variance and examine the specific case of the critical actor method by viewing it as a resolution of a Monte Carlo simulation problem through optimal control variables. We conclude by comparing reinforcement learning and supervised learning, with a focus on gradient descent policy methods in RL and its connection with cross-entropy supervised learning.

We end this thesis by summarising our contributions and present future developments, whether by natural extensions or new questions that this thesis raises.

Contents

List of Symbols

Algebra

$\mathbb{R}$	Set of real numbers
$\mathbb{N}$	Set of natural integers
$\mathbb{R}^d$	Set of d-dimensional real-valued vectors
x^+	Positive part of the real number x defined as $max(0, x)$
$\|.\|_1$	L_1 norm
$\|.\|_2$	Euclidean or L_2 norm
M^T	Transpose of the matrix M
M^{-1}	Inverse of the matrix M
$Tr(M)$	Trace of the matrix M
$\mathcal{S}^p_{++}$	Symmetric definite positive matrices
$\text{vec}(M)$	Vectorization of the matrix M meaning we stack each column on top of each other starting by the left-most column
$\mathbb{P}$	Probability operator
$\mathbb{P}(X\|Y)$	Probability of X conditional of Y
$\mathbb{E}[.]$	Expectation operator
$(\Omega, \mathcal{F}, \mathcal{P})$	Probability space
$\mathcal{A}(\Omega, \mathcal{G}, \mathcal{P})$	σ – algebra of the probability space
$\mathcal{L}^2(\Omega, \mathcal{F}, \mathcal{P})$	Space of square integrable random variables
$\mathbb{E}[. \mid Y]$	Conditional expectation operator with respect to Y
$D_{\text{KL}}(p\|q)$	Kullback–Leibler divergence of distribution p from a
$H(p, q)$	Relative entropy of distribution p with respect to q
$\text{StdDev}(\{x_1, \ldots, x_n\})$	Standard deviation of the n uplet $\{x_1, \ldots, x_n\}$

Reinforcement Learning

s_t	State at time step t
a_t	Action at time step t
o_t	Observation at time step t
$\mathcal{S}$	Space space
$\mathcal{A}$	Action space
$\mathcal{O}$	Observation space
$\mathcal{T}(s, a, s')$	Transition probability function from state s, action a to next state s'
$\mathcal{R}(s, a)$	Reward function from state s and with action a
t	discrete time step t
γ	Discount factor
α_k	Learning rate in Gradient descent at step k
$\nabla_\theta F$	Gradient with respect to θ of the vectorial function F
π	Agent policy
τ	Trajectory starting at time $t = 0$
τ_t	Trajectory starting at time t
$R(\tau)$	Cumulated reward of the trajectory τ
$V^\pi(s^)$	V-value function according to policy π starting in state s
$Q^\pi(s, a)$	Q-value function or states-action value function according to policy π starting in state s and with first action a
$A^\pi(s, a)$	Advantage function according to policy π starting in state s and with first action a
$J(\theta)$	Expected value of the cumulative rewards

Financials

DD	Drawdowns
MDD	Maximum Drawdowns
SR	Sharpe Ratio
YF	Year fraction

Abbreviations

RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
SL	Supervised Learning
UL	Unsupervised Learning
MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process
PGM	Policy Gradient Methods not be confused with Probabilistic Graphical Models
AC	Actor-Critic
A-AC	Advantage Actor-Critic
ACM	Actor-Critic Methods
TD	Temporal Difference

List of Figures

List of Tables

Forewords

The only thing that makes life possible is permanent, intolerable uncertainty; not knowing what comes next

Le Guin (1969), an American author best known for her works of speculative fiction

Motivations

Among the types of machine learning methods, reinforcement learning (RL), is considered one of the most fascinating forms of machine learning[1]. This is due to its close resemblance to the human learning process, where an agent learns through experience and reinforcement. The key concept is to be *reinforced* by the experience. Unlike supervised learning, it does not require prior knowledge of the solution, enabling the discovery of solutions to problems where the answer is unknown. Furthermore, it demonstrates the ability to uncover novel approaches to problem-solving, as demonstrated by the game of Go (Silver et al. 2017), where the machine played in a very different way from traditional human players.

Reinforcement Learning (RL) has been a subject of growing interest among researchers, and this is not surprising given its long history and strong foundations. The origins of reinforcement learning (Sutton & Barto 2018), can be traced back to the late 1950s with the development of optimal control theory (Bellman 1957*a*) or (Pontryagin et al. 1962). Although the theory was not

[1]For instance, Richard Sutton said in a 2018 interview with Wired: "Reinforcement learning is the most powerful form of machine learning." Likewise, Demis Hassabis said in a 2017 interview with The Verge: "Reinforcement learning is the key to artificial general intelligence.". Last but not least David Silver said in a 2016 interview with MIT Technology Review: "Reinforcement learning is one of the most exciting areas of machine learning research today, and it has the potential to revolutionize many different industries."

presented with the modern formalism of reinforcement learning, it contained many of the essential elements that are still present today.

One of the most important elements of current reinforcement learning is the Markov Decision Process (MDP), which provides a mathematical framework for modeling decision-making problems. The MDP is a stochastic process that models the interactions between an agent and its environment, where the agent makes decisions based on its current state and receives rewards in response. The Bellman equation, which is a key part of the MDP, provides a way to quantify the expected value of taking a particular action in a given state.

Another key aspect of reinforcement learning is the mathematical justification of convergence, which is a critical issue in the field. The convergence of reinforcement learning algorithms ensures that the RL algorithm will eventually find the optimal policy, even in complex or high-dimensional environments. This is why reinforcement learning is such an attractive method for solving problems in fields as diverse as robotics, control systems, game theory, and artificial intelligence.

However, despite some interesting applications like the TD-Gammon, a Backgammon-playing program (Tesauro 1992, 1994), reinforcement learning remained quite unknown to the public until the last decade. The success of the DQN agent in playing Atari games from raw pixels and inputs marked a major turning point in the field of reinforcement learning (Mnih et al. 2013). Prior to this breakthrough, RL had been relatively unknown to the public and was used primarily for academic purposes. However, the demonstration of the DQN's ability to learn and play multiple games without prior knowledge of the rules was a game-changer, bringing the potential of reinforcement learning to the forefront of public attention. Indeed the combination of Deep learning with RL that created a new field called Deep Reinforcement Learning (DRL) was a game changer.

The DQN success paved the way for further explorations of DRL both in terms of algorithms and potential applications in other fields. The versatility and robustness of DRL make it a promising candidate for solving complex problems that involve decision-making, such as portfolio allocation.

In this thesis, we investigate the potential of DRL to provide new solutions to the challenging problem of portfolio allocation.

Layout

This thesis includes selected contributions that have resulted in conference publications and participations, as listed in the publications section. While these contributions may not be major, they are part of the ongoing efforts of the wider machine-learning community in exploring the usage of machine learning in finance. It is hoped that these contributions will add to the existing body of knowledge in the field of machine learning in finance and help to further our understanding of the potential of deep reinforcement learning for portfolio allocation.

In part one, we conduct experiments demonstrating that Deep Reinforcement Learning (DRL) is a viable approach for portfolio allocation.

Chapter 2 is focused on exploring the use of Deep Reinforcement Learning (DRL) in the context of asset management. The standard financial methods are based on forecasting expected returns and risks and finding the best portfolio allocation. However, they do not take into account the dynamic relationship between market conditions and hedging strategies. This is where DRL can

provide a significant advantage as it is capable of creating a dynamic connection between market information and allocation decisions (Xiong et al. 2018). Besides taking additional contextual information and implementing a one-day lag between observations and actions to stick to reality, we show that it is easy in this method to incorporate the management of the leverage of strategies. Experiments show that it can outperform traditional methods in terms of returns and risk, which is confirmed by a walk-forward analysis, which is similar to cross-validation but without any knowledge of future information. The findings of this chapter demonstrate that DRL can play a crucial role in solving the portfolio allocation problem in a noisy and non-stationary environment, especially when it comes to hedging strategies where regime changes and noisy data are significant drivers of asset dynamics. Although the experiment is focused on hedging strategies, the results can be easily translated to other allocation problems that face similar challenges.

Chapter 3 proposes to use DRL to select volatility-targeting models (as presented in Hocquard et al. (2013) or Perchet et al. (2016)). Volatility targeting models are a traditional portfolio allocation method that rescales portfolio weights according to some volatility forecast to maintain portfolio volatility equal to a specific target (Dreyer & Hubrich 2017). We add an extra overlay with a model-free DRL step to select the proportion or allocation in each volatility targeting model, as there are numerous choices for doing volatility targeting models. This DRL model can be considered as a meta-model or in pure RL world as a mix between a quantitative approach that gives the action dynamics and a model free. Although the final result is a portfolio allocation like for the previous chapter, this approach differs substantially as it combines a volatility-targeting layer with a DRL approach. As in previous chapters, we confirm that contextual information such as volatility and other contextual data play a crucial role in the DRL's capacity to learn. The profitability and ability of this method to reduce risk are verified as we can overperform various benchmarks. Like in chapter 2, we employ the walk-forward analysis method to assess the robustness of the DRL agent and ensure that the results are not due to chance. The features sensitivity analysis performed supports the importance of volatility and contextual variables, and helps to explain why the DRL agent outperforms other methods. Statistical tests are then conducted to confirm the statistical significance of the results.

In the second part of the thesis, we delve into the theoretical foundations of the DRL approach and provide arguments to support its use.

Chapter 4 focuses on comparing traditional portfolio allocation methods, such as the Markowitz efficient frontier, minimum variance, maximum diversification, and equal risk parity, with Deep Reinforcement Learning (DRL) techniques. The main goal is to emphasize the differences between these two approaches and showcase how DRL can be viewed as an extension of traditional portfolio allocation methods. The traditional methods mentioned, such as the Markowitz efficient frontier and minimum variance, have been affected by errors in optimal portfolio choice (Chopra & Ziemba 1993) or when considering practical risk (Kritzman 2014). On the other hand, maximum diversification and equal risk parity (Roncalli & Weisang 2016) have also been explored. By transforming the optimization problem into a continuous control optimization with delayed rewards, DRL offers several advantages. It can adapt to changing market conditions, incorporate additional data, and does not solely rely on traditional financial risk assumptions. This demonstrates how DRL expands the possibilities beyond the limitations of traditional methods. Additionally, the chapter includes an experiment that highlights the importance of convolutional networks in DRL, showcasing their significance in portfolio allocation within the DRL framework.

In the chapter 5, it is shown that the Actor Critic methods presented in Sutton, McAllester, Singh &

Mansour (1999), later modernized with parallelization in Mnih et al. (2016), can be reformulated as control variate estimators. Using the projection theorem, we can establish that the Q (QAC) and Advantage Actor Critic (A2C) methods are optimal in terms of L^2 norm for control variance estimators based on functions conditional on the current state and action. This direct application of the Pythagorean theorem provides a theoretical explanation for the improved performance of QAC and A2C methods in deep policy gradient methods over TD AC and REINFORCE.

Chapter 6 explores the similarities between reinforcement learning (RL) and supervised learning (SL). RL is a decision-making process where the current state influences the next state, whereas SL aims to establish a mapping between input and output by discovering the relationship within a training dataset and generalizing it to new, unseen inputs where output solutions are unavailable (Hastie et al. 2009). Although SL and RL have fundamental differences, there are connections between the two that have often been overlooked (Bishop 2007). This chapter aims to shed light on the relationship between gradient policy methods and supervised learning. By framing gradient policy methods as a supervised learning problem, where discounted rewards replace true labels, we present a fresh proof of policy gradient methods that emphasizes the strong connection between cross-entropy and policy gradient methods.

We end the thesis with a conclusion chapter. It provides a concise summary of all the key concepts and findings presented throughout the thesis and highlights potential avenues for future research.

Publications

This PhD thesis has led to the following publication at conferences.

1. **MIDAS 2021: best paper award**
   E Benhamou, D Saltiel, S Tabachnik, C Bourdeix and F Chareyron (Benhamou, Saltiel, Tabachnik, Bourdeix & Chareyron (2021))
   Adaptive Supervised Learning for Financial Markets Volatility Targeting Models
   http://midas.portici.enea.it/midas2021

2. **ICPR 2021: full paper**
   E Benhamou, D Saltiel, JJ Ohana, J Atif (Benhamou, Saltiel, Ohana & Atif (2021))
   Detecting and adapting to crisis patterns with context-based Deep RL
   https://www.micc.unifi.it/icpr2020/

3. **Global Quant Network 2021: full paper**
   S Ungari, E Benhamou (Ungari & Benhamou (2021)),
   Deep Reinforcement Learning for Portfolio Allocation
   https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3886804

4. **Risk Forum 2021 (March 25): full paper**
   S Ungari, E Benhamou, D SALTIEL, B GUEZ, A MUKHOPADHYAY (Ungari et al. (2021)),
   From Markowitz to Deep Reinforcement Learning for portfolio allocation
   https://www.risks-forum.org/RF2019.html

5. **ESANN 2020: full paper**
   E Benhamou, D Saltiel (Benhamou & Saltiel (2020)),
   Similarities between policy gradient methods in RL and SL
   https://www.esann.org/sites/default/files/proceedings/2020/ES2020-34.pdf

6. **ESANN 2020: full paper**
   C Beji, E Benhamou, M Bon, F Yger, J Atifl (Beji et al. (2021)),
   Estimating Individual Treatment Effects through Causal Populations Identification
   https://www.esann.org/sites/default/files/proceedings/2020/ES2020-110.pdf

7. **Risk Forum 2019 (March 19): full paper**
   D Saltiel, E Benhamou (Saltiel & Benhamou (2019)),
   Trade Selection with Supervised Learning and OCA
   https://www.risks-forum.org/RF2019.html

8. **AAMAS 2021: EXTRAAMAS workshop**
   JJ Ohana, S Ohana, E Benhamou, D Saltiel and B Guez (Ohana et al. (2021))
   Explainable AI (XAI) models applied to the multi-agents environment of financial markets
   https://extraamas.ehealth.hevs.ch/archive.html#presentations-2021

9. **AAMAS 2021: ALA workshop**
   E Benhamou, D Saltiel, S Tabachnik, S-K Wong and F Chareyron (Benhamou, Saltiel, Tabachnik, Wong & Chareyron (2021))

Adaptive learning for financial markets mixing model-based and model-free RL
https://ala2021.vub.ac.be/

10. **AAAI 2021: KDF workshop**
E Benhamou, D Saltiel, S Ungari, J. Atif (Benhamou, Saltiel, Ungari & Abhishek Mukhopadhyay (2021)),
Knowledge discovery with Deep RL for selecting financial hedges
https://aaai.org/Conferences/AAAI-21/ws21/

11. **DIMACS 2021: From forecast to decisions workshop**
E. Benhamou, D. Saltiel, B. Guez, J. Atif, R. Laraki (Benhamou, Saltiel, Guez, Atif & Laraki (2021)),
From forecast to decisions in graphical models: a natural gradient optimization approach
http://dimacs.rutgers.edu/events/details?eID=1864

12. **ICAPS 2020: Finplan workshop**
E Benhamou, D Saltiel, S Ungari, A Mukhopadhyay (Benhamou, Saltiel, Ungari & Mukhopadhyay (2020*b*)),
Time your hedge with Deep RL
https://icaps20subpages.icaps-conference.org/workshops/finplan/

13. **ICAPS 2020: PRL workshop**
E Benhamou, D Saltiel, S Ungari, A Mukhopadhyay (Benhamou, Saltiel, Ungari & Mukhopadhyay (2020*a*)),
Bridging the gap between Markowitz planning and deep reinforcement learning
https://icaps20subpages.icaps-conference.org/workshops/prl/

14. **ECML-PKDD 2020: MIDAS workshop**
D Saltiel, E Benhamou, R Laraki, and J Atif (Saltiel & Benhamou (2019)),
Trade Selection with Supervised Learning and OCA
http://midas.portici.enea.it/midas2020

15. **GECCO 2020: poster**
E Benhamou, D Saltiel, S Verel (Benhamou, Saltiel & Verel (2020)),
Bayesian CMA-ES: a new approach
https://dl.acm.org/doi/10.1145/3377929.3389913

16. **ECML PKDD 2020: demo paper**
E Benhamou, D Saltiel, JJ Ohana, R. Laraki, J Atif (Benhamou, Saltiel, Ohana, Atif & Laraki (2021)),
DRLPS: Deep Reinforcement Learning for Portfolio Selection
https://ecmlpkdd2020.net/programme/accepted/#allTab

17. **ECML PKDD 2023: LIDTA workshop**
R. Krief, E. Benhamou, B. Guez, J-J. Ohana, D. Saltiel, R. Laraki and J. Atif. Krief et al. (2023) FSDA: Tackling Tail-Event Analysis in Imbalanced Time Series Data with Feature Selection and Data Augmentation https://lidta.github.io/

18. **ECML PKDD 2023: MIDAS workshop**
E. Benhamou, B. Guez, J-J. Ohana, D. Saltiel, R. Laraki and J. Atif. Benhamou et al. (2023) Comparing Deep RL and Traditional Financial Portfolio Methods http://midas.

portici.enea.it/

Software

2021 ICPR Deep RL experiment	demo link
2020 ECML PKDD Deep RL experiment	demo link
2020 ICAPS PRL Deep RL experiment	demo link
2020 ICAPS Finplan Deep RL experiment	demo link
2021 GECO Bayesian CMAES experiment	source code
pytorch implementation of A2C	source code
pytorch implementation of DDPG	source code
pytorch implementation of DQN	source code
pytorch implementation of HER	source code
pytorch implementation of PPO	source code
pytorch implementation of SAC	source code
pytorch implementation of TD3	source code

PART I

Deep Reinforcement Learning (DRL) for portfolio allocation

Overview

> Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur
>
> *Thorndike (1911), an American psychologist that worked on comparative psychology and the learning process to lay the scientific foundation for educational psychology*

In the first part of the thesis, we aim to demonstrate the practical application of Deep Reinforcement Learning (DRL) in solving the portfolio allocation problem. This part is composed of three chapters that explore the use of DRL to solve the complex task of portfolio allocation in a noisy and non-stationary environment. We begin by introducing the concepts of DRL and the portfolio allocation problem and formulating four questions that the thesis seeks to answer in chapter 1, Introduction.

Chapter 2, Creating a hedging portfolio with DRL, focuses on the use of additional and contextual information in portfolio allocation by deep reinforcement learning (DRL). Its objective is to show how the use of additional data can significantly improve traditional portfolio allocation methods. We also provide experimental evidence to highlight the importance of contextual data in portfolio allocation with DRL. The experiments indicate that the use of contextual data can lead to a significant improvement over traditional methods that do not consider such information. For instance, the experiments may demonstrate that incorporating additional data such as economic indicators can result in better portfolio outcomes. Additionally, we illustrate the potential capability of DRL to adapt to fluctuating market conditions through the use of convolutional networks. These networks can effectively capture the relationship between market data and adjust to changing market circumstances. In other words, the DRL system can modify its portfolio allocation approaches based on market changes, leading to improved portfolio performance.

Chapter 3, Selecting the experts with DRL, delves into the concept of combining two approaches in portfolio allocation - a volatility-based approach and a deep reinforcement learning (DRL) approach, resulting in a two-step approach. The main goal of the first step, the volatility-based method, is to adjust the weights of different investments in the portfolio so that they align with a target level of volatility. The second step, the DRL approach, acts as an overlay on top of the volatility-based approach. Its objective is to select the best-performing volatility-targeting models among all the models considered. This DRL overlay uses reinforcement learning algorithms to determine which of the volatility targeting models are most likely to achieve the best Sharpe ratio, and allocates the portfolio accordingly.

In essence, the DRL overlay is a decision-making mechanism that selects the "experts" among

the different volatility targeting models. This two-step approach to portfolio allocation differs from traditional methods because the DRL component acts as a building block within the overall allocation process. This chapter highlights the potential for DRL to play a role in traditional portfolio allocation. Using DRL as an overlay to weigh the various volatility targeting models can provide a more informed and sophisticated approach to portfolio allocation methods based on volatility targeting methods.

Overall, this initial section of the thesis provides a comprehensive overview of the practical application of DRL in portfolio allocation, demonstrating its potential to offer dynamic portfolio allocations, adapt to changing market conditions, and combine DRL with traditional financial methods such as volatility targeting.

1 Introduction

> Reinforcement learning is the framework for people who are interested in trying to solve the big problems of artificial intelligence
>
> *Silver (2015), a British computer scientist who leads research at DeepMind on AlphaGo*

1.1 Summary

This chapter serves as a comprehensive introduction to Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) with a focus on providing a thorough understanding of the motivations and underlying principles behind them. To begin, the standard framework for RL is presented, followed by an overview of DRL, which is considered as an extension of RL.

The chapter then delves into the key characteristics of DRL that have led to its exceptional growth over the past decade. This includes a discussion of the improvements that DRL brings to RL and how it differs from traditional methods.

The chapter then shifts its focus to the problem of portfolio allocation and the limitations of traditional financial methods. This leads to the formulation of the fundamental question of what and how the computer should learn to effectively solve the problem of portfolio allocation.

Finally, the chapter concludes by summarizing the potential new questions and contributions that this thesis introduces on portfolio allocation using DRL. The materials discussed in this chapter will be explored in greater detail throughout the manuscript.

1.2 Deep Reinforcement Learning

Reinforcement learning (RL) is a central topic in machine learning that addresses the task of making sequential decisions in an uncertain environment to achieve a specific goal. This approach draws on principles from behavioural psychology and provides a structured framework for decision-making (Sutton & Barto 2018).

In RL, the decision-making process is modelled as an interaction between an agent and an environment. The agent performs actions in the environment in order to maximize a reward signal that reflects its progress toward the goal. Over time, the agent learns from the results of its actions and adjusts its behavior to maximize its reward.

The decision-making process in RL is often viewed as a game between the agent and the environment. The agent must choose actions based on its current state, while the environment determines the next state and the reward based on the action taken. The agent's goal is to learn a policy, which is a correspondence between states and actions, that maximizes its expected reward over time.

RL is one of the three basic and traditional machine learning paradigms, alongside supervised learning and unsupervised learning. RL has been applied to a wide range of problems, including playing Atari games from the raw pixels (Mnih et al. 2013), or other video games like Starcraft II (Vinyals et al. 2019), mastering more human oriented games like Go (Silver et al. 2017), or Poker (Brown et al. 2020), training robots to move in rich environment (Heess et al. 2017) and (Andrychowicz et al. 2020), using hand-eye coordination for robotic grasping (Levine et al. 2016), teaching aerial Vehicle to avoid obstacles (Gandhi et al. 2017), training self-driving cars (You et al. 2017), doing finance (Deng et al. 2016), optimising smart grids (François-Lavet et al. 2016), solving healthcare problems (Schaefer et al. 2005), answering general visual question on complex scenes by visualising objects (Antol et al. 2015), guessing objects (de Vries et al. 2017), solving energy management problems for micro-grids with an agent based control (Dimeas & Hatziargyriou 2007) to highlight a few of the many achievements of Deep Reinforcement Learning (DRL), as the list of accomplishments continues to grow at an unprecedented rate

1.2.1 Mathematical Formulation

Traditionally, a Reinforcement Learning (RL) problem involves an agent making decisions in an environment that is described by a set of states, actions, and rewards. The agent represents the decision maker, such as a robot, software, or any system that needs to make decisions. The environment at each time step t is fully described by some state s_t, such as the velocity, acceleration, and mass of a car or the value of assets and other financial variables in finance. These states (s_t) belong to a set of possible values denoted by $\mathcal{S}$. The agent interacts with the environment by choosing an action a_t from a set of possible actions $\mathcal{A}$ and receiving a reward r_t whose values belong to a set $\mathcal{R}$. This process repeats as the state transitions to a new state $s_{t+1} \in \mathcal{S}$, and the sequential decision problem continues.

The formal control setting for Reinforcement Learning (RL) was initially introduced in Bellman & Kalaba (1957) and further developed for machine learning by Barto et al. (1983). This setting is typically illustrated in a diagram, as shown in Figure 1.1, which emphasizes the dynamic nature of the sequential decision problem through the use of arrows.

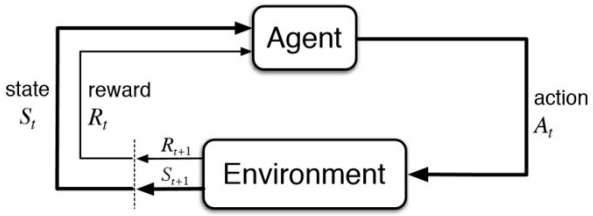


Figure 1.1: Reinforcement learning seminal diagram summarising the chronological order of states, actions and rewards.

The sequential decision problem can be made more realistic by considering the distinction between states s_t and observations o_t and using the concept of Partially Observable Markov Decision Process as described below in the section 1.2.3. The observations represent the information that the agent has access to, while the states represent the real underlying environment. In this context, the environment is always fully described by a state, but the agent does not observe it directly. Instead, the agent receives an observation, o_t, which is a reduced representation of the environment and depends on the state. This setup highlights the uncertainty the agent faces when making decisions, as the actual state of the environment is not directly known. For example, in the case of a car, the observation may include the angular velocity, acceleration, and mass of the car, while the state may include additional factors such as the age, make, and robustness of the engine, as well as the oil level in the engine. In finance, the state can also include variables such as the health of the economy and the current business cycle.

To evaluate the value of future rewards resulting from our actions, we introduce the concept of a discount factor, denoted by $\gamma \in [0, 1]$. The discount factor, also known as the discount rate, is a concept that is familiar to financial specialists, as it represents the factor by which a future variable must be multiplied to obtain its present value. It reflects the foresightedness of the agent and their interest in receiving rewards sooner or later. A value of $\gamma = 1$ implies that all rewards are considered equal, regardless of when they are received, while a value of $\gamma = 0$ represents absolute short-sightedness, with the agent only concerned with immediate rewards. To avoid mathematical issues and infinite sums in an infinite horizon setting, we impose that the discount rate is strictly less than one in the rest of the manuscript.

1.2.2 Markov Decision Process (MDP)

We define a Markov Decision Process (MDP) (Bellman 1957*b*) as a discrete time stochastic control process defined as follows:

Definition 1.1. ***MDP****: an MDP is a quintuplet (five-tuple)* $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ *where:*

- $\mathcal{S}$ *is the state space,*
- $\mathcal{A}$ *is the action space,*
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ *is the transition function: set of conditional transition probabilities between states, actions to next states,*
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ *is the (immediate) reward function, where* $\mathbb{R}$ *represents the set of real numbers,*

- $\gamma \in [0, 1)$ *is the discount factor.*

If we use graphical models, we can represent our MDP by a sequential graphical model as shown in figure 1.2.

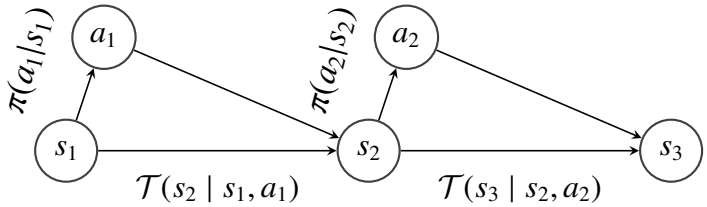


Figure 1.2: MDP graphical model representation. The nodes s_1, s_2 represent the state nodes, while the ones denoted by a_1, a_2 are the action nodes. Once the agent has taken an action, the environment moves to the next state. To make a decision and take an action, an agent follows a policy π whose value depends on the states and is denoted by $\pi(a_t \mid s_t)$. The transition from state s_t and action a_t nodes to the next state s_{t+1} node is denoted by $\mathcal{T}(s_{t+1} \mid s_t, a_t)$

.

1.2.3 Partially Observable Markov Decision Process (POMDP)

The concept of Partially Observable Markov Decision Process (POMDP) extends the traditional Markov Decision Process (MDP) by considering a set of observations, $\mathcal{O}$, and the conditional transition function $\mathcal{T}_O$ from observation to next observation as presented in Astrom (1969). This addition allows for a more realistic representation of an agent's decision-making process in which the agent must make decisions based on reduced information, rather than complete knowledge of the environment's state.

At each time step t, the environment is in some state $s_t \in \mathcal{S}$. The agent then takes an action $a_t \in \mathcal{A}$, causing the environment to transition to state s_{t+1} with probability $\mathcal{T}(s_{t+1} \mid s_t, a_t)$. The agent also receives an observation $o_t \in \mathcal{O}$, dependent on both the new state of the environment and the taken action, with transition probability $\mathcal{T}_O(o_{t+1} \mid s_{t+1}, a_t)$. The agent is then rewarded based on the reward function $r(s_t, a_t)$ and the process repeats. To simplify notations, we will use equivalently $\mathcal{R}(s_t, a_t)$ or r_t to denote the immediate reward.

The POMDP can be visualized using the modified graphical model represented by figure 1.3. This setting provides a more nuanced and dynamic representation of the agent's sequential decision-making process, taking into account the uncertainty of the true environment state.

The introduction of the Partially Observable Markov Decision Process (POMDP) highlights the challenge of obtaining a reliable state representation and the limitations introduced by observations. In this thesis, we will utilize both the Markov Decision Process (MDP) and POMDP frameworks as appropriate. The use of POMDP will serve to emphasize the complexity associated with obtaining an accurate state representation and the reliance on observations. In real-world scenarios, where only observations are available, the use of MDP introduces a limitation, resulting in an approximation of the true problem. This underscores the difficulty of solving the portfolio allocation problem due to the inherent constraints of relying solely on observations.

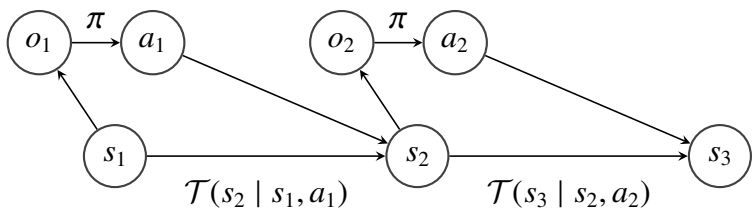


Figure 1.3: POMDP graphical model representation. Nodes s_1, s_2, o_1, o_2 and a_1, a_2 represent state, observation and action nodes. In POMDP, the agent receives an observation o_t derived from a state s_t and makes an action a_t. The agent uses only the observation to make an action using a policy π. The environment moves to the next state and the sequential decision process carries on.

1.2.4 Trajectory, Policy, Return

In an MDP, the sequence of states and actions generated by the interaction between the agent and the environment is referred to as a trajectory or an episode. This sequence is denoted by $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$ if the trajectory starts at time step $t = 0$, or more generally, $\tau_t = (s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \ldots)$ if the trajectory starts at a time step t. The state-action pair of a time step t, $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$, is referred to as the state-action pair of time step t.

A policy π defines how an agent selects actions. Policies can be categorized under the criterion of being either stationary or non-stationary.

A stationary policy is a mapping from states to actions, $\pi : \mathcal{S} \to \mathcal{A}$. It is denoted by $\pi(a \mid s)$ and represents the probability of taking action a in state s. The policy is stationary because it does not change over time, meaning that the action chosen by the agent in a given state will always be the same, regardless of the time step. In contrast, a non-stationary policy depends on the time-step t and is useful only for finite horizon context where the cumulative rewards that the agent seeks to optimize are limited to a finite number of future time steps.

In the context of reinforcement learning, the goal is to find an optimal policy that maximises the expected cumulative reward over time. The optimal policy, denoted by π^*, is the policy that maximises the expected cumulative reward, given by the expected discounted reward, $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_t \mid s_0 = s \right]$, where T is the end of the horizon, finite or not.

The distinction between stationary and non-stationary policies is important because the choice of a policy affects the way an agent interacts with the environment and the type of learning algorithms that can be used to find an optimal policy. In this thesis, we will focus on finding optimal stationary policies as they provide a natural and effective framework for learning and planning in Reinforcement Learning.

Policies can be differentiated based on their second characteristic of being either deterministic or stochastic. A deterministic policy is defined as $\pi : \mathcal{S} \to \mathcal{A}$ where the action taken at each state is determined and represented by a deterministic function $\pi : s \mapsto \pi(s) = a$, where for each state, there is only one single possible action. On the other hand, a stochastic policy is characterized by the probabilities of taking an action at each state. This is defined as a new function taht now depends on two variables as follows $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, with $\pi(s, a) : s \times a \mapsto \pi(s, a) = \mathbb{P}(s_t = s, a_t = a)$, representing the probability of taking action a in state s at time t. We often denote it by $\pi(a \mid s)$ to emphasize that the action a is conditioned by the state s.

In full generality, a policy π could be designed based on the complete history of observations or states during the agent-environment interactions, such as in POMDPs or MDPs. However, for the purpose of considering stationary policies, we limit our focus to policies that only depend on the current state s_t. The starting state s_0 is characterized by the initial probability distribution $\mu(s_0)$, and the length of the trajectory τ is denoted by T which can be either a finite value, meaning $T < \infty$ or an infinite value, meaning $T = \infty$. The overall probability distribution of trajectories, which takes into account both the policy and the MDP, can be expressed as follows:

$$\mathbb{P}(\tau_0 \mid \pi) = \mu(s_0) \prod_{t=0}^{T} \pi(a_t \mid s_t) \mathcal{T}(s_{t+1} \mid s_t, a_t). \tag{1.1}$$

If we start at t, this changes to

$$\mathbb{P}(\tau_t \mid \pi) = \mu(s_t) \prod_{k=t}^{T} \pi(a_k \mid s_k) \mathcal{T}(s_{k+1} \mid s_k, a_k). \tag{1.2}$$

The return of a trajectory τ_t starting at time t, also known as the cumulative reward from that time t forward, also referred to as the reward to go, is calculated as the sum of all discounted rewards along the trajectory. This can be represented as:

$$R(\tau_t) = \sum_{k=0}^{T} \gamma^k r_{t+k} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots, \tag{1.3}$$

where the individual reward term is abbreviated for simplicity as $r_t = \mathcal{R}(s_t, a_t)$.

1.2.5 RL Optimization Problem

In reinforcement learning, the goal is to find a policy, which may not be unique, that maximizes the expected return over all trajectories. To do this, we consider the set of all possible policies, denoted by Π, and we use the notation $\tau \sim \pi$ to indicate that the distribution of the trajectory is determined by equation (1.1), meaning that the trajectory τ follows the policy π. The optimization problem can then be expressed as follows:

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} \mathbb{E}_{\tau \sim \pi} \left[R(\tau) \right]. \tag{1.4}$$

1.2.6 Value Function

The expected return also called the value or V-value function $V^\pi : \mathcal{S} \mapsto \mathbb{R}$ is defined as follows:

$$V^\pi(s) = \mathbb{E}\left[\sum_{k=0}^{k=T} \gamma^k r_{t+k} \mid s_t = s, \pi \right] = \mathop{\mathbb{E}}_{\tau \sim \pi} \left[R(\tau) \mid s_t = s \right]. \tag{1.5}$$

Using the definition of the expected return, we can define the optimal expected return, also known as the value function, as:

$$V^*(s) = \max_{\pi \in \Pi} V^\pi(s). \tag{1.6}$$

Along with the value function, we can also define the Q function, also known as the Q-value function or states-action value function, as $Q^\pi(s,a) : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, as follows:

$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{k=0}^{k=T} \gamma^k r_{t+k} \mid s_t = s, a_t = a, \pi\right] = \mathop{\mathbb{E}}_{\tau \sim \pi}\left[R(\tau) \mid s_t = s, a_t = a\right]. \tag{1.7}$$

If we assume an infinite horizon, we can take advantage of the recursive relationships satisfied by value and Q functions and write the Bellman's equation as follows:

$$Q^\pi(s,a) = \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s' \mid s,a) Q^\pi(s', a = \pi(s')). \tag{1.8}$$

The Bellman equation is a highly valuable tool for showcasing the convergence of both the Q function and the value function through fixed point iteration. Under mild conditions, such as a stationary policy and the ability to sample all states repeatedly, the Bellman equation ensures sufficient exploration even without knowledge of the transition model. The convergence of these functions can be readily established by leveraging the Banach theorem on contraction mappings, particularly when the discount factor γ is less than 1.

Similarly to the V-value function, the optimal Q-value function $Q^*(s,a)$ can also be defined as

$$Q^*(s,a) = \max_{\pi \in \Pi} Q^\pi(s,a). \tag{1.9}$$

The interest of the Q-value function compared to the V-value function is its capacity to provide the optimal policy directly as follows:

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(s,a). \tag{1.10}$$

It is worth noting that V-value and Q-value functions are related thanks to the equation

$$V^\pi(s) = \mathop{\mathbb{E}}_{a \sim \pi(.|s)}\left[Q^\pi(s,a)\right] \tag{1.11}$$

$$Q^\pi(s,a) = \mathop{\mathbb{E}}_{s' \sim \mathcal{T}(.|s,a)}\left[R(s,a) + \gamma V^\pi(s')\right]. \tag{1.12}$$

In conclusion, we have presented the basics of reinforcement learning, highlighting that RL algorithms can choose to either calculate the value function, and variation around the Q function or the policy π, or a combination of both. This leads to the classification of RL algorithms into value-based, policy-based, and combinations of these two methods, as well as model-based algorithms that are not discussed in this thesis as they are problem specific, except for the specific case of financial models in chapter 3. We will delve into actor-critic methods, which are a combination of value and policy methods later during this thesis in chapter 5. Finally, we will move on to discussing Deep Learning and how it can be integrated with RL to create DRL methods.

1.2.7 Deep Learning

The motivation behind deep learning is to be able to represent any arbitrary function through a function $f : \mathcal{X} \to \mathcal{Y}$, which is parameterized with a large set of parameters, commonly represented as $\theta \in \mathbb{R}^{n_\theta}$, as follows:

$$y = f(x, \theta). \tag{1.13}$$

A deep neural network typically utilizes multiple processing layers to handle a given function. Each layer is comprised of non-linear transformations, with a common example being the Relu (rectified linear unit) function, defined as follows:

Definition 1.2. *The rectifier or ReLU (Rectified Linear Unit) activation function is defined as follows:* $f(x) = (ax + b)^+ = \max(0, ax + b)$

Interestingly, the application of multiple transformations can result in varying degrees of abstraction as presented in Erhan et al. (2009) or more recently in Olah et al. (2017). To gain a better understanding, let's consider a straightforward fully connected neural network with three hidden layers, as illustrated in figure 1.4

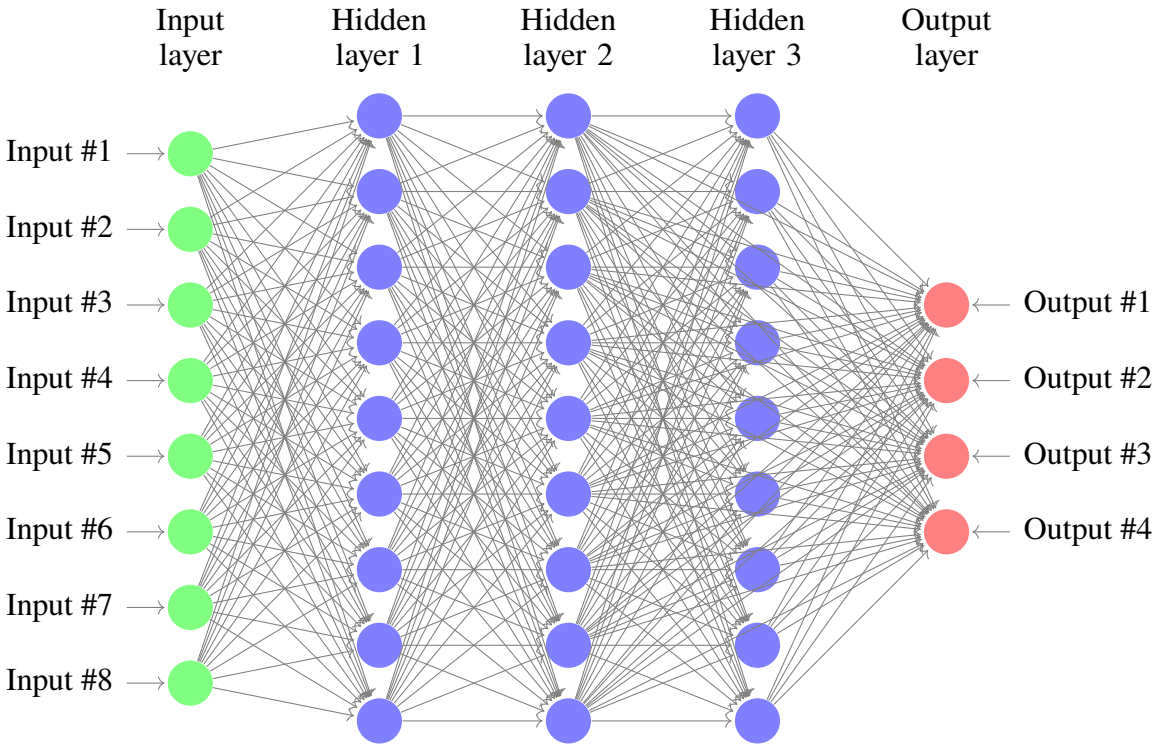


Figure 1.4: Very basic neural network with three single hidden layers. The input layer receives the input value x while the output layer computes the final variable y. In this specific network, the size of the input layer is eight: $n_x = 8$ while the output layer is 4: $n_y = 4$

In a neural network, the first layer is responsible for taking in the input values x, which are typically presented as column vectors of size n_x. The next layer, which is commonly the hidden

layer, then computes its input values through a nonlinear transformation of the previous layer's output values. This transformation involves an initial linear transformation, which is achieved by multiplying the input values with a weight matrix W_1 of size $n_h \times n_x$ and adding a bias term b_1 of size n_h. The resulting affine transformation is then passed through a nonlinear activation function A, which applies a nonlinear function to each element of the output values, resulting in the following calculation:

$$h = A(W_1 x + b_1), \tag{1.14}$$

where h represents the output values of the hidden layer. It is worth noting that all size inputs such as n_θ, n_x, and n_h are integers, where n_θ represents the number of parameters in the neural network.

The nonlinear activation function A is an essential component of the neural network, as it introduces nonlinearity into each layer's transformation, allowing the network to represent a wide range of functions. Commonly used activation functions include the sigmoid, ReLU, and tanh functions, each of which has its own advantages and disadvantages depending on the task at hand. By applying a nonlinear function to the output of the affine transformation, the hidden layer can learn complex representations of the input data and extract meaningful features.

Theoretically, the universal approximation theorem, that is the machine learning equivalent of the Arnold–Kolmogorov representation theorem in real analysis and approximation theory, states that a feedforward neural network with a single hidden layer and a sufficient number of neurons can approximate any continuous function on a compact input space to arbitrary precision, given appropriate activation functions (Cybenko 1989) and Hornik (1991). This theorem provides the foundation to use deep networks whenever we need to approximate a function in machine learning.

In a neural network, the intermediate layers, which are also referred to as hidden layers, play a crucial role in transforming the input data into output values. The hidden layers receive the output from the previous layer and generate new values that are passed onto the next layer. This process continues until the final transformation that produces the output values, denoted by y.

For a neural network with n hidden layers, the full transformation can be represented mathematically as follows:

$$y = f_n(f_{n-1}(\cdots f_2(f_1(x;\theta_1);\theta_2)\cdots;\theta_{n-1});\theta_n), \tag{1.15}$$

where x represents the input data, θ_i are the parameters of the ith layer, and f_i denotes the activation function of the ith layer. The activation function is applied to the output of the previous layer to generate new values that are used as inputs to the next layer.

The process of passing information through the intermediate layers is known as forward propagation, and it is an essential step in the neural network's learning process. The hidden layers allow the network to learn complex representations of the input data by extracting relevant features and patterns. These features are then used to make predictions or perform other tasks, such as classification or regression.

Overall, the intermediate layers play a critical role in transforming the input data into meaningful output values. By applying activation functions to the outputs of the previous layers, the hidden layers enable the network to learn complex representations of the input data and make accurate

predictions. This process is fundamental to the success of neural networks in a wide range of applications, such as image recognition, natural language processing, and robotics.

In deep learning and deep reinforcement learning, the process of determining the optimal parameters for a neural network is known as learning. This involves using a backpropagation algorithm, which is commonly found in multiple sources, including (Kelley 1960), (Werbos 1974), and (Rumelhart et al. 1985). The goal of the learning procedure is to adjust the internal parameters of the network, denoted by θ_k, in order to minimize a loss function L. This is typically achieved by using a gradient descent algorithm with backpropagation, which calculates the gradient of the loss function with respect to the internal parameters.

The loss function L is used to evaluate the error or distance between the target function f and the approximated function generated by the neural network. During each iteration k, the learning algorithm updates the internal parameters θ_k by computing the gradient of the loss function and using it to adjust the parameters in a direction that reduces the error between the predicted outputs and the target outputs. This is done using the following equation:

$$\theta_{k+1} \Longleftarrow \theta_k - \alpha \nabla_\theta L, \tag{1.16}$$

where α_k represents the learning rate at step k and $\nabla_\theta L$ the gradient of the loss L with respect to the parameters θ of the neural network. The learning rate determines the size of the step taken during each iteration and is typically chosen to be a small positive value.

Overall, the backpropagation algorithm with gradient descent is a powerful tool for learning in neural networks. It allows the network to iteratively adjust its parameters to better approximate the target function and reduce the error between the predicted and target outputs. This procedure is widely used in deep learning and deep reinforcement learning to train models for a wide range of tasks, such as image classification, natural language processing, and game playing.

In the field of deep learning, and consequently deep reinforcement learning (DRL), stochastic environments are commonly encountered. This means that noisy gradient evaluations are used to speed up computations, leading to a process known as stochastic gradient descent. However, choosing an appropriate learning rate for the algorithm is less straightforward than determining the exact gradient setting (Bertsekas 1997, Nocedal & Wright 2006). Consequently, the process of determining the learning rate often relies on empirical methods, with different guidelines available for setting learning schedules.

According to classical theory, as presented in (Robbins & Monro 1951), if the loss function is sufficiently smooth and the learning rate is set based on the following conditions:

$$\sum_{k=1}^{\infty} \alpha_k = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty \tag{1.17}$$

then the algorithm converges in expectation (Bottou et al. 2018). The first condition requires that the sum of the learning rates over an infinite number of iterations is infinite. The second condition states that the sum of the squared learning rates over an infinite number of iterations is finite.

Furthermore, if the loss function is strongly convex, then the update of the stochastic gradient update $\theta_{k+1} \Longleftarrow \theta_k - \alpha \nabla_\theta L$ converges to the optimal values. This is because the strongly convex property of the loss function guarantees that the optimization problem has a unique global minimum, and the convergence of the update process to the optimal values is guaranteed.

In practice, determining the appropriate learning rate often requires trial and error. There are several techniques available for setting the learning rate, such as learning rate schedules that gradually decrease the learning rate over time, or adaptive learning rate methods that adjust the learning rate based on the gradient's magnitude. However, the choice of learning rate ultimately depends on the specific problem being tackled and the characteristics of the data being used.

The field of deep learning is guided by two fundamental concerns that shape the different strategies used for training neural networks. The first concern is to train the network as efficiently as possible by avoiding getting stuck in narrow valleys or poor minima of the cost function. The second concern is to control the representativeness of the neural network according to the amount of training data, thus avoiding overfitting and minimizing generalization errors.

Over the years, various types of neural networks have been proposed, each with specific advantages depending on the application. For example, some architectures may provide a good tradeoff between bias and overfitting in a supervised learning setting. Within a given neural network architecture, an arbitrarily large number of layers is possible. In recent years, there has been a trend towards using ever-growing numbers of layers, with some supervised learning tasks using more than 100 layers and millions and potentially billions of parameters, like for instance for Large Language Models (LLMs).

The number of layers in a neural network is crucial as it impacts the network's representational power and its ability to learn complex functions. However, adding too many layers can lead to overfitting and slower training times. To address this issue, regularization techniques such as dropout and weight decay are often used to prevent overfitting.

In addition to the number of layers, the choice of activation function, optimization algorithm, and learning rate can also impact the neural network's performance. Choosing appropriate values for these parameters often requires experimentation and fine-tuning.

In the field of deep reinforcement learning, convolutional layers, as introduced in (Fukushima 1980) and popularized later in (LeCun et al. 1995), have gained particular interest due to their efficiency in processing images and sequential data with a translation-invariant nature (see figure 1.5). At its core, a convolutional neural network (CNN) utilizes a specific type of neural network layer called a convolutional layer. The convolutional layer applies a mathematical operation known as convolution to the input data (see figure 1.6). This operation involves sliding a small window, called a filter or kernel, across the input data and performing element-wise multiplication and summation. The purpose of the convolution operation is to extract local features from the input data. The filters within the convolutional layer learn to detect various patterns, such as edges, textures, or more complex structures, by identifying similarities between neighboring pixels. The convolutional layer generates a set of feature maps, which are essentially transformed representations of the input data.

Typically, a CNN consists of multiple convolutional layers, interspersed with other types of layers like pooling layers and fully connected layers (see figure 1.7). Pooling layers are responsible for reducing the spatial dimensions of the feature maps while retaining the most important information.

They achieve this by downsampling and selecting the most dominant features within a local region. Usually, the last layers are fully connected layers. These layers are employed at the end of the CNN to perform high-level reasoning and make predictions based on the extracted features. These layers are similar to those found in traditional neural networks, where each neuron is connected to every neuron in the previous layer. During the training process, a CNN learns to recognize and classify patterns by adjusting the weights and biases of its neurons through a technique called backpropagation. By leveraging the hierarchical nature of CNNs and their ability to capture spatial dependencies, these networks have proven to be highly effective in various computer vision tasks. They have achieved state-of-the-art performance in image classification competitions and have been widely adopted in many real-world applications involving visual data analysis (Erhan et al. 2009, Olah et al. 2017).

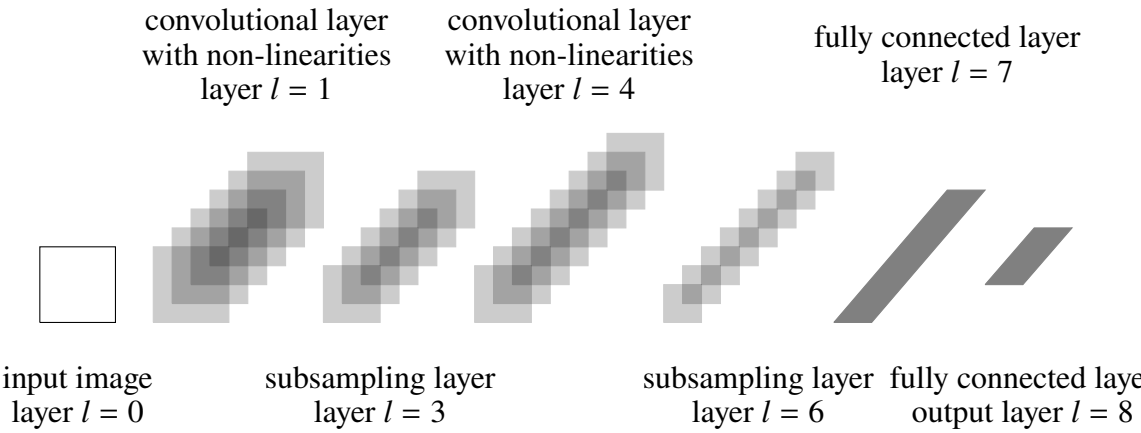


Figure 1.5: The architecture of the original convolutional neural network, as introduced by LeCun et al. (1989), alternates between convolutional layers including hyperbolic tangent non-linearities and subsampling layers. In this illustration, the convolutional layers already include non-linearities and, thus, a convolutional layer actually represents two layers. The feature maps of the final subsampling layer are then fed into the actual classifier consisting of an arbitrary number of fully connected layers. The output layer usually uses softmax activation functions.

The CNN layer is one type of feedforward layer, with the specificity that many weights are null, and other weights are shared. This implementation leads to a drastic dimension reduction, making the learning process much more efficient than fully connected networks. In addition to convolutional neural networks, other architectures have been proposed to address the issue of sequential data. Recurrent neural networks (RNNs) are a type of neural network that processes sequential data by passing information from one time step to the next. This allows the network to encode temporal information and make predictions based on previous inputs. However, basic RNNs suffer from the vanishing gradient problem, where gradients become too small to be useful during backpropagation, limiting their ability to encode information from long sequences.

To address this issue, long short-term memory networks (LSTMs) were introduced by (Hochreiter et al. 2001). LSTMs are a type of RNN that uses a gated cell structure to selectively remember and forget information. This allows the network to encode information from long sequences, making them well-suited for processing time-series data.

In practice, the choice of architecture depends on the specific problem being tackled and the

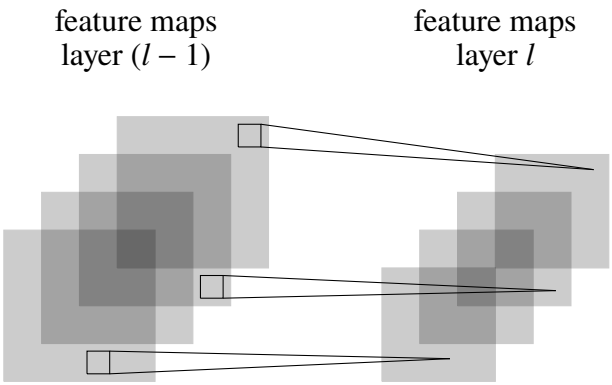


Figure 1.6: Illustration of a pooling and subsampling layer. If layer l is a pooling and subsampling layer and given $m_1^{(l-1)} = 4$ feature maps of the previous layer, all feature maps are pooled and subsampled individually. Each unit in one of the $m_1^{(l)} = 4$ output feature maps represents the average or the maximum within a fixed window of the corresponding feature map in layer $(l-1)$.

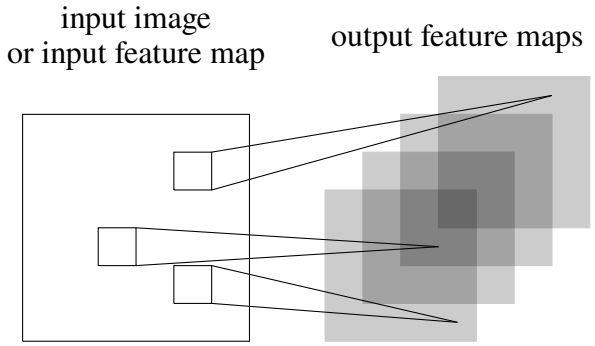


Figure 1.7: Illustration of a single convolutional layer. If layer l is a convolutional layer, the input image (if $l = 1$) or a feature map of the previous layer is convolved by different filters to yield the output feature maps of layer l.

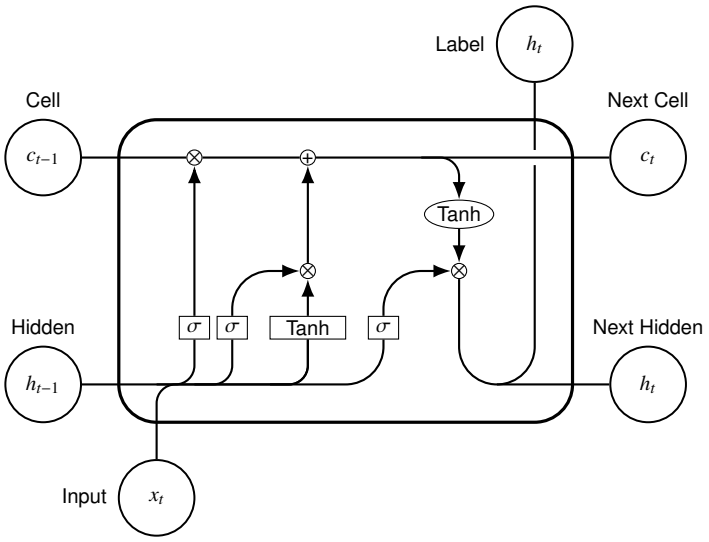


Figure 1.8: The figure represents the architecture of a Long Short-Term Memory (LSTM) cell. It consists of input nodes (x, h, c) (as input, hidden and cell), gates (σ), multiplication nodes ($\times$), an addition node (+), activation functions (Tanh), and output nodes (Net hidden, Next Cell, And Label). The gates control the flow of information and selectively update the cell state. The multiplication nodes combine gate outputs, and the addition node combines retained and new information. The Tanh activation functions transform input values. The outputs are the updated hidden state (h) and cell state (c).

characteristics of the data being used. Convolutional neural networks are efficient for processing images and sequential data, while recurrent neural networks, such as LSTMs, are better suited for processing time-series data. The transformer architecture has shown promising results in processing sequential data through self-attention mechanisms, making it well-suited for tasks involving long-range dependencies.

1.2.8 Policy gradient

Policy gradient methods are a class of reinforcement learning algorithms that directly optimize the policy of an agent. They differ from value-based methods, which learn to estimate the value function of a given policy and then iteratively improve the policy based on the estimated values.

The objective of policy gradient methods is to find a policy, π, that maximizes the expected total reward, J, over some task, where the expected total reward is given by

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}[R(\tau)], \tag{1.18}$$

where τ is a trajectory of the agent and $R(\tau)$ is the total reward received over that trajectory. The objective is to find a policy that maximizes the expected total reward.

One common way to achieve this is by using gradient ascent on the policy parameter space. The gradient of the expected total reward with respect to the policy parameters, θ, can be estimated using the score function gradient estimator, also known as the REINFORCE algorithm:

$$\nabla_\theta J(\pi_\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau_i), \tag{1.19}$$

where N is the number of trajectories, T is the length of each trajectory (finite or not), $\pi_\theta(a_t|s_t)$ is the policy at time t given state s_t and action a_t, and $R(\tau_i)$ is the total reward received in the ith trajectory.

This estimator is then used to update the policy parameters using gradient ascent:

$$\theta \leftarrow \theta + \alpha \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau_i), \tag{1.20}$$

where α is the learning rate.

There are several variants of the policy gradient method, including actor-critic algorithms, which incorporate value function estimates to reduce the variance of the policy gradient estimator, as well as trust region optimization and natural gradient optimization, which address the issue of curvature in the policy parameter space.

Overall, policy gradient methods offer a direct approach to optimizing policies in reinforcement learning, and have been applied to a wide range of tasks, from playing games to robotic control. However, they can be sensitive to the choice of hyperparameters, such as the learning rate, and can have high variance in the gradient estimator, leading to slow convergence and the risk of getting stuck in suboptimal local minima.

1.2.9 Actor Critic methods

Actor-critic is a class of reinforcement learning algorithms that combines value-based and policy-based methods to optimize the policy of an agent. The algorithm consists of two components: the critic and the actor. The critic evaluates the state-value function, which represents the expected return starting from a given state and following a specific policy. The actor, on the other hand, determines the policy to be followed by the agent based on the state-value function.

The idea of actor-critic algorithms dates back to the 1980s with the work of Sutton and Barto (Sutton 1984) and Konda and Tsitsiklis (Konda & Tsitsiklis 2003). However, the first successful implementation of an actor-critic algorithm was the Asynchronous Advantage Actor-Critic (A3C) algorithm proposed by Mnih et al. (Mnih et al. 2016) in 2016. The A3C algorithm was able to achieve state-of-the-art performance on several benchmark tasks in the Atari game environment.

The intuition behind the actor-critic algorithm is to combine the strengths of both value-based and policy-based methods. Value-based methods, such as Q-learning and SARSA, learn an optimal action-value function that maps states to values. However, these methods can be slow to converge and are often sensitive to the choice of hyperparameters. Policy-based methods, on the other hand, directly learn the policy of the agent and can handle high-dimensional continuous action spaces. However, they can suffer from high variance in the gradient estimates and can be slow to learn.

Actor-critic algorithms address these limitations by using a critic to estimate the value function, which reduces the variance in the policy gradient estimates, and an actor to directly optimize the policy, which handles continuous action spaces efficiently.

The underlying mathematics of actor-critic algorithms can be described as follows. The critic estimates the state-value function $V_\theta(s)$, which represents the expected return starting from state s and following policy π_ϕ with parameters ϕ. The critic is trained to minimize the mean-squared error between the estimated value and the true value:

$$\mathcal{L}V(\theta) = \mathbb{E}_{s\sim p_{\text{data}}}[(V_\theta(s) - V^\pi(s))^2], \tag{1.21}$$

where p_{data} is the distribution of states encountered during training, and $V^\pi(s)$ is the true value of state s under policy π and the notation $s \sim p_{\text{data}}$ means that states are distributed according to the data.

The actor determines the policy π_ϕ by maximizing the expected return, or the so-called "advantage" function $A^\pi(s, a)$, which represents the expected additional return of taking action a in state s compared to following the current policy:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s), \tag{1.22}$$

where $Q^\pi(s, a)$ is the action-value function, representing the expected return of taking action a in state s and following policy π thereafter.

The actor is trained to maximize the expected return, or equivalently, the policy gradient:

$$\nabla_\phi J(\phi) = \mathbb{E}_{s\sim p_{\text{data}}}[\nabla_\phi \log \pi_\phi(a|s) A^\pi(s, a)], \tag{1.23}$$

where $J(\phi)$ is the objective function, representing the expected return under policy π_ϕ.

The above equations represent the basic actor-critic algorithm. Several extensions and variations of actor-critic have been proposed to improve its performance and stability, such as trust region policy optimization (TRPO) (Schulman, Levine, Abbeel, Jordan & July(2015), proximal policy optimization (PPO) (Schulman et al. 2017), and soft actor-critic (SAC) (Haarnoja et al. 2018). These algorithms introduce modifications to the basic algorithm to improve convergence speed and stability, handle continuous action spaces more effectively, and incorporate entropy regularization to encourage exploration.

TRPO modifies the policy update rule to ensure that the new policy is not too far from the old policy, thus avoiding large policy updates that can cause instability. PPO simplifies the trust region constraint in TRPO and adds a clipping mechanism to the policy update to further improve stability. SAC incorporates entropy regularization to encourage exploration and prevent premature convergence to suboptimal policies.

Overall, actor-critic algorithms have proven to be powerful and effective for solving complex reinforcement learning problems with high-dimensional state and action spaces.

1.3 Portfolio allocation

The portfolio allocation problem has been a long-standing issue in finance, and it requires a strategic approach for making investment decisions. In this problem, an asset manager has the ability to invest in N financial assets, and the prices of these assets at time t are represented by $(P_t^i)_{i=1,\dots,N}$. The returns on these assets are calculated as the percentage change in their prices, which is denoted by $(r_t^i)_{i=1,\dots,N}$ and defined as the percentage change of the financial asset prices: $r_t^i = P_t^i / P_{t-1}^i - 1$. Although the concepts of return in portfolio allocation and rewards in reinforcement learning are distinct, it is customary to denote both by the symbol r. To prevent any confusion, we will clarify the meaning of r whenever there is ambiguity, indicating whether it represents returns or rewards.

At each time step, the asset manager must decide the proportion of investment to be made in each asset, which is referred to as the portfolio weights $w_t = (w_t^1, w_t^2, \dots, w_t^N)$. This decision which is defined as the portfolio allocation question is traditionally guided by a risk-reward function or objective, and various risk measures may be used. For the purpose of this thesis, the Sharpe ratio is assumed to be the guiding objective in most cases. Later on, we will provide a more detailed definition of the Sharpe ratio, but in summary, the Sharpe ratio is a financial metric used to evaluate the risk-adjusted return of a portfolio thanks to the ratio of its average excess return over its volatility or the standard deviation of the returns.

1.3.1 Sharpe Ratio (SR)

Definition 1.3. *Let's consider a portfolio with financial daily returns denoted by r_t and daily market prices by P_t at time t. To keep things simple, we assume a risk-free rate of zero and look at a time horizon from t to T. The Sharpe ratio, originally defined by (Sharpe 1966) and later in (Sharpe 1975), can be expressed as the ratio of the annual financial return $R_{t,T}$ over the annual standard deviation for the considered period $\sigma_{t,T}$:*

$$SR = \frac{R_{t,T}}{\sigma_{t,T}}$$

The annual return is calculated as follows: $R_{t,T} = (\frac{P_T}{P_t})^{\frac{1}{YF(t,T)}} - 1$, where $YF(t,T)$ is the year fraction of the period from t to T. The annual return's standard deviation is given by $\sigma_{t,T} = StdDev(r_t, \dots, r_T) \times \sqrt{252}$, where StdDev denotes the traditional standard deviation of a time series, and $\sqrt{252}$ accounts for the number of trading days in a year.

In essence, the Sharpe ratio is a measure of the portfolio's (excess) return per unit of risk. The higher the Sharpe ratio, the better the portfolio's risk-adjusted performance. The risk-free rate is assumed to be zero in this above definition, but in financial literature, it is also taken as any benchmark interest rate, such as the 3-month Treasury bill rate.

In the context of portfolio allocation, the objective of the asset manager is to select the portfolio weights w_t at each time step that maximize the Sharpe ratio. This can be done by solving an optimization problem with the Sharpe ratio as the objective function and constraints that ensure that the weights sum up to 1 and are within some bounds, such as a minimum investment level or a maximum risk tolerance.

In the rest of the thesis, we will delve deeper into this problem and explore how we can use deep reinforcement learning to solve this question. In particular, we will focus on using policy gradient methods, to optimize the Sharpe ratio and find the best portfolio weights. These methods have shown great promise in recent years and offer a more flexible and adaptive approach to portfolio construction.

The factor $\sqrt{252}$ indicates that we assume 252 days per year. The Sharpe ratio is in some sense a modified z-score of our portfolio financial return.

The Sharpe ratio is widely used in finance as a measure of risk-adjusted performance that helps compare investment strategies and portfolios, as well as make informed investment decisions. It is praised for its simplicity and versatility, and has been extended to better account for drawdowns (Challet 2017) and target diversified portfolios that perform well out of sample (Lopez de Prado 2016).

Because of its tractability, the Sharpe ratio is also commonly used to evaluate hedge funds and mutual funds, as noted by Sharpe (1975) and Sharpe (1992). Moreover, one can compute the statistics of having a specific Sharpe ratio at a given time horizon, which enables inferences about whether the asset manager has real skill or is simply lucky with their reported Sharpe ratio (Benhamou et al. 2019).

However, it is important to acknowledge that the Sharpe ratio has certain limitations (Lopez de Prado 2016, Challet 2017, Benhamou & Guez 2018). For example, it assumes that returns are normally distributed, which is not always true in financial markets. Moreover, the Sharpe ratio does not account for skewed returns or other non-normal return distributions, which can lead to a distorted view of an investment's risk-return characteristics.

1.3.2 Transaction costs

Once the weights are chosen and taken the convention that the initial allocation w_{-1} is filled with zeros (which simply means that before making portfolio allocation decisions, we did not take any previous allocation), the value of the portfolio assuming transaction costs $(b_i)_{i=1,\dots,N}$ proportional to the absolute difference of weights can easily be computed as follows (Lobo et al. 2007)

Definition 1.4. *The portfolio valuation with transaction costs is given as the cumulative products of realised returns minus the transaction costs:*

$$P_T = P_0 \prod_{t=0}^{T-1} \left(1 + \sum_{i=1}^{N} w_t^i \times r_{t+1}^i - \sum_{i=1}^{N} b_i \mid w_{t-1}^i - w_t^i \mid \right). \tag{1.24}$$

This can be reformulated in vectorial form as follows

$$P_T = P_0 \prod_{t=0}^{T-1} \left(1 + w_t \cdot r_{t+1} - b \cdot \mid w_{t-1} - w_t \mid \right), \tag{1.25}$$

where $x \cdot y$ represents the standard dot product of two vectors x, y.

It is worth noting the offset difference between weights and returns. This is to account for the fact that portfolio returns happen in the period following our portfolio allocation decision. Extra costs listed below can be added but to keep things simple, we ignore these additional costs.

While the model selected to factor in transaction costs is typical in the financial sector, particularly when managing significant portfolios, there exist numerous mathematical models for transaction costs in financial literature, each with its own benefits and drawbacks. The following are some of the most frequently employed models for the purpose of reference.

- linear impact also called proportional transaction cost model as presented in (Lobo et al. 2007): this model assumes that the market impact of a trade is proportional to the trade size. This model is realistic for large trades but may not be accurate for smaller trades. This is what we will be using in the rest of the thesis.
- affine transaction costs model: in this model, the transaction cost is an affine function of the traded amount. This model captures the effects of market impact as well as slippage cost due to the transaction size, assuming a linear impact from the traded amount. It could be quite realistic for retail traders who face fixed costs as well as linear cost but is not widely used for institutional trades.
- square-root impact model: this model assumes that the market impact of a trade is proportional to the square root of the trade size. This model has been shown to be a good representation of the market impact faced by institutional traders (Almgren & Chriss 2005). It takes into account the market liquidity and the size of the trade and can be used to estimate the market impact of a trade. High-frequency trading firms, which execute trades frequently throughout the day, rely on this model to obtain a more accurate understanding of their trades' impact on the broader market.

1.3.3 Optimal Weights in a Financial Portfolio with Transaction Costs

Optimizing the weights w_t of a financial portfolio is a classical problem in finance that aims to maximize the expected return of the portfolio while minimizing its risk. Moreover, this problem becomes more complex when transaction costs are taken into account, as the buying and selling of assets incurs additional fees and slippage costs that can significantly impact the portfolio's performance and are non-trivial to include in traditional financial problem.

This explains why the problem of choosing the optimal weights in a financial portfolio with transaction costs can be formulated as an optimization problem. The objective function is typically the expected portfolio return, which is a linear combination of the expected returns of the assets weighted by their respective portfolio weights. The constraints include the portfolio's total value, which must be equal to the initial investment, as well as the individual weight limits for each asset.

Traditional portfolio methods for asset allocation, such as mean-variance optimization, rely on statistical models and assumptions about the distribution of asset returns, which may not hold in practice due to non-stationarity, non-linearity, and other sources of uncertainty and risk. Moreover, these methods often require significant manual tuning and expertise to achieve satisfactory results, which may limit their applicability and scalability.

Deep Reinforcement Learning (DRL) offers a promising alternative to traditional portfolio methods by leveraging the power of neural networks and reinforcement learning to learn optimal policies directly from data and feedback. DRL can handle high-dimensional state and action spaces, nonlinear dynamics, and non-stationary and non-Markovian processes, making it suitable for a wide range of financial applications, including portfolio optimization.

DRL-based portfolio optimization can be framed as a Markov Decision Process (MDP), where the portfolio weights are the actions, and the portfolio performance is the reward function. The neural network policy is then trained using a combination of Monte Carlo sampling and gradient-based methods to maximize the expected long-term reward. The use of deep neural networks allows the policy to capture complex patterns and relationships among assets, which may be missed by traditional methods.

One advantage of DRL-based portfolio optimization is that it can adapt to changing market conditions and new information in real-time, without requiring manual intervention or parameter tuning. DRL can also handle different investment objectives and constraints, such as transaction costs, liquidity, and risk management, by incorporating them into the reward function or as additional constraints. Furthermore, DRL-based portfolio optimization can learn from multiple data sources and modalities, including historical prices, news feeds, social media, and other unstructured data, which may provide additional insights and predictive power.

Overall, DRL-based portfolio optimization holds great potential as an alternative to traditional portfolio methods, offering more flexibility, adaptability, and scalability in the face of complex and uncertain financial markets. However, it is important to note that DRL-based portfolio optimization is still an active area of research and development, and there are many challenges and limitations that need to be addressed before it can be widely adopted in practice.

1.4 Questions of this thesis

Having put the different notations and formulations in place, we can now formulate the questions addressed in this thesis. This introduction explains that the portfolio allocation problem can be framed as an optimization problem where the goal is to find the portfolio weights that maximize a performance objective. Traditional approaches to portfolio allocation rely on mathematical optimization techniques and statistical models. However, these methods often assume simplistic assumptions and do not fully capture the complex dynamics of financial markets.

We have seen that deep reinforcement learning (DRL) provides a framework for addressing this challenge by treating the portfolio allocation problem as a decision-making problem in a dynamic environment. In this framework, the portfolio manager takes actions (i.e., adjusts the portfolio weights) based on the current market state, receives feedback in the form of rewards (i.e., portfolio returns), and learns to optimize its decision-making process over time through trial-and-error.

In this way, the portfolio allocation problem can be viewed as a reinforcement learning problem, where the goal is to learn a policy (i.e., a mapping from market states to portfolio weights) that maximizes the cumulative reward (i.e., the portfolio return) over the long term. The use of deep neural networks in reinforcement learning allows the policy to be learned in a data-driven manner, enabling the portfolio manager to adapt to changing market conditions and learn more complex patterns and relationships in the data.

Hence, the deep reinforcement learning framework provides a way to tackle the portfolio allocation problem in a more dynamic, data-driven, and flexible manner, with the potential to achieve better risk-adjusted returns than traditional portfolio methods.

We will see in this thesis that compared to traditional portfolio methods, the usage of deep reinforcement learning for portfolio allocation has several potential advantages:

- flexibility: traditional portfolio methods typically rely on a fixed set of heuristics and assumptions that may not be well-suited for all market conditions. In contrast, deep reinforcement learning can adapt to changing market conditions and learn more complex patterns and relationships in the data. We will in particular highlight the importance of contextual features that have proven to be instructive in various experiments.
- data-driven: deep reinforcement learning methods are driven by data and can learn from historical patterns and trends in the market. This can help to identify opportunities and risks that may not be apparent through traditional portfolio methods.
- real-time decision-making: deep reinforcement learning can make portfolio allocation decisions in real-time based on the current market conditions, allowing for more timely and informed decision-making.

However, there are also potential drawbacks to using deep reinforcement learning for portfolio allocation, such as the need for large amounts of data to train the neural network, the risk of overfitting to historical data, and the complexity and computational cost of training the neural network. Additionally, the lack of interpretability of the neural network's decision-making process may make it difficult to understand and justify portfolio allocation decisions. Ultimately, the suitability of deep reinforcement learning for portfolio allocation will depend on the specific goals, constraints, and characteristics of the investment strategy and market environment.

In this thesis, we will explore various questions related to portfolio allocation and its connection to deep reinforcement learning. These questions will be addressed in the different chapters:

- Chapter 2 will demonstrate the application of deep reinforcement learning as an independent solution for the portfolio allocation problem. The chapter will emphasize the significance of contextual features and validate the robustness of the approach through walk-forward analysis.
- Chapter 3 will investigate how deep reinforcement learning can be applied as a building block to other portfolio allocation methods, such as selecting the best volatility targeting model. In contrast to the previous chapter that investigate how to use DRL alone, this chapter will explore how DRL can be employed in conjunction with conventional portfolio methods.
- Chapter 4 aims to compare and contrast deep reinforcement learning-based portfolio allocation methods with traditional methods. The chapter will demonstrate that deep reinforcement learning-based methods extend the framework of traditional portfolio allocation methods.
- Chapter 5 will explore the efficiency of deep reinforcement learning methods in sampling and solving expectation calculations, particularly in the context of actor-critic models.
- Last but not least, chapter 6 explores the comparison between Deep Reinforcement Learning (DRL) and Supervised Learning (SL) in terms of their learning approaches. The chapter focuses on understanding the distinctions and similarities between these two learning paradigms.

In terms of significance and contribution to the existing literature, the goal of this thesis is to provide some answers to these questions that are not easy. Indeed, the question of how DRL can be applied as an independent solution for the portfolio allocation problem, considering the significance of contextual features and the robustness of the approach through walk-forward analysis, may advance

our state of knowledge in several ways. Firstly, it could address the limitations of traditional portfolio allocation methods by leveraging the flexibility and adaptability of DRL algorithms. Secondly, it could help to exploit contextual features in the decision-making process to capture the complexity of financial markets. Thirdly, through the walk-forward validation, discussed at length in this thesis, the DRL approach could provide more details on when the model remains effective over time and adapt to changing market conditions.

2 Creating a hedging portfolio with Deep Reinforcement Learning

I visualize a time when we will be to robots what dogs are to humans, and I'm rooting for the machines

Shannon (1948), an American mathematician, electrical engineer, and cryptographer known as a father of information theory

2.1 Introduction

The asset management industry is a prime candidate for the application of machine learning due to the availability of ample data and the strategic importance of quantitative strategies, which do not suffer from emotional or behavioural biases as highlighted by Kahneman (2011). Despite this potential, the use of machine learning in investment decision-making remains limited. This is due to the complexity of the learning environment, causing asset managers to continue relying heavily on traditional methods based on human decisions.

However, recent advances in deep reinforcement learning (DRL) demonstrate its potential in a variety of applications such as game playing (Atari games from raw pixel inputs (Mnih et al. 2013, 2015), Go (Silver et al. 2016), StarCraft II (Vinyals et al. 2019)), but also application in robotics like advanced locomotion and manipulation skills from raw sensory inputs with end to end training of deep visuomotor policies (Levine et al. 2015), or by learning hand-eye coordination for robotic grasping (Levine et al. 2016), thanks to better methods like trust region policy optimization (Schulman, Levine, Abbeel, Jordan & July(2015) or using general advantage estimation (Schulman, Moritz, Levine, Jordan & Abbeel 2015) or proximal optimization (Schulman et al. 2017), or continuous control (Lillicrap et al. 2015), or by learning how to teach car autonomous driving

(Wang et al. 2018).

In this context, we investigate the potential of DRL in solving a classical portfolio allocation problem - finding hedging strategies for an existing portfolio, specifically the MSCI World index in this example. The challenge lies in determining when to add, remove, increase, or decrease these hedging strategies, which perform well under different market conditions, for an augmented asset manager.

2.1.1 Related works

Reinforcement Learning has found applications in various domains in finance, such as electronic markets and market microstructure Spooner et al. (2018), Spooner & Savani (2020), optimal execution Nevmyvaka et al. (2006), Ye et al. (2020), portfolio optimization, option pricing and hedging Du et al. (2020), market making and order book trading Guéant & Manziuk (2019), Jerome et al. (2022), Zhang et al. (2019*b*), Zhang & Chen (2020), robo-advising Capponi et al. (2021), and smart order routing (Hambly et al. 2021).

Given that this thesis specifically applies reinforcement learning to portfolio optimization, it is crucial to examine the extensive body of literature that has already explored this topic. In the subsequent discussion, we will primarily focus on prior studies that have utilized reinforcement learning techniques for portfolio optimization and note that the literature in this field is substantial.

There is a significant body of literature on the application of RL algorithms to portfolio optimization, and the algorithms used depend on the nature of the problem. For instance, some authors have used value-based methods like Q-learning (Du et al. 2016, Pendharkar & Cusatis 2018) SARSA (Pendharkar & Cusatis 2018), and DQN (Park et al. 2020) in discrete conditions. On the other hand, some have employed continuous action models and policy gradients methods like DPG and DDPG (Xiong et al. 2018, Jiang et al. 2017, Yu et al. 2019, Liang et al. 2018, Aboussalah 2020, Cong et al. 2021) along the traditional model-free DRL approach. Also, Zhang et al. (2019*b*) and Zhang et al. (2019*a*) have used Deep RL for very general trading purpose algorithm that can ultimately be used in portfolio allocation.

The modeling of states is an essential aspect in portfolio optimization using RL algorithms, and it is interesting to analyze how different authors approach this task. In most works, the state variables used include time, asset prices, asset past returns, current holdings of assets, and remaining balance. For instance, (Jiang et al. 2017) used historical data, which includes the highest, lowest, and closing prices of portfolio components, to predict the potential growth of assets in the immediate future, and (Xiong et al. 2018) used daily prices of the 30 stocks to study the portfolio selection. Moreover, some authors also incorporate transaction costs (Jiang et al. 2017, Liang et al. 2018, Du et al. 2016, Park et al. 2020, Yu et al. 2019, Aboussalah 2020) and investments in the risk-free asset (Yu et al. 2019, Wang & Zhou 2020, Wang 2019, Du et al. 2016, Jiang et al. 2017) in their models. For example, Yu et al. (2019) proposed a DDPG model that includes prediction of asset price movements based on historical prices and synthetic market data generation using a GAN.

In addition to examining the modeling of the states, understanding the reward signals used in these algorithms is also crucial. The reward signals typically include various performance measures such as portfolio return, Sharpe ratio, and profit. For instance, some works use portfolio return as a reward signal Jiang et al. (2017), Pendharkar & Cusatis (2018), Yu et al. (2019), while others use (differential) Sharpe ratio (Du et al. 2016, Pendharkar & Cusatis 2018). Some works also

consider profit as a reward signal (Du et al. 2016).

To evaluate the performance of the RL algorithms, various benchmark strategies are used. The Constantly Rebalanced Portfolio (CRP) and the buy-and-hold or do-nothing strategy are commonly used as benchmark strategies in these works. For instance, Yu et al. (2019), Jiang et al. (2017) used the CRP as a benchmark, while the buy-and-hold or do-nothing strategy was used as a benchmark (Park et al. 2020, Aboussalah 2020).

In value-based algorithms, Du et al. (2016) compared the performance of Q-learning and Recurrent RL (RRL) algorithms under a value function either as a Sharpe ratio or the total net profit. Pendharkar & Cusatis (2018) tested the performance of three algorithms, SARSA and Q-learning methods with discrete state and action space that maximize either the portfolio return or differential Sharpe ratio, and a TD learning method with discrete state space and continuous action space that maximizes the portfolio return. In addition, Park et al. (2020) proposed a portfolio trading strategy based on DQN, while Dixon & Halperin (2020) applied a G-learning-based algorithm, a probabilistic extension of Q-learning, for wealth management problems.

For policy-based algorithms, researchers have proposed various methods for portfolio optimization. For instance, Jiang et al. (2017) proposed a framework combining neural networks with Deterministic Policy Gradient (DPG) algorithm. The Deep Deterministic Policy Gradient (DDPG) algorithm was explored for portfolio selection of 30 stocks by Xiong et al. (2018). Liang et al. (2018) considered adversarial learning and investigated the performance of DDPG, Proximal Policy Optimization (PPO), and policy gradient method. Yu et al. (2019) used DDPG algorithm to predict asset future price movements based on historical prices and generate synthetic market data using a Generative Adversarial Network (GAN). Furthermore, Cong et al. (2021) embedded alpha portfolio strategies into a deep policy-based method and designed a framework that is easy to interpret. Aboussalah (2020) combined the mean-variance framework and the Kelly Criterion framework using Actor-Critic methods for portfolio optimization.

Furthermore, Wang & Zhou (2020) investigated the entropy-regularized continuous-time mean-variance framework, considering a setup with one risky asset and one risk-free asset. They introduced the Exploratory Mean-Variance (EMV) algorithm, which demonstrated superior performance compared to two benchmarks, including a DDPG algorithm, in terms of annualized sample mean, sample variance of terminal wealth, and Sharpe ratio in their simulations. The continuous-time framework presented by Wang & Zhou (2020) was later extended in Wang (2019) to a larger scale portfolio selection scenario involving d risky assets and one risk-free asset. The generalized EMV algorithm was evaluated using price data from stocks in the S&P 500, with $d \geq 20$, and it outperformed several algorithms, including DDPG.

Additionally, a notable number of Deep RL methods have been applied to non-conventional markets, such as cryptocurrencies, as well as emerging markets like the Chinese market. For example, Jiang & Liang (2016) employed a deep neural network to learn an optimal portfolio allocation policy using historical price data of cryptocurrencies. They formulated the problem as a Markov Decision Process and utilized the Q-learning algorithm to determine the optimal action policy. The proposed strategy's performance was evaluated using a dataset of 11 cryptocurrencies and compared against the performance of a traditional buy-and-hold strategy. Similarly, Zhengyao et al. (2017) applied DRL techniques to Chinese stocks.

Last but not least, there is ongoing research in developing more efficient and effective algorithms for portfolio optimization, such as those that can handle high-dimensional data or those that can

incorporate various constraints and risk factors in a multi-agent reinforcement learning (Mao et al. 2020, Ma et al. 2023), and adversarial learning Liang et al. (2018).

2.1.2 Contributions

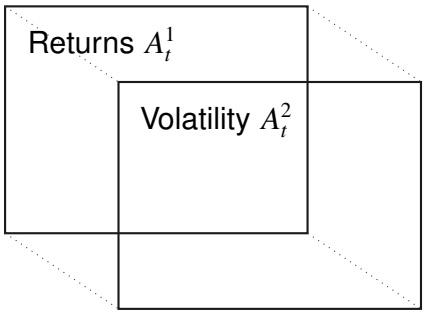
We present three main contributions in this chapter:

- We incorporate contextual information into our deep reinforcement learning model. Simply using past observations is not sufficient for learning in a dynamic and noisy environment. Therefore, we include additional information such as risk aversion levels in financial markets, early warning indicators for future recessions, and corporate earnings by creating two sub-networks. One sub-network takes direct observations (past prices and standard deviation) while the other takes contextual information.
- We introduce a one-day lag between price observation and action in our model to reflect the realistic trading environment. This makes the reinforcement learning problem more challenging but also more realistic.
- We perform a cross-validated empirical study to evaluate the stability of our deep reinforcement learning model. As financial data is non-stationary, it is crucial to validate the selected hyperparameters and ensure that the resulting models are stable over time. To address this challenge, we use a walk-forward analysis approach where the model is iteratively trained and tested on an increasing data set, similar to cross-validation for time series.

2.2 Background and mathematical formulation

The traditional Reinforcement Learning (RL) framework is based on Markov Decision Process (MDP), which assumes that the agent has complete knowledge of all states and can make optimal decisions in every state. However, in real-world scenarios, external events can introduce unpredictable noise into financial market data, making it difficult for the agent to have access to complete state information. Therefore, we prefer to use Partially Observable Markov Decision Process (POMDP) for our financial market model as presented in chapter 1.

To ensure that the observation follows the Markov property, theoretically, we could use all the values of our features seen so far as the observation variable. However, in practice, this would quickly result in a rapidly growing observation as we would add one more dimension at each time step. Hence, this would become rapidly too large to be manageable. To address this issue, we choose to reduce the observation to a subset of past values of our features. These past values will be taken with a set of lags denoted by δ such that $\delta = (0 < i_1 < .. < i_j)$, where i_1 to i_j are a set of increasing and distinct integers that represent the lag of our features from the current value. To make things more concrete, in our experiment, we will select the value of our features over a lag of $(0, 1, 2, 3, 4, 20, 60)$ periods, where a lag of 0 corresponds to the current value of our feature, a lag of 1 to the previous value, and lags of 5, 20, and 60 to values observed one week, one month, and one quarter ago, respectively. While the specific choice of these lags may seem arbitrary, we have found that they work reasonably well in capturing information over the last week, month, and quarter. It is important to note that the machine learning model used in this experiment is highly adaptable to various lags, which makes the specific periods chosen for the lag less important.



with $A_t^1 = \begin{pmatrix} r^1_{t-i_j} & \dots & r^1_t \\ \dots & \dots & \dots \\ r^m_{t-i_j} & \dots & r^m_t \end{pmatrix}$, $A_t^2 = \begin{pmatrix} \sigma^1_{t-i_j} & \dots & \sigma^1_t \\ \dots & \dots & \dots \\ \sigma^m_{t-i_j} & \dots & \sigma^m_t \end{pmatrix}$

Table 2.1: Representation of the regular observations

2.2.1 Observations

Regular observations

There are two types of observations used in this portfolio allocation problem, namely regular and contextual information. Regular observations consist of features that are directly related to the portfolio allocation problem. These features include past prices of the portfolio constituents observed over a set $\delta = (0 < i_1 < .. < i_j)$ of different increasing past periods or any derived features computed over the same set of different increasing past periods. We will denote the past prices by p_{t-i_k} for $i_k \in \delta$ for the asset i. To normalize prices, the past returns of the assets in the portfolio are used instead. For the asset i, this return r_t^i is simply computed as the price percentage change over one period: $r_t^i = \frac{p_t^i}{p_{t-1}^i} - 1$. Moreover, to capture information about regime changes, the volatility of returns over a rolling period of d steps is also included in the regular observations. This rolling volatility is computed at the same time steps as the returns. Hence, we will compute the rolling volatility at the time steps $t, t-i_1, \dots, t-i_j$ as follows: $\sigma_t^i = \sqrt{\frac{1}{d-1}\sum_{u=t-d+1}^{t}\left(r_u^i - \mu_t^i\right)^2}$, where the empirical rolling mean μ_t^i is computed as $\mu_t^i = \frac{1}{d}\sum_{u=t-d+1}^{t} r_u^i$. Therefore, the regular observations form a three-dimensional tensor as described below:

The two-layer setting, which considers past returns and volatilities, is particularly noteworthy as it aims to capture the dynamics of not only returns, but also volatilities. This is different from the features used in (Jiang & Liang 2016) or (Liang et al. 2018), which utilize separate layers to represent open, high, low, and close prices. However, these cannot be calculated for portfolio strategies that only have daily observations, which is the case for most funds, and gives some limitations. Although (Jiang & Liang 2016) or (Liang et al. 2018) do not use directly volatility, the inclusion of open, high, low, and close prices could somehow capture the variation of price as explained in (Rogers & Satchell 1991).

Context observation

Contextual observations refer to supplementary information that offers insights into the current financial context. Practically, contextual observations are various financial data that are not directly related to the portfolio but are assumed to have some predictive power for the portfolio assets. These contextual observations provide valuable information about the broader economic environment that may impact the performance of the portfolio assets and are chosen as follows:

- market sentiment, which is a measure of the level of risk aversion in financial markets. It ranges from 0 (maximum risk aversion) to 1 (maximum risk appetite). Market sentiment is an important factor to consider when making investment decisions, as it can influence the behavior of investors and the direction of market trends.

- bond/equity historical correlation, which is a classical ex-post measure of the diversification benefits of a duration hedge. It is measured on a rolling window of 1-month, 3-month, and 1-year. The correlation between bonds and equities is an essential factor to consider when building a diversified portfolio, as it can provide insights into the risk and return characteristics of different asset classes.

- credit spreads of global corporate - investment grade, high yield, in Europe and the US. These spreads are known to be an early indicator of potential economic tensions. Credit spreads measure the difference in yield between corporate bonds and government bonds of the same maturity. They are an important factor to consider when assessing the credit risk of a company or a country.

- equity implied volatility, which is a measure of the "fear factor" in the financial market. Implied volatility is the market's expectation of the future volatility of an asset. High levels of implied volatility indicate that investors are uncertain about the future direction of the market, which can lead to increased risk and potential losses.

- the spread between the yield of Italian government bonds and the German government bond, which is a measure of potential tensions in the European Union. The spread between Italian and German government bonds reflects the perceived credit risk of Italy compared to Germany. It is an important factor to consider when assessing the stability of the European Union and the potential impact on financial markets.

- the US Treasury slope, which is a classical early indicator for US recession. The Treasury slope measures the difference in yield between long-term and short-term US Treasury bonds. A flattening or inversion of the yield curve can indicate a recession in the US economy.

- other financial variables, such as the dollar, the level of rates in the US, and the estimated earnings per share (EPS), which are often used as a gauge for global trade and activity. These variables provide insights into the health of the global economy and can help investors identify potential investment opportunities or risks.

In addition to the previously mentioned contextual observations, we also consider the maximum and minimum portfolio strategies' returns and the maximum portfolio strategies' volatility. The inclusion of these observations is motivated by the fact that they are useful features in detecting crises.

To store the contextual observations, we use a 2D matrix denoted by C_t. This matrix is composed of stacked past p individual contextual observations, creating a time series of contextual features that can be used as inputs to our asset management agent. The use of a time series of contextual observations enables the agent to consider the historical context when making investment decisions, providing a more nuanced understanding of the market conditions that may impact portfolio performance.

The contextual state writes as $C_t = \begin{pmatrix} c^1_{t-i_k} & \dots & c^1_t \\ \dots & \dots & \dots \\ c^p_{t-i_k} & \dots & c^p_t \end{pmatrix}$.

The fact that contextual states C_t are represented as a matrix implies that when using convolutional layers, 1D convolutions will be employed. Thus, the augmented observations can be written as $O_t = [A_t, C_t]$, where $A_t = [A^1_t, A^2_t]$, and this data will be fed into the two sub-networks of the global network shown in figure 2.1.

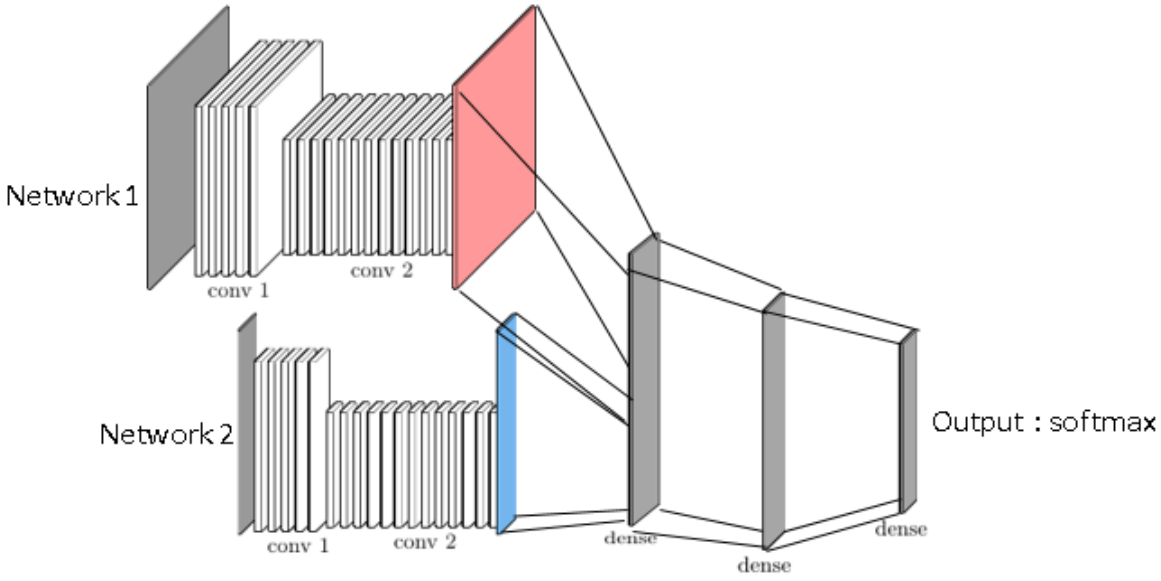


Figure 2.1: Network architecture

2.2.2 Action

In the problem of deep reinforcement learning for asset management, the agent must determine the optimal allocation of the portfolio among the available assets at each time step. However, it is crucial to consider that the actual trading occurs only at the closing price of that day, which creates a one-period delay between the decision and its implementation. This delay is often ignored in research papers, but it is a critical factor that affects the agent's performance, as we will demonstrate in our experiments.

The one-period delay is necessary to mimic the real-world scenario, where asset managers require time to adjust their positions and cannot execute their trading decision immediately. Ignoring this delay can lead to suboptimal investment decisions, as market conditions may have changed during the one-day interval. Hence, the agent needs to consider not only the current market conditions but also their potential evolution during the next day.

Accounting for the one-period delay between the decision and implementation is a crucial factor that significantly influences the effectiveness of the agent's investment strategy. Our experiments will demonstrate the importance of considering this delay and highlight the critical role it plays in achieving optimal investment decisions in a deep reinforcement learning framework.

The output of the agent is an n-dimensional vector that specifies the percentage of the portfolio to invest in each asset. The final layer of our deep network is a softmax layer, which ensures that the portfolio weights lie between 0 and 100% and sum up to 1. The softmax layer is required to create continuous allocations between all the portfolio's assets. The fact that the weights are always positive implies that short selling is not permitted. The agent's objective is to maximize the expected reward over a given period while considering constraints such as risk management and regulatory requirements. Therefore, the agent's investment decision involves striking a balance between maximizing the expected return and managing associated risk.

2.2.3 Reward

There are multiple options available for our reward function. One straightforward reward function is to calculate the net performance of the portfolio at the last train date t_T over the initial value of the portfolio t_0, minus one: $P_{t_T}/P_{t_0} - 1$. However, this reward function would be inadequate if there were no constraints on the weights, as it would encourage the agent to take unlimited risks. In our setting, since the weights are always positive and must sum up to one, these two constraints ensure that the agent does not take unlimited risks.

Another natural reward function is to use the Sharpe ratio, which was introduced in chapter 1. Additionally, we can use the Sortino ratio (Sortino & Price 1994) as another natural reward function, which is a variation of the Sharpe ratio where risk is computed using the downside deviation (instead of the regular standard deviation). The downside deviation is calculated by taking the square root of the ratio of the sum of the squared negative daily returns over the length of all returns multiplied by the annual factor. It's worth noting that there are two common variants of the downside deviation: full and subset, which are closely related. For our experiments, we used the full method as it is the most commonly used.

2.2.4 Policy Gradient with Noisy Observations

A policy is a mapping from the observation space to the action space, $\pi : \mathcal{O} \to \mathcal{A}$. To achieve this, a policy is specified by a deep network with a set of parameters θ. The action is a vector function of the observation given the parameters: $a_t = \pi_\theta(o_t)$. The performance metric of π_θ for time interval $[0, t]$ is defined as the corresponding total reward function of the interval $J_{[0,t]}(\pi_\theta) = R\left(o_1, \pi_\theta(o_1), \cdots, o_t, \pi_\theta(o_t), o_{t+1}\right)$.

After being randomly initialized, the parameters in our deep reinforcement learning algorithm are continuously updated along the gradient direction with a learning rate α through the following equation: $\theta_{k+1} \longrightarrow \theta_k + \alpha \nabla_{\theta_k} J_{[0,t]}(\pi_{\theta_k})$. To perform the gradient ascent optimization, we use the standard Adam optimizer, which combines adaptive gradient descent with root mean square propagation (Kingma & Ba 2015). Let us revisit and explain in more details what policy gradient is doing.

The objective of Reinforcement Learning is to find the optimal policy π_θ that maximizes the expected sum of rewards received by an agent while interacting with an environment. This can be expressed as an optimization problem: where θ represents the parameters of the policy π_θ, τ is a trajectory of states and actions, and $r(s_t, a_t)$ is the reward received by taking action a_t in state s_t.

We use notations introduced in chapter 1 and denote by $r(\tau)$ the sum of reward incurred in this trajectory, so it can be equivalently defined as $r(\tau) = \sum_{t=1}^{T} r(s_t, a_t)$

To optimize this objective, we can use the policy gradient theorem to compute the gradient of the expected reward with respect to the policy parameters θ. We define a function $J(\theta)$ as the expected reward:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi\theta}[r(\tau)]$$

In other words, the J function is an integral of the product of policy and reward:

$$J(\theta) = \int \pi_\theta r(\tau) d\tau. \tag{2.1}$$

2.2.5 Policy gradient theorem

Using the policy gradient theorem, we can compute the gradient of $J(\theta)$ easily. Let us recall its derivation. We can simplify the expression for the gradient of $J(\theta)$ using the log trick identity (also referred to as the connection between the expectation of the logarithmic derivative of the policy and its gradient) which writes as follows:

$$\pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) = \pi_\theta(\tau)\frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} = \nabla_\theta \pi_\theta(\tau). \tag{2.2}$$

Substituting this expression into the previous equation 2.1 and being able to interchange the grad and integral operator (because of dominated convergence), we can compute the gradient of $J(\theta)$ as follows:

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta r(\tau) d\tau = \int \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)] \tag{2.3}$$

The equation

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}[\nabla_\theta \log \pi_\theta(\tau) r(\tau)] \tag{2.4}$$

is referred to as the policy gradient theorem (see (Sutton & Barto 2018)).

In order to simplify the expression $\log \pi_\theta(\tau)$ in equation 2.3, we can use the fact that a trajectory τ is a list of states and actions, and write the probability of performing the trajectory $(s_1, a_1, ..., s_T, a_T)$ using Bayes' rule:

$$\pi_\theta(s_1, a_1, ..., s_T, a_T) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) p(s_{t+1}|s_t, a_t). \tag{2.5}$$

We can then take the logarithm on both sides to obtain:

$$\log \pi_\theta(\tau) = \log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t). \tag{2.6}$$

The next step is to substitute the expression we derived for $\log \pi_\theta(\tau)$ in equation 2.6 into the gradient expression we derived earlier (equation 2.3):

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\nabla_\theta\left(\log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \log p(s_{t+1}|s_t, a_t)\right) r(\tau)\right]. \tag{2.7}$$

We can further simplify the expression by removing the terms $\log p(s_1)$ and $\log p(s_{t+1}|s_t, a_t)$ as they do not depend on θ and we are precisely taking the derivative with respect to θ. Hence,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t=1}^{T} r(s_t, a_t)\right)\right]. \tag{2.8}$$

So far we are left with an expression for policy gradient, which involves calculating an expectation. However, in most cases and in full generality, we will not be able to compute explicitly this expectation because it is intractable. We will therefore approximate the expectation thanks to Monte Carlo sampling, by generating N trajectories and computing their average as follows:

$$\nabla_\theta J(\theta) \simeq \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\right)\left(\sum_{t=1}^{T} r(s_{i,t}, a_{i,t})\right). \tag{2.9}$$

where the subscripts i, t means time step t in the i-th rollout. With the above gradient, we can do gradient descent (ascent) on the parameter θ by updating our parameter along the gradient with a learning rate α:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta). \tag{2.10}$$

This basic algorithm called the vanilla policy gradient algorithm by direct gradient ascent on the Monte Carlo-approximated policy gradient parameters is called the REINFORCE algorithm.

It was originally derived for MDPs in (Sutton, McAllester, Singh & Mansour 1999). However, in the above equation, the transition function does not appear, which means that the Markovian property is not utilized. We use the Bayes'rule in equation 2.5. This implies that the policy gradient can be applied to a Partially Observable Markov Decision Process (POMDP) without requiring any modifications, except for replacing the state s_t with the observation o_t. Therefore, the policy gradient can be used to optimize policies in POMDPs in a straightforward manner.

Let us continue our investigation of Policy Gradient method and highlight the high variance problem. Recall the intuition behind the policy gradient update: making trajectories with higher rewards more probable. However, a potential problem arises when two trajectories with similar positive rewards and a third one with a low negative reward are considered. In this case, the policy gradient algorithm assigns high probabilities to the two trajectories with positive rewards and zero probability to the trajectory with the negative reward. Now, consider adding a large constant number to the reward function, which will come in our Monte Carlo simulations as it generates biased terms from time to time. This will not change the relative ranking between different trajectories' rewards because we are only adding in a constant. But in this case, the negative reward will become positive, and the policy gradient is likely to spread out the likelihood

over the three trajectories since the three rewards are now positive. This will lead to a significant change in the reward distribution, which is not desirable.

This illustrates the high variance in the naive policy gradient algorithm, which can be mitigated by developing methods to reduce the introduced variance. We will also revisit this in greater length in chapter 5.

It is useful here to introduce the concept of causality. The causality fact says that the policy at time t' cannot affect the reward at t if it is in the future: that is if $t < t'$. This simple, commonsensical idea allows us to discard some operands in the summation 2.9 and start the third sum from t and not 1 as initially stated, leading to :

Proposition 2.1. *The on-policy policy gradient term can be estimated thanks to a Monte Carlo simulation as follows:*

$$\nabla_\theta J(\theta) \simeq \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t}) \hat{Q}_{i,t}, \tag{2.11}$$

where the estimated $\hat{Q}_{i,t}$ function is given by the sum of future rewards

$$\hat{Q}_{i,t} = \sum_{t'=t}^{T} r(s_{i,t'}, a_{i,t'}). \tag{2.12}$$

The item $\hat{Q}_{i,t}$ is referred to as the *reward-to-go*. This can alleviate the induced variance as we are multiplying the likelihood by smaller numbers due to the reduction of the summation terms (from t to T, hence decreasing as t increases. We can expect to reduce the variance to some extent.

2.2.6 Baselines

To reduce variance, a commonly used technique is to incorporate baselines. Baselines refer to adjusting the probability of trajectories not solely based on high rewards, but only for those that perform better than the average. Therefore, a baseline, denoted by b, is defined as the average reward.

$$b = \frac{1}{N} \sum_{i=1}^{N} r(\tau). \tag{2.13}$$

We can then incorporate a deterministic baseline b into our original policy gradient expression as follows:

$$\nabla_\theta J(\theta) \simeq \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta \log \pi_\theta(\tau) \left[r(\tau) - b \right]. \tag{2.14}$$

We can justify incorporating the baseline b into the policy gradient equation because we can demonstrate that the expected value remains unchanged with the inclusion of the baseline b as

follows:

$$\mathbb{E}_{\pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau) b\right] = \int \pi_\theta(\tau) \nabla \log \pi_\theta(\tau) b \, d\tau = \int \nabla_\theta \pi_\theta(\tau) b \, d\tau = b \nabla_\theta \int \pi_\theta(\tau) \, d\tau = b \nabla_\theta 1 = 0, \tag{2.15}$$

where we have used in the previous equations the fact that dominated convergence justifies that we can interchange the integration and grad operator. Thus, by subtracting a deterministic baseline, our policy gradient remains unbiased in expectation. Now the question is which baseline is the best? Let us assume in full generality that we use a baseline that is a deterministic expression, hence validates the condition 2.15, let us see which baseline is the best.

2.2.7 Analyzing the Variance with Baselines

Recall the definition of variance:

$$\text{Var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2. \tag{2.16}$$

Hence we can compute the variance of the policy gradient with baseline by taking the expected value of the squared difference between the policy gradient with baseline and its expected value as follows:

$$\text{Var} = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\left[\left(\nabla_\theta \log \pi_\theta(\tau) \left(r(\tau) - b\right)\right)^2\right] - \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau) \left(r(\tau) - b\right)\right]^2. \tag{2.17}$$

Because the baseline is unbiased (meaning $\mathbb{E}_{\pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau) b\right] = 0$) as a consequence of being deterministic (see equation 2.15), the second term in the variance equation 2.17 is a constant term equal to $\mathbb{E}_{\tau \sim \pi_\theta(\tau)}\left[\nabla_\theta \log \pi_\theta(\tau) r(\tau)\right]^2$ that does not depend in our baseline b. Therefore, when solving for the optimal baseline b, denoted by b^*, that minimizes the variance, we need to compute the gradient with respect to b of the first term.

The above equation 2.17 is a quadratic form with respect to b, hence a necessary and sufficient condition to solve for the optimal b is by setting the gradient of the variance to zero as follows (and using the traditional argument that we can interchange derivation and expectation operators thanks to dominated convergence):

$$\begin{aligned}
0 &= \frac{d\text{Var}}{db} && (2.18)\\
&= \frac{d}{db}\mathbb{E}\left[\left(\nabla_\theta \log \pi_\theta(\tau)\right)^2 \left(r(\tau) - b\right)^2\right] && (2.19)\\
&= \frac{d}{db}\mathbb{E}\left[\left(\nabla_\theta \log \pi_\theta(\tau)\right)^2 r(\tau)^2\right] - 2\mathbb{E}\left[\left(\nabla_\theta \log \pi_\theta(\tau)\right)^2 r(\tau) b\right] + b^2 \mathbb{E}\left[\left(\nabla_\theta \log \pi_\theta(\tau)\right)^2\right] && (2.20)\\
&= -2\mathbb{E}\left[\left(\nabla_\theta \log \pi_\theta(\tau)\right)^2 r(\tau)\right] + 2b\mathbb{E}\left[\left(\nabla_\theta \log \pi_\theta(\tau)\right)^2\right] && (2.21)
\end{aligned}$$

The solution of equation 2.21 is simply given by:

$$b^* = \frac{\mathbb{E}\left[\left(\nabla_\theta \log \pi_\theta(\tau)\right)^2 r(\tau)\right]}{\mathbb{E}\left[\left(\nabla_\theta \log \pi_\theta(\tau)\right)^2\right]}. \tag{2.22}$$

The optimal baseline b^* is intuitive: it is the expected reward but weighted by the gradient magnitude $w_{g.m.}$ defined by

$$w_{g.m.} = \frac{\left(\nabla_\theta \log \pi_\theta(\tau)\right)^2}{\mathbb{E}\left[\left(\nabla_\theta \log \pi_\theta(\tau)\right)^2\right]}. \tag{2.23}$$

Therefore, the theoretical optimal baseline b^* writes as follows:

$$b^* = \mathbb{E}\left[w_{g.m.} r(\tau)\right]. \tag{2.24}$$

In practice, this is a cyclical problem as we are precisely computing the gradient of the policy with a baseline that requires to compute the gradient. This explains why practically, we use the average reward as the baseline.

2.2.8 On-Policy vs. Off-Policy

In the domain of reinforcement learning (RL), two significant concepts are introduced: on-policy learning and off-policy learning. On-policy learning refers to the approach of exclusively learning from the current policy π_θ, whereas off-policy learning involves learning from other policies in addition to the current one. In simpler terms, off-policy learning does not require solving for the optimal policy at each step, unlike on-policy learning. It is worth noting that the policy gradient method is an example of an on-policy technique, as it utilizes the gradient with a baseline provided by:

$$\nabla_\theta J(\theta) \simeq \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[\nabla_\theta \log \pi_\theta(\tau) \left(r(\tau) - b\right)\right]. \tag{2.25}$$

In the policy gradient method, the expectation is taken over the current, known trajectory of interest. As a result, each time a new policy is introduced, new samples must be obtained. Given that θ is subject to change and that the policy π_θ evolves over time in the policy gradient approach, the process can be extremely inefficient when implemented using neural networks. This inefficiency arises because, in neural networks, small changes in θ occur, and the overhead for updating the policy can be significant.

2.2.9 Off-policy Learning and Importance Sampling

An efficient solution to this problem is to employ an off-policy learning technique. To accomplish this, we utilize an important tool known as importance sampling. This method enables us to compute the expectation of a distribution $p(x)$ using samples drawn from another distribution $q(x)$ by leveraging an importance ratio as follows

$$\mathbb{E}_{x \sim p(x)}\left[f(x)\right] = \int p(x) f(x)\, dx = \int \frac{q(x)}{q(x)} p(x) f(x)\, dx = \mathbb{E}_{x \sim q(x)}\left[\frac{p(x)}{q(x)} f(x)\right] \tag{2.26}$$

With importance sampling, we can integrate it into the off-policy policy gradient. Suppose we have a trained policy $\pi_\theta(\tau)$ and samples from another policy $\bar{\pi}(\tau)$. We can use the samples from $\bar{\pi}(\tau)$ to compute the $J(\theta)$ function using importance sampling in the following manner:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[r(\tau)\right] = \mathbb{E}_{\tau\sim\bar{\pi}(\tau)}\left[\frac{\pi_\theta(\tau)}{\bar{\pi}(\tau)}r(\tau)\right] \tag{2.27}$$

Let's examine the importance ratio more closely. Recall that $\pi_\theta(\tau) = p(s_1)\prod_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)$. We can simplify the importance ratio as follows:

$$\frac{\pi_\theta(\tau)}{\bar{\pi}(\tau)} = \frac{p(s_1)\prod_{t=1}^{T}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)}{p(s_1)\prod_{t=1}^{T}\bar{\pi}(a_t|s_t)p(s_{t+1}|s_t,a_t)} = \frac{\prod_{t=1}^{T}\pi_\theta(a_t|s_t)}{\prod_{t=1}^{T}\bar{\pi}(a_t|s_t)} \tag{2.28}$$

Interestingly, the policy gradient theorem as presented in equation 2.4 can be extended to off-policy learning via importance sampling. Specifically, we can generalize the policy gradient theorem using importance sampling for off-policy learning as follows. Recall the objective of RL is to find the optimal parameters of our network such that it maximises $J(\theta)$:

$$\theta^* = \arg\max_\theta J(\theta). \tag{2.29}$$

Recall also the expression for $J(\theta)$ as $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[r(\tau)\right]$. To estimate J with some new parameter θ' using importance sampling, we can use the following expression:

$$J(\theta') = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)}r(\tau)\right]. \tag{2.30}$$

Then, we can take the gradient (provided that we can interchange grad and expectation operators thanks to dominated convergence) as follows :

$$\nabla_{\theta'}J(\theta') = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\frac{\nabla_{\theta'}\pi_{\theta'}(\tau)}{\pi_\theta(\tau)}r(\tau)\right] = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)}\nabla_{\theta'}\log\pi_\theta(\tau)r(\tau)\right] \tag{2.31}$$

The latter expression is referred to as the offline policy gradient theorem. We can validate that this is consistent with the original policy gradient theorem as setting θ' equal to θ cancels out the importance ratio $\frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)}$ term and gives us back the policy gradient theorem 2.4:

$$\mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\nabla_{\theta'}\log\pi_{\theta'}(\tau)r(\tau)\right] \tag{2.32}$$

This states that locally when $\theta = \theta'$, an offline policy is equivalent to online policy gradient. When θ' is not equal to θ ($\theta' \neq \theta$), we can compute a first-order approximation for our importance sampling based offline policy gradient as follows. Suppose that $\theta \neq \theta'$, we have:

$$\nabla_{\theta'}J(\theta') = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\frac{\pi_{\theta'}(\tau)}{\pi_\theta(\tau)}\nabla_{\theta'}\log\pi_{\theta'}(\tau)r(\tau)\right] \tag{2.33}$$

$$= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\left(\frac{\prod_{t=1}^{T}\pi_{\theta'}(a_t|s_t)}{\prod_{t=1}^{T}\pi_\theta(a_t|s_t)}\right)\left(\sum_{t=1}^{T}\nabla_{\theta'}\log\pi_{\theta'}(a_t|s_t)\right)\left(\sum_{t=1}^{T}r(s_t,a_t)\right)\right] \tag{2.34}$$

There is a problem in the above equation 2.31 due to the fact that the three terms in the product may be of very different scales (one being very small compared to the other or equally T being so large that the sum depending on T exploding). This would result in high variance. To mitigate the problem of high variance in terms of the expectation, we can make use of the causality fact as before and get:

$$\nabla_{\theta'} J(\theta') = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t)\left(\prod_{t'=1}^{t} \frac{\pi_{\theta'}(a_{t'}|s_{t'})}{\pi_\theta(a_{t'}|s_{t'})}\right)\left(\sum_{t'=t}^{T} r(s_{t'}, a_{t'})\left(\prod_{t''=t}^{t'} \frac{\pi_{\theta'}(a_{t''}|s_{t''})}{\pi_\theta(a_{t''}|s_{t''})}\right)\right)\right] \quad (2.35)$$

We can go further and rely on the causality fact that states that future actions do not affect past weights. Hence we can simplify the product term

$$\prod_{t'=1}^{t} \frac{\pi_{\theta'}(a_{t'}|s_{t'})}{\pi_\theta(a_{t'}|s_{t'})} \qquad \text{into} \qquad \frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}$$

In addition, if we assume that the last term $\prod_{t''=t}^{t'} \frac{\pi_{\theta'}(a_{t''}|s_{t''})}{\pi_\theta(a_{t''}|s_{t''})}$ can be approximated to 1, we get an off-policy policy iteration algorithm as follows:

$$\nabla_{\theta'} J(\theta') = \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\sum_{t=1}^{T} \nabla_{\theta'} \log \pi_{\theta'}(a_t|s_t) \frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\left(\sum_{t'=t}^{T} r(s_{t'}, a_{t'})\right)\right]$$

Hence the on-policy policy gradient seen in proposition 2.1 and given by:

$$\nabla_\theta J(\theta) \simeq \frac{1}{N}\sum_{i=1}^{T}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_{i,t}|s_{i,t})\hat{Q}_{i,t}$$

is changed into

$$\nabla_{\theta'} J(\theta') \simeq \frac{1}{N}\sum_{i=1}^{T}\sum_{t=1}^{T} \frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \nabla_{\theta'} \log \pi_{\theta'}(a_{i,t}|s_{i,t})\hat{Q}_{i,t}$$

which is referred to as the off-policy policy gradient method.

2.2.10 Advanced Policy Gradients

Recall that the update rule for policy gradients is expressed as follows:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

In many cases, certain parameters may have a greater impact on the overall outcome compared to others. Therefore, it would be beneficial to assign a higher learning rate to parameters with less

impact and a lower learning rate o those with more impact. To achieve this, the covariant/natural policy gradient is used. We can consider the constrained view of iterative gradient descent, where we seek to maximize a new set of parameters θ' subject to the constraint that the distance between the two sets of parameters θ and θ' is limited by a threshold ϵ.

$$\begin{aligned} \theta' \leftarrow \arg\max_{\theta'} & (\theta' - \theta)^T \nabla_\theta J(\theta) \\ \text{s.t. } & ||\theta' - \theta||^2 \leq \epsilon \end{aligned} \tag{2.36}$$

where this ϵ controls how far we should go.

However, this constraint is defined in the parameters' space, which limits our control over individual parameters. To address this issue, we can rescale the constraint to limit the step size in terms of the policy space, allowing for greater control over individual parameters. For instance, the following approach can be used:

$$\begin{aligned} \theta' \leftarrow \arg\max_{\theta'} & (\theta' - \theta)^T \nabla_\theta J(\theta) \\ \text{s.t. } & D(\pi_{\theta'} - \pi_\theta) \leq \epsilon \end{aligned}, \tag{2.37}$$

where $D(\cdot, \cdot)$ is a parameterization-independent divergence measure, which usually is the KL-divergence: $D_{KL}(\pi_{\theta'} - \pi_\theta) = \mathbb{E}_{\pi_{\theta'}}[\log \pi_\theta - \log \pi_{\theta'}]$.

It is non-trivial to compute the KL divergence and hence, a typical approximation is to estimate the KL divergence locally using second-order Taylor expansion by:

$$D_{KL}(\pi_{\theta'} - \pi_\theta) \approx (\theta' - \theta)^T F(\theta' - \theta)$$

where F is the Fisher-information matrix defined as:

$$F = \mathbb{E}_{\pi_{\theta'}}[\nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^T]$$

Thus, with F, the rescaled constraint optimization problem can be equivalently rewritten as:

$$\begin{aligned} \theta' \leftarrow \arg\max_{\theta'} & (\theta' - \theta)^T \nabla_\theta J(\theta) \\ \text{s.t. } & ||\theta' - \theta||_F^2 \leq \epsilon \end{aligned}. \tag{2.38}$$

Using Lagrangian, one could solve this optimization problem iteeratively as follows:

$$\theta \leftarrow \theta + \alpha F^{-1} \nabla_\theta J(\theta)$$

The latter equation is the basic idea for the TRPO and PPO algorithms (see (Schulman, Levine, Abbeel, Jordan & July(2015) and (Schulman et al. 2017)).

2.2.11 Implementation

Our algorithm is summarized in algorithm 1, which we call Policy Gradient with Noisy Observations because we introduce random noise in the observations to make the training process harder and thus more robust. The introduction of noise in observations has previously been suggested to improve training in (Liang et al. 2018). It's worth noting that in (Liang et al. 2018), this policy gradient algorithm is referred to as an adversarial policy gradient algorithm. However, we prefer to use the terminology of Policy Gradient with Noisy Observations to emphasize that there is no adversary that actually tries to hurt the agent, but rather there is only random noise in the observations. We fix the number of maximum iterations in our Deep Reinforcement Learning algorithm to 500 (see table 2.8).Furthermore, an early stopping criterion is employed to terminate the training if there is no improvement in performance on the training set over the last 50 iterations. This practice is common in DRL to avoid lengthy computations. We see in practice that the early stopping criterion is triggered in approximately 60 percents of the cases. This suggests that running 500 iterations for a DRL experiment may sometimes be too much. This is because the model has already converged or reached its optimal performance before the maximum number of iterations is reached. Therefore, reducing the number of iterations can potentially save computation time without significantly affecting the model's performance.

Algorithm 1 Policy Gradient with Noisy Observations

Input: initial policy parameters θ $\mathcal{D}$
repeat
  while Not terminal **do**
    observe observation o and select action $a = \pi_\theta(o)$ with probability p and a random action with probability $1 - p$
    for the random action, use a uniform Dirichlet distribution, ensuring that actions reside within the conventional Simplex
    execute a in the environment
    observe next observation o', reward r, and done signal d to indicate whether o' is terminal
    apply noise to next observation o'
    if terminal **then**
      for updates in $\mathcal{D}$ **do**
        compute final reward R
      end for
      update network parameter with Adam gradient ascent $\theta \longrightarrow \theta + \lambda \nabla_\theta J_{[0,t]}(\pi_\theta)$
    end if
  end while
until convergence

2.2.12 Walk forward analysis

In machine learning, k-fold cross-validation is a commonly used approach to evaluate model performance. However, this method can potentially use past data in the test set and break the chronology of data, which is not recommended in finance literature (Lopez de Prado 2018, 2020). To address this issue, we can use a sliding test set and take past data as training data. Two variations

of this approach are the extending walk forward (see figure 2.3) and the sliding walk forward (see figure 2.4).

The idea of walk-forward analysis, also referred to as time series cross-validation, is not a recent innovation. It was initially presented in Pardo (1992), where the author coined the term "walk-forward," despite the approach having existed since the early days of time series analysis and prediction as a discipline. Hyndman & Athanasopoulos (2018) also discuss this method, referring to it as "time series cross-validation," and it has been incorporated into scikit-learn as the TimeSeriesSplit function.

In the extending walk-forward approach, we start with a fixed amount of training data and incrementally add more data to the training set at each new training step. This results in more stable models as we share all past data, but the downside is that the model may adapt slowly to new information. On the other hand, in the sliding walk-forward approach, we always use the same amount of data for training and slide the training data, giving more weight to recent data. This results in more rapidly changing models that may be less stable.

Based on our experience, we found that extending walk-forward is better suited for our DRL model as we have limited data to train our model. Furthermore, as the test set is always after the training set, the walk-forward analysis gives fewer steps compared to cross-validation. In our case, we train our models from 2000 to the end of 2006 to have at least seven years of data and use an extended test period of one year.

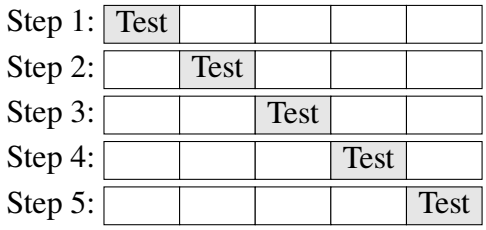


Figure 2.2: K-fold cross validation

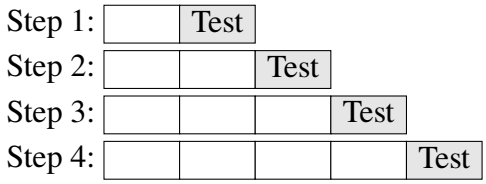


Figure 2.3: Extending Walk Forward

2.3 Experiments

2.3.1 Goal of the experiment

We are interested in finding a hedging strategy for a risky asset. The experiment is using daily data from 01/05/2000 to 19/06/2020. The risky asset is the MSCI world index (see

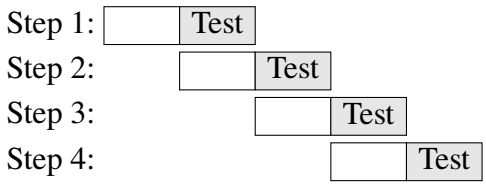


Figure 2.4: Sliding Walk Forward

https://www.msci.com/ - data source: Societe Generale CIB). We choose the MSCI world index because it is a good proxy for a wide range of asset manager portfolios. The hedging strategies are 4 Societe Generale CIB proprietary systematic strategies (see https://sgi.sgmarkets.com/ - data source: Societe Generale CIB), computed and executed by trading bots and further described below. As we use extending walk-forward training, the train set is initially from 2000 to 2006 with a test set in 2007, then the training set is from 2000 to 2007, with a test set in 2008, and so on up to the last training set which is from 2000 to 2019 with a test set in 2020.

2.3.2 Data-set description

Let us first describe the portfolio strategies. The dataset utilized in this study consists of four Societe Generale CIB proprietary 'hedging strategies' that are classified as systematic quantitative strategies. These strategies operate in financial markets according to pre-defined and adaptive trading rules. The four strategies are designed to perform well when the stock markets experience negative returns, crashes, or other specific market configurations.

The four strategies are:

- Directional hedges - designed to react to small negative returns in equities.
- Gap risk hedges - designed to perform well in sudden market crashes.
- Proxy hedges - designed to perform in specific market configurations, such as when highly indebted stocks under-perform other stocks.
- Duration hedges - invest in the bond market, which is a classical diversifier to equity risk in finance.

The financial instruments used in these strategies range from put options, listed futures, single stocks, to government bonds. Some of these strategies function like insurance contracts and bear a negative cost over the long run, which requires balancing costs versus benefits. In practice, asset managers have to determine how much of these hedging strategies are necessary to achieve a better risk-reward balance on top of an existing portfolio. This task will be accomplished by a deep reinforcement learning agent whose ultimate objective is to assign weights to the four hedging strategies in a positive manner that sums up to one, without permitting shorting.

The second part of our data set consists in the features that the DRL agent will use to make its portfolio allocations. The categories chosen include market sentiment, historical correlation between bonds and equities, credit spreads of global corporate, equity implied volatility, the spread between the yield of Italian government bonds and the German government bond, the US Treasury slope, and other financial variables such as the dollar, the level of rates in the US, and the estimated

earnings per share (EPS).

We classify each variable based on their nature as either always positive or not always positive. The variables that are not always positive include rates, spreads, and rolling correlation between equity and bonds. On the other hand, the variables that are always positive include the value of traditional assets like the VIX, dollar index value, GDP, and inflation forecast from economists gathered by the FED.

For the always positive variables, we compute their percentage change and standard deviation over 5, 10, 20, 60, 120, and 250 days, as well as technical value indicators on these values. This information will be used as input for the DRL agent to optimize the portfolio weights of the four hedging strategies.

We can represent the always positive variables as follows: VIX_t: The value of the VIX at time t $DollarIndex_t$: The value of the dollar index at time t GDP_t: The GDP at time t $InflationForecast_t$: The inflation forecast from economists gathered by the FED at time t

For these variables, we compute their percentage change over 5, 10, 20, 60, 120, and 250 days, denoted by $\Delta_{t,5}, \Delta_{t,10}, \Delta_{t,20}, \Delta_{t,60}, \Delta_{t,120}$, and $\Delta_{t,250}$, respectively. We also calculate their standard deviation over the same time periods, denoted by $\sigma_{t,5}, \sigma_{t,10}, \sigma_{t,20}, \sigma_{t,60}, \sigma_{t,120}$, and $\sigma_{t,250}$. For the variables of the portfolio, we compute additionally several technical indicators to gain insights into their behavior over time. Specifically, we calculate the Relative Strength Index (RSI) and Stochastic RSI with periods of 14 and 30, as well as the Moving Average Convergence Divergence (MACD) difference and signal for the traditional MACD settings of 12, 26, and 9. The RSI is a momentum oscillator that measures the strength of price changes over a given period. It ranges from 0 to 100, with readings above 70 indicating overbought conditions and readings below 30 indicating oversold conditions.

The Stochastic RSI is a variation of the RSI that uses the Stochastic oscillator to identify overbought and oversold conditions. It ranges from 0 to 100 and is calculated by comparing the current RSI value to its high and low values over a given period.

The MACD is a trend-following momentum indicator that shows the relationship between two moving averages of a security's price. It is calculated by subtracting the 26-day exponential moving average (EMA) from the 12-day EMA, and then plotting a 9-day EMA of the MACD as a signal line.

By calculating these technical indicators for the portfolio constituents, we can gain a better understanding of its price momentum, overbought or oversold conditions, and trend-following behavior. This information can be useful in making investment decisions and optimizing portfolio weights.

We can represent the not always positive variables as follows: $Rates_t$: Interest rates at time t $Spreads_t$: Credit spreads at time t $RollCorr_{t,w}$: Rolling correlation between equity and bonds over a window w at time t For these variables, we also compute their percentage change over 5, 10, 20, 60, 120, and 250 days, denoted by $\Delta_{t,5}, \Delta_{t,10}, \Delta_{t,20}, \Delta_{t,60}, \Delta_{t,120}$, and $\Delta_{t,250}$. However, since these variables are not always positive, we do not compute their standard deviation. Instead, we use technical value indicators to analyze these variables over time.

By considering the percentage change and standard deviation of the always positive variables, as well as the technical value indicators of the not always positive variables, the DRL agent can

optimize the portfolio weights of the four hedging strategies to achieve the desired risk-reward balance.

The table 2.2 provides a more detailled list of contextual data used

Variable	Ticker	Type	Variations
VIX	VIX Index	Always positive	$\Delta_{t,5}, \Delta_{t,10}, \Delta_{t,20}, \Delta_{t,60}, \Delta_{t,120}, \Delta_{t,250}$, RSI, Stochastic RSI, MACD
Dollar Index	DXY Index	Always positive	$\Delta_{t,5}, \Delta_{t,10}, \Delta_{t,20}, \Delta_{t,60}, \Delta_{t,120}, \Delta_{t,250}$, RSI, Stochastic RSI, MACD
GDP Forecast	ECGDUS 23 Index	Always positive	$\Delta_{t,5}, \Delta_{t,10}, \Delta_{t,20}, \Delta_{t,60}, \Delta_{t,120}, \Delta_{t,250}$, RSI, Stochastic RSI, MACD
Inflation Forecast	ECPIUS 23 Index	Always positive	$\Delta_{t,5}, \Delta_{t,10}, \Delta_{t,20}, \Delta_{t,60}, \Delta_{t,120}, \Delta_{t,250}$, RSI, Stochastic RSI, MACD
Rates	USGG 2,5,10Y Index	Can be negative	$\Delta_{t,5}, \Delta_{t,10}, \Delta_{t,20}, \Delta_{t,60}, \Delta_{t,120}, \Delta_{t,250}$, technical value indicators
Spreads	CDX HY and ITRX XOVER	Can be negative	$\Delta_{t,5}, \Delta_{t,10}, \Delta_{t,20}, \Delta_{t,60}, \Delta_{t,120}, \Delta_{t,250}$, technical value indicators
Equity Bond Correl	Bespoke	Can be negative	$\Delta_{t,5}, \Delta_{t,10}, \Delta_{t,20}, \Delta_{t,60}, \Delta_{t,120}, \Delta_{t,250}$, technical value indicators

Table 2.2: Contextual data used. Provided tickers are Bloomberg tickers. The symbol $\Delta_{t,5}$ means either the percentage change over 5 days if the variable is always positive or the difference over the same period if the variable can be negative.

2.3.3 Evaluation metrics

Asset managers use a wide range of metrics to gauge the success of their investment decision. For a thorough review of those metrics, see for example (Cogneau & Hübner 2009). To keep things simple, we use the following metrics:

- annualised return defined as the average annualised compounded return,
- annualised daily based Sharpe ratio defined as the ratio of the annualised return over the annualised daily based volatility μ/σ,
- Sortino ratio computed as the ratio of the annualised return over the downside standard deviation,
- maximum drawdown denoted by max DD in table 2.6.

Let P_T be the final value of the portfolio at time T and P_0 its initial value at time $t = 0$. Let τ be the year fraction of the final time T. The annualised return is defined as $\mu = (P_T/P_0)^{1/\tau} - 1$. The maximum drawdown is computed as the maximum of all daily drawdowns. The daily drawdown is computed as the ratio of the difference between the running maximum of the portfolio value ($RM_T = \max_{t=0..T}(P_t)$) and the portfolio value over the running maximum of the portfolio value. Hence $DD_T = (RM_T - P_T)/RM_T$ and $MDD_T = \max_{t=0..T}(DD_t)$.

2.3.4 Baseline

Pure risky asset

This first evaluation is to compare our portfolio composed only of the risky asset (in our case, the MSCI world index) with the one augmented by the trading bot and composed of the risky asset and the hedging overlay. If our asset management agent is successful in identifying good hedging strategies, it should improve the overall portfolio and have a better performance than the risky asset.

Markowitz theory

The standard approach for portfolio allocation in finance is the Markowitz model (Markowitz 1952). It computes the portfolio with minimum variance given an expected return which is taken in our experiment to be the average return of the hedging strategies over the last year. The intuition in Markowitz's theory is that an investor wants to have the lowest risk for a given return. The investor is therefore interested in the optimal mean-variance portfolio. In practice, Markowitz's portfolio is the solution of a quadratic program that finds the minimum portfolio variance under the constraint that the expected return is greater or equal to the minimum return. In our baseline, Markowitz's portfolio is recomputed every 6 months to have something dynamic to cope with regime changes.

Follow the winner

This is a simple strategy that consists in selecting the hedging strategy that was the best performer in the past year. If there is some persistence over time in the hedging strategies' performance, this simple methodology should work well. It replicates standard investors' behaviour that tends to select strategies that performed well in the past.

Follow the loser

Following the loser is the opposite of following the winner. It assumes that there is some mean reversion in strategies' performance, meaning that strategies tend to perform equally well in the long term and mean revert around their trend. Hence if a strategy did not perform well in the past, and if there is mean reversion, there is a lot of chance that this strategy will recuperate with its peers.

Hyperparameters tuning

Hyperparameters are important parameters in machine learning that control the behavior of the learning algorithm and have a significant impact on model performance. These parameters are set before training the model and cannot be learned from the data. In order to achieve good performance, it is crucial to find the optimal values for these hyperparameters.

For our deep reinforcement learning agent, we use a Bayesian optimization approach to tune the hyperparameters. We rely on the state-of-the-art Optuna library for this purpose (Akiba et al. 2019). This is preferred over a naive grid search method, as we have a large number of hyperparameters that would make a grid search prohibitive. Similarly, random search would not be ideal as it would not leverage the previous runs.

Optuna search involves constructing a probabilistic model of the performance of the model as a function of the hyperparameters. This model is then used to guide the search for the optimal hyperparameters.

In the case of our deep reinforcement learning agent, hyperparameter tuning is even more critical as it can significantly impact the agent's performance. The agent has several hyperparameters that need to be tuned, such as the learning rate, discount factor, exploration rate, and batch size. Finding the optimal values for these hyperparameters can significantly improve the agent's performance and the resulting portfolio weights. Therefore, using advanced techniques like Bayesian optimization with Optuna is essential for achieving optimal results. The resulting hyperparameters are provided in table 2.8.

2.3.5 Analysis of learned policy

The process of analyzing the data utilized by a model for learning can provide valuable insights into how the model is making its predictions. One notable finding from such an analysis is the importance of contextual variables in the learning process. In particular, it has been observed that models without contextual data generally perform worse than models with contextual data.

To further investigate the importance of different categories of contextual variables, a trial and error analysis was conducted where one category of contextual variables was removed at a time from the model. This allowed for the identification of the most significant contextual category, which was found to be the VIX category. This is because the removal of the VIX category generates the smallest return (15.3 % versus 22.5% for the model with full contextual variables). This is also verified on the Sortino, Sharpe ratio and maximum drawdown and highlighted in bold. For each number, we provide the standard deviation due to the fact that DRL produces results with a certain amount of accuracy.

Table 2.7 provides more detailed information on the significance of the VIX category and its impact on the model's performance, relative to the removal of other categories of features. It is worth noting, however, that more sophisticated feature analyses, such as Shapley values or lime, can also be employed to gain a better understanding of how individual features impact the model's predictions. These methods are discussed further in Chapter 4 of the thesis.

2.3.6 Results and discussion

We compare the performance of the following 5 models: the DRL model based on convolutional networks with contextual states (Sentiment indicator, 6-month correlation between equity and bonds and credit main index), the same DRL model without contextual states, follow the winner, follow the loser and Markowitz portfolio. The resulting graphics are displayed in figure 2.5 with the risky asset position alone in blue and the models in orange. Out of these 5 models, only DRL and Follow the winner can provide significant net performance increase thanks to an efficient hedging strategy over the 2007 to 2020 period. The DRL model is in addition able to better adapt to the Covid crisis and to have better efficiency in net return but also in Sharpe and Sortino ratios over 3 and 5 years as shown in table 2.6. In terms of the smallest maximum drawdown, the Follow the loser model can significantly reduce the maximum drawdown but at the price of a lower return, Sharpe and Sortino ratios. Removing contextual information deteriorates model performances significantly and is illustrated by the difference in terms of return, Sharpe, Sortino

ratio and maximum drawdown between the DRL model with and without context. Last but not least, Markowitz's model is not able to adapt to the new regime change of 2015 onwards despite its good performance from 2007 to 2015. It is the worst performer over the last 3 and 5 years because of its lack of adaptation. For all models, we use the walk-forward analysis as described in the corresponding section. Hence, we started training the models from 2000 to the end of 2006 and used the best model on the test set in 2007. We then train the model from 2000 to the end of 2007 and use the best model on the test set in 2008. In total, we do 14 training sessions (from 2007 to 2020). This process ensures that we detect models that are unstable over time. It is similar in spirit to delayed online training.

2.3.7 Impact of context

In table 2.3, we provide a list of 32 models based on the following choices: network architecture (LSTM or CNN), training with noise in data or not, use of contextual states, and reward function (net profit and Sortino), use of day lag between observations and actions. We see that the best DRL model with the day-lag turnover constraint is the one using convolutional networks, noisy training, contextual states and net profit reward function. These 4 parameters are meaningful for our DRL model and change model performance substantially as illustrated by the table. We also compare the same model with and without contextual state and see in table 2.5 that the use of contextual state improves model performance substantially. This is quite intuitive as we provide more meaningful data to the model.

2.3.8 Impact of one day lag

Reminding the fact that asset managers cannot immediately change their position at the close of the financial markets, modelling the one-day lag turnover to account is also significant as shown in table 2.4. It is not surprising that a delayed action after observation makes the learning process more challenging for the DRL agent as the influence of variables tends to decrease very rapidly with time. Surprisingly, this salient modelling characteristic is ignored in the DRL literature applied to finance.

Table 2.3: Model comparison based on reward function, network (CNN or LSTM units) training with noisy observation and use of contextual state

reward	network	noise in observations training	contextual states	performance with 1 day lag	performance with 0 day lag
Net_Profit	CNN	Yes	Yes	81.8%	123.8%
Net_Profit	CNN	No	Yes	75.2%	112.3%
Net_Profit	LSTM	Yes	Yes	65.9%	98.8%
Net_Profit	LSTM	No	Yes	64.5%	98.5%
Sortino	LSTM	No	Yes	61.8%	87.4%
Net_Profit	LSTM	No	No	56.6%	59.8%
Sortino	LSTM	No	No	48.5%	51.4%
Net_Profit	LSTM	Yes	No	47.5%	50.8%
Sortino	LSTM	Yes	Yes	29.6%	47.6%
Sortino	LSTM	Yes	No	28.4%	47.0%
Sortino	CNN	No	Yes	26.5%	45.3%
Sortino	CNN	Yes	Yes	26.3%	29.3%
Sortino	CNN	Yes	No	-16.7%	16.9%
Net_Profit	CNN	Yes	No	-29.5%	13.9%
Sortino	CNN	No	No	-45.0%	10.6%
Net_Profit	CNN	No	No	-47.7%	8.6%

Table 2.4: Impact of day lag

reward	network	noise in observations training	contextual states	day lag impact
Net_Profit	CNN	Yes	Yes	-42.0%
Net_Profit	CNN	No	Yes	-37.2%
Net_Profit	LSTM	Yes	Yes	-32.9%
Net_Profit	LSTM	No	Yes	-34.0%
Sortino	LSTM	No	Yes	-25.6%
Net_Profit	LSTM	No	No	-3.2%
Sortino	LSTM	No	No	-2.9%
Net_Profit	LSTM	Yes	No	-3.3%
Sortino	LSTM	Yes	Yes	-18.0%
Sortino	LSTM	Yes	No	-18.7%
Sortino	CNN	No	Yes	-18.8%
Sortino	CNN	Yes	Yes	-3.0%
Sortino	CNN	Yes	No	-33.6%
Net_Profit	CNN	Yes	No	-43.4%
Sortino	CNN	No	No	-55.6%
Net_Profit	CNN	No	No	-56.3%

Table 2.5: Impact of contextual state

reward	network	noise in observations training	contextual states impact
Net_Profit	CNN	Yes	111.4%
Net_Profit	CNN	No	122.9%
Net_Profit	LSTM	Yes	18.5%
Net_Profit	LSTM	No	7.9%
Sortino	LSTM	No	13.3%
Sortino	LSTM	Yes	1.2%
Sortino	CNN	No	71.5%
Sortino	CNN	Yes	43.0%

Table 2.6: Models comparison over 3 and 5 years. In each of our experiments using Deep Reinforcement Learning (DRL), we include the standard deviation of the relevant metrics, including return, Sortino ratio, Sharpe ratio, and maximum drawdown, in a separate row below the reported values. These values are enclosed in parentheses and provide insight into the variability of the results, allowing for a more comprehensive assessment of the performance of the DRL model.

	3 Years			
	Return	Sortino	Sharpe	max DD
DRL full context	**22.5%**	**0.95**	**1.17**	-27%
	(6.2%)	(0.23)	(0.31)	(8%)
DRL no context	8.1%	0.34	0.47	-34%
	(2.5%)	(0.07)	(0.08)	(2%)
Loser	9.3%	0.70	0.89	**-15%**
Markowitz	-0.3%	-0.02	-0.01	-41%
Risky asset	10.3%	0.27	0.38	-34%
Winner	13.2%	0.53	0.72	-35 %
	5 Years			
	Return	Sortino	Sharpe	max DD
DRL full context	**16.4%**	**0.77**	**0.96**	-27%
	(4.2%)	(0.16)	(0.22)	(8%)
DRL no context	6.9%	0.35	0.47	-34%
	(1.8%)	(0.05)	(0.05)	(1.3%)
Loser	7.0%	0.63	0.76	**-15%**
Markowitz	-0.1%	-0.00	-0.00	-41%
Risky asset	9.1%	0.43	0.57	- 34%
Winner	10.8%	0.51	0.68	-35%

Table 2.7: Impact of contextual variables for the DRL with contextual variables. We have sorted in increasing order along the return the removal of one category. We can see that the removal of the VIX has the most impact on the DRL model with contextual variables. To emphasize this, we put this model without VIX category in bold.

	3 Years			
	return	Sortino	Sharpe	max DD
No VIX	**15.3%**	**0.68**	**0.83**	**-19%**
	4.4%	**0.16**	**0.22**	**6%**
No Credit spread	17.9%	0.75	0.95	-21%
	4.9%	0.18	0.25	6%
No Dollar Index	18.4%	0.78	0.96	-21%
	5.%	0.18	0.25	7%
No GDP Forecast	18.8%	0.79	1.02	-23%
	5.4%	0.19	0.26	7%
No Inflation Forecast	19.5%	0.82	1.02	-23%
	5.4%	0.2	0.27	7%
No Rates Forecast	19.9%	0.84	1.02	-24%
	5.6%	0.2	0.27	7%
No Eq-Bond correlation	20.3%	0.85	1.04	-24%
	5.4%	0.2	0.27	7%
	5 Years			
	return	Sortino	Sharpe	max DD
No VIX	**11.8%**	**0.55**	**0.69**	**-18%**
	2.9%	**0.11**	**0.15**	**6%**
No Credit spread	12.6%	0.62	0.75	-21%
	3.3%	0.13	0.18	6%
No Dollar Index	12.7%	0.60	0.74	-21%
	3.4%	0.13	0.18	6%
No GDP Forecast	13.7%	0.66	0.82	-23%
	3.5%	0.14	0.19	7%
No Inflation Forecast	14.3%	0.68	0.85	-23%
	3.6%	0.14	0.19	7%
No Rates Forecast	14.3%	0.68	0.83	-24%
	3.8%	0.14	0.2	7%
No Eq-Bond correlation	14.4%	0.69	0.84	-25%
	3.7%	0.15	0.2	7%

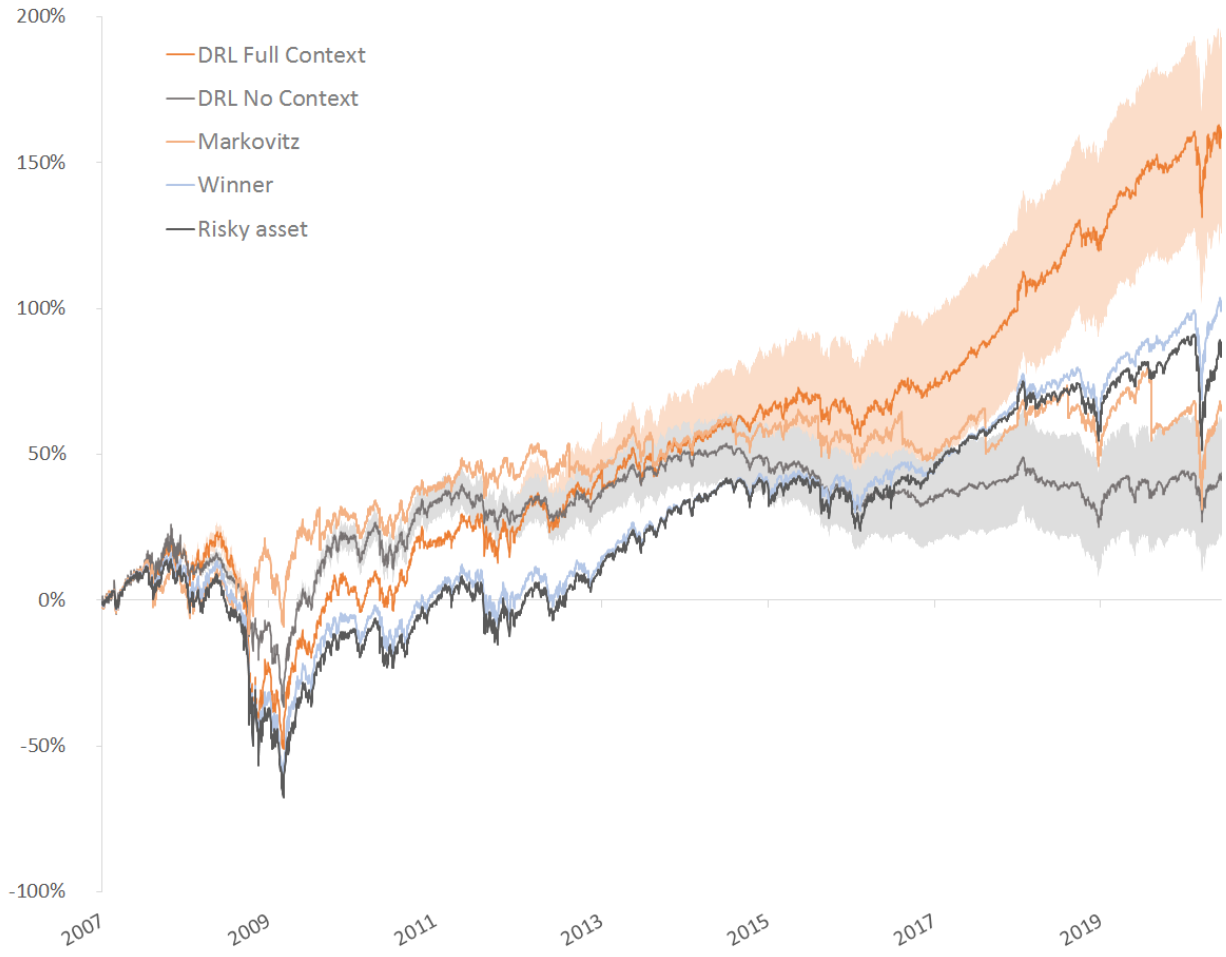


Figure 2.5: Comparison of the various models (DRL with and without context, Follow the winner and Markowitz versus the risky asset). As these strategies are hedging strategies that should improve the risky asset, they should perform better than the risky asset. Among all these models, the best model is the DRL model with contextual features. Most models are not able to continuously adapt to regime changes and consequently under perform compared to the standalone risky asset position over a long period like 2007 to 2020. They initially work well in the initial period from 2007 to 2015 but are not able to adapt after 2015.

Table 2.8: Hyper parameters used

hyper-parameters	value	description
batch size	50	mini-batch size during training
regularisation coefficient	1e-8	L_2 regularisation coefficient applied to network training
learning rate	0.01	Step size parameter in Adam
standard deviation period	20 days	period for standard deviation in asset states
commission	30 bps	commission rate
stride	2,1	stride in convolution networks
conv number 1	5,10	number of convolutions in sub-network 1
conv number 2	2	number of convolutions in sub-network 2
lag period 1	[60, 20, 4, 3, 2, 1, 0]	lag period for asset states
lag period 2	[60, 20, 4, 3, 2, 1, 0]	lag period for contextual states
noise	0.002	Gaussian noise deviation
max iterations *	500	maximum number of iterations
early stop iterations *	50	early stop criterion
random seed	12345	random seed

2.4 Summary

The focus of this chapter is on teaching an asset management agent to learn in a complex and dynamic environment, where observations are sequential, non-stationary and non-homogeneous. Our approach is based on deep reinforcement learning, which is enhanced by contextual information obtained from a second sub-network. We also demonstrate that the delay in taking action after observing has a significant impact on the performance of the model.

To ensure that our learning agent is effectively cross-validated, we implement the concept of walk-forward analysis that has been used extensively in financial statistics. This approach is particularly crucial in a regime-changing environment where a simple train-validation or k-fold cross-validation procedure that disregards the chronological order of observations may not be sufficient. Through the use of "walk forward," which involves conducting a series of train-test iterations where the test data always comes sequentially after the train data, we can effectively evaluate the robustness of the deep RL model and obtain a more precise assessment of its performance.

To improve the learning process of our asset management agent, we use both historical portfolio performance and standard deviation to generate predictive variables for regime changes as well as contextual information. The inclusion of contextual information leads to a more efficient and effective learning process in the presence of noise. Our experiment shows that our approach outperforms baseline methods such as Markowitz or simple winner/loser strategies. However, fine-tuning the numerous hyper-parameters is critical for optimal performance and non-trivial as it involves many parameters: the various lags (lags period for the sub-network fed by portfolio strategies past returns, lags period for common contextual features referred to as the common features in the chapter), standard deviation period and learning rates to cite the main ones.

Although our contextual-based deep RL model is currently efficient, there is still ample room for improvement. To begin with, we need to conduct these extensive experiments on a wider range of examples. Furthermore, we should consider integrating additional information, such as news or social media sentiment, to further enhance its performance. Our work has been presented at three conferences: (Benhamou, Saltiel, Ohana, Atif & Laraki 2021), (Benhamou, Saltiel, Ohana, Laraki & Atif 2020), and (Benhamou, Saltiel, Ungari & Abhishek Mukhopadhyay 2021).

In the following chapter, we will examine the potential of DRL in enhancing traditional volatility targeting techniques by selecting the most optimal model for superior performance.

3 Selecting the experts with Deep Reinforcement Learning

> Uncertainty and expectation are the joys of life. Security is an insipid thing.
>
> *Congreve (1755), a British playwright and poet of the Restoration period and a political figure*

3.1 Introduction

In the previous chapters, we used a model-free Deep Reinforcement Learning (DRL) approach to generate a portfolio allocation from raw data.

In this chapter, we will investigate a hybrid approach that integrates quantitative methods for portfolio weight sizing with DRL for model selection to achieve a target level of portfolio volatility. This combination of quantitative methods and DRL is a compelling approach that can handle varying regimes effectively. By leveraging the strengths of both approaches, we can identify the best-performing model among the different quantitative methods for predicting future volatility and use it to size our portfolio weights. This strategy enables us to maintain a target level of portfolio volatility while adjusting to changing market conditions.

To be more specific, the quantitative approach is employed to maintain a consistent level of volatility in our portfolio, while the DRL approach addresses the need to adapt the model to changing market conditions. This strategy can be described as a DRL expert selection technique, which combines different models to impose the dynamics of our actions. This approach can also be considered as a hybrid of model-based reinforcement learning and DRL model-free techniques, as it selects the best quantitative models to achieve the desired outcome.

The proposed approach utilizes a combination of a model-based RL method and a quantitative approach. While the second step may not strictly be considered a model-based approach in RL

terminology, it still follows a similar path by imposing action dynamics through a model. Thus, the quantitative models used in the approach can be seen as a model-based RL technique, where the aim is to learn and optimize the parameters to impose certain dynamics in the system. On the other hand, the DRL component selects the optimal quantitative model to achieve the desired outcome and can therefore be considered a DRL model-free technique. The optimization of the quantitative models' parameters is similar to training an RL model, where the model imposes its dynamics on the system. By combining these two techniques, the approach can adapt to changing market conditions and maintain a consistent level of volatility in the portfolio, making it suitable for portfolio allocation strategies based on volatility targeting methods.

3.1.1 Combining Model-based and Model-free RL

More generally, as a matter of fact, the combination of model-free and model-based RL approaches is not a new concept in the field of RL. For example, Chebotar et al. (2017) aimed to combine model-based and model-free updates for trajectory-centric RL in robotics. Similarly, Pong et al. (2018) used temporal difference models to develop a model-free deep RL approach for model-based control. van Hasselt et al. (2019) addressed the question of when to use parametric models in RL, while Janner et al. (2019) provided insights into when to trust model-based policy optimization over model-free techniques. Finally, Feinberg et al. (2018) demonstrated how to use model-based value estimation for efficient model-free RL. While these studies are not finance-specific, they offer valuable insights into the use of model-free and model-based RL approaches and their combination mostly in robotics.

To our knowledge, the combination approach of DRL has been predominantly applied in robotics and virtual environments, but has not yet been utilized in financial time series analysis. The aim of this study is to differentiate between various financial models that can be considered as model-based RL methods, which focus on predicting volatility in financial markets for portfolio allocation based on volatility target methods. These models are diverse and include statistical models based on historical data, such as moving average models, GARCH models, HEAVY models, and forward-looking models such as implied volatility or PCA decomposition of implied volatility indices. To determine the best allocation between these models, we utilize deep model-free RL. However, relying solely on the last price data is insufficient as these volatility models exhibit similar behaviors. Thus, we also integrate contextual information such as macro signals and risk appetite indices to provide additional information to our DRL agent, enabling it to select the best pre-trained models for a given environment.

3.1.2 Contributions

Our contributions are five-fold:

- **The combination of RL with quantitative model to select the most appropriate one**. To cope with the challenges of noisy and regime-changing financial time series, practitioners often use a model to represent the dynamics of the market. In our approach, we combine RL with quantitative models to select the most appropriate one. We use a model-free approach to learn from states which model will perform better.
- **The use of contextual information to provide rich or augmented states**. To enhance the state representation, we incorporate additional contextual information, similar to what

we did in Chapter 2. This approach combines RL with quantitative models, and can be considered a hybrid of model-based and model-free RL, although the overall approach is not strictly speaking a model based and model free RL in the traditional sense. In a volatile and constantly changing financial market environment, practitioners often rely on models to capture the dynamics of the market. By using a model-free approach, we can learn which models are more effective at predicting future market behavior.

- **The application of walk-forward cross-validation in DRL testing**. To ensure the stability and robustness of our DRL model in the non-stationary and time-dependent financial data environment, we adopt a walk-forward procedure, a widely used methodology in algorithmic trading but not commonly used in DRL model evaluation. This approach involves iteratively training and testing models on extending data sets to validate the effectiveness of selected hyper-parameters and ensure model stability over time. Walk-forward analysis can be seen as a time series equivalent of cross-validation, and it provides us with the confidence that the resulting models will perform well in real-world scenarios.
- **A features sensitivity procedure.** Drawing inspiration from the feature importance concept used in gradient boosting methods, we have developed a similar feature importance measure for our deep RL model. Our approach involves assessing the sensitivity of our model to input features, enabling us to rank each feature at each date and provide insight into why our DRL agent selects a particular action. This information can be used to identify which features are most influential in driving our model's decision-making process and can aid in understanding the behavior of the agent in different market conditions. Overall, this feature importance measure enhances the interpretability of our DRL model and can assist in identifying important patterns and trends in financial data.
- **A statistical approach to test model stability.** We aim to address a commonly overlooked issue in RL literature which is the statistical difference between the actions generated by our model and predefined baselines or benchmarks. As a matter of fact, it is essential to evaluate the statistical significance of the results obtained from the proposed model in comparison to the baselines to determine the practical usefulness of the proposed approach. We address this gap by introducing the concept of statistical difference to our evaluation metrics. Specifically, we compare the performance of our DRL model with that of several predefined baselines and use statistical tests such as the t-test and p-value to determine the significance of the difference between the two.

  Our approach ensures that the resulting model is not only performing better than the baselines but also that the improvement is statistically significant. This is important because it indicates that the improvement is not just due to chance or randomness. Additionally, our evaluation approach also enables us to identify which features or factors are responsible for the significant difference in performance between the proposed model and the baselines. This information can be used to further improve the model or to gain insights into the underlying dynamics of the financial time series.

3.2 Problem formulation

The asset management industry faces the challenge of determining the optimal investment strategy through asset allocation. This involves adjusting the percentage of investment in each portfolio

asset based on factors such as risk tolerance, investment goals, and horizons, in order to balance risk and reward. Volatility targeting is a frequently used strategy in this regard. It involves forecasting the appropriate amount to invest in various assets based on their level of risk, to achieve a consistent level of volatility over time. This approach is based on the observation that a constant level of volatility offers benefits such as higher returns and lower risk, resulting in higher Sharpe ratios and lower drawdowns compared to a buy-and-hold strategy (Hocquard et al. 2013, Perchet et al. 2016, Dreyer & Hubrich 2017). This evidence can be found not only in stocks but also in bonds, commodities and currencies. Hence, a common model-based RL approach for solving the asset allocation question is to model the dynamics of future volatility.

The persistent nature of volatility, where past volatility predicts future near-term volatility, is a key factor in this approach. While volatility targeting is not considered a model-based reinforcement learning (RL) approach per se, it does share some similarities with it. Specifically, it involves specifying the dynamics of a portfolio in a fully deterministic manner based on the historical trajectories of various assets. This approach can be applied to a diverse range of asset classes such as stocks, bonds, commodities, and currencies, making it highly adaptable to various investment strategies. In contrast to traditional model-based RL approaches, which explicitly model the dynamics of the environment and not only a portfolio, volatility targeting focuses on achieving a consistent level of volatility through the allocation of assets. This involves forecasting the amount to invest in each asset based on its level of risk, in order to maintain a targeted level of volatility over time. By relying on historical data and the persistent nature of volatility, this approach aims to provide higher returns and lower risk compared to a buy-and-hold strategy.

The challenge in the volatility targeting approach revolves around predicting volatility. This volatility is intuitively the degree of variation in the returns of an asset, expressed as its standard deviation. Accurately forecasting volatility is a critical aspect of this method as it ensures effective asset management.

There are multiple approaches to predicting volatility, each with its own strengths and weaknesses. Many of these approaches rely on proprietary models developed by experts in the field, such as those used by the Lombard Odier quantitative team. These models incorporate a range of parameters and methods that give non-identical results that are hard to compare as their ranking and efficiency highly depends on the time window selected. In other words, the best model for a given period is rarely the best model for another method. This variation of efficiency of these methods makes them an ideal candidate for a Reinforcement Learning step to select the best experts among these models. Overall, predicting volatility is a complex and ever-evolving challenge in the field of asset management, requiring sophisticated modeling techniques and a deep understanding of market trends and economic indicators.

We will employ various methods to predict volatility, using a range of techniques and tools as follows:

- Moving average: This model, inspired by Chan (2013) (Chan 2013), utilizes moving averages to predict volatility. The concept behind employing moving averages is to smoothen the volatility signal by averaging the volatility over a specific time period. This denoised volatility signal is then used for forecasting. By doing so, the model aims to filter out noise in the data and offer a clearer depiction of volatility trends. The prediction of volatility using moving averages is based on calculating the average volatility over a fixed period, which serves as a proxy for future volatility. The length of the moving average window can

vary depending on the asset under analysis and the desired level of smoothing. Generally, longer moving average windows yield smoother and less volatile predictions, while shorter windows provide more detailed and higher-frequency predictions. One notable advantage of employing moving averages for volatility prediction is their relative simplicity in calculation and comprehension.

- Level shift: The level shift model is a popular approach to volatility targeting that allows for the incorporation of abrupt jumps in the volatility signal that has been introduced in (Lu & Perron 2010). This two-step approach involves first modeling the trend in the volatility data and then adding in the jumps as a separate component. In the first step of the level shift model, a standard time series model such as an autoregressive integrated moving average (ARIMA) model is used to estimate the trend in the volatility data. This provides a baseline prediction for the future volatility of the asset. In the second step, the model incorporates the jumps in the volatility signal by adding a separate component to the baseline prediction. This component is usually modeled using a Poisson process, which allows for the creation of abrupt changes in the volatility level. The level shift model has been shown to be effective in capturing the stylized facts of volatility, including both the persistence and the abrupt jumps that are often observed in financial markets. It is also relatively simple to implement and can be applied to a wide range of asset classes. However, like any model, the level shift model has its limitations and may not always provide accurate or reliable predictions. It is important to carefully evaluate the assumptions and limitations of any model before applying it to real-world investment decisions.

- GARCH: the generalised auto-regressive conditional heteroskedasticity model (see (Bollerslev 1986)) assumes that the return r_t can be modelled by a time series $r_t = \mu + \epsilon_t$ where μ is the expected return and ϵ_t is a zero-mean white noise, and $\epsilon_t = \sigma_t z_t$, where $\sigma_t^2 = \omega + \alpha\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2$. The parameters $(\mu, \omega, \alpha, \beta)$ are estimated simultaneously by maximising the log-likelihood.

- GJR-GARCH: the Glosten-Jagannathan-Runkle GARCH (GJR-GARCH) model is a variation of the GARCH model (Glosten et al. 1993) with the difference that σ_t, the variance of the white noise ϵ_t, is modelled as: $\sigma_t^2 = \omega + (\alpha + \gamma_{t-1})\epsilon_{t-1}^2 + \beta\sigma_{t-1}^2$ where $I_{t-1} = 1$ if $r_{t-1} < \mu$ and 0 otherwise. The parameters $(\mu, \omega, \alpha, \gamma, \beta)$ are estimated simultaneously by maximising the log-likelihood.

- HEAVY: the HEAVY model utilises high-frequency data for the objective of multi-step volatility forecasting (Noureldin & Shephard 2012).

- HAR: this model is a heterogeneous auto-regressive (HAR) model that aims at replicating how information flows in financial markets from long-term to short-term investors (Corsi 2009). The HAR model is indeed a popular econometric model used to forecast volatility in financial markets, particularly in high-frequency trading. The model is based on the idea that information flows differently among investors with different investment horizons, with long-term investors reacting more slowly to news than short-term investors.

  The HAR model captures this by incorporating three different autoregressive components in its specification, each capturing a different investment horizon: a short-term component (1 day), a medium-term component (1 week), and a long-term component (1 month). The model is thus said to be "heterogeneous" because it allows for differences in how investors process information depending on their investment horizons.

The HAR model can be written as:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \alpha_i r_{t-i}^2 + \sum_{j=1}^{q} \beta_j \sigma_{t-j}^2 + \sum_{k=1}^{m} \gamma_k \sigma_{t-kD}^2, \tag{3.1}$$

where σ_t^2 is the variance of returns at time t, r_{t-i}^2 is the squared return at time $t-i$, σ_{t-j}^2 is the variance of returns j days ago, and σ_{t-kD}^2 is the variance of returns k months ago (where D is the number of trading days per month).

The HAR model has been shown to outperform other popular volatility forecasting models, such as the GARCH and EGARCH models, particularly in high-frequency trading applications.

- Adjusted TYVIX: The Adjusted TYVIX model uses the TYVIX index, which measures the implied volatility of options on the 10-year Treasury note, as a predictor of volatility in the bond future market. The model adjusts the TYVIX index by incorporating information from other relevant variables, such as the level of interest rates and the shape of the yield curve.

- Adjusted Principal Component: The Adjusted Principal Component (APC) model is a method for decomposing a set of historical realized volatility indices into its main eigenvectors and adjusting the resulting volatility proxy to match a realized volatility metric. The model is based on Principal Component Analysis (PCA), a statistical technique that identifies the underlying structure of a dataset by finding the main components that explain the majority of the variance. In the context of financial markets, rolling historical volatilities may not always accurately reflect the future volatility of the underlying asset, as it can be affected by various market factors, such as changes in the correlation structure of financial assets, changing regimes, to name a few. The APC model aims to address this issue by using PCA to identify the main components that explain the variation in a set of implied volatility indices. These components are then used to construct a volatility proxy that captures the underlying structure of the implied volatility indices. The model then adjusts this volatility proxy to match a realized volatility metric, such as the historical volatility of the underlying asset or a benchmark index. By capturing changes in the correlation structure of financial assets, the APC model can be a useful tool for predicting changes in volatility in financial markets. Overall, the APC model provides a powerful way to analyze and model the complex dynamics of financial markets by using statistical techniques to identify the underlying structure of financial data and adjust it to match real-world metrics.

- RM2006: the RM2006 methodology is a method introduced by Risk Metrics and detailed in (Zumbach 2007). It is a method to predict volatility forecasts. It is designed to be more accurate than the previous risk metrics methods. Consistency across risk horizons is obtained by building the methodology using a long memory ARCH process to compute the required forecasts. A large data set covering the main asset classes and geographical areas is used to validate the various sub-components of the methodology. Extensive backtesting using probtiles is done to assess the final performance, as well as the contributions of the various parts.

3.2.1 Mathematical formulation

In this experiment, we are working with a set of nine models ($n = 9$) that each make predictions about the volatility of the rolled U.S. 10-year note future contract, which we will refer to as the "bond future". The daily returns of the bond future are represented by $r^{bond}t$. To determine how much weight to give to each model's prediction, we calculate an allocation for each model based on their predicted volatility. Specifically, if a given model i predicts a volatility for the bond future of $\sigma^{i,pred}t$, and the target volatility of our strategy is denoted by σ_{target}, then the allocation to that model at time t is given by the ratio of the target volatility to the predicted volatility: $b_t^i = \frac{\sigma_{target}}{\sigma_t^{i,pred}}$.

Using the allocation calculated for each bond future volatility model, we can determine the daily amounts invested in each model and create a corresponding time series of returns, denoted by $r_t^i = b_t^i \times r^{bond}t$. This involves investing in bonds future according to the allocation computed by the volatility targeting model i. This results in n time series of compounded returns, denoted by $P_t^i = \prod_{u=t_1}^{t} \left(1 + r_u^i\right)$, where the values of each time series represent the compounded returns of investing according to the allocation of that model.

Our Reinforcement Learning (RL) problem is then to select the optimal portfolio allocation in each model-based RL strategy a_t^i, with respect to the cumulative reward. The portfolio weights must sum up to one and be non-negative, represented by $\sum_{i=1}^{n} a_t^i = 1$ and $a_t^i \geq 0$ for any $i = 1..n$. These allocations are precisely the continuous actions of the DRL model. However, this is not a simple problem, as the different volatility forecasts are quite similar, resulting in the n time series of compounded returns appearing almost identical. This makes the RL problem non-trivial. Our goal is, in a sense, to distinguish between the indistinguishable strategies that are presented in Figure 3.1.

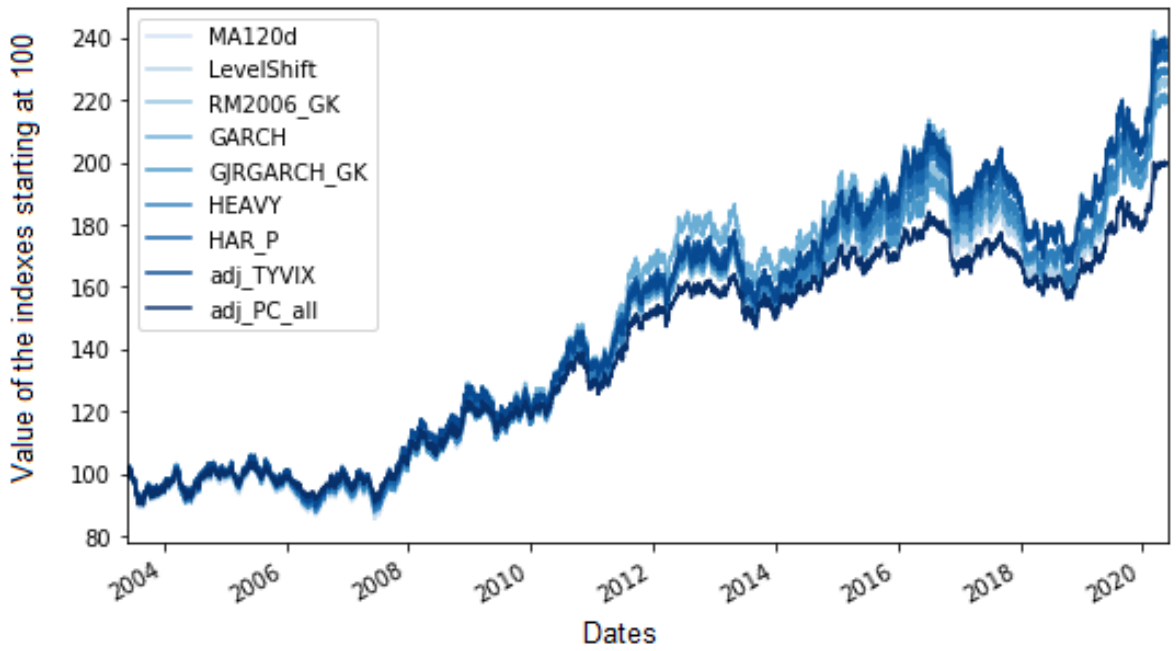


Figure 3.1: Volatility targeting model price evolution. To standardize the different volatility targeting approaches, we have reindexed them all starting at 100.

The returns of the strategies in this problem are highly correlated and similar, as shown by the correlation matrix in Figure 3.2, with the lowest correlation being 97%. This high degree of similarity makes this problem different from standard portfolio allocation problems.

As discussed in Chapter 1, we assume that our process follows a Markov Decision Process (MDP).

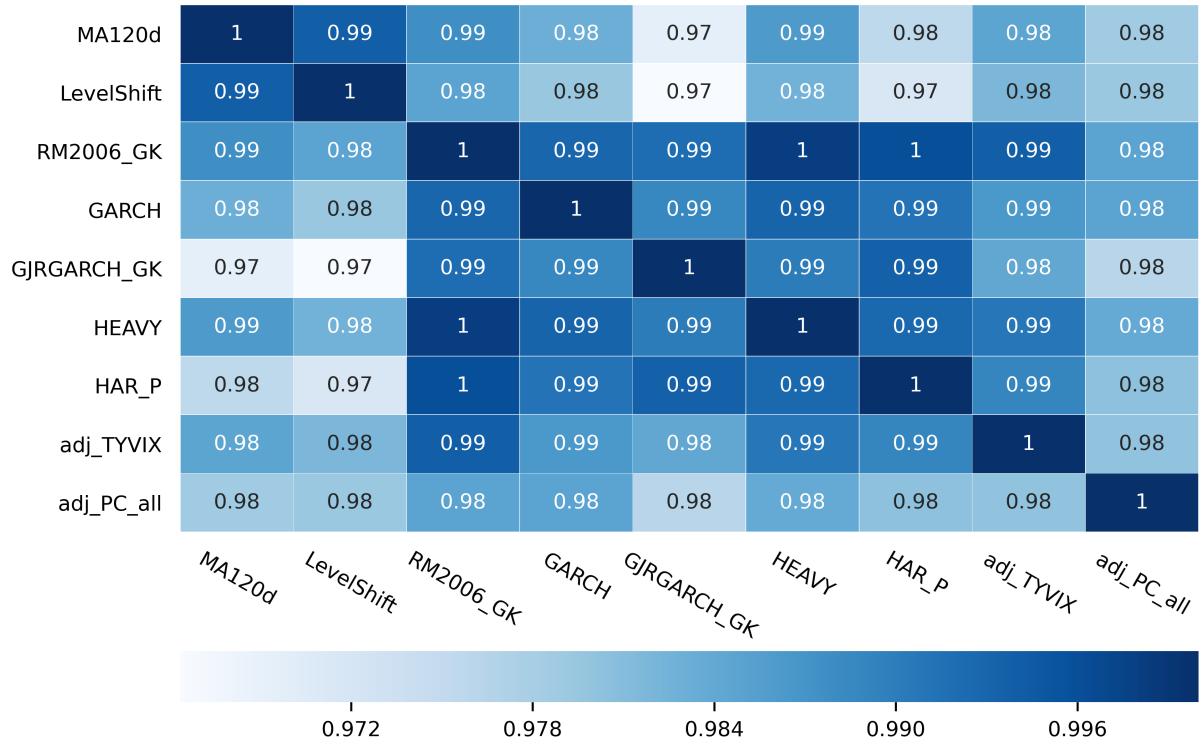


Figure 3.2: Correlation between the different volatility targeting models' returns

The agent's objective is to maximise its expected cumulative returns, given the start of the distribution. If we denote by π the policy mapping specifying the action to choose in a particular state, $\pi : \mathcal{S} \rightarrow \mathcal{A}$, the agent wants to maximise the expected cumulative discounted returns. This is written as: $J^{\pi} = \mathbb{E}_{s_t \sim P, a_t \sim \pi} \left[\sum_{t=1}^{T} \gamma^{t-1} r(s_t, a_t) \right]$.

MDP assumes that we know all the states of the environment and have all the information to make the optimal decision in every state.

From a practical standpoint, there are a few limitations to accommodate. First of all, the Markov property implies that knowing the current state is sufficient. Hence, we modify the RL setting by taking a pseudo-state formed with a set of past observations $(o_{t-n}, o_{t-n-1}, \ldots, o_{t-1}, o_t)$. The trade-off is to take enough past observations to be close to a Markovian status without taking too many observations which would result in noisy states.

In our settings, the actions are continuous and consist in finding at time t the portfolio allocations a_t^i in each volatility targeting model. We denote by $a_t = \left(a_t^1, ..., a_t^n\right)^T$ the portfolio weights vector.

Likewise, we denote by $p_t = \left(p_t^1, ..., p_t^n\right)^T$ the closing price vector, and by $u_t = p_t \oslash p_{t-1} = \left(p_t^1/p_{t-1}^1, ..., p_t^n/p_{t-1}^n\right)^T$ the price relative difference vector, where $\oslash$ denotes the element-wise division, and by $r_t = \left(p_t^1/p_{t-1}^1 - 1, ..., p_t^n/p_{t-1}^n - 1\right)^T$ the returns vector which is also the percentage change of each closing price $p_t^1, ..., p_t^n$. Due to price change in the market, at the end of the same period, the weights evolve according to $w_{t-1} = (u_{t-1} \odot a_{t-1})/(u_{t-1}.a_{t-1})$ where $\odot$ is the element-wise multiplication, and . the scalar product, as shown by Figure 3.3.

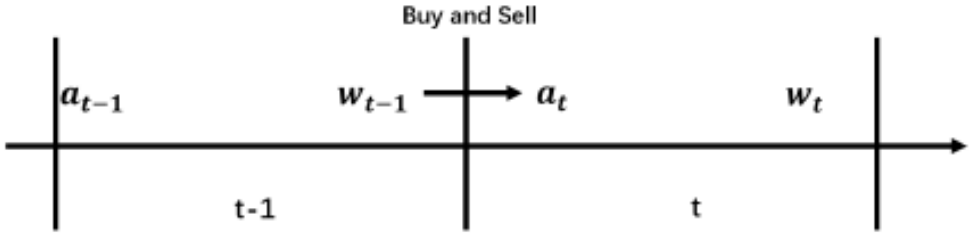


Figure 3.3: Weights evolution due to cost

The agent's objective at time t is to adjust their portfolio vector from previous weights w_{t-1} to a_t by buying and selling assets, while considering transaction costs. These costs are determined by the percentage cost for a transaction, denoted by α, and the L_1 norm operator. The portfolio return at the end of time t is given by $a_t.u_t - \alpha|a_t - w_{t-1}|_1$ as we apply the same transaction costs as in chapter 2, namely proportional cost to the weights.

The cumulative reward of the portfolio strategy is the sum of the logarithmic returns and can be expressed as an expected value of the logarithm of the portfolio returns. This expression is easier to process as a log sum using a TensorFlow graph and can be written as $\mathbb{E}[\log(\sum_{t=1}^{T} a_t.u_t - \alpha|a_t - w_{t-1}|_1)]$.

The agent's actions are modeled using a multi-input, multi-layer convolution network, which is better suited for selecting features in deep reinforcement learning (DRL) for portfolio allocation problems. The aim of the model-free RL method is to find the network parameters using the randomized policy gradient method, summarized by Algorithm 1 presented in the previous chapter, and optimized using traditional Adam optimization with a learning rate of 5%. The training process involves several iterations of a few thousands steps (strictly speaking 250 times the number

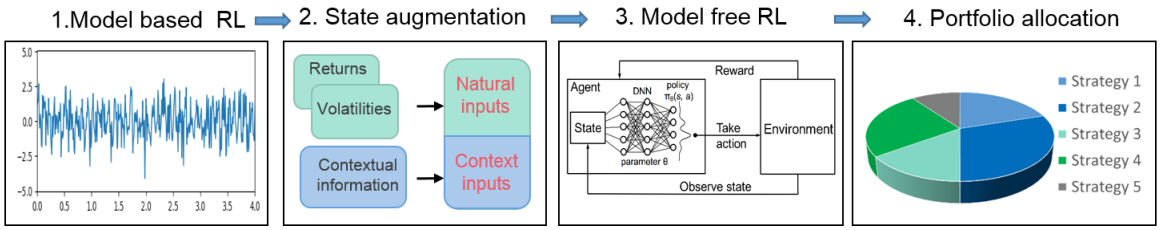


Figure 3.4: Overall architecture

of years of the training data), with an early stop criterion if the cumulative reward does not improve after 15 iterations. This method allows for adaptive gradient descent with root mean square propagation.

3.2.2 Benchmarks

Markowitz

To evaluate the effectiveness of our DRL approach for portfolio allocation, it is essential to compare it with traditional financial methods. One such method is the Markowitz allocation, which was introduced in (Markowitz 1952). It is widely used as a benchmark in portfolio allocation due to its simplicity and intuitive approach to balancing performance and risk.

The Markowitz allocation approach represents risk using the variance of the portfolio. The goal of the Markowitz portfolio is to minimize variance while achieving a given expected return. This is achieved through standard quadratic programming optimization. If we have n model strategies with expected returns denoted by $\mu = (\mu_1, ..., \mu_n)^T$ and covariance matrix of returns denoted by Σ, and we set a targeted minimum return of r_{min}, then the Markowitz optimization problem can be formulated as follows:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2} w^T \Sigma w \\ \text{subject to} \quad & w^T \mu \geq r_{min} \quad \text{and} \sum_{i=1}^{n} w_i = 1, w \geq 0, \end{aligned}$$

where $w = (w_1, ..., w_n)^T$ represents the portfolio weights of the n model strategies.

Effectively, the Markowitz allocation provides a benchmark for portfolio allocation that balances performance and risk by minimizing variance for a given expected return. This traditional financial method serves as a useful point of comparison for evaluating the performance of our DRL approach.

Average

Another benchmark model for indistinguishable strategies is the arithmetic average of all the volatility targeting models. This seemingly simple benchmark is surprisingly effective for indistinguishable models, as it combines diversification and the mean reversion effects of these strategies.

Follow the winner

Another common strategy is to select the best performer of the past year and use it in the subsequent year, known as the "follow the winner" or "the winner" strategy. If there is persistence in the models' performance over time, this simple methodology can work well. It replicates the standard investor's behavior of selecting strategies that have performed well in the past.

The overall procedure of our approach is summarized in Figure 3.4. We start with n models that represent the dynamics of the market volatility. We then augment the states with volatility and contextual information, yielding augmented states. Using a model-free RL approach, we determine the portfolio allocation among the various volatility targeting models. To test the robustness of our resulting DRL model, we introduce a new methodology called walk-forward analysis.

3.2.3 Possible alternate benchmarks

While we did not consider them as potential benchmarks for our analysis, several other approaches could be explored in future experiments. One such method is the Multiplicative Weights Update Method (MWUM) Arora et al. (2012). This method is a meta-algorithm used to optimize a wide range of problems in machine learning, optimization, and game theory. The core idea of MWU is to iteratively update a set of weights in a way that maximizes or minimizes a certain objective function. In each iteration, the algorithm updates the weights based on the gradient of the objective function. The update rule involves multiplying the current weight by a factor that depends on the sign of the gradient. Specifically, if the gradient is positive, the weight is multiplied by a number greater than 1, and if the gradient is negative, the weight is multiplied by a number less than 1. Another interesting approach that we left for further research is to compare our approach with the AdaBoost method Freund & Schapire (1997). In short, the overall objective is to make predictions based on a sequence of examples, without knowledge of the future examples, using AdaBoost. In each iteration, AdaBoost assigns a weight to each training example based on how well the current classifier predicts that example. The algorithm then trains a new weak classifier on the weighted training data and adds it to the ensemble. The final classifier is a weighted sum of the weak classifiers, where the weights are determined by their accuracy and the weights assigned to their corresponding training examples. Last but not least, another interesting approach is the Multiplicative Updates (MU) framework, as presented in Helmbold et al. (1996). The method called the Multiplicative Updates for Online Portfolio Selection (MUPS) algorithm, solves the portfolio problem by iteratively updating the portfolio weights based on the past performance of the assets. In each iteration, the MUPS algorithm calculates the performance of each asset and updates the portfolio weights based on the performance of each asset relative to the other assets. Specifically, the algorithm updates the portfolio weights using a multiplicative rule that depends on the logarithm of the asset's return. The logarithmic rule ensures that the portfolio weights do not become too concentrated in a single asset, which can lead to excessive risk.

3.2.4 Procedure and walk forward analysis

Figure 3.4 provides a summary of the entire process. Initially, we have n models that describe the market volatility dynamics. Next, we combine the volatility and contextual information with the states, resulting in augmented states. This step is presented as the second phase of the process. Subsequently, we employ a model-free RL technique to determine the allocation of the portfolio among the different models targeting volatility, which corresponds to steps 3 and 4. To assess the resilience of our DRL model, we follow the walk forward analysis as presented in chapter 2.

The method of using a sliding test set and incrementally extending the training set is commonly known as anchored walk-forward analysis. It is also referred to as extending walk-forward analysis since the training set is progressively extended. While this approach provides more training data, it can result in slower adaptation to new information, which may negatively impact the model's

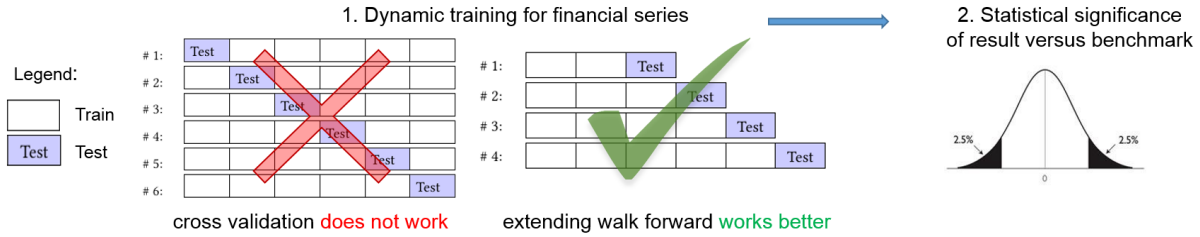


Figure 3.5: Overall training process

performance.

In our case, since we have limited data to train our DRL model, we opted for the anchored walk-forward approach to ensure sufficient training data. It is worth noting that walk-forward analysis typically involves fewer steps compared to traditional cross-validation since the test set is always after the training set.

To validate the selected model, we employ a repetitive test period of one year from 2014 onward. We train our models using data from 2000 to the end of 2013, which provides at least 14 years of data. We then assess the statistical significance of the selected model by comparing its returns with another time series using a T-test. This process is illustrated in Figure 3.5.

Model architecture

The states used in our model consist of two distinct types of data: asset inputs and contextual inputs.

- Asset inputs are derived from a truncated portion of the time series of financial returns of the volatility targeting models. The volatility of these returns is computed over a sliding window of 20 observations. Specifically, if we denote the returns of model i at time t by r_t^i and the standard deviation of returns over the last $d = 20$ periods by σ_t^i, the asset inputs are represented by a 3-D tensor denoted by $A_t = [R_t, V_t]$. Here, R_t represents the returns of all n models at time t, while V_t denotes the corresponding volatilities, with $R_t = \begin{pmatrix} r_{t-n}^1 \ldots r_t^1 \\ \ldots \ldots \ldots \\ r_{t-n}^m \ldots r_t^m \end{pmatrix}$, and $V_t = \begin{pmatrix} \sigma_{t-n}^1 \ldots \sigma_t^1 \\ \ldots \ldots \ldots \\ \sigma_{t-n}^m \ldots \sigma_t^m \end{pmatrix}$.

  Our approach, which involves using two layers to represent past returns and past volatilities, differs significantly from the approach presented in (Jiang & Liang 2016, Zhengyao et al. 2017, Liang et al. 2018). In those works, the models use layers representing open, high, low, and close prices, which may not be available for volatility targeting models.

  The absence of volatility in those works is surprising because it is a crucial factor for detecting regime change in financial markets. By including the past volatilities in our model, we can better capture the dynamics of the market and make more informed decisions about portfolio allocation.

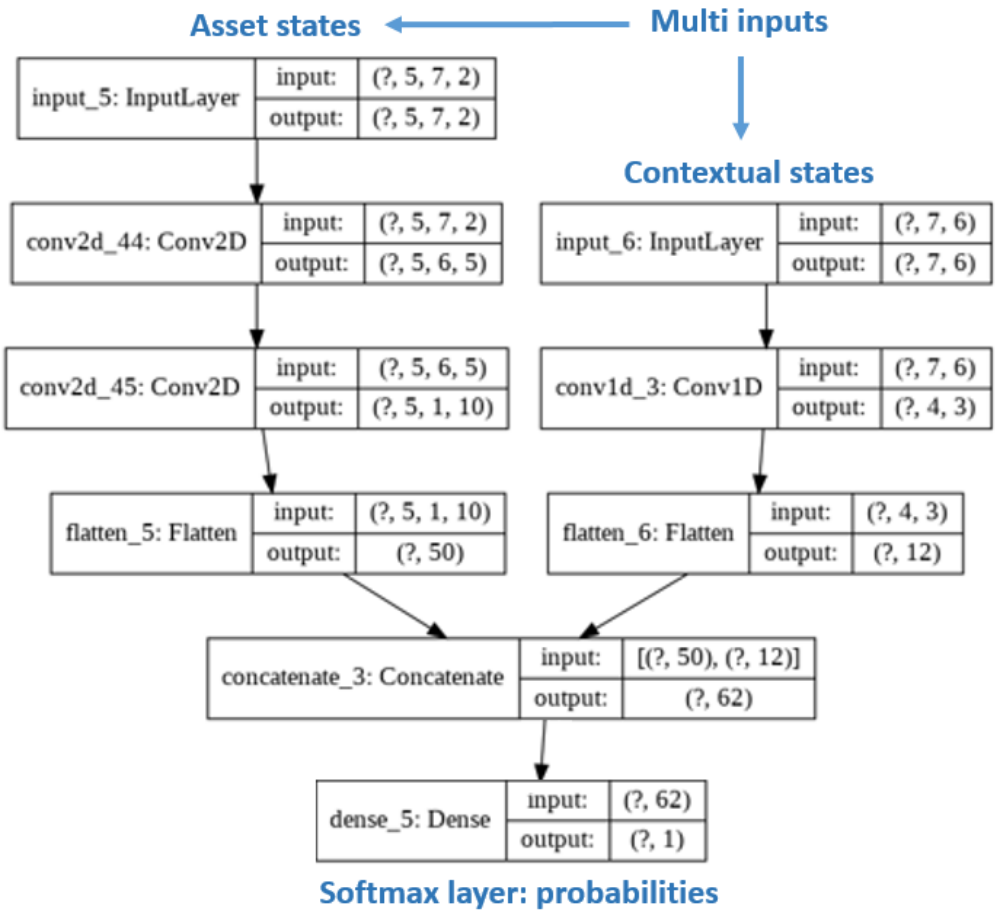


Figure 3.6: Multi-input DRL network

- In our model, contextual inputs consist of a truncated portion of the time series of additional data that provide relevant contextual information. This information enables our DRL agent to learn about the context and make informed portfolio allocation decisions. In our problem, we use short-term and long-term risk appetite indices, short-term and long-term macro signals, as well as the maximum and minimum portfolio strategies' return and volatility as contextual inputs. The standard deviations of these inputs are particularly useful for detecting regime changes.

  The contextual observations are stored in a 2D matrix denoted by C_t, which includes stacked past p individual contextual observations. Specifically, the contextual state at time t is given by $C_t = \left[c_{t-p+1}, \ldots, c_t\right]$, where c_t represents the contextual observations at time t.

  The inclusion of contextual inputs in our model enables our DRL agent to take into account various market factors and trends that may impact volatility and make more informed portfolio allocation decisions. By using both asset and contextual inputs in the state representation, our model can capture a more comprehensive understanding of market dynamics and provide more accurate predictions.

$$C_t = \begin{pmatrix} c^1_{t-n} \ \ldots \ c^1_t \\ \ldots \ \ldots \ \ldots \\ c^p_{t-n} \ \ldots . \ c^p_t \end{pmatrix}.$$

The output of our DRL model is a softmax layer that provides the allocation weights for the various volatility targeting models in the portfolio. Since the dimensions of the asset and contextual inputs are different, the network is a multi-input network that includes several convolutional layers and a final softmax dense layer. Figure 3.6 illustrates the architecture of our model.

The multi-input architecture of our model allows us to process both asset and contextual inputs simultaneously, which is crucial for capturing the complex relationships between market factors and making informed portfolio allocation decisions. The convolutional layers in the network are designed to extract features from both the asset and contextual inputs, which are then used to inform the final portfolio allocation decision. The softmax dense layer at the end of the network provides the allocation weights for each volatility targeting model in the portfolio.

The use of a softmax function ensures that the sum of the allocation weights adds up to 1, which is a desirable property for portfolio allocation decisions. By optimizing the softmax layer using the RL framework, our model can learn to allocate the portfolio weights based on the current market conditions and maximize the expected returns while controlling for risk.

3.2.5 Features sensitivity analysis

Neural networks often present a challenge in terms of their explainability, making it difficult to understand their behaviors. To address this challenge, we employ a methodology inspired by computer vision, which allows us to relate features to actions by performing feature sensitivity analysis.

In our DRL model, the neural network is a multivariate function that takes as input all the relevant features, including historical performances, standard deviations, contextual information, short-term and long-term macro signals, and risk appetite indices. We denote these inputs by X, which is an element of $\mathbb{R}^k$, where k is the number of features. The output of the network is the action vector Y,

which is an n-dimensional array with elements between 0 and 1. The action vector is a subset of $\mathbb{R}^n$, denoted by $\mathcal{Y}$. Thus, the neural network is a function $\Phi : \mathbb{R}^k \to \mathcal{Y}$, where $\Phi(X) = Y$.

To relate the features to the actions, we perform sensitivity analysis by computing the partial derivatives of the output vector with respect to each input feature. We then take the L1 norm of the partial derivatives to project the sensitivity of each feature onto the action vector. Specifically, we compute $|\frac{\partial \Phi(X)}{\partial X}|_1$, where $|\cdot|_1$ denotes the L1 norm. The choice of the L1 norm is arbitrary, but it is intuitively motivated by the desire to scale the distance of the gradient linearly.

By performing feature sensitivity analysis, we can gain insights into how the input features affect the output actions of the model. This approach provides a more interpretable understanding of the DRL model's decision-making process and allows us to better understand how the model allocates portfolio weights based on the input features.

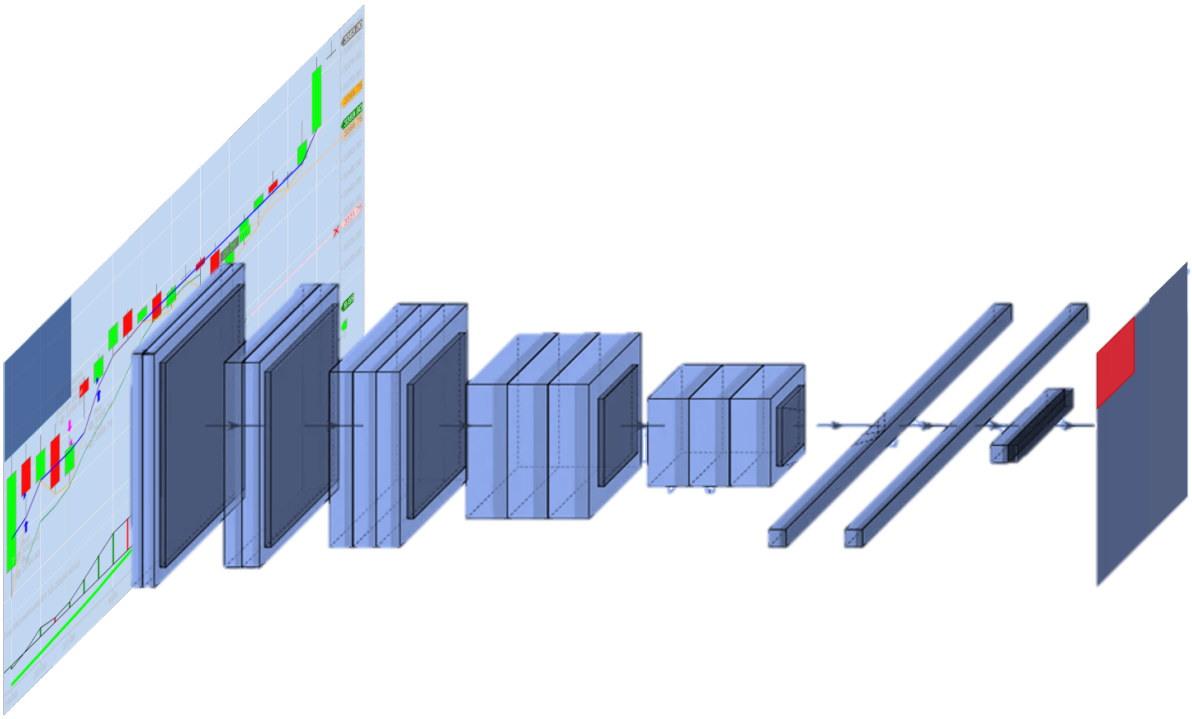


Figure 3.7: Features sensitivity summary

To measure the sensitivity of the DRL model's outputs, we perform a "what if" analysis, where we change the initial feature by its mean value over the last d periods. This approach allows us to measure the impact of changing a feature from its mean value to its current value on the model's decision-making process. Inspired by computer vision, we do not switch off the pixel, but rather set the feature to its mean value to avoid favoring large features. We are interested in measuring the sensitivity of our actions when a feature deviates from its mean value, as shown in Figure 3.7.

We compute the resulting value numerically to determine the importance of each feature in the decision-making process. We rank the importance of these features and assign the value 100 to the most important feature and 0 to the least important. The resulting feature importance plot is presented in Figure 3.8.

The analysis reveals that the historical average of returns (HAR) and volatility are the most important features, followed by various returns and volatilities for the TYVIX model. Although returns and volatility are the dominating features among the most important, macro signals 0d observations come in as the 12th most important feature out of over 70 features, with a very high score of 84.2.

The feature sensitivity analysis confirms two important points: firstly, it is useful to include volatility features as they are good predictors of regime changes, and secondly, contextual information, such as the macro signals, plays a role in the decision-making process. These findings provide valuable insights into the DRL model's decision-making process and highlight the importance of including relevant features to improve its performance.

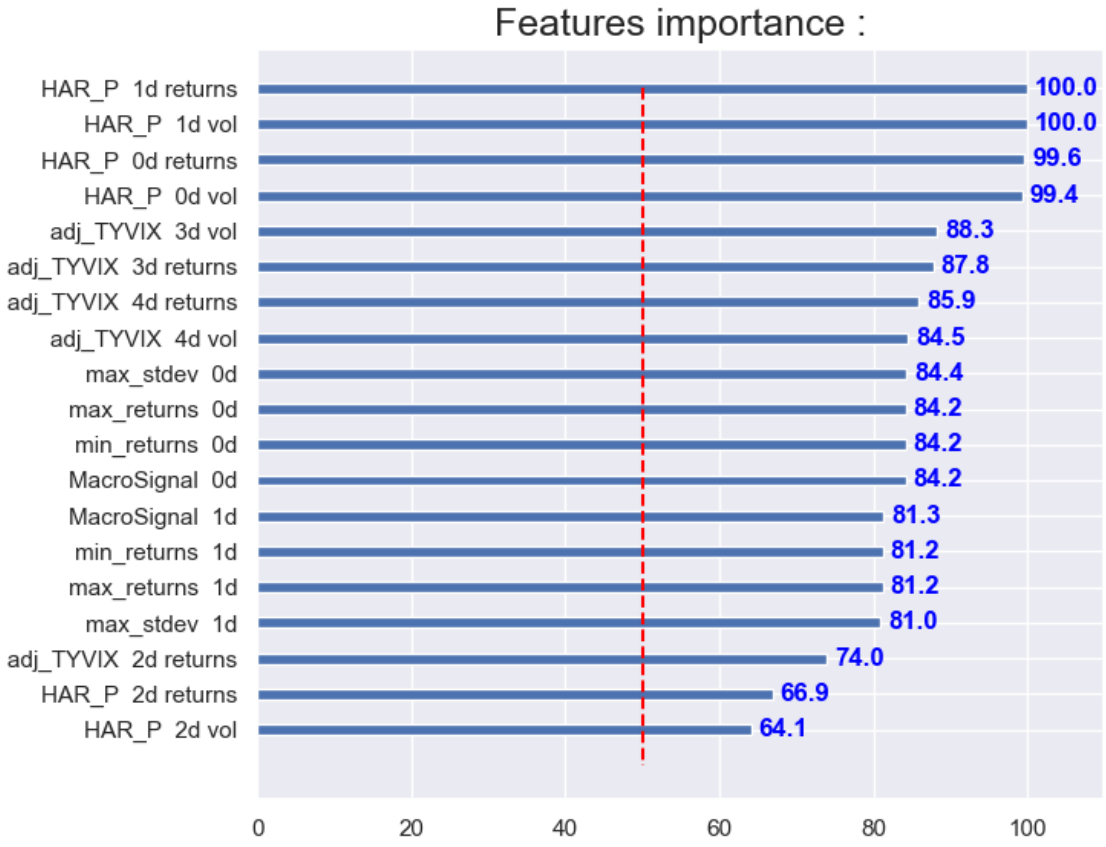


Figure 3.8: Model explainability

3.3 Out of sample results

In this section, we compare the performance of several portfolio allocation models, including the deep RL model with contextual inputs and standard deviation, the deep RL model without contextual inputs and standard deviation, the average strategy, the Markowitz portfolio, and the

"winner" strategy. The performance results are based on the combination of seven distinct test periods, each year from 2014 to 2020. The resulting performance is plotted in figures 3.9, 3.11, 3.10 and 3.12, where the first two figures present the DRL model with contextual features called the DRL1 model while the last two figures the DRL model without contextual features and later called DRL2 model.

The comparison reveals that the deep RL model with contextual information and standard deviation significantly outperforms the other models in terms of performance. This model ends with a final return of 157, while the deep RL model without contextual inputs and standard deviation, the average strategy, the Markowitz portfolio, and the "winner" strategy end with final returns of 147.6, 147.8, 145.5, and 143.4, respectively.

These results suggest that the inclusion of contextual information and standard deviation in the DRL model significantly improves its performance in terms of portfolio allocation decisions. The performance of the deep RL model without contextual inputs and standard deviation is only slightly better than the other traditional portfolio allocation models, suggesting that the inclusion of these features is crucial for the DRL model's superior performance.

Overall, these findings demonstrate the potential of using DRL models with contextual inputs and standard deviation for portfolio allocation decisions, particularly in the context of volatile and complex financial markets. By incorporating relevant features and using the RL framework, these models can learn to make informed portfolio allocation decisions that maximize returns while controlling for risk, outperforming traditional portfolio allocation methods.

To make such a performance, the DRL model needs to frequently rebalance between the various models (Figure 3.13) with dominant allocations in GARCH and TYVIX models (Figure 3.14).

3.3.1 Results description

Risk metrics

We present various statistics in Table 3.1 for different time horizons, including 1, 3, and 5 years. For each horizon, we highlight the best-performing model in bold, according to the column's criterion. The Sharpe and Sortino ratios are computed based on daily returns, and the maximum drawdown (denoted by mdd in the table) is also calculated on daily returns.

DRL1 is the DRL model with standard deviations and contextual information, while DRL2 is a model with no contextual information and no standard deviation. The comparison shows that, overall, DRL1, the DRL model with contextual information and standard deviation, outperforms other models for 1, 3, and 5 years, except for the three-year maximum drawdown. This model provides a 1% increase in annual net return for a 5-year horizon and increases the Sharpe ratio by 0.1. It also reduces most of the maximum drawdowns, except for the 3-year horizon.

In contrast, traditional financial methods, such as Markowitz portfolio selection and the "winner" strategy, do not perform as well compared to a naive arithmetic average and when compared to the DRL model with contextual and standard deviation inputs. One potential explanation for this finding may be that these volatility targeting strategies are very similar, making the diversification effects ineffective.

These results highlight the potential benefits of using DRL models with contextual information

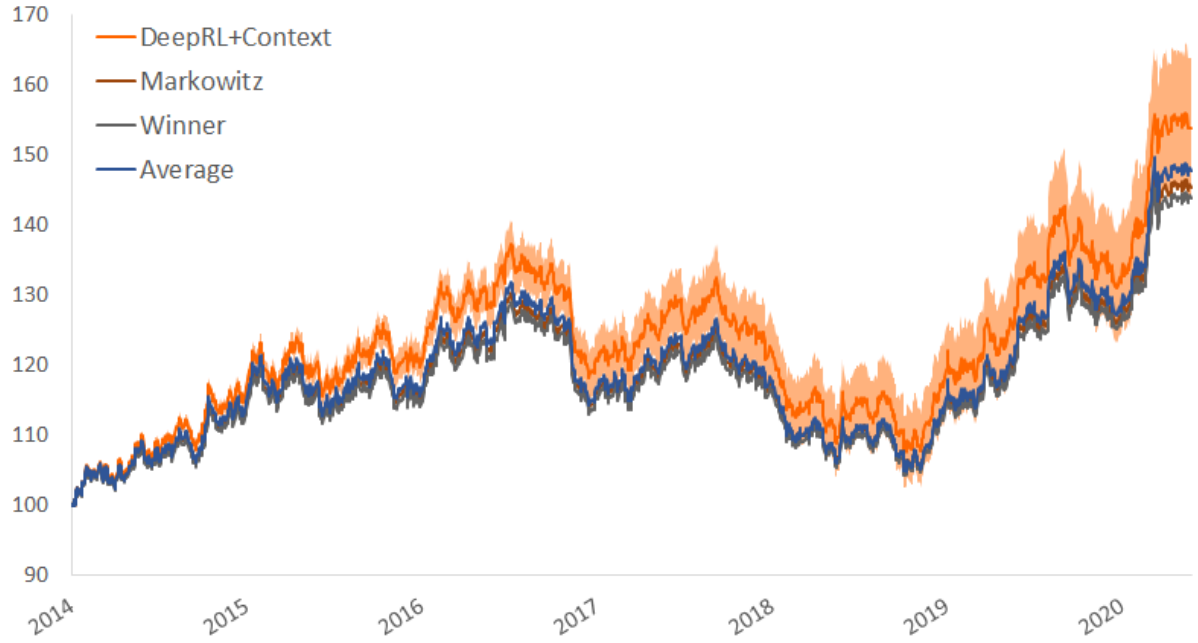


Figure 3.9: Comparison of the DRL model with contextual variables with other models: the average of all the strategies, the follow the winner model and the Markowitz portfolio. We can see that DRL model with context is reasonably able to improve over either the follow the winner and the Markowitz approach. The orange area plot is computed as plus or minus one standard deviation computed over the last 30 runs in the DRL method.

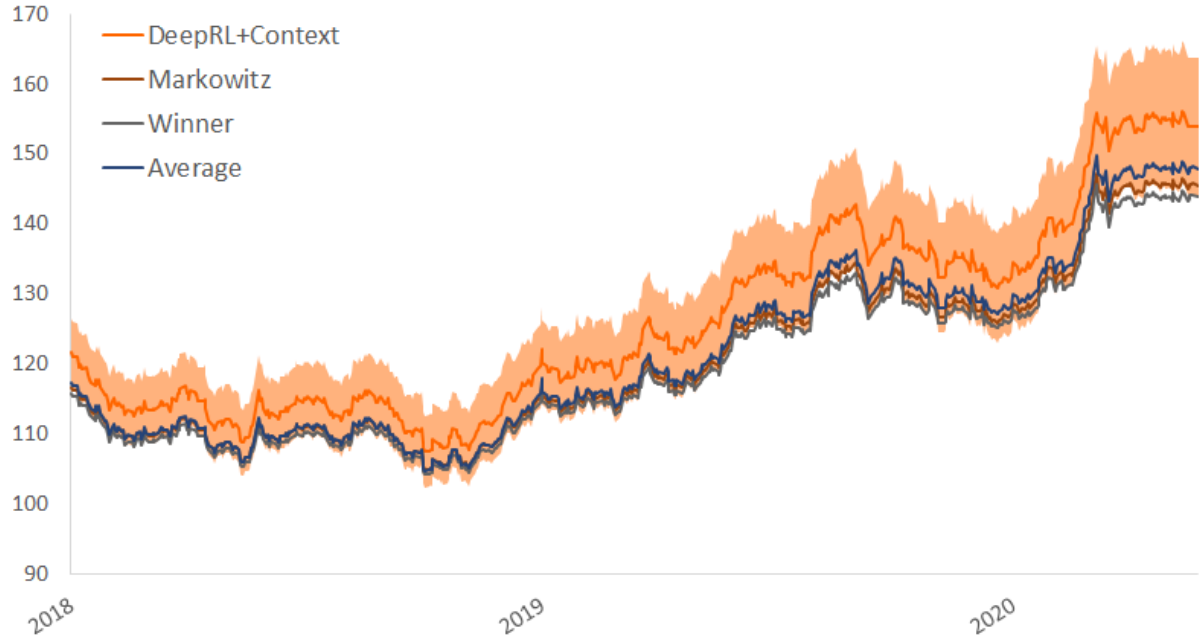


Figure 3.10: Zoom of the DRL model with contextual variables over the period of 2018 up to end of 2020.

and standard deviation inputs for portfolio allocation decisions. These models can outperform traditional methods and provide higher returns while reducing risks. The findings also suggest the importance of including relevant features, such as contextual information and volatility, in the input features to improve the DRL model's performance. Overall, these insights can be useful for practitioners and investors looking to improve their portfolio allocation strategies in volatile and complex financial markets.

Statistical significance

Following the methodology described in Figure 3.5, we conduct a T-statistic test to validate the significance of the results obtained for the various walk-forward test periods. The overall idea is to test if the difference in returns between our Deep RL model is really significant or not as we can see that the uncertainty curve of figures 3.9, 3.11, 3.10 and 3.12 seem to indicate that the DRL model with contextual features can be considered to overperform other models while this is not the case for the DRL model without contextual features. So specifically, we test in our statistical test the null hypothesis that the difference in returns between two models is equal to 0. To do so, we compute the T-statistic and the p-value for each pair of models and present the results in Table 3.2. We set a p-value threshold of 5% and highlight the cases where we can reject the null hypothesis in bold. We not only do this for the DRL model but also for the winner model to see how the statistical significance test applies on a non DRL model;

The results reveal that the DRL1 model is statistically different from the DRL2 (DRL2 model is the DRL model without contextual features) and "winner" models, as the p-values are below the threshold. However, we fail to reject the null hypothesis for the other models, indicating that

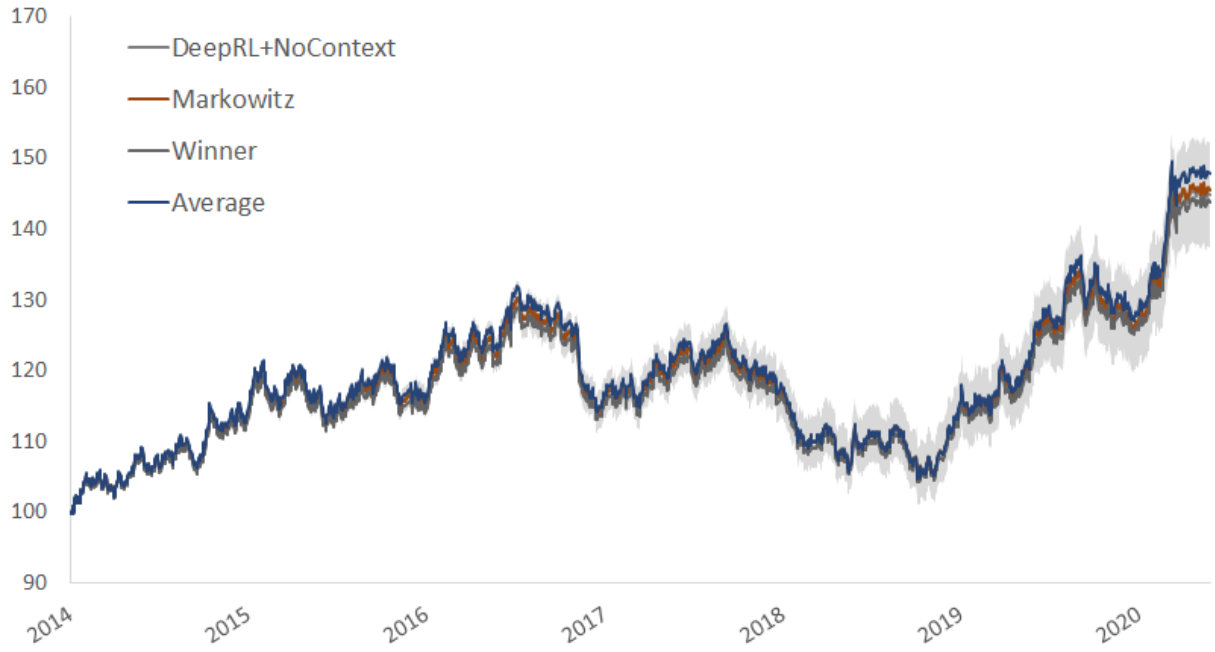


Figure 3.11: Comparison of the DRL model without contextual variables with other models: the average of all the strategies, the follow the winner model and the Markowitz portfolio. We can see that DRL model without context is very comparable to the follow the winner and the Markowitz approach. The grey area plot is computed as plus or minus one standard deviation computed over the last 30 runs in the DRL without contextual variables model.

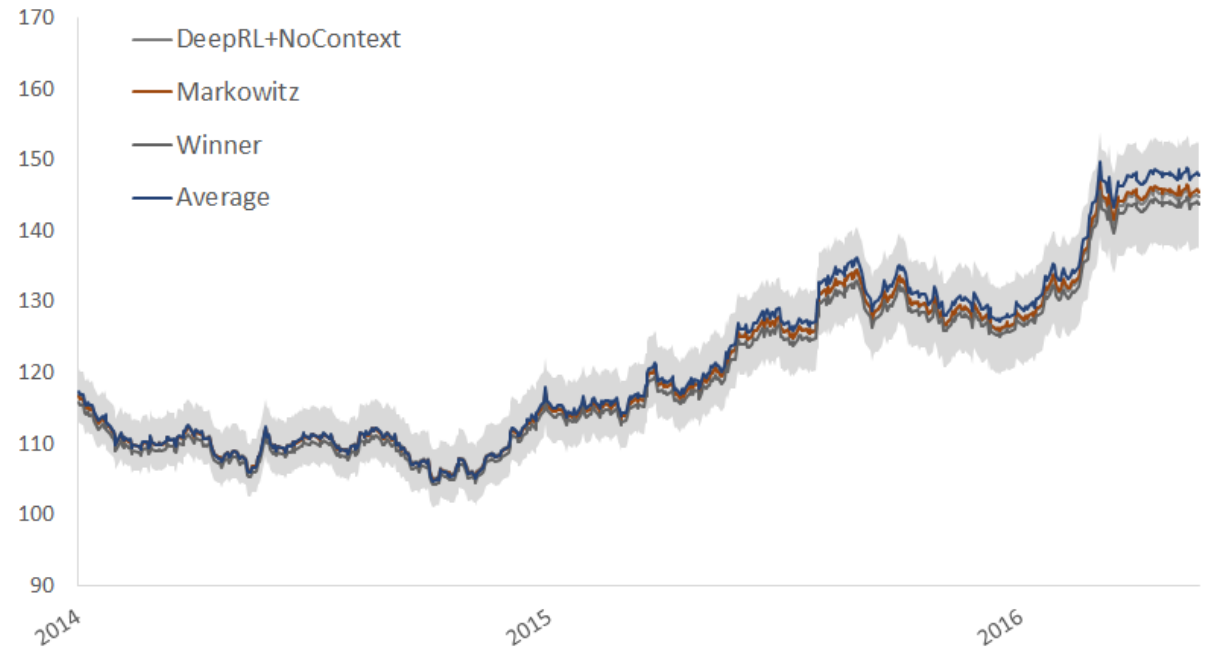


Figure 3.12: Zoom of the DRL model without contextual variables over the period of 2018 up to end of 2020.

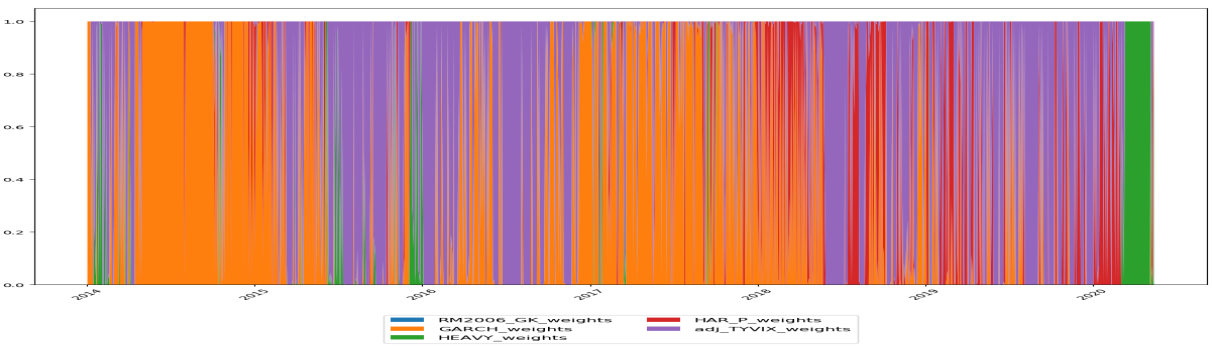


Figure 3.13: DRL portfolio allocation

Table 3.1: Models comparison over 1, 3, 5 years

	return	sharpe	sortino	mdd	mdd/vol
			1 Year		
DRL1	**22.659**	**2.169**	**2.419**	- 6.416	**- 0.614**
DRL2	20.712	2.014	2.167	- 6.584	- 0.640
Average	20.639	2.012	2.166	- 6.560	- 0.639
Markowitz	19.370	1.941	2.077	- 6.819	- 0.683
Winner	17.838	1.910	2.062	**- 6.334**	- 0.678
			3 Years		
DRL1	**8.056**	**0.835**	**0.899**	- 17.247	**- 1.787**
DRL2	7.308	0.783	0.834	- 16.912	- 1.812
Average	7.667	0.822	0.876	**- 16.882**	- 1.810
Markowitz	7.228	0.828	0.891	- 16.961	- 1.869
Winner	6.776	0.712	0.754	- 17.770	- 1.867
			5 Years		
DRL1	**6.302**	**0.651**	**0.684**	**- 19.794**	**- 2.044**
DRL2	5.220	0.565	0.584	- 20.211	- 2.187
Average	5.339	0.579	0.599	- 20.168	- 2.187
Markowitz	4.947	0.569	0.587	- 19.837	- 2.074
Winner	4.633	0.508	0.526	- 19.818	- 2.095

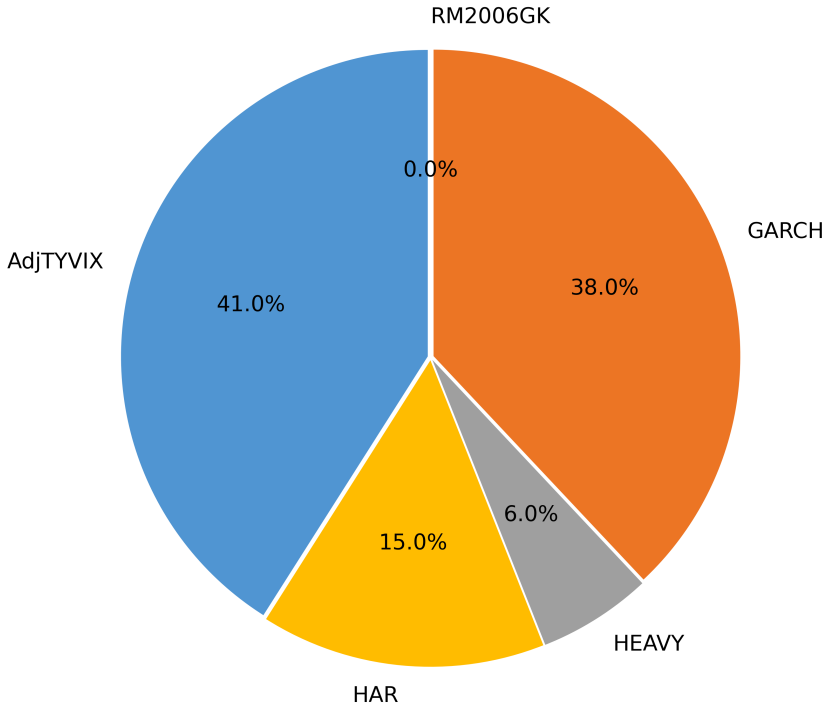


Figure 3.14: Average model allocation for the DRL model with context

they are not statistically different from each other. These findings confirm the superiority of the DRL1 model over other models, particularly the DRL2 and "winner" models, and highlight the robustness of our results.

Overall, the T-statistic test provides a useful validation method for assessing the statistical significance of our DRL model with contextual variables named DRL1 model.

Table 3.2: T stat and P-values (in parenthesis) for the statistical difference between returns.

Returns	DRL2	Average	Markowitz	Winner
DRL1	**2.7 (0.7%)**	0.2 (85%)	1.5 (14.2%)	**2 (4.9%)**
DRL2		0 (99.8%)	0.4 (71.5%)	0.7 (48.2%)
Average			0.1(95.4%)	0.1 (92.7%)
Markowitz				0.2 (84.1%)

If we conduct a T-statistic test on the returns running average, computed as $(\sum_{u=0}^{t} r_u/t)$ for various times t, the conclusions of the test results are markedly different, as shown in Table 3.3. For a p-value threshold of 5%, we conclude that all the models are statistically different, except for the comparisons between the DRL2 and Average models or the Average and "winner" models, where we fail to reject the null hypothesis that they are statistically different. These results on the running average are intuitive as it is not so easy to distinguish the curves in Figure 3.9.

These findings suggest that the choice of the performance metric can significantly affect the conclusions drawn from the statistical tests. In this case, using the returns running average instead of the raw returns leads to different conclusions about the performance differences between the models. Therefore, it is crucial to carefully select appropriate performance metrics that accurately reflect the investment objectives and strategies when conducting such tests.

The T-statistic tests on the returns running average provide robust evidence of the superior performance of the DRL1 model compared to other portfolio allocation strategies, including the DRL2 model, the Average strategy, the Markowitz portfolio, and the "winner" strategy. These findings can be highly valuable for investors and practitioners seeking to optimize their portfolio allocation strategies and achieve improved investment outcomes.

The DRL1 model's outperformance can be attributed to its ability to incorporate both asset and contextual information, as well as standard deviation, in its decision-making process. The sensitivity analysis results confirm the importance of these features in predicting regime changes and identifying profitable investment opportunities. The use of walk-forward analysis and T-statistic tests further enhances the model's robustness and reliability,

Table 3.3: T-statistics and P-values (in parenthesis) for running average returns difference

Avg Return	DRL2	Average	Markowitz	Winner
DRL1	**72.1 (0%)**	**14 (0%)**	**44.1 (0%)**	**79.8 (0%)**
DRL2		1.2 (22.3%)	**24.6 (0%)**	**10 (0%)**
Average			**7.6(0%)**	0.9 (38.7%)
Markowitz				**-13.1 (0%)**

Results discussion

It is noteworthy to investigate how the DRL model achieves superior performance compared to other portfolio allocation strategies, yielding an additional 1

In a volatility targeting framework, the capital weights of each model are inversely proportional to their volatility estimates. As a result, lower volatility estimates result in higher weights, yielding higher returns in bullish markets. Conversely, higher volatility estimates lead to lower capital weights, resulting in better performance in bearish markets. The allocation of these models evolves over time, as demonstrated in Figure 3.15, which plots the rank of the first five models.

These observations suggest that the DRL model's superior performance is due to its ability to dynamically adjust the portfolio allocation weights based on the changing market conditions, leading to better risk-adjusted returns. The use of contextual information, standard deviation, and sensitivity analysis further enhances the model's ability to detect regime changes and select profitable investment opportunities.

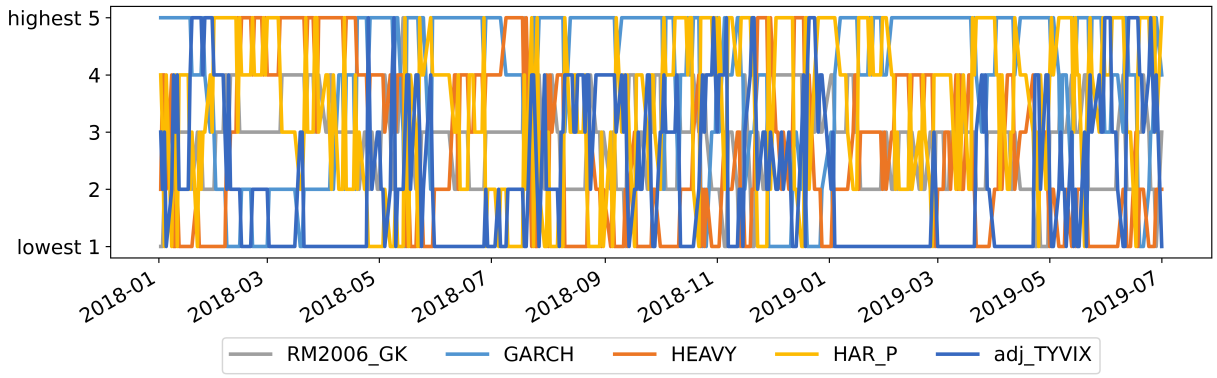


Figure 3.15: Volatility estimates rank

We can investigate whether the DRL model has a tendency to select volatility-targeting models that favour lower volatility estimates. To do so, we can plot the occurrence of rank by the dominant model for the DRL model. Figure 3.16 reveals that the DRL model selects the model with the lowest volatility estimate quite often, accounting for 38.2

These observations indicate two things: firstly, the DRL model tends to select either the lowest or highest volatility estimates models, known to perform well in bullish or bearish markets, respectively. Nevertheless, the model does not blindly select these models and can time when to choose the lowest or highest volatility estimates. Secondly, the DRL model can simultaneously decrease maximum drawdowns and increase net annual returns, as observed in Table 3.1. This ability to improve both performance metrics suggests that the model can detect regime changes. In contrast, a random selection of models would increase leverage when selecting the model with the lowest volatility estimate, leading to higher maximum drawdowns.

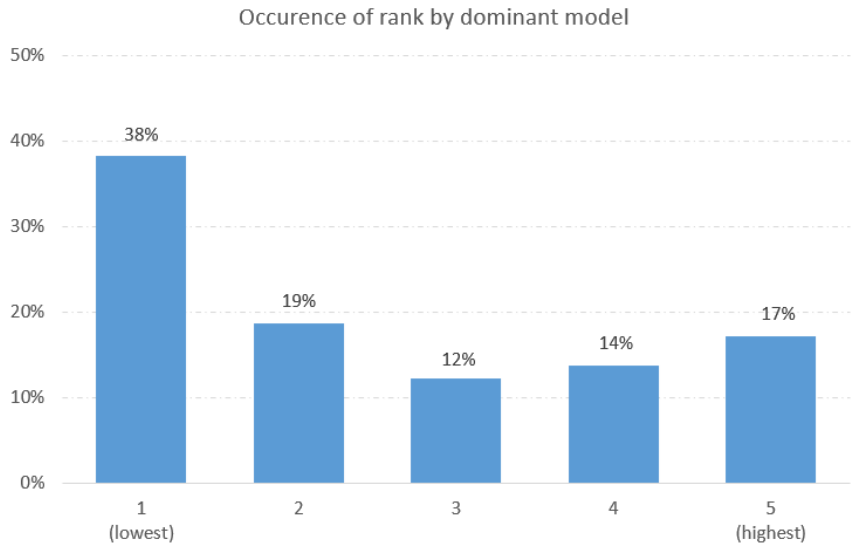


Figure 3.16: Occurrence of rank for the DRL model

3.3.2 Benefits of DRL and future works

The contextual DRL approach has several advantages over traditional optimization methods. Firstly, it can directly map market conditions to actions and thus adapt to regime changes in the market. Secondly, DRL can incorporate additional data and be a multi-input method, which is not possible with more traditional optimization methods. However, there is still room for improvement in this area. For instance, more contextual data could be incorporated into the model to improve its accuracy and predictive power. Additionally, different architectural network choices could be explored to find the best one for the problem at hand. Overall, the contextual DRL approach shows great potential for improving portfolio allocation strategies and investment outcomes, but further research and development are needed to fully realize its benefits.

3.4 Summary

In this chapter, a novel approach was proposed for selecting volatility-targeting models thanks to a DRL model whose task is to allocate among the different quantitative models. The model-free step aims to improve the quantitative model by adapting it to the current environment and selecting the best models. The addition of volatility and contextual variables enables an efficient allocation among the volatility-targeting models, leading to better performance and risk reduction compared to various benchmarks. The approach's robustness was tested through successive training and testing sets, and statistical tests were used to validate the results' significance.

Feature sensitivity analysis confirmed the importance of volatility and contextual variables, explaining the DRL agent's better performance. The chapter's research has been extensively elaborated in two papers, (Benhamou, Saltiel, Tabachnik, Wong & Chareyron 2021) and (Benhamou, Saltiel, Tabachnik, Bourdeix & Chareyron 2021), which delve into additional details beyond the scope of this thesis. Additionally, the second paper received the Best Paper Award at the MIDAS 2021 conference. This illustrates the interest from the research community in the approach of selecting quantitative models using DRL, which is quite new and different from the traditional usage of DRL in portfolio allocation.

In the upcoming chapters, we will delve into the theoretical foundations of the DRL approach. This will involve comparing it with more traditional methods and examining its mathematical rigour to provide justification for its use in the field.

PART II

Theoretical considerations for using DRL in portfolio allocation

Overview

> One of the most gratifying results of intellectual evolution is the continuous opening up of new and greater prospects.
>
> *Tesla (1915), a Serbian-American inventor, electrical and mechanical engineer known for his contributions to the design of alternating electricity*

In the second part of this thesis, we focus on understanding the reasons and benefits of using the DRL approach.

First, in chapter 4, Comparison of DRL and Traditional Financial Methods, we compare DRL for portfolio allocation with traditional financial methods. We explain that DRL adds a new layer to usual portfolio strategies by changing the original problem into a multi-step control problem, which expands the possibilities of portfolio optimization. This approach can include more financial factors when deciding how to allocate investments, making it an exciting and fresh idea. DRL also considers not just immediate rewards, but future ones too, creating a stronger model that can handle changes in the market. In the end, DRL tries to find a function instead of just simple portfolio percentages, making it a more complete and adaptable model.

Next, in chapter 5, Variance reduction in Actor critic methods, we look at actor-critic methods and show that these methods can be understood as solving a Monte Carlo simulation problem using optimal control variables. This helps explain why actor-critic methods naturally reduce variance.

Lastly, in chapter 6, Similarities between RL and Supervised Learning, we explore the analogies between reinforcement learning and supervised learning. We demonstrate that gradient descent reinforcement learning can be seen as supervised learning with a cross-entropy loss function and labels equal to optimal rewards, as long as actions do not affect future states. This theoretical finding shows a deep connection between the two learning methods, even though we can't know the optimal rewards beforehand.

In summary, this part of the thesis gives theoretical reasons and explanations for why the DRL approach works well, which can be helpful for professionals and researchers looking to improve portfolio allocation strategies and investment results.

4 Comparison of Deep Reinforcement Learning and Traditional Financial Methods

> Computer languages of the future will be more concerned with goals and less with procedures specified by the programmer
>
> *Minksy (1970), an American cognitive and computer scientist concerned largely with the research of artificial intelligence*

4.1 Introduction

In the first part of this thesis, various applications of DRL for portfolio allocation were presented. This chapter aims to explore the relationship between these new machine-learning techniques and more traditional financial methods used in asset management.

Traditionally, financial methods for portfolio optimization rely on risk optimization and treat the planning problem of finding the optimal portfolio as a single-step optimization question. On the other hand, DRL methods do not make any assumptions about risk and involve a more complex multi-step optimization process. Hence, it makes sense to compare the two approaches and see what they have in common and what their differences are. Indeed, both methods aim to solve the decision-making problem of finding the optimal portfolio allocation weights and understanding the relationships between these approaches can be valuable in developing better portfolio allocation strategies.

4.2 Traditional methods

4.2.1 Markowitz

In portfolio allocation, practitioners rely on the so-called financial risk-reward paradigm which is the core subject of the Markowitz portfolio model. In a nutshell, Markowitz's main idea is that by diversifying across a portfolio of assets with different expected returns and risks, investors can reduce the overall portfolio risk without sacrificing returns. This approach involved calculating the expected return and variance for each asset in the portfolio, as well as the correlations between them and finding the optimal portfolio according to the efficient frontier. Indeed, by analyzing the returns and risks of different financial assets, we can construct an efficient frontier, which represents the optimal portfolio that offers the highest expected return for a given level of risk or the lowest risk for a given expected return.

The efficient frontier, represented by the red dot line in figure 4.1, illustrates the various possible combinations of assets that an investor can hold in a portfolio. The goal is to find the optimal mix of assets that provides the desired level of return while minimizing risk. This is achieved by diversifying the portfolio across various assets to reduce the impact of individual asset risks.

Markowitz's portfolio theory highlights the importance of considering both return and risk when making investment decisions. By analyzing and comparing the potential returns and risks of various assets, investors can construct an optimal portfolio that balances the desired level of return with the level of risk they are willing to take on. The theory provides a useful framework for portfolio construction and management, which has been widely adopted in the financial industry.

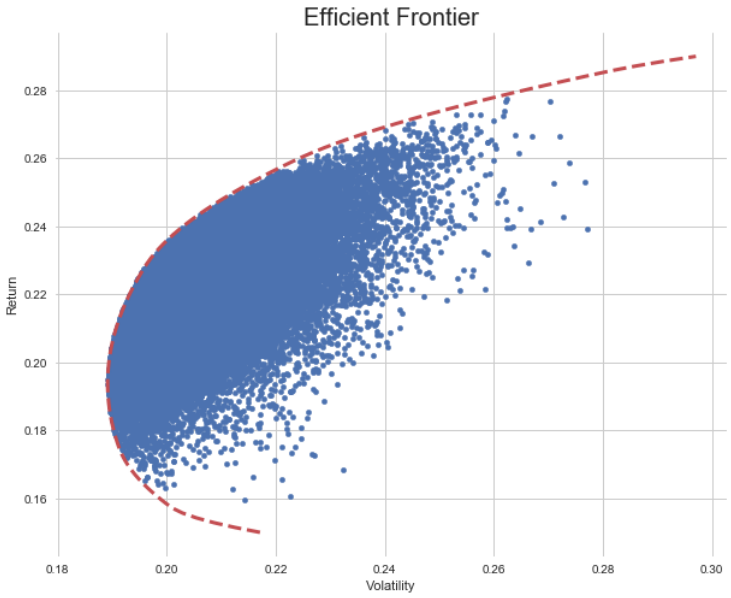


Figure 4.1: Markowitz efficient frontier for the GAFA: returns taken from 2017 to end of 2019

The allocation of a portfolio can be mathematically formulated as an optimization problem. Let us consider a portfolio consisting of n different assets, with each asset represented by its expected return μ_i and its variance σ_i^2. We define a weight w_i for each asset in the portfolio, where

$1 \geq w_i \geq 0$ and $\sum_{i=1}^{n} w_i = 1$. We write the vectors w respectively μ as the vector whose coordinates are $(w_i)_{i=1..n}$, respectively $(\mu_i)_{i=1..n}$.

The goal is to construct an optimal portfolio that minimizes the risk while achieving a target minimum expected return. The risk of a portfolio is measured by its variance, which is given by the quadratic form $w^T \Sigma w$, where Σ is the covariance matrix of the asset returns.

Thus, the Markowitz optimization problem can be formulated as follows:

$$\begin{aligned} &\underset{w}{\text{Minimize}} \quad w^T \Sigma w \\ &\text{subject to} \quad \mu^T w \geq r_{min}, \sum_{i=1}^{n} w_i = 1, 1 \geq w \geq 0, \end{aligned} \tag{4.1}$$

Here, the objective is to minimize the risk, subject to the constraint that the expected return of the portfolio should be greater than or equal to the target minimum expected return r_{min}, and the weights must sum to 1, with each weight w_i between 0 and 1. Standard quadratic programming techniques can solve this optimization problem. Thanks to duality, there is an equivalent maximisation with a given maximum risk σ_{max} for which the problem writes as follows:

$$\begin{aligned} &\underset{w}{\text{Maximize}} \quad \mu^T w \\ &\text{subject to} \quad w^T \Sigma w \leq \sigma_{max}, \sum_{i=1}^{n} w_i = 1, 1 \geq w \geq 0. \end{aligned} \tag{4.2}$$

Standard quadratic programming techniques can be used to solve the following optimization problem: maximize the expected return, with the condition that the portfolio risk, as measured by the variance of the portfolio, should not exceed a given maximum risk σ_{max}.

4.2.2 Minimum variance portfolio

The minimum variance portfolio is another well-known financial model that aims to construct a portfolio with the minimum possible risk, regardless of the expected returns. The idea behind this approach is to create an efficient portfolio that provides the lowest possible risk for a given level of expected returns or to achieve a target level of expected returns with the lowest possible risk.

Various extensions have been made to this model over the years. For instance, Bernstein & Sharpe (1990) and Chopra & Ziemba (1993) proposed an approach where the objective is to minimize risk while ignoring expected returns. This approach makes sense as cap-weighted stock portfolios have been shown to be inefficient as illustrated in (Haugen & Baker 1991)), and a risk-minimization approach could be more effective.

From a practical perspective, minimizing risk could be useful as well. As Kritzman (2014) suggests, minimizing risk can help investors avoid the negative impact of large losses in their portfolio, which can significantly affect their long-term investment goals.

The optimization problem for the minimum variance portfolio is as follows:

$$\begin{aligned} \underset{w}{\text{Minimize}} \quad & w^T \Sigma w \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i = 1, 1 \geq w \geq 0. \end{aligned} \tag{4.3}$$

It is important to highlight that the minimum variance portfolio optimization (optimization program 4.3), in essence, corresponds to Markowitz's approach (optimization program 4.1) but without the requirement of achieving a minimum return. Here, the objective is to minimize the portfolio risk, measured by the variance of the portfolio. The constraints ensure that the weights of the assets sum to 1 and each weight is between 0 and 1. Various optimization techniques, such as quadratic programming can solve this optimization problem.

Maximum diversification portfolio

The maximum diversification portfolio is another widely adopted approach in finance that aims to construct a portfolio that maximizes diversification. The intuition behind this approach is that diversification can lead to a more robust and stable portfolio that can better withstand market turbulence and mitigate risks.

To define diversification, we first consider the volatilities of the n different assets in the portfolio, which are the diagonal elements of the covariance matrix Σ. We denote this vector of volatilities as $\sigma = (\Sigma_{i,i})_{i=1\ldots n}$.

The diversification of a portfolio is then defined as the ratio of the weighted average of the asset volatilities to the overall portfolio volatility, given by the square root of the quadratic form $w^T \Sigma w$. Specifically, we define the diversification of a portfolio as follows:

$$D = \frac{w^T \sigma}{\sqrt{w^T \sum w}} \tag{4.4}$$

Here, w is the vector of weights assigned to each asset in the portfolio, and $\sum$ is the covariance matrix of the asset returns. The goal is to maximize the diversification of the portfolio by finding the weights that maximize the above expression.

Choueifaty & Coignard (2008) introduced the concept of a maximum diversification portfolio and argued that it is a more robust and stable portfolio that can better withstand market turbulence and mitigate risks. Choueifaty et al. (2012) further showed that the maximum diversification portfolio is robust out of sample.

The optimization problem for the maximum diversification portfolio can be formulated as follows:

$$\begin{aligned} \underset{w}{\text{Maximize}} \quad & \frac{w^T \sigma}{\sqrt{w^T \sum w}} \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i = 1, 1 \geq w \geq 0. \end{aligned} \tag{4.5}$$

Here, the objective is to maximize the diversification of the portfolio, subject to the constraint that the weights of the assets sum to 1 and each weight is between 0 and 1. This optimization problem can be solved using various optimization techniques, such as quadratic programming.

As a matter of fact, maximum diversification portfolio provides a robust and stable approach to constructing a portfolio that maximizes diversification. By optimizing the allocation of assets in a portfolio to maximize diversification, investors can achieve a more stable and resilient portfolio that can better withstand market turbulence and mitigate risks.

Maximum decorrelation portfolio

Another approach in finance to construct an optimal portfolio is the maximum decorrelation portfolio, which aims to maximize diversification by minimizing the correlation between the assets in the portfolio. The idea behind this approach is to identify assets that are not highly correlated with each other so that their collective behaviour is less prone to extreme fluctuations.

The maximum decorrelation portfolio is obtained by finding the weights that provide the maximum decorrelation or equivalently the minimum correlation between the portfolio assets. To do this, we first calculate the correlation matrix C of the portfolio strategies. The optimization problem for the maximum decorrelation portfolio can then be formulated as follows:

$$\begin{aligned} &\underset{w}{\text{Minimize}} \quad w^T C w \\ &\text{subject to} \quad \sum_{i=1}^{n} w_i = 1, 1 \geq w \geq 0. \end{aligned} \tag{4.6}$$

Here, the objective is to minimize the correlation between the assets in the portfolio, subject to the constraint that the weights of the assets sum to 1 and each weight is between 0 and 1. This optimization problem can also be solved using various optimization techniques, such as quadratic programming.

Christoffersen et al. (2010) argue that the maximum decorrelation portfolio provides a more robust and stable approach to constructing a diversified portfolio compared to traditional approaches, as it focuses on identifying assets that are not highly correlated with each other.

Intuitively, the maximum decorrelation portfolio provides a powerful approach to constructing a diversified portfolio that minimizes the correlation between assets. By optimizing the allocation of assets in a portfolio to minimize correlation, investors can achieve a more robust and stable portfolio that can better withstand market turbulence and mitigate risks.

Risk parity portfolio

Last but not least, another approach to constructing an optimal portfolio is to follow the principle of risk parity. The idea behind risk parity is to allocate the portfolio weights in a way that balances the risk contribution of each asset in the portfolio. This approach is based on the assumption that each asset in the portfolio contributes equally to the overall portfolio risk, regardless of its expected return.

Maillard et al. (2010) and Roncalli & Weisang (2016) have proposed the risk parity approach and argued that it can provide a more stable and robust portfolio that can better withstand market turbulence and mitigate risks.

The risk parity optimization problem can be formulated as follows:

$$\begin{aligned} \underset{w}{\text{Minimize}} \quad & \frac{1}{2} w^T \Sigma w - \frac{1}{n} \sum_{i=1}^{n} \ln(w_i) \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i = 1, 1 \geq w \geq 0. \end{aligned} \tag{4.7}$$

Here, the objective is to minimize the portfolio risk, measured by the variance of the portfolio, subject to the constraint that the risk contribution of each asset in the portfolio is equal. The risk contribution of each asset is calculated as the product of the asset weight and the corresponding element in the covariance matrix Σ. The constraint ensures that the weights of the assets sum to 1 and each weight is between 0 and 1.

The second term in the objective function, $\frac{1}{n} \sum_{i=1}^{l} \ln(w_i)$, is a logarithmic penalty that enforces equal risk contribution among the assets in the portfolio. This term penalizes any weight that deviates from the equal risk contribution rule.

The risk parity approach is often used in combination with the maximum diversification portfolio, as they both aim to provide a more balanced and stable portfolio. By optimizing the allocation of assets in a portfolio using risk parity, investors can achieve a more stable and robust portfolio that can better withstand market turbulence and mitigate risks.

Common characteristics among these methods

These traditional portfolio methods, such as Markowitz portfolio, minimum variance portfolio, maximum diversification portfolio, and risk parity, share in common various characteristics. They all aim to balance the trade-off between expected return and risk while ensuring diversification of the portfolio. There are all mathematically expressed as a convex optimization problem under some constraints. They are all only using the portfolio assets but nothing else hence can be quite problematic to detect regime changes. We will see in the coming section that this motivates for a more general optimisation problem which can be formulated as a reinforcement learning problem.

4.3 Reinforcement learning

Previous financial methods have treated the portfolio question as a one-step optimization problem with convex objective functions, which has several limitations. These methods do not relate market conditions to portfolio allocation dynamically, do not take into account the delayed evaluation of portfolio allocation, and make strong assumptions about risk.

To address these limitations, we can cast the portfolio allocation planning question as a dynamic control problem, where we have market information at each time step and need to decide the optimal portfolio allocation problem while evaluating the result with a delayed reward. This

approach would move from static portfolio allocation to optimal control territory, allowing us to change the portfolio allocation dynamically as market conditions change.

Despite the potential benefits of this approach, it has been largely ignored by the portfolio allocation community due to the differences between the two fields and the additional complexity introduced by DRL. However, there has been a growing interest in the use of reinforcement learning and deep reinforcement learning for portfolio optimization in recent years either using Q-learning Du et al. (2016), Pendharkar & Cusatis (2018), SARSA Pendharkar & Cusatis (2018), DQN Park et al. (2020), or policy-based algorithms such as DPG and DDPG Xiong et al. (2018), Yu et al. (2019), Liang et al. (2018), Aboussalah (2020), Cong et al. (2021). We refer the reader to the section 2.1.1 for more details.

Unlike supervised learning, which predicts future returns, reinforcement learning learns the optimal policy for portfolio allocation in connection with dynamically changing market conditions. It directly learns the optimal policy instead of implicitly learning the structure of the market, as in supervised learning. By leveraging reinforcement learning and deep reinforcement learning, we can develop more dynamic, adaptive, and flexible approaches to portfolio optimization that can adapt to changing market conditions and evaluate portfolios over longer time horizons.

Casting the portfolio allocation planning question as a dynamic control problem using reinforcement learning and deep reinforcement learning offers promising avenues for constructing more efficient and effective portfolios. By directly learning the optimal policy for portfolio allocation in connection with dynamically changing market conditions, we can develop more dynamic, adaptive, and flexible approaches to portfolio optimization that can adapt to changing market conditions and evaluate portfolios over longer time horizons.

However, this approach also requires careful validation and testing to ensure its effectiveness and avoid overfitting. Since reinforcement learning and deep reinforcement learning are relatively new to the field of portfolio allocation, it is crucial to thoroughly validate these methods to ensure that they produce effective and robust portfolios. Furthermore, overfitting can be a significant problem in deep reinforcement learning, as these models can be highly complex and difficult to interpret. Therefore, it is important to use proper validation techniques and carefully consider the architecture and hyperparameters of the model to avoid overfitting.

Overall, while reinforcement learning and deep reinforcement learning offer promising avenues for portfolio optimization, it is important to approach this area of research with care and caution to ensure that these methods are effective, robust, and reliable.

4.3.1 Deep Reinforcement Learning Intuition

Deep Reinforcement Learning (DRL) combines Reinforcement Learning (RL) and Deep Learning (D) to optimize portfolio allocation. In DRL, deep learning is used to represent the policy function in RL. The goal is to find an optimal portfolio allocation by taking actions on the market state, which is denoted by s_t. These actions denoted by a_t, represent the decision of the current portfolio allocation or weights. After the action is taken, the next state, s_{t+1}, is observed. The performance of the actions is evaluated using a reward, which can be computed only at the final time of the episode, denoted by T. The reward, denoted by R_T, is similar to the objective function used in traditional methods, such as final portfolio net performance or other financial performance evaluation criteria like Sharpe or Sortino ratios.

This approach allows for a more dynamic and adaptive approach to portfolio optimization that can adapt to changing market conditions and evaluate portfolios over longer time horizons. However, the use of deep learning can also make the models highly complex and difficult to interpret, and overfitting can be a significant problem. Therefore, it is important to carefully validate and test DRL models to ensure their effectiveness and reliability.

Following standard RL, we can model our problem using a Markov Decision Process (MDP), as introduced in chapter 1. The objective of the agent is to learn a policy π that maps the state s_t to the optimal action a_t^*, which maximizes the expected cumulative discounted rewards $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_t]$.

The use of deep neural networks in RL is to represent the function that maps the state to the action taken by the agent, i.e., the policy function denoted by $a_t = \pi(s_t)$. This function is represented by a deep neural network because of the universal approximation theorem, which states that any function can be approximated by a deep neural network provided that the network has enough layers and nodes. Compared to traditional methods that only solve for a one-step optimization, we are solving for the following dynamic control optimization problem:

$$\begin{aligned} \underset{\pi(.)}{\text{Maximise}} \quad & \mathbb{E}[R_T] \\ \text{subject to} \quad & a_t = \pi(s_t). \end{aligned} \tag{4.8}$$

4.3.2 How does DRL compare with traditional methods?

It is noteworthy that traditional financial methods and Deep Reinforcement Learning (DRL) share several similarities. Firstly, they are both framed as optimization problems where the objective is to identify the optimal allocation based on a criterion function. In conventional financial methods, this criterion involves a tradeoff between risk and reward, which is achieved through fixed weights that are independent of time. The optimal weights are dependent on our assumptions regarding the mean of the asset (μ), their variance (Σ), and the correlation between assets (C). Therefore, if these asset characteristics remain constant over time, the optimal weights will also remain the same. However, if these characteristics vary over time, the weights must change accordingly. Unfortunately, obtaining the accurate statistical properties of assets, such as their mean and variance, is practically impossible, as it is subject to a noisy estimation that fails to capture the true dynamics of assets.

In contrast, DRL aims to solve a multi-time step problem where the model is required to provide optimal weights at each time based on the state. It is evident that DRL and traditional methods differ significantly from several angles. While traditional financial methods focus on a single time-step optimization, DRL involves taking various actions from $t = 1$ to $t = T$ depending on the state. If we consider the DRL problem as a single time-step problem, where the states represent the mean (μ), variance (Σ), and correlation of assets (C), and use the opposite of risk ($-w^T \Sigma w$) as rewards and enforce that the actions should take a value between 0 and 1, sum to one and potentially validate a constraint $\mu^T a \geq r_{min}$, we obtain a DRL problem whose optimal solution is the Markowitz portfolio. If we only impose a constraint that the action coordinates should take value between 0 and 1 and sum to one, we get the minimum variance portfolio. Hence, it is clear that traditional portfolio methods are merely a simple one-time-step DRL problem where the states are statistical properties of the portfolio assets and the reward is a convex function whose solution can be obtained through closed form.

Therefore, the primary distinction between traditional portfolio methods and DRL is that DRL is a generalization of traditional portfolio methods that computes optimal weights at each time step, enabling them to adapt to changing environments. Moreover, if we take very simple states that are precisely chosen to be the statistical properties at stake when computing traditional portfolio and imposing appropriate constraints, we can create a DRL problem whose solution is at each time step the rolling version of either Markowitz portfolio or the minimum variance portfolio or any of the previous financial methods presented above. In order to be more rigorous, let us reformulate this with more mathematical formalism and prove the following statement:

Proposition 4.1. *For a given investment universe, the traditional Markowitz portfolio optimization problem can be formulated as a one-step Deep Reinforcement Learning (DRL) problem.*

Proof. Let n be the number of assets in the investment universe, and let $\mathbf{w} = [w_1, w_2, \ldots, w_n]^T$ be the vector of weights assigned to each asset. Then, the Markowitz portfolio optimization problem can be formulated as follows:

$$\begin{aligned} &\underset{w}{\text{Minimize}} \quad w^T \Sigma w \\ &\text{subject to} \quad \mu^T w \geq r_{min}, \sum_{i=1}^{n} w_i = 1, 1 \geq w \geq 0. \end{aligned} \tag{4.9}$$

where Σ is the covariance matrix of the asset returns, $\boldsymbol{\mu}$ is the vector of expected returns. The above problem is a standard convex optimization problem whose closed-form solution is a function of μ and Σ. To reformulate this problem as a DRL problem, we can consider a one-time step problem where an agent needs to select a weight vector $\mathbf{w}$ that determines the optimal allocation. These actions or weights need in addition to satisfy some constraints:

$$\mu^T w \geq r_{min}, \sum_{i=1}^{n} w_i = 1, 1 \geq w \geq 0.$$

The state s is defined as the vector of the mean of the asset μ and their variance

$$s = [\boldsymbol{\mu}, \Sigma].$$

The reward R is defined as the negative of the portfolio variance $R = -\mathbf{w}_t^T \Sigma_t \mathbf{w}_t$. It is a quadratic function of the action w. Hence the one-time step DRL problem, which consists in maximizing the reward, is given by

$$\underset{w}{\text{Maximize}} \quad -w^T \Sigma w \tag{4.10}$$

$$\text{subject to} \quad \mu^T w \geq r_{min}, \sum_{i=1}^{n} w_i = 1, 1 \geq w \geq 0 \tag{4.11}$$

$$\text{with} \quad w = \pi(s). \tag{4.12}$$

The last line states that the action w is a function of the state or more explicitly of μ and Σ as these two variables are the underneath variable for our state. By construction, the two problems are the same as a maximization and a minimization on the opposite function are equivalent and as the final constraint in the DRL problem that states that the actions are a function of μ and Σ is trivially verified in Markowitz. Therefore, we have shown that the Markowitz portfolio optimization problem can be reformulated as a one-time step DRL problem. □

Corollary 4.2. *For a given investment universe, the traditional minimum variance portfolio optimization problem can be formulated as a one-time step Deep Reinforcement Learning (DRL) problem.*

Proof. The previous proof remains unchanged, except for the modification of the constraint on the action in a new one-time step Deep Reinforcement Learning (DRL) problem.Recall that the optimization of the minimum variance portfolio is essentially identical to Markowitz's portfolio optimization, with the key distinction being the absence of a constraint on the minimum desired return. The optimization problem, in this case, can be expressed as follows:

$$\underset{w}{\text{Maximize}} \quad -w^T \Sigma w \tag{4.13}$$

$$\text{subject to} \quad \sum_{i=1}^{n} w_i = 1, 1 \geq w \geq 0 \tag{4.14}$$

$$\text{with} \quad w = \pi(s). \tag{4.15}$$

□

We can continue this process for each of the previous traditional portfolio allocation methods, resulting in various corollaries, as follows:

Corollary 4.3. *For a given investment universe, the traditional maximum diversification portfolio respectively maximum decorrelation portfolio, respectively risk parity portfolio can be formulated as an equivalent one-time step Deep Reinforcement Learning (DRL) problem.*

Proof. The previous proof remains unchanged, except that for the case of the traditional maximum diversification portfolio, the reward is modified into :

$$R = \frac{w^T \sigma}{\sqrt{w^T \sum w}}.$$

In the case of the traditional maximum decorrelation portfolio, the reward is also modified into

$$R = -w^T C w.$$

Last but not least, in the case of the risk parity portfolio, the reward is given by

$$\frac{1}{2} w^T \Sigma w - \frac{1}{n} \sum_{i=1}^{n} \ln(w_i).$$

□

We can take things a step further and frame the rolling Markowitz method as a Deep Reinforcement Learning (DRL) problem. In this approach, we calculate the empirical mean and variance of the asset μ_t over a historical period (e.g. the last two years) at each time step t. Unlike the previous approach, this reformulation casts the rolling Markowitz problem as a multi-time step problem rather than a single-time step problem. To formalize this mathematically, we introduce the following definitions

Definition 4.4. *The rolling Markowitz problem is a portfolio optimization method where, at each time step t, the mean μ_t and variance Σ_t of an asset are computed over a historical period by looking backwards in time (e.g. over the last two years). The rolling Markowitz problem involves solving multiple optimization problems:*

$$\begin{aligned} &\underset{w_t}{\textit{Minimize}} && w_t^T \Sigma_t w_t \\ &\textit{subject to} && \mu_t^T w_t \geq r_{min}, \sum_{i=1}^{n} w_{t,i} = 1, 1 \geq w_{t,i} \geq 0. \end{aligned} \tag{4.16}$$

Here, w_t is the weight vector of the portfolio at time step t, r_{min} is a minimum return threshold, and n is the number of assets in the portfolio. The optimization problem seeks to minimize the portfolio variance subject to the constraint that the portfolio return is at least r_{min} and the weights sum up to 1.

We can further extend our initial proposition that states that the Markowitz portfolio can be cast as a DRL problem as follows:

Proposition 4.5. *For a given investment universe, the rolling Markowitz portfolio optimization problem can be formulated as a multi-time step Deep Reinforcement Learning (DRL) problem. In this specific problem, the reinforcement learning discount factor γ has to be set to zero.*

Proof. Let n be the number of assets in the investment universe, and let $\mathbf{w}_t = [w_{t,1}, w_{t,2}, \ldots, w_{t,n}]^T$ be the vector of weights assigned to each asset at time step t. Then, the Markowitz portfolio optimization problem can be formulated as a time-dependent optimization problem as follows:

$$\begin{aligned} &\underset{w_t}{\text{Minimize}} && w_t^T \Sigma_t w_t \\ &\text{subject to} && \mu_t^T w_t \geq r_{min}, \sum_{i=1}^{n} w_{t,i} = 1, 1 \geq w_{t,i} \geq 0. \end{aligned} \tag{4.17}$$

where Σ_t is the covariance matrix of the asset returns at time step t, $\boldsymbol{\mu_t}$ is the vector of expected returns at time step t, and r_{min} is a minimum return threshold. The solution to the above problem is a function of μ_t and Σ_t and is known as the optimal weight vector $\mathbf{w}_t^*$.

To reformulate this problem as a multi-time step DRL problem, we can consider a sequence of time steps $t = 1, 2, \ldots, \tau$, where at each time step t, the agent needs to select a weight vector $\mathbf{w}_t$ that determines the optimal allocation for that time step. τ represents the final time horizon in this problem. These actions or weight vectors need to satisfy the same constraints as in the

time-dependent optimization problem. The state s_t is defined as the vector of the mean of the asset returns $\boldsymbol{\mu}_t$ and their covariance matrix Σ_t at time step t, i.e.,

$$s_t = [\boldsymbol{\mu}_t, \Sigma_t].$$

The reward R_t at time step t is defined as the negative of the portfolio variance at that time step, i.e.,

$$R_t = -\mathbf{w}_t^T \Sigma_t \mathbf{w}_t. \tag{4.18}$$

As we are now in a multi-time step DRL problem, to compute the sum of the future rewards, we introduce the concept of discounted rewards. Hence, we denote by γ a discount factor. The multi-time step DRL problem is given by:

$$\begin{aligned}
\underset{\mathbf{w}}{\text{Maximize}} \quad & \sum_{k=0}^{\tau-t} \gamma^k R_{t+k} && (4.19)\\
\text{subject to} \quad & \mu_t^T w_t \geq r_{min}, \sum_{i=1}^{n} w_{t,i} = 1, 1 \geq w_{t,i} \geq 0 && (4.20)\\
\text{with} \quad & \mathbf{w}_t = \pi(s_t), && (4.21)
\end{aligned}$$

where $\pi(s_t)$ is the policy at time step t that maps the state s_t to an action or weight vector $\mathbf{w}_t$. If we replace the reward R_t by its expression (equation 4.18) and use as well the setting of the states, we get

$$\begin{aligned}
\underset{\mathbf{w}}{\text{Maximize}} \quad & \sum_{k=0}^{\tau-t} -\gamma^k \mathbf{w}_{t+k}^T \Sigma_{t+k} \mathbf{w}_{t+k} && (4.22)\\
\text{subject to} \quad & \mu_t^T w_t \geq r_{min}, \sum_{i=1}^{n} w_{t,i} = 1, 1 \geq w_{t,i} \geq 0 && (4.23)\\
\text{with} \quad & \mathbf{w}_t = \pi(\mu_t, \Sigma_t). && (4.24)
\end{aligned}$$

At this stage, we can notice that the DRL formulation equation 4.19 and the rolling Markowitz formulation equation 4.17 are quite different as the DRL version contains additional terms $\sum_{k=1}^{T} \gamma^k R_{t+k}$ which are future rewards and not only the immediate reward. But if we further impose that the discount factor γ is zero, we recover in the DRL problem the rolling Markowitz optimization, which concludes the proof. □

Remark 4.6. It is important to highlight that in order to recover a similar optimization problem as the rolling Markowitz problem using reinforcement learning, we have been forced **to set the discount factor to zero**. However, setting the discount factor to zero is a strong requirement and needs to be understood. Recall that in Deep Reinforcement Learning (DRL), the discount factor is a parameter that determines the relative importance of immediate rewards versus future

rewards. When the discount factor is set to zero, the agent becomes myopic and only considers the immediate reward at each time step. This means that the agent does not take into account any future rewards or consequences of its actions, and can lead to suboptimal policies.

This implies that the application of Deep Reinforcement Learning (DRL) methods, specifically when the discount factor is non-zero, will diverge significantly from traditional methods. In this context, the DRL agent will be incentivized to strike a balance between short-term and long-term rewards, in contrast to traditional methods that primarily focus on maximizing immediate gains. By choosing a discount factor $0 < \gamma < 1$, the agent would be encouraged to optimize a balance between immediate and future rewards, potentially leading to better investment decisions in the long run. In particular, for a non-zero discount factor, the reward would become:

$$R_t = \mathbb{E}\left[\sum_{k=0}^{\tau-t} \gamma^k R_{t+k}\right]. \tag{4.25}$$

In this case, the agent would consider both short-term gains and long-term consequences, which can lead to a more sophisticated and effective portfolio optimization strategy. Furthermore, the discount factor can be tuned to emphasize the desired balance between short-term and long-term rewards, allowing for a more flexible and adaptable investment strategy.

In conclusion, while setting the discount factor to zero in a DRL framework recovers a problem similar to the rolling Markowitz problem, this result highlights the key difference between a general DRL approach and traditional portfolio methods that results in a myopic agent that is focused solely on immediate rewards. By using a non-zero discount factor, the agent can balance both short-term and long-term rewards, potentially leading to better and more flexible investment strategies. In financial terms, this observation demonstrates that the DRL approach exhibits a more forward-looking perspective, addressing one of the well-known limitations of traditional portfolio methods. By incorporating a non-zero discount factor, DRL considers the long-term implications and potential future gains, thereby overcoming the myopic nature often associated with conventional portfolio strategies.

Remark 4.7. Note that the above proposal is not limited to the rolling Markowitz's method but can be extended to any other traditional rolling portfolio allocation methods, where the optimization problem is defined over a rolling window of historical returns and covariances.

4.3.3 Optimizing variables versus a function

A second major difference between Deep Reinforcement Learning (DRL) and traditional methods lies in the objective they seek to optimize. While traditional financial methods primarily focus on optimizing simple weights, DRL aims to optimize a policy function denoted by π. The previous proofs have demonstrated that when actions are a function of the first two statistical moments, such as mean (μ_t) and covariance (Σ_t), the distinction between optimizing variables and functions becomes inconsequential. However, this distinction becomes crucial when dealing with more complex states. Also, although practically, the policy function is ultimately represented by a deep neural network with weights in some space $\mathbb{R}^d$ where d is the number of parameters of the deep neural network, the optimization problem in DRL is conceptually very different from traditional methods, as we are optimizing in the space of policy functions $\pi : \mathcal{S} \to \mathcal{A}$, that map states to

action. This space of function is a much larger space than simply the n-simplex where weights live. More rigorously, we define the n-simplex as follows:

Definition 4.8. *An n-simplex, denoted by Δ^n, is a geometric object defined in an n-dimensional space. It can be mathematically defined as follows:*

$$\Delta^n = \left\{ x \in \mathbb{R}^n : x = \sum_{i=0}^{n} \lambda_i \vec{v}_i, \sum_{i=0}^{n} \lambda_i = 1, \lambda_i \geq 0 \right\}, \tag{4.26}$$

where x is a point in the n-dimensional space $\mathbb{R}^n$, the vectors $(\vec{v}_i)_{i=1,\dots,n}$ form the canonical orthonormal basis of the n-dimensional space $\mathbb{R}^n$, and λ_i are the convex combination coefficients satisfying $\lambda_i \geq 0$ and $\sum_{i=0}^{n} \lambda_i = 1$.

Let's consider a simple feedforward neural network with L layers, activation functions $\sigma_l(\cdot)$, and weights w_{ij}^l, which represents the policy function $\pi : \mathcal{S} \to \mathcal{A}$. The input space is $\mathcal{S}$ and the output space is $\mathcal{A}$. The total number of weights in the neural network is p. The output of the network is given by:

$$\pi(s) = \sigma_L \left(W^L \sigma_{L-1} \left(W^{L-1} \cdots \sigma_1 \left(W^1 s \right) \cdots \right) \right). \tag{4.27}$$

Here, W^l represents the weight matrix for the layer l, and the activation functions $\sigma_l(\cdot)$ are applied element-wise.

Now, consider the set of all possible neural networks with the same architecture but different weights, denoted by $\mathcal{N}$:

$$\mathcal{N} = \{\pi_w : w \in \Delta^n\}. \tag{4.28}$$

Since each weight can take any value in the simplex Δ^n, the total number of possible weight configurations is uncountably infinite, which shows that the space of all functions $\pi : \mathcal{S} \to \mathcal{A}$ is much larger than the simplex. Let us make a proposition and add a proof to it. In order to simplify our presentation, we will assume that weights do not live in the n-simplex but in the set of real numbers $\mathbb{R}^n$. We have the following proposition that emphasises a difference between the DRL and the traditional financial approach.

Proposition 4.9. *The space of functions $\pi : \mathcal{S} \to \mathcal{A}$ is larger than the set of real numbers $\mathbb{R}^n$.*

Proof. To show that the space of functions $\pi : \mathcal{S} \to \mathcal{A}$ is larger than the set of real numbers $\mathbb{R}^n$, we will use the traditional Cantor's diagonalization argument.

Suppose there is a one-to-one correspondence between the functions $\pi : \mathcal{S} \to \mathcal{A}$ and the elements of $\mathbb{R}^n$. Then, for each element $w \in \mathbb{R}^n$, we could associate a unique function $\pi_w : \mathcal{S} \to \mathcal{A}$.

However, we can construct a new function $\pi^* : \mathcal{S} \to \mathcal{A}$ that is not in the list of functions corresponding to the elements of $\mathbb{R}^n$ by changing the output of each π_w for a particular state $s \in \mathcal{S}$. Specifically, let $\pi^*(s) = \pi_w(s) + 1$ for all $s \in \mathcal{S}$. Now, π^* is different from every π_w in the list of functions corresponding to the elements of $\mathbb{R}^n$, because it differs from each π_w by 1 for the state s.

This contradicts our assumption that there is a one-to-one correspondence between the functions $\pi : \mathcal{S} \to \mathcal{A}$ and the elements of $\mathbb{R}^n$. Thus, the space of functions $\pi : \mathcal{S} \to \mathcal{A}$ is much larger than the set of real numbers $\mathbb{R}^n$. □

In summary, the optimization problem in Deep Reinforcement Learning (DRL) is conceptually different from traditional financial methods because it involves optimizing a policy function π in the space of functions $\pi : \mathcal{S} \to \mathcal{A}$, which is much larger than the set of real numbers $\mathbb{R}^n$. This difference in the size of the search spaces has important implications for the complexity of the optimization problem and the expressive power of the learned policies.

4.3.4 Key differences between DRL and traditional financal methods

A compelling approach to understanding the potential of Deep Reinforcement Learning (DRL) in the realm of portfolio allocation is to explore its similarities and differences with traditional financial methods. Such an analysis can provide insights into when DRL might outperform traditional methods or, conversely, fall short of expectations. By examining the ways in which both approaches utilize past data, identify optimal allocations based on a criterion function, and formulate optimization problems, we can gain an intuition for the relative strengths and limitations of each method. Additionally, considering how DRL's ability to learn from experience and adapt to changing environments may enable it to develop more complex and adaptive strategies compared to traditional methods, while traditional methods may be constrained by a state that only accounts for mean and variance of assets, highlights the potential for DRL to outperform traditional methods in dynamic and uncertain market conditions. Let us hence summarize the key findings of this chapter.

Deep Reinforcement Learning (DRL) and traditional financial methods share a commonality in using past data to make assumptions. However, traditional methods rely on the estimation of asset characteristics such as mean and variance, while DRL can have much richer states beyond simple statistical assumptions. This allows DRL to learn from experience and adapt to changing environments. However, the large usage of data and large feature space in DRL could also lead to potential limitations. One issue is the possibility of spurious correlations between the much richer state and the action, which can lead to ineffective strategies and wrong connections between features and allocations. This behavior can be particularly challenging when DRL overconcentrates on a few assets since traditional methods' diversification principle could mitigate this.

Another similarity is that both DRL and traditional methods aim to identify optimal allocations based on a criterion function. However, DRL surpasses traditional methods when the discount factor is not set to zero. DRL's forward-looking nature allows it to balance short-term and long-term rewards, while traditional methods often focus solely on immediate rewards. However, this introduces a new challenge as setting an optimal discount factor can be challenging, and poorly designed criterion functions can lead to DRL generating ineffective strategies that focus too much on the future and provide poor short-term results.

Both DRL and traditional methods can be formulated as optimization problems. However, traditional methods typically perform a one-time-step optimization, while DRL optimizes a policy function in the space of functions. This allows DRL to handle dynamic and uncertain market conditions, whereas traditional methods may struggle to adapt as they only offer fixed weights.

However, DRL's optimization in the space of functions can also lead to overfitting, increased computational resources, and implementation challenges.

In summary, DRL and traditional financial methods share similarities but DRL goes beyond traditional methods by learning from experience and adapting to changing environments, being forward-looking, and optimizing in the space of functions. While DRL may perform better in the long term, traditional methods may be more focused on short-term gains. These findings have been summarized in Table 4.1.

Similarities	Differences	Implications
Both uses past data to make their assumptions. Traditional financial methods require estimation of asset characteristics such as mean and variance	DRL learns from experience and adapts to changing environments, thanks to potentially much richer states than simply statistical assumptions on the asset of the portfolio	DRL can learn more complex and adaptive strategies, while traditional methods may be limited by a state that is relying only mean and variance of assets
Both aim to identify optimal allocations based on a criterion function	DRL is able to encompass traditional financial methods but goes much beyond when the discount factor is not set to zero, being forward-looking and able to balance short-term and long-term rewards, while traditional methods are often myopic and focus solely on immediate rewards	DRL may lead to better long-term performance, while traditional methods may be more focused on short-term gains
Both can be formulated as optimization problems. In traditional methods, this optimization is traditionally a one-time-step optimization	DRL optimizes a policy function in the space of functions, while traditional methods optimize fixed weights	DRL may be better equipped to handle dynamic and uncertain market conditions, while traditional methods may struggle to adapt as they only offer fixed weights

Table 4.1: Similarities and differences between Deep Reinforcement Learning (DRL) and traditional portfolio methods.

4.4 Experiments

4.4.1 Objective of the Experiment

The main objective of this experiment is to investigate and validate the differences between Deep Reinforcement Learning (DRL) and traditional financial methods. Specifically, we aim to compare the performance of DRL in portfolio allocation with a focus on long-only allocations among 11 different assets. These assets consist of rolling futures contracts from various categories, including equity indexes, bond indexes, and commodities indexes.

The 11 assets considered in this experiment are as follows:

- Four major equity indexes: S&P 500, Eurostoxx 50, Nikkei 225, FTSE 100.
- Four major bond indexes: US 10-year TNote, European Bund, UK 10-year Gilt, Japanese Government Bond 10-year.
- Three major commodities indexes: Brent Oil, Gold, and Copper.

In the context of this experiment, the daily returns of these 11 assets are denoted by $(r_t^i)_{i=1..11}$, following the notation introduced in Chapter 2. The initial investment at time $t = 0$ is $I_0 = 100$. The objective is to determine the optimal allocation weights for the 11 strategies, represented by a vector $W_t = (w_t^1, w_t^2, ..., w_t^{11})^T$. These weights should be non-negative and sum up to 1, satisfying the constraint $\sum_{i=1}^{11} w_t^i = 1$. The model will be asked to provide the allocation weights on a daily basis, and these weights will be implemented at the end of each trading day. The model's performance will be evaluated based on the returns obtained the following day, taking into account transaction costs as described in Chapter 2. It is important to note that there is a one-day lag between taking the action (selecting the weights) and implementing it.

The resulting investment value at each day, denoted by I_t, will be calculated using Equation 4.29 as follows:

$$I_{t+1} = I_t \times \left(1 + \sum_{i=1}^{11} w_t^i r_{t+1}^i - b|w_t^i - w_{t+1}^i|\right), \tag{4.29}$$

where the parameter b represents the transaction cost associated with each rolling future and is assumed to be 0.0002, which can be equivalently stated as two basis points (bps). The transaction model employed is a linear model, as defined in 1.4. This choice is justified by the same argument presented in Chapter 2, where it is explained that linear models are more suitable and realistic when dealing with portfolios of considerable size. Here, the one-day lag between action and implementation is reflected in the time offset between the weights w_t^i and the returns r_{t+1}^i.

The ultimate goal of this experiment is to identify a model that achieves the highest Sharpe ratio out of sample, indicating superior risk-adjusted performance in portfolio allocation.

4.4.2 Reward

In this experiment, the reward is naturally chosen to be the Sharpe ratio as presented in definition 1.3. The Sharpe ratio is a widely used measure of risk-adjusted performance in finance. It quantifies the excess return of an investment per unit of risk taken. The Sharpe ratio is computed from the trajectory on the training dataset for the investment index. By maximizing the Sharpe ratio, we aim to find a model that achieves the highest risk-adjusted return on investment.

4.4.3 States

Similar to the approach described in Chapter 2, we create two types of inputs for our model: regular observations features and contextual observations features.

The regular observations are directly related to the 11 assets in the portfolio. In addition to the setup discussed in Chapter 2, we include the rolling maximum drawdowns as part of the regular observations. The rolling maximum drawdowns provide information about the historical maximum loss experienced by the portfolio. The rolling returns are computed in the same manner as described in Section 2.2.1.

These regular observations capture crucial information about the performance and risk characteristics of the assets, allowing the model to make informed decisions regarding portfolio allocation.

Furthermore, we incorporate contextual observations to provide additional context for the model. These contextual observations may include factors such as macroeconomic indicators, market sentiment, or any other relevant information that can potentially impact the asset returns. By including contextual observations, the model can consider the broader market conditions and adapt its decision-making accordingly.

Mathematically, we can specify more the setting as follows. We first define the regular observations features as follows:

$$\text{Rolling returns}_t = \begin{pmatrix} r^1_{t-i_j} & \dots & r^1_t \\ \dots & \dots & \dots \\ r^m_{t-i_j} & \dots. & r^m_t \end{pmatrix}, \tag{4.30}$$

likewise, rolling volatilities are given by:

$$\text{Rolling volatility}_t = \begin{pmatrix} \sigma^1_{t-i_j} & \dots & \sigma^1_t \\ \dots & \dots & \dots \\ \sigma^m_{t-i_j} & \dots. & \sigma^m_t \end{pmatrix}, \tag{4.31}$$

and rolling maximum drawdowns as follows:

$$\text{Rolling maxdrowdawns}_t = \begin{pmatrix} \text{mdd}^1_{t-i_j} & \dots & \text{mdd}^1_t \\ \dots & \dots & \dots \\ \text{mdd}^m_{t-i_j} & \dots. & \text{mdd}^m_t \end{pmatrix}, \tag{4.32}$$

where the maximum drawdown *mdd* is easily computed by first computing the cumulative maximum cum_max = df.cummax(), then computing the drawdown as $drawdown = \frac{df - \text{cum\_max}}{\text{cum\_max}}$ and then computing the maximum drawdown max_drawdown = drawdown.min(). The regular observation lags and contextual lags are given as follows:

$$\left(i_j,\, i_{j-1},\, \dots, 1, 0\right) = (60, 20, 4, 3, 2, 1, 0) \tag{4.33}$$

$$(i_k,\, i_{k-1},\, \dots, 1, 0) = (60, 20, 4, 3, 2, 1, 0)\,. \tag{4.34}$$

The list of all features used in the experiment is provided in tables 4.3 and 4.4. The regular observations or features used in the experiment, as shown in Figure 4.2, are directly related to the 11 assets in the portfolio while the contextual observations, depicted in Figure 4.3, consist of various economic and market indicators. These indicators include interest rates (such as US and European rates), market sentiment (Cboe Volatility Index), credit default swaps (CDX HY index and MARKIT Credit Europe Itraxx Crossover), currency rates (Dollar index), economic forecasts (GDP and CPI), commodity market data (crude oil, gold, and copper inventories), and economic surprise indices (global, USD, EUR, JPY, and EM).

By incorporating both regular and contextual observations, we aim to provide the model with a comprehensive set of information that captures the performance and risk characteristics of the assets, as well as the broader market and economic conditions. This enables the model to make informed decisions regarding portfolio allocation, taking into account both asset-specific factors and external market influences.

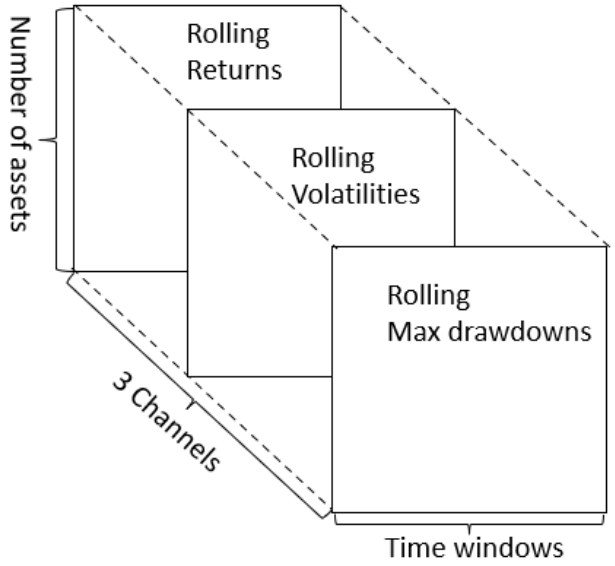


Figure 4.2: Regular observations

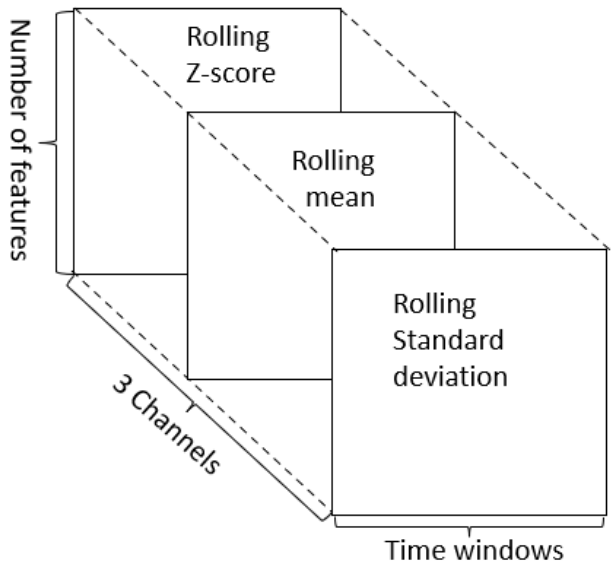


Figure 4.3: Contextual observations

4.4.4 Deep Network

For this experiment, we utilize the same algorithm as described in Chapter 2, specifically Algorithm 1. The hyperparameters used in the experiment are provided in Table 4.5, and they are similar to the ones used in the previous experiment in Chapter 2.

The details of the network architecture used in this experiment are illustrated in Figure 4.4. The network consists of several layers, including fully connected layers and Convolutional layers. The Convolutional layers allow the network to capture patterns when making portfolio allocation decisions. As already mentioned several time in this thesis, we use a two inputs networks which is something to our knowledge original and never used in previous works using DRL for portfolio allocations like Xiong et al. (2018), Jiang et al. (2017), Yu et al. (2019), Liang et al. (2018), Aboussalah (2020), Cong et al. (2021)

Figure 4.7 provides a more detailed overview of the network components. Each input, whether it is a regular observation or a contextual observation, goes through separate branches in the network. These branches process the inputs independently and then merge at a later stage, allowing the model to consider both asset-specific and contextual information.

The network architecture is designed to learn the optimal allocation weights by incorporating the historical performance of the assets and the contextual factors that can influence their returns. By training the network on a large dataset and fine-tuning the weights through the reinforcement learning process, we aim to achieve improved portfolio allocation decisions.

Through this deep network approach, we can effectively leverage both regular observations and contextual observations to enhance the model's ability to make informed and adaptive portfolio allocation decisions.

4.4.5 Testing the model

4.4.6 Walk-Forward Analysis and Results

As described in Chapter 2, a walk-forward analysis is conducted to stress test the Deep Reinforcement Learning (DRL) model and evaluate its out-of-sample performance, with the details of the dates provided in figure 4.5. The table 4.2 provides a comparison of a DRL (Deep Reinforcement Learning) model with traditional financial models, specifically focusing on various performance metrics such as annual return, annual volatility, Sharpe ratio, maximum drawdown (Max DD), and Calmar ratio.

In terms of annual return, the DRL model achieves a rate of 10.7%, which is higher than most traditional models such as Markovitz (8.6%), minimum variance (9.6%), maximum diversification (8.7%), maximum decorrelation (8.5%), and Risk Parity (8.4%). This suggests that the DRL model has been able to generate higher average returns compared to these traditional models.

The Sharpe ratio, a measure of risk-adjusted return, is significantly higher for the DRL model (1.50) compared to the traditional models like Markovitz (0.67), minimum variance (1.09), maximum diversification (1.33), maximum decorrelation (0.73), and Risk Parity (0.83). This suggests that the DRL model has achieved superior risk-adjusted returns compared to the traditional models.

In terms of maximum drawdown (Max DD), which measures the maximum loss from a peak to a subsequent trough, the DRL model experiences a drawdown of 16.4%, which is lower than most

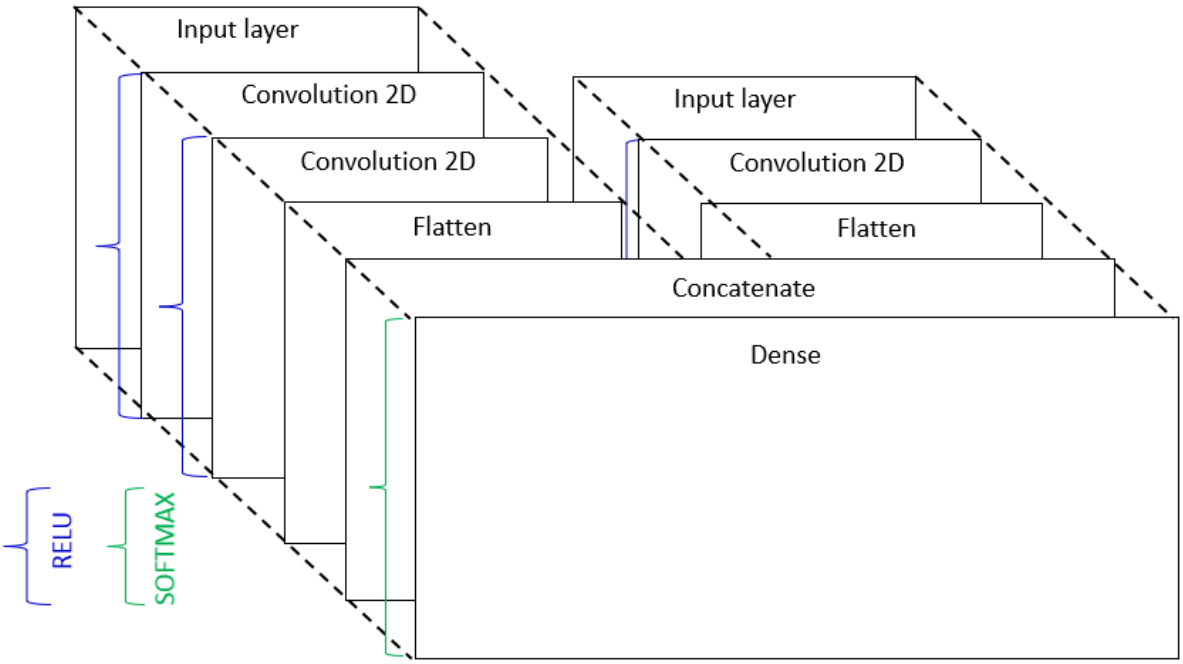


Figure 4.4: Network overall architecture

Figure 4.5: Walk forward analysis

of its financial models peers such as Markovitz (30.0%), Maximum decorrelation (24.6%), and Risk Parity (31.9%) or comparable to the Minimum variance (16.0%) model. This indicates that the DRL model has managed to limit losses during downturns better than these traditional models. However, the maximum diversification model is able to have a reduced maximum drawdown (10.0%), which is not a surprise given its specific nature.

The Calmar ratio, which assesses the risk-adjusted return relative to the maximum drawdown, is higher for the DRL model (0.65) compared to most of the traditional models like Markovitz (0.29), minimum variance (0.60), maximum decorrelation (0.35), and Risk Parity (0.26), except for the maximum diversification (0.87), due to its capacity to reduce drawdown. A higher Calmar ratio suggests better risk-adjusted performance in the DRL model.

Overall, based on the provided metrics, the DRL model appears to outperform the traditional financial models in terms of annual return, risk-adjusted return (Sharpe ratio), and for most of them maximum drawdown, and risk-adjusted return relative to drawdown (Calmar ratio). However, further analysis and evaluation are necessary to fully assess the effectiveness and robustness of the DRL model compared to traditional models in the specific financial context.

Additionally, two other models, namely DRL Max and DRL Min, are derived from the best and worst-performing DRL models, respectively, over the last 50 trials. Confidence intervals are computed based on these models, providing a range of potential outcomes.

To further illustrate the performance of the different strategies, figure 4.6 presents a plot showcasing their cumulative returns over the course of the experiment.

The results highlight the superior performance of the DRL model with Full Context, demonstrating its ability to generate higher returns with lower volatility compared to the other investment models. The maximum diversification model also proves to be a viable alternative with favorable Sharpe ratio of 1.33. Markowitz and maximum deccorrelation turns out to be the worst choices. These findings underscore the effectiveness of deep reinforcement learning in portfolio allocation tasks.

Table 4.2: Model comparison for the DRL model and the presented traditional financial models. For the DRL model, we provide its mean, maximum and minimum over the last 50 epochs. We also compute the Calmar ratio which is simply the ratio of the annual return over the maximum drawdown.

	DRL Mean	DRL Max	DRL Min	Markovitz	Minimum variance	Maximum diversification	Maximum decorrelation	Risk Parity
Ann. return	10.7%	12.6%	8.3%	8.6%	9.6%	8.7%	8.5%	8.4%
Ann. Volatility	7.1%	5.8%	9.6%	12.7%	8.8%	6.5%	11.6%	10.0%
Sharpe ratio	1.50	2.17	0.86	0.67	1.09	1.33	0.73	0.83
Max DD	16.4%	13.3%	21.7%	30.0%	16.0%	10.0%	24.6%	31.9%
Calmar ratio	0.65	0.95	0.38	0.29	0.60	0.87	0.35	0.26

4.4.7 Benefits of DRL

Overall, Deep Reinforcement Learning (DRL) model offers an interesting alternative to traditional methods in the following aspects:

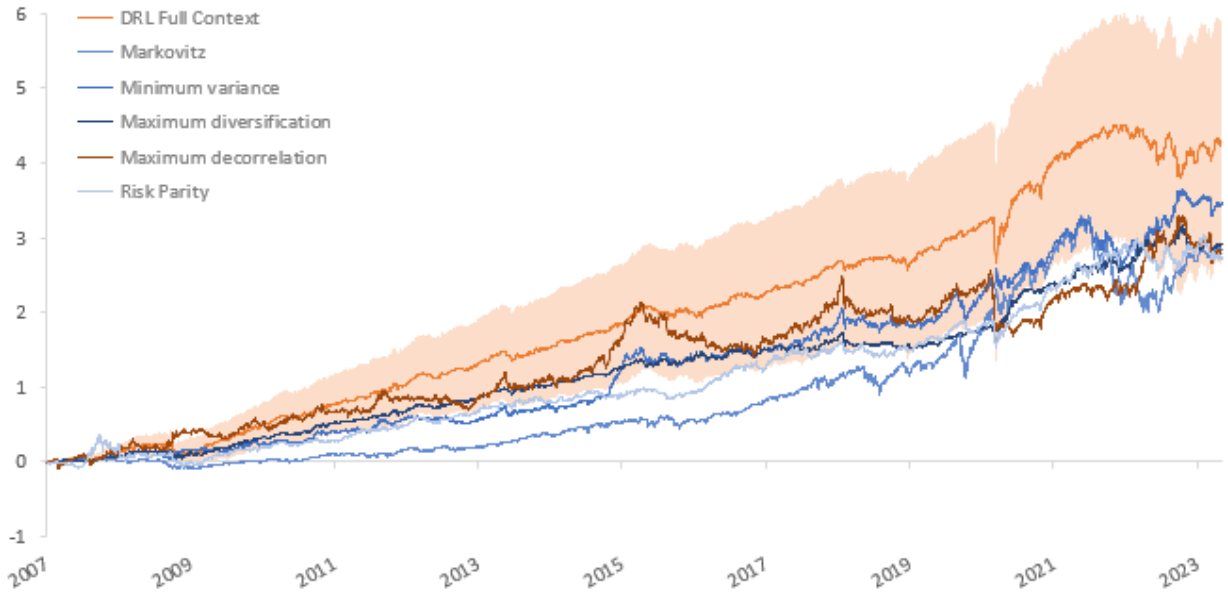


Figure 4.6: Comparison of DRL with confidence interval with traditional financial methods

1. Adaptability to changing environments: The DRL model is specifically designed to adapt to dynamic market conditions. By mapping market conditions directly to actions, the model can adjust its strategies and policies in response to changing investment regimes. This adaptability enables the DRL model to effectively navigate different market environments and potentially exploit emerging opportunities. This may explain why the model achieves higher Sharpe ratio on average.

2. Independence from traditional financial risk assumptions: Unlike traditional methods that rely on specific risk assumptions and the first two moments of the portfolio assets, the DRL model can capture and incorporate a broader range of information and factors into the decision-making process. By not being constrained by traditional risk assumptions, the DRL model can potentially uncover new patterns and relationships in the data, leading to improved performance.

3. Incorporation of additional data and multi-input capability: The DRL model has the advantage of being able to incorporate additional data beyond traditional financial indicators. This flexibility allows the model to consider a wider range of factors and information, providing a more comprehensive view of the market. By utilizing multiple inputs, the DRL model can capture complex relationships and patterns, potentially enhancing its decision-making capabilities and performance.

4. Incorporation of additional data and multi-input capability: The DRL model has the advantage of being able to incorporate additional data beyond traditional financial indicators. This flexibility allows the model to consider a wider range of factors and information, providing a more comprehensive view of the market. By utilizing multiple inputs, the DRL model can capture complex relationships and patterns, potentially enhancing its decision-making capabilities and performance.

Overall, the experiment highlighted the adaptability, independence from traditional risk assumptions, and multi-input capability of the DRL model as key factors contributing to its superiority over traditional methods. These advantages make the DRL model a promising approach for portfolio management and investment decision-making in various financial contexts.

4.5 Summary

This chapter provides a comprehensive exploration of the process involved in converting a traditional portfolio allocation problem into a Deep Reinforcement Learning (DRL) problem. We highlight the similarities between DRL and conventional financial methods, as both strive to achieve optimal allocations by solving optimization problems and relying on historical data. In fact, by utilizing states that include the first two moments of the portfolio assets and applying a one-step Reinforcement Learning approach, we can derive a Reinforcement Learning reformulation of traditional financial techniques. Extending this basic framework to a multi-timestep optimization yields an RL reformulation of rolling versions of Markowitz, minimum variance, risk parity, and numerous other traditional financial methods, given that a myopic reward is employed.

When all constraints are removed, we arrive at the general DRL setting, which reveals that DRL is a more diverse yet more challenging framework for determining optimal portfolio allocations. It is also important to note that, unlike traditional financial methods that merely optimize a set of numerical values, DRL seeks to establish a mapping or a function. We apply this concept to a new experiment where the trading agent is tasked with allocating resources among 11 major assets. The results demonstrate that DRL outperforms conventional methods. A condensed version of this chapter can be found in (Benhamou, Saltiel, Ungari & Mukhopadhyay 2020*a*).

In the following chapter, we will maintain our focus on the more theoretical aspects of DRL, delving into the implicit variance reduction theory employed by actor-critic methods.

4.6 Appendix

Table 4.3: Regular observations. List of all portfolio assets

Bloomberg Ticker	Description	Category
ES1 Index	E-Mini S&P 500 Futures	Equity
NH1 Index	Nikkei 225 Futures (Yen)	Equity
VG1 Index	Euro Stoxx 50 Futures	Equity
Z 1 Index	FTSE 100 Futures	Equity
TY1 Index	10Y T-Note Futures	Rates
RX1 Index	Euro-Bund Futures	Rates
G1 Index	ICE Long Gilt Futures	Rates
JB1 Index	10Y JGB Futures	Rates
GC1 Index	Gold Futures	Commodity
CO1 Index	Brent Crude Futures	Commodity
HG1 Index	High Grade Copper Futures	Commodity

Table 4.4: List of all the features used as contextual variables

Bloomberg Ticker	Description	Category
USGG10YR Index	US Rates 10 year	Interest Rates
USGG2YR Index	US Rates 2 year	Interest Rates
GDBR10 Index	Europe Rates 10 year	Interest Rates
GDBR2 Index	Europe Rates 2 year	Interest Rates
VIX INDEX	Cboe Volatility Index	Market Sentiment
CDX HY CDSI GEN 5Y SPRD Corp	MARKIT Credit CDX HY index	Credit Default Swaps
ITRX XOVER CDSI GEN 5Y Corp	MARKIT Credit Europe Itraxx Crossover	Credit Default Swaps
DXY Index	Dollar index	Currency Rates
PFRGGDP6 Index	Survey of Professional Forecaster on GDP	Economic Forecast
PHFFCPI1 Index	Survey of Professional Forecaster on CPIT	Economic Forecast
DOESTCRD Index	Crude Oil Total Inventory	Commodity Market
COMXGOLD Index	Comex Gold Inventory Data	Commodity Market
COMXCOPR Index	Comex Copper Inventory Data	Commodity Market
CESIGL Index	Global Economic Surprise	Economic Indicators
CESIUSD Index	USD Economic Surprise	Economic Indicators
CESIEUR Index	EUR Economic Surprise	Economic Indicators
CESIJPY Index	JPY Economic Surprise	Economic Indicators
CESIEM Index	EM Economic Surprise	Economic Indicators

Table 4.5: Hyper parameters used in the experiment

hyper-parameters	value	description
batch size	50	mini-batch size during training
regularisation coefficient	1e-8	L_2 regularisation coefficient applied to network training
learning rate	0.01	Step size parameter in Adam
standard deviation period	20 days	period for standard deviation in asset states
commission	30 bps	commission rate
stride	2,1	stride in convolution networks
conv number 1	5,10	number of convolutions in sub-network 1
conv number 2	2	number of convolutions in sub-network 2
lag period 1	[60, 20, 4, 3, 2, 1, 0]	lag period for asset states
lag period 2	[60, 20, 4, 3, 2, 1, 0]	lag period for contextual states
noise	0.002	Gaussian noise deviation
max iterations *	1000	maximum number of iterations
early stop iterations *	50	early stop criterion
random seed	12345	random seed

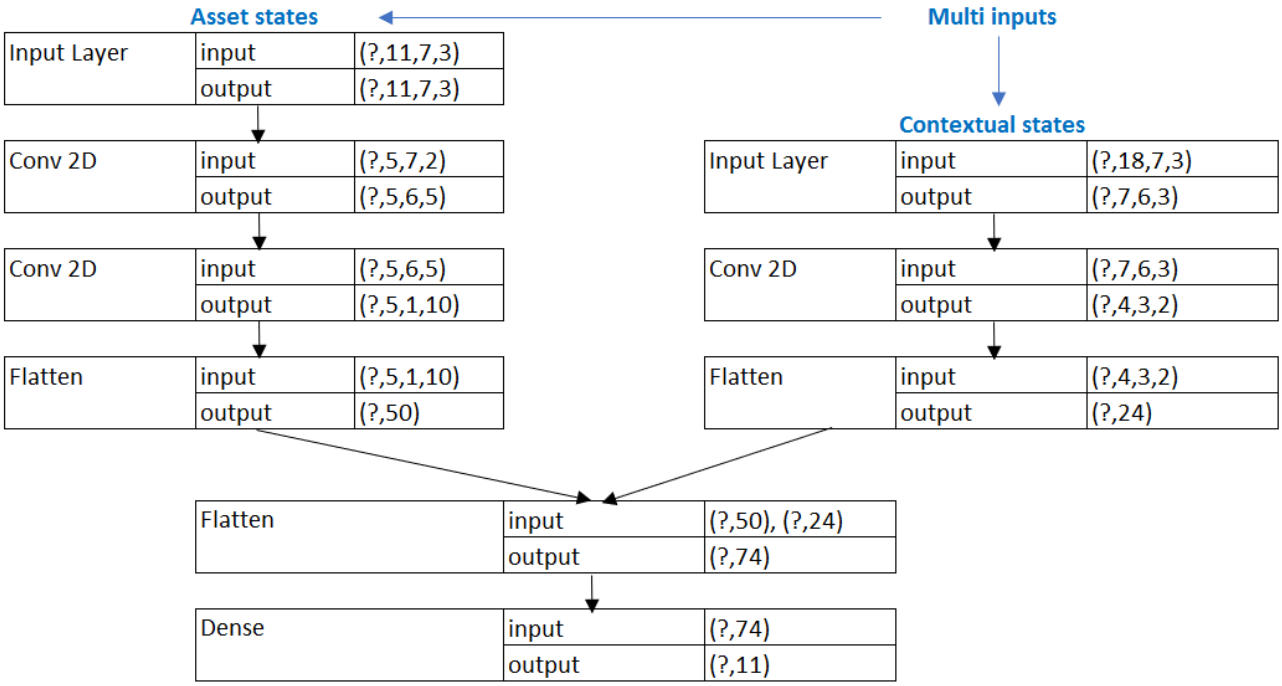


Figure 4.7: Network architecture details

5 Variance reduction in Actor critic methods

Mankind's history has been a struggle against a hostile environment. We finally have reached a point where we can begin to dominate our environment... As soon as we understand this fact, our mathematical interests necessarily shift in many areas from descriptive analysis to control theory.

Bellman (1968), an American applied mathematician, who introduced dynamic programming, an IEEE medal of honor recipient

5.1 Introduction

In the previous chapter, it was demonstrated that Deep Reinforcement Learning (DRL) is an extension of traditional financial methods that seeks to maximize the expected cumulated rewards. It is a non myopic optimization and uses a function to relate states to actions. However, computing the expectation of cumulated rewards and the resulting policy gradient is a non-trivial task. This chapter delves into how policy gradient and actor critics are efficient in calculating this expectation and how they are related to variance reduction techniques. Although most of the results are well known, the presentation as a control variate is somehow new as the traditional deep reinforcement learning focus on using a baseline to reduce the variance without explicitly stating that this is a control variate.

Actor-Critic Methods (ACM) have emerged as the state-of-the-art methods in DRL problems, as reported by (Jaderberg et al. 2016) and (Espeholt et al. 2018). With the introduction of Deep Reinforcement Learning methods, the scope of RL has been expanded to include a wide range

of domains, such as atari games (Mnih et al. 2016), Go (Silver et al. 2016), image recognition (Zoph et al. 2017), physics tasks, including classic problems such as cart pole swing-up, dexterous manipulation, legged locomotion, and car driving (Lillicrap et al. 2015), traffic signal control (Mannion, Duggan & Howley 2016), electricity generator scheduling (Mannion, Mason, Devlin, Duggan & Howley 2016), water resource management (Mason et al. 2016), controlling robots in real environments (Levine et al. 2016), and other transport problems (Talpaert et al. 2019), as well as control and manipulation tasks for robotics (Barth-Maron et al. 2018, Horgan et al. 2018).

Advantage Actor Critic (A2C) and Asynchronous Advantage Actor Critic (A3C) methods were originally derived from the REINFORCE algorithm (Williams 1992) and are an extension of the policy gradient descent method. Although their efficiency has been very extensively verified experimentally on test-bed environments such as Open AI gym and Atari games, the variance reduction question has been very rarely presented as a control variate question as this approach is mostly used in Monte Carlo simulation but not so much in the Reinforcement learning literature.

This chapter addresses this challenging question and provides a theoretical justification for the efficiency of Advantage Actor-Critic methods. It begins by presenting the key motivation for A2C methods and, in particular, the concept of baseline. It then proves the existence of optimal baselines according to the L^2 norm. The chapter also explains the concept of variance reduction and relates Q and Advantage Actor-Critic methods to conditional expectations that can be interpreted as projection in the L^2 norm of the initial logarithmic gradient in REINFORCE, as proposed by (Schulman, Levine, Abbeel, Jordan & July(2015).

5.1.1 Related Work

Multiple works have focused on presenting and comparing the Actor-Critic Method (ACM) in the context of Reinforcement Learning (RL). Some notable examples include (Grondman et al. 2012) and more recent works such as (Jaderberg et al. 2016, Lillicrap et al. 2015, Zoph et al. 2017, Barth-Maron et al. 2018, Horgan et al. 2018), and (Espeholt et al. 2018). The primary goal of these studies is to demonstrate the convergence of ACM as an improvement over the REINFORCE method.

Efficient optimization of ACM from both CPU and GPU perspectives has also been a subject of research. (Mnih et al. 2016) proposed an asynchronous version of the A2C algorithm, which uses multiple actors learning from a set of critics. This approach distributes the computation workload across multiple CPUs, enabling efficient parallel computation. (Babaeizadeh et al. 2017) further improved the algorithm by distributing computation across multiple GPUs, leveraging the highly parallelizable nature of the actor-critic method. (Espeholt et al. 2018) introduced the V-trace algorithm, which enables learners to learn asynchronously from different workers.

Despite advancements in certain domains, the foundational aspect of variance reduction in Actor-Critic Methods (ACM) has been relatively overlooked. Researchers in the field of reinforcement learning employ a baseline function to decrease variance by subtracting a term, with the objective of obtaining a smaller term. The underlying rationale is that a reduced term should correspond to diminished variance. The term "baseline function" denotes the function subtracted from return estimates during the computation of policy gradients. Given a state-dependent baseline function $b(\mathbf{s})$, the gradient estimator is represented as $\nabla \log \pi(\mathbf{s}, \mathbf{a}) \left(\hat{Q}(\mathbf{s}, \mathbf{a}) - b(\mathbf{s}) \right)$. The optimal state-dependent baseline is identified as the squared-gradient-norm weighted average of the state-action values (Weaver & Tao 2001, Greensmith et al. 2004, Zhao et al. 2011).

In fact, baseline functions have significantly contributed to the success of policy gradients in contemporary deep Reinforcement Learning (RL) (Schulman et al. 2017, Mnih et al. 2016, Chung et al. 2020), with numerous subsequent studies. One such line of research involves the design of state-action dependent baselines, as opposed to state-only dependent baselines, in order to achieve further variance reduction Liu et al. (2018), Grathwohl et al. (2018), Wu et al. (2018). However, previous studies have not presented this as a control variate estimator, thus not revealing certain theoretical results regarding optimal estimators.

This chapter seeks to emphasize the crucial role of variance reduction in the method and elucidate its fundamental principles through control variate theory. We demonstrate the applicability of multi-dimensional control variates and extend the work of (Schulman, Levine, Abbeel, Jordan & July(2015), who introduced a generalized advantage estimation utilizing a trust region optimization procedure for the policy and value functions represented by neural networks. Although the optimal control variate is not particularly practical, as it would necessitate a control variate fully correlated with the estimator to calculate, thus implying prior knowledge of the solution, this chapter sheds new light on the foundations of variance reduction in actor-critic methods.

5.1.2 Mathematical background

In this chapter, like in the rest of the thesis, we adopt the standard framework of Reinforcement Learning (RL). In RL, a learning agent interacts with an environment $\mathcal{E}$ to select rational or optimal actions and receives rewards in return. The rewards may not necessarily be positive and serve as feedback to guide the agent towards the best possible action.

To formalize this process, we utilize the Markov decision process (MDP) framework, which has been described at length in this thesis and specifically in the introductory Chapter 1. It consists in a 4-tuple: $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R})$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ the set of actions, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the transition function: set of conditional transition probabilities between states, actions to next states and finally, $\mathcal{R}$ the immediate reward received after state s and action a.
The requirement of a Markovian state is not a strong constraint as we can stack observations to enforce that the Markov property is satisfied. We adopt the approach of (Mnih et al. 2016) and (Jaderberg et al. 2016) by introducing the concept of observations and grouping them together to form states. At time t, the agent receives an observation o_t and a reward r_t before selecting an action a_t. The state s_t of the agent is a function of its experiences up to time t, specifically, $s_t = f(o_1, r_1, a_1, ..., o_t, r_t)$. This construction ensures that the states are Markovian. In practice, we only store a limited set of historical experiences and not the full history.

At time t, the n-step return $R_{t:t+n}$ is defined as the discounted sum of rewards over n steps, where $\gamma \in [0, 1)$ is the discount factor. That is, $R_{t:t+n} = \sum_{i=1}^{n} \gamma^i r_{t+i}$.

We recall that the value function $V^\pi(s)$ is the expected return from a state s under a policy π, that is, $V^\pi(s) = \mathbb{E}\left[R_{t:\infty} | s_t = s, \pi\right]$, where actions are selected according to the policy $\pi(a|s)$. The action-value function $Q^\pi(s, a)$ is the expected return from state s following action a, when actions are selected according to policy π. The goal of the agent is to find a policy π that maximizes the value function.

5.2 Variance Reduction

In the case where states and/or actions have high dimensions, we adopt a parametric approach to represent our policy using parameters denoted by θ. These parameters are typically the weights of a deep neural network that is used to approximate the policy function. The Actor-Critic Method (ACM) aims to leverage policy gradient descent with reduced variance to achieve efficient and stable learning.

To explain this in more detail, we recall that the policy gradient is computed as the logarithmic gradient of the policy weighted by the sum of future discounted rewards (Williams 1992, Sutton, McAllester, Singh & Mansour 1999).

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_\tau\Bigg[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \underbrace{\left(\sum_{t'=t+1}^{T} \gamma^{t'-t} r_{t'}\right)}_{R_t}\Bigg] \\ &= \mathbb{E}_\tau\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) R_t\right], \end{aligned} \tag{5.1}$$

where, τ denotes a trajectory, R_t represents the sum of future discounted rewards, and $\gamma \in [0, 1)$ is the discount factor. Equation (5.1) shows that the policy deep network parameters are updated through Monte Carlo updates computed as an expectation. It is worth noting that the accuracy of the estimation of this expectation is critical, as a large variance in the estimator provided by the standard empirical mean can result in high variability in our gradient update, leading to slow convergence of the policy method.

To overcome this challenge, it is useful to seek alternative expressions of the policy gradient with lower variance. This approach, known as variance reduction, aims to modify the expression inside the expectation such that it has the same expected value but a lower variance (Hammersley 1964). One common method of variance reduction is to use control variate(s) (see, for instance, (Ross 2002)). A lower variance in the gradient produces a less noisy gradient and leads to more stable learning, resulting in a policy distribution that is skewed towards the optimal direction more rapidly.

5.2.1 Control Variate

Let us begin by introducing the concept of control variates. The control variates (CV) method, a well-established variance reduction technique in Statistics (Rubinstein & Marcus 1985, Nelson 1990), is a representative approach (Mnih et al. 2016, Schulman, Levine, Abbeel, Jordan & July(2015, Schulman et al. 2017). In order to reduce the variance of an unbiased Monte Carlo estimator, a control variate with known expectation under the randomness is subtracted from (and the expectation added back to) the original estimator. The resulting estimator would still be unbiased but have lower variance than the original estimator if the control variate correlates with the original estimator.

More formally, in the scenario where we wish to estimate a quantity μ defined as an expectation $\mu = \mathbb{E}[\hat{m}]$ of an estimator $\hat{m}$, we can improve our estimation of the expectation of our initial estimator

by considering another statistic or estimator, denoted by $\hat{t}$ for which we know it expectation, defined as $\tau = \mathbb{E}[\hat{t}]$. This new estimator is built such that is still unbiased and potentially and improved estimator of μ as follows: $\hat{m'} = \hat{m} - \alpha(\hat{t} - \tau)$, where $\alpha \in \mathbb{R}$. The new estimator is referred to in the Monte Carlo literature (Glasserman 2004) as the "control variate" estimator of $\hat{m}$. It involves subtracting a zero-expectation term, also often referred to as an unbiased term. More formally we define a control variate as follows:

Definition 5.1 (Control Variate)**.** *Let $\hat{m}$ be a random estimator with unknown expected value $\mathbb{E}[\hat{m}]$, and let t be a control variate with known expected value $\tau = \mathbb{E}[\hat{t}]$. The estimator of $\mathbb{E}[\hat{m}]$ using the control variate $t = \hat{t} - \tau$ is defined as:*

$$\hat{m}_{CV} = \hat{m} - \alpha(\hat{t} - \tau), \tag{5.2}$$

where α is a real number.

The following control variate proposition holds:

Proposition 5.2. *Optimal Control Variate - Among all possible values of α, the optimal one (in the sense that it produces the estimator with minimum variance) is given by*

$$\alpha^* = \frac{\mathrm{Cov}(\hat{m}, \hat{t})}{\mathbb{V}\mathrm{ar}(\hat{t})} = \frac{\sigma_{\hat{m}}}{\sigma_{\hat{t}}}\rho_{\hat{m},\hat{t}}. \tag{5.3}$$

The corresponding control variate estimator is given by $\hat{m}^* = \hat{m} - \alpha^*(\hat{t} - \tau)$ and has a variance given by

$$\mathbb{V}\mathrm{ar}(\hat{m}^*) = (1 - \rho^2_{\hat{m},\hat{t}})\sigma^2_{\hat{m}} \leq \sigma^2_{\hat{m}}. \tag{5.4}$$

Proof. We have:

$$\mathbb{E}[\hat{m}_{CV}] = \mathbb{E}[\hat{m}] - \alpha\mathbb{E}[\hat{t} - \tau] = \mathbb{E}[\hat{m}] = \mu, \tag{5.5}$$

since $\mathbb{E}[\hat{t}] = \tau$. This demonstrates that the control variate estimator is unbiased for any value of α. The variance of this estimator can be easily calculated and is given by:

$$\mathbb{V}\mathrm{ar}(\hat{m}_{CV}) = \mathbb{V}\mathrm{ar}(\hat{m}) - 2\alpha\,\mathrm{Cov}(\hat{m}, \hat{t}) + \alpha^2\mathbb{V}\mathrm{ar}(\hat{t}). \tag{5.6}$$

This is a second-order parabolic function of α. It attains its minimum value for the unique value of its stationary point given by:

$$\alpha^* = \frac{\mathrm{Cov}(\hat{m}, \hat{t})}{\mathbb{V}\mathrm{ar}(\hat{t})} = \frac{\sigma_{\hat{m}}}{\sigma_{\hat{t}}}\rho_{\hat{m},\hat{t}}. \tag{5.7}$$

The resulting control variate variance is given by:

$$\mathbb{V}\mathrm{ar}(\hat{m}^*) = (1 - \rho^2_{\hat{m},\hat{t}})\sigma^2_{\hat{m}} \leq \sigma^2_{\hat{m}}. \tag{5.8}$$

□

Remark 5.3. This proposition readily implies that the most effective control variate estimators are attained when control variates exhibit high positive correlation ($\rho_{\hat{m},\hat{t}} \approx 1$) or high negative correlation ($\rho_{\hat{m},\hat{t}} \approx -1$), as this leads to a reduction in variance.

Remark 5.4. For a given control variable $\hat{t}$ and a given variable $\hat{m}$ whose aim is to reduce its variance $\hat{m}$, the optimal control variate corresponding to this specific control variable is unique, as it is derived from the minimum of a parabolic equation. The value of the floating number α^* is provided by proposition 5.2. However, when considering a target variance after applying control variates, the potential control variates $\hat{t}$ are not unique. Among all control variates, those with the same squared correlation yield obviously the same reduction in variance. In practice, there is usually only one variable that is highly positively or negatively correlated with our estimator of interest $\hat{m}$. The essential aspect of control variates lies in identifying another estimator that exhibits strong positive or negative correlation with our estimator of interest $\hat{m}$. It is important to note that the theoretical case of perfect correlation, $\rho_{\hat{m},\hat{t}} = 1$, is purely hypothetical because it would imply that our estimation problem is already solved, as we would have another estimator with a correlation of 1 whose expectation is known. This would suggest that we already possess a rescaled version of our desired estimator.

Intuitively, the more correlated (positively or negatively) the estimators $\hat{m}$ and $\hat{t}$ are, the better we can utilize the knowledge of our control variate zero-expectation estimator $\hat{t} - \tau$ to reduce the variance of our initial estimator $\hat{m}$. In essence, control variates allow us to exploit information about the errors in estimates of known quantities $(\hat{t} - \tau)$ to reduce the error of an estimate of an unknown quantity $(\hat{m})$. The optimal control variate weight $\alpha^* = \frac{\text{Cov}(\hat{m},\hat{t},)}{\mathbb{V}\text{ar}(\hat{t})}$ can be analyzed as a regression coefficient of our unknown estimator $\hat{m}$ over the space of linear combinations of the zero-expectation estimator $\hat{t} - \tau$. Therefore, control variates help us focus on the orthogonal part of our unknown estimator $\hat{m}$ with respect to the space of linear combinations of the zero-expectation estimator $\hat{t} - \tau$. If our unknown estimator is highly correlated with the control variate, this orthogonal part is close to zero, and we obtain a much lower variance estimator.

In practice, in deep RL, we do not know the correlation between the two estimators $\rho_{\hat{m},\hat{t}}$, nor the variances of our two estimators: $\sigma_{\hat{m}}$ or $\sigma_{\hat{t}}$. Therefore, instead of being able to use the optimal control variate estimator, we create a pseudo-control variate estimator given by:

$$\hat{m} - \hat{t}, \tag{5.9}$$

provided $\mathbb{E}[\hat{t}] = 0$. The latter formulation takes a control variate coefficient of $\alpha = 1$, which implicitly assumes that the optimality should be obtained for $\text{Cov}(\hat{m}, \hat{t}) \approx \mathbb{V}\text{ar}(\hat{t})$. We will use this setting to interpret various Actor-Critic (AC) methods as control variate estimators for standard AC methods for policy gradients.

In order to have some mathematical formalism, let us recall some definitions even if we have already presented actor-critic methods in Section 1.2.9.

Definition 5.5. *The REINFORCE estimator is a popular method for training the agent's policy, which is a function that maps states to actions. The REINFORCE estimator works by estimating the gradient of the log policy times the future rewards. It is mathematically given by:*

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a) R(s)\right]. \tag{5.10}$$

Apart from the traditional REINFORCE estimator to compute the gradient policy, there is a large body of literature on policy gradient methods (Williams 1992, Sutton et al. 2000, Konda

& Tsitsiklis 2000, Kakade 2002, Peters & Schaal 2006, Mnih et al. 2016, Schulman, Levine, Abbeel, Jordan & July(2015, Schulman et al. 2017). So we also define the following actor-critic estimators:

Definition 5.6. *We can at least present the following baseline estimators:*

- *Q actor-critic abbreviated as Q-AC:*

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a) Q(s,a)\right], \tag{5.11}$$

  where $Q(s,a)$ is the action-value function defined as $Q(s,a) = \mathbb{E}_{\pi\theta}[R_t|s_t = s, a_t = a]$.

- *Advantage actor-critic abbreviated as A-AC:*

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a) A(s,a)\right], \tag{5.12}$$

  where $A(s,a)$ is the Advantage function defined as $A(s,a) = \mathbb{E}_{\pi\theta}[R_t|s_t = s, a_t = a] - V^\pi(s)$.

- *TD-AC:*

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a) TD(s)\right], \tag{5.13}$$

  where $TD(s)$ is the traditional Temporal Difference function defined as $TD(s) = r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$

In each case, the method is an estimator of the policy gradient and can be analyzed as a control variate estimator of the REINFORCE policy gradient estimator. The Q-AC method uses the action-value function, the A-AC method uses the Advantage function, and the TD-AC method uses the Temporal Difference function. This leads to the following proposition:

Proposition 5.7. *Actor Critic Methods - The previous estimators of the policy gradient defined in definition 5.6 are unbiased and can be analysed as control variate estimators of REINFORCE policy gradient estimator*

Proof. Let us tackle one by one the various estimators given in proposition 5.7. The **Q Actor-Critic (Q-AC)** estimator is given by $\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a) Q(s,a)\right]$. It writes also as

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a) (\underbrace{R(s)}_{\hat{m}} - \underbrace{(R(s) - Q(s,a))}_{\hat{t}})\right]$$

which shows that it is a control variate estimator (as explained in equation (5.9)) provided we prove that

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a) \left(R(s) - Q(s,a)\right)\right] = 0. \tag{5.14}$$

The equation (5.14) is trivially verified as the Q function writes as an expectation of the reward function:

$$Q(s,a) = \mathbb{E}_{\pi_\theta}[R(s) \mid s_t = s, a_t = a]$$

The law of total expectation states that if X is a random variable and Y any random variable on the same probability space, then $\mathbb{E}[\mathbb{E}[X \mid Y]] = \mathbb{E}[X]$ or in other words $\mathbb{E}[\mathbb{E}[X \mid Y] - X] = 0$, which concludes the proof for the Q Actor Critic (Q-AC) method with $X = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a) R(s)$ and $Y = (s_t = s, a_t = a)$.

As for the **Advantage Actor Critic (A-AC)** method, the same reasoning shows that it suffices to prove that

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a)(R(s) - A(s,a))\right] = 0$$

to show that this is also a control variate method. Recall that the advantage function is given by $A(s,a) = Q(s,a) - V(s)$. Using the result previously proved for the Q AC method (equation (5.14)), it suffices to prove that

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a) V(s)\right] = 0. \tag{5.15}$$

To prove the latter, recall that the gradient of the log policy is given by:

$$\nabla_\theta \log \pi_\theta(s,a) = \frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)}. \tag{5.16}$$

Hence we have,

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a) V(s)\right] = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \pi_\theta(s,a) \frac{V(s)}{\pi_\theta(s,a)}\right] = \nabla_\theta \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \pi_\theta(s,a) \frac{V(s)}{\pi_\theta(s,a)}\right], \tag{5.17}$$

where we have used the linearity of expectation and move the gradient term outside of the expectation. To conclude, we can further simplify the expression inside the expectation and notice that the term inside the expectation does not depend on the policy π_θ, so its gradient with respect to θ is null:

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a) V(s)\right] = \nabla_\theta \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} V(s)\right] = 0, \tag{5.18}$$

Finally, let us prove that the **TD Actor-Critic (TD-AC)** method is also a control variate. Recall that the Temporal Difference term is given by

$$TD(s_t) = r_t + \gamma V(s_{t+1}) - V(s_t)$$

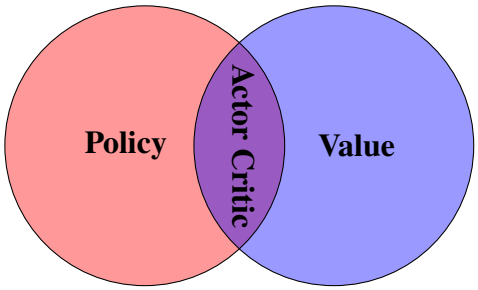


Figure 5.1: The different RL methods. ACM combine policy and value computations, making them mixed methods

Using equation (5.15), it suffices to prove that

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a)(r_t + \gamma V(s_{t+1}) - R(s_t))\right] = 0. \tag{5.19}$$

This is again a straightforward application of the law of total expectation as

$$\gamma V(s_{t+1}) = \gamma \mathbb{E}_{\pi_\theta}[\sum_{s=t+1}^{T} \gamma^{s-(t+1)} r_s \mid s_{t+1}]$$

while

$$-(R(s_t) - r_t) = -\sum_{s=t+1}^{T} \gamma^{s-t} r_s$$

Hence we can apply the law of total expectation with $X = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(s,a) \sum_{s=t+1}^{T} \gamma^{s-t} r_s$ and $Y = s_{t+1}$. This concludes the proof. □

Traditionally, methods for solving Reinforcement Learning (RL) are either categorised as policy methods if they aim to find the optimal policy $\pi^\star$ or as value methods if they aim to find the optimal 'Q' function. Somehow, Q-AC and A-AC methods aim to find the optimal policy thanks to gradient ascent computation but use in their gradient ascent term a 'Q' function, making these methods a mix between policy and value methods. This is summarised by figure 5.1.

5.2.2 Conditional expectation, projection and optimality

So far, we have discussed AC methods as control variates. However, there is a strong connection between conditional expectation and projection that we can explore. Before delving into this connection, let us first recall a fundamental property of conditional expectation, which states that the conditional expectation with respect to a sub σ-algebra $\mathcal{G} \in \mathcal{F}$ of a stochastic variable X, which is $\mathcal{L}^2$ measurable on a probability space $(\Omega, \mathcal{F}, \mathcal{P})$, is the best predictor of the subspace spanned by this sub σ-algebra. In other words, the conditional expectation provides the best estimate of X given the information contained in $\mathcal{G}$.

Proposition 5.8. *Conditional expectation and Pythagoras - Let $(\Omega, \mathcal{F}, \mathcal{P})$ be a probability space, $X : \Omega \to \mathbb{R}^n$ a random variable on that probability space square integrable and $\mathcal{G} \subseteq \mathcal{F}$ is a sub σ-algebra of $\mathcal{F}$, then we have that*

- *$X - \mathbb{E}[X \mid \mathcal{G}]$ is orthogonal to any element Y of $\mathcal{L}^2(\Omega, \mathcal{G}, \mathcal{P})$ where $\mathcal{L}^2(\Omega, \mathcal{G}, \mathcal{P})$ is the space of random variable on the sub σ-algebra $\mathcal{A}(\Omega, \mathcal{G}, \mathcal{P})$ that are square integrable.*
- *$\mathbb{E}[X \mid \mathcal{G}]$ is the best prediction in the sense that $\mathbb{E}[X \mid \mathcal{G}]$ minimises its variance with X: $\mathbb{E}[(X - Y)^2]$ among any element $Y \in \mathcal{L}^2(\Omega, \mathcal{G}, \mathcal{P})$.*

Proof. Let us first prove that for any $Y \in \mathcal{L}^2(\Omega, \mathcal{G}, \mathcal{P})$, we have

$$\mathbb{E}[Y\dot{(}X - \mathbb{E}[X \mid \mathcal{G}]] = 0$$

This is straight application of the law of total expectation (also referred to as the law of iterated expectation or also the tower property) as follows

$$\begin{aligned}
& \mathbb{E}[Y \cdot (X - \mathbb{E}[X \mid \mathcal{G}])] \\
= \; & \mathbb{E}[\mathbb{E}[Y \cdot (X - \mathbb{E}[X \mid \mathcal{G}]) \mid \mathcal{G}]] \\
= \; & \mathbb{E}\Big[Y \cdot \Big(\underbrace{\mathbb{E}[X \mid \mathcal{G}] - \mathbb{E}[\mathbb{E}[X \mid \mathcal{G}] \mid \mathcal{G}]}_{=0}\Big)\Big] \\
= \; & 0,
\end{aligned}$$

which proves the orthogonality.

For $Y \in \mathcal{L}^2(\Omega, \mathcal{G}, \mathcal{P})$, we have

$$\begin{aligned}
& \mathbb{E}[(X - Y)^2] \\
= \; & \mathbb{E}[(X - \mathbb{E}[X \mid \mathcal{G}] + \mathbb{E}[X \mid \mathcal{G}] - Y)^2] \\
= \; & \mathbb{E}[(X - \mathbb{E}[X \mid \mathcal{G}])^2 + (\mathbb{E}[X \mid \mathcal{G}] - Y)^2] \\
& +2\underbrace{\mathbb{E}[(X - \mathbb{E}[X \mid \mathcal{G}])(\mathbb{E}[X \mid \mathcal{G}] - Y)]}_{=0} \\
\geq \; & \mathbb{E}[(X - \mathbb{E}[X \mid \mathcal{G}])^2],
\end{aligned}$$

which concludes the proof that the squared L^2 norm of $X - Y$ for any element Y of $\mathcal{L}^2(\Omega, \mathcal{G}, \mathcal{P})$ is lower bounded by the squared L^2 norm of $X - \mathbb{E}[X \mid \mathcal{G}]$ □ □

The different Actor-Critic Methods (ACM) can be compared from a variance perspective by using propositions 5.2, 5.7, and 5.8. The intuition behind these methods is worth noting. TD-AC uses the Temporal Difference, which projects the cumulative discounted rewards onto the sub σ-algebra generated by the knowledge of the state s_{t+1}. Q-AC uses the action value 'Q' function, which is its projection onto the sub σ-algebra generated by the knowledge of the state s_t and action a_t. A-AC relies on the advantage function, which is the difference between the action value 'Q' function and the value function, which is the cumulative discounted rewards projected onto the sub σ-algebra generated by the knowledge of the state s_t.

As the sub σ-algebras become larger (in the sense that each one is contained by the next one), the corresponding AC methods become more effective. TD-AC can be improved by Q-AC, which in turn can be improved by the A-AC method, as shown in proposition 5.9.

From a variance point of view, TD-AC is less efficient than A-AC, and REINFORCE is less efficient than Q-AC. This is because the variance of the control variate is due to the residuals computed as the initial estimator minus the control variate. The control variate is the orthogonal projection of the initial estimator on the linear subspace spanned by the control variate. Proposition 5.8 shows that within the possible sub σ-algebras, the conditional expectation is the orthogonal projection and is the best estimator.

The Advantage Actor-Critic (A-AC) method has lower variance than the Temporal Difference Actor-Critic (TD-AC) method because the corresponding sub σ-algebra obtained by conditioning with respect to s_t, a_t used for A-AC contains the one obtained by conditioning with respect to s_{t+1} and used for TD-AC. Similarly, as the 'Q' function is a conditional expectation of the future discounted rewards, the Q Actor critic method performs better than just REINFORCE as it has lower variance.

More formally, we can prove the following:

Proposition 5.9. *AC Methods Comparison - From a variance point of view, TD-AC is less efficient than A-AC, and REINFORCE is less efficient than Q-AC.*

Proof. As explained in Proposition 5.2, the variance of the control variate is due to the residuals computed as the initial estimator minus the control variate. The control variate is the orthogonal projection of the initial estimator on the linear subspace spanned by the control variate. Proposition 5.8 shows that within the possible sub σ-algebras, the conditional expectation is the orthogonal projection and is the best estimator.

Recall that the Temporal difference writes as

$$r_{t+1} + \gamma V(s_{t+1}) - V(s_t). \tag{5.20}$$

Recall also that the value function is a conditional expectation

$$V(s_{t+1}) = \mathbb{E}[R_{t+1} \mid s_{t+1}], \tag{5.21}$$

while the state-action value 'Q' function is also a conditional expectation but with respect to s_t, a_t:

$$Q(a_t, s_t) = \mathbb{E}[R_t \mid s_t, a_t]. \tag{5.22}$$

Last but not least, the Advantage function writes as

$$A(a_t, s_t) = Q(a_t, s_t) - V(s_t). \tag{5.23}$$

From equations (5.20), (5.22), and (5.23), we can deduce that the Advantage Actor-Critic (A-AC) method has lower variance than the Temporal Difference Actor-Critic (TD-AC) method, as the corresponding sub σ-algebra obtained by conditioning with respect to s_t, a_t used for A-AC contains the one obtained by conditioning with respect to s_{t+1} and used for TD-AC.

Likewise, as the 'Q' function is a conditional expectation of the future discounted rewards, the Q Actor-Critic method should perform better than just REINFORCE as it has lower variance. □

5.3 Summary

Throughout this chapter, we have revisited actor-critic (AC) methods, which can be seen as control variate estimators of REINFORCE. We have demonstrated that these methods can reduce the variance of our estimator, leading to a more stable and faster learning process. By leveraging the sub σ-algebra generated by the state and action space, these methods project the cumulative discounted rewards onto a lower-dimensional subspace, resulting in a more informative estimate of the expected return. Specifically, TD-AC relies on the Temporal Difference, which is the projection of the cumulative discounted rewards onto the sub σ-algebra generated by the knowledge of the state s_{t+1}. Q-AC utilizes the action value 'Q' function, which is its projection onto the sub σ-algebra generated by the knowledge of the state s_t and action a_t. A-AC, on the other hand, utilizes the advantage function, which is the difference between the action value 'Q' function and the value function, projected onto the sub σ-algebra generated by the knowledge of the state s_t. Thanks to these properties, we have shown that the Q Actor critic respectively the Advantage Actor-Critic method is more efficient than the REINFORCE, respectively the TD-Actor Critic method, based on the property of conditional expectation.

In the next chapter, we will explore the relationship between supervised and reinforcement learning under certain conditions. This will further our understanding of the similarities and differences between these two paradigms and help us to develop more effective and efficient learning algorithms.

6 Similarities between Reinforcement Learning and Supervised Learning

> It is a good thing to have two ways of looking at a subject, and to admit that there are two ways of looking at it.
>
> *Maxwell (1873), a Scottish mathematician and physicist who developed the classical theory of electromagnetic radiation*

6.1 Introduction

The previous chapters of this thesis have primarily focused on reinforcement learning, which involves sequential decision-making. In contrast, supervised learning is a more common approach used in machine learning, where the objective is to predict a target output given a set of input features.

Although there is a fundamental difference between RL and SL, we will investigate potential connections between the two in this chapter. Specifically, we demonstrate that the gradient policy method can be viewed as a supervised learning problem, where true labels are replaced with discounted rewards. By establishing this link, we provide a new perspective on policy gradient methods (PGM) and their connection to cross entropy and supervised learning.

To further explore this connection, we conduct a simple experiment where we interchange labels and pseudo-rewards. This experiment illustrates the potential for modifying reward functions to establish other relationships between RL and SL, suggesting that these two paradigms may be more interconnected than previously thought.

Overall, this chapter sheds light on the potential similarities and connections between RL and SL,

offering insights into how these two approaches can be combined and leveraged to develop more effective and efficient learning algorithms.

6.1.1 Motivations

Reinforcement Learning (RL) and its policy gradient methods (PGM) have been widely used in machine learning (Williams 1992, Sutton & Barto 2018, Silver et al. 2014, Lillicrap et al. 2015). PGM is an RL technique that optimizes the parameters of policies to maximize the expected cumulative reward using gradient descent optimization. In contrast, value learning methods (Watkins & Dayan 1992) improve the value function until convergence or optimize the parameters of the value function. The value function represents the expected cumulative reward given an initial state and action.

PGM principles are straightforward: improve the policy gradually through gradient descent. PGM involves two important concepts: policy and gradient descent methods. Policy means we observe and act, while gradient descent methods use the fact that the best move locally is along the gradient. A learning rate is necessary to ensure that policy improvement is not too large at each step. PGM has been popularized by REINFORCE (Williams 1992) and Actor-Critic methods (Konda & Tsitsiklis 2003, Peters & Schaal 2008) and has received wider attention with deep PGM (Mnih et al. 2016) that combines policy and value methods. In addition, recently, it has been found that an entropy regularization term may accelerate convergence (O'Donoghue et al. 2016, Nachum et al. 2017, Schulman et al. 2017).

Looking more closely at REINFORCE (Williams 1992), we can observe that the gradient term concerning the policy can be interpreted as the logarithmic term in the cross-entropy function in supervised learning (SL). If we make the bridge between RL and SL, emphasizing that RL problems can be reformulated as SL problems where true labels are replaced by expected discounted future rewards, and estimated probabilities by policy probabilities, the connection between RL and SL becomes apparent. Furthermore, leveraging the tight relationship between cross-entropy and Kullback Leibler divergence, we can interpret the entropy regularization terms naturally. Therefore, this chapter aims to highlight the close connection between RL and SL to provide theoretical justifications for some of the techniques used in PGMs.

The chapter is organized as follows. In Section 6.3.1, we recapitulate the various choices of functional losses in SL and show that cross-entropy is one of the main possibilities for loss functions. We emphasize the relationship between cross-entropy and Kullback Leibler divergence, highlighting the additional entropy term. In Section 6.3.2, we present PGMs and interpret RL problems as a modified cross-entropy SL problem.

6.1.2 Related Work

RL and SL have traditionally been viewed as distinct fields with little overlap. However, Barto & Dietterich (2004) has discussed at length the key differences between reinforcement learning and supervised learning and whether these differences are fundamental or superficial. The author notes that it is possible to convert any supervised learning task into a reinforcement learning task by using the loss function of the supervised task as a reward function. However, it is not clear why one would want to do this as it converts the supervised problem into a more difficult reinforcement learning problem.

Conversely, there is no general way to convert a reinforcement learning task into a supervised learning task because the goals of the two types of learning are different. In supervised learning, the goal is to reconstruct the unknown function f that assigns output values y to data points x, while in reinforcement learning, the goal is to find the mapping from states x to action $a(x)$, where actions are a deterministic function of the states, that gives the maximum reward provided the current state and action $R(x, a(x))$. The assumption that actions are a function of the states implies in particular that we limit ourselves to deterministic policies.

Continuing on this idea, the author explores whether ideas from supervised learning can be applied to perform reinforcement learning. For instance, given a set of training examples of the form $(x_i, R(x_i))$, where x_i are points and $R(x_i)$ are the corresponding observed rewards, we would attempt to find a function h that approximates R well. If h were a perfect approximation of R, we could find the optimal action x to map the state to the optimal reward just by applying standard optimization algorithms to h. However, h could be a very poor approximation of R and still be very helpful for finding the optimal action provided their optimum remains the same.

Indeed, the article notes that we seek a function h whose maxima are good approximations of R's maxima. There are a variety of ways of formulating this problem as an optimization problem that can be solved. For example, we can require h to be a good approximation of R but also satisfy a monotony constraint: for any two training examples $(x_1, R(x_1))$ and $(x_2, R(x_2))$, if $R(x_1) > R(x_2)$ then $h(x_1)$ must be greater than $h(x_2)$.

The author concludes that techniques of this kind are an area of active research, and these optimization problems are not equivalent to supervised learning problems.

Another way to explore the relationship between reinforcement learning and supervised learning has been the emergence of Imitation Learning (Schaal 1996), which bridges the gap between the two. Imitation Learning is a supervised learning approach that involves imitating an expert's behaviour, making it possible to accomplish an RL task as a supervised one. One of the most popular Imitation Learning algorithms is DAGGER (Ross et al. 2011), which is based on requesting an action from the expert at each step and combining the observed states and demonstrated actions to train the agent successively. This approach has led to numerous extensions, including a deep version of DAGGER that leverages deep neural networks and continuous action spaces (Sun et al. 2017).

Other approaches have sought to combine RL and SL techniques more directly (Hester et al. 2017). For example, the Deep Q-learning from Demonstrations (DQfD) method combines temporal difference updates from RL with supervised classification of the demonstrator's actions, training the target network in Deep Q-learning with supervised learning.

These developments in imitation learning and incorporating SL into deep RL demonstrate that the boundaries between RL and SL are not as clear-cut as previously thought. In fact, it can be argued that PGMs can be reformulated as SL tasks, with true labels replaced by future expected rewards and the policy gradient interpreted as minimizing cross-entropy loss. This highlights the underlying similarities and synergies between RL and SL and shows that both approaches can be used to solve similar problems.

6.2 Contributions

The following are the main contributions of this chapter:

1. We have identified the specific conditions under which supervised learning and reinforcement learning are equivalent for policy gradient methods. The equivalence holds true if the reward does not change future states, and if the supervised learning problem has a label equal to the optimal rewards. It is important to note that the assumptions required for this equivalence differ from those used in Barto & Dietterich (2004), which focuses on using losses that are good approximators of reward functions, rather than on finding a one to one mapping between supervised and reinforcement learning.

2. We have established that the cross-entropy loss-based supervised learning and reinforcement learning can be equated by modifying the gradient of the policy gradient logarithm as the supervised learning cross-entropy loss, and by modifying the reward as the label. However, to map policy gradient methods in RL to regression in supervised learning, we would need to know the optimal rewards expected for each mapping, which makes the exercise more theoretical than practical.

3. Our work demonstrates that a RL problem where actions have no impact on future states can be rigorously considered as a supervised learning problem, aimed at learning the mapping from x to y. In RL terms, this involves learning the mapping from state s to the optimal reward r.

6.3 Relations between SL and RL

6.3.1 Supervised Learning

Supervised learning (SL) is a broad class of machine learning algorithms that aims to infer a function from labelled training data, which maps input variables to labelled output variables (Bishop 2007). SL encompasses both classification and regression tasks. In classification, the goal is to learn a function that maps inputs to one of several possible classes. In regression, the goal is to learn a function that maps inputs to a continuous output variable.

To obtain the optimal solution of the optimization program, the parameters of the SL algorithm are traditionally optimized through the minimization of a loss function. In binary classification problems, the label variable takes two different values, $\mathcal{Y} = -1, 1$, while multi-class classification problems assume $\mathcal{Y} = 1, \ldots, K$. In this context, the goal is to find a function $f : \mathcal{X} \mapsto \mathbb{R}$ that best maps input variable X to output variable Y. A loss function $\ell : -1, 1 \times -1, 1 \mapsto \mathbb{R}$ is used to measure the error of a specific prediction, where the loss function value at an arbitrary point $(Y, \hat{Y})$ represents the cost incurred when predicting $\hat{Y}$ while the true label is Y. In classification, the loss function is often a zero-one loss, where $\ell(Y, \hat{Y})$ is zero when the predicted label matches the true label $Y = \hat{Y}$ and one otherwise.

The best classifier is obtained by looking for the classifier with the smallest expected loss. In other words, the function f that minimizes the expected ℓ-risk, denoted by $\mathcal{R}\ell(f) = \mathbb{E}X, Y[\ell(Y, f(X))]$, is the optimal solution. Here, X is the input variable, and Y is the corresponding output variable.

In SL, the goal is to learn from a set of labeled examples, $D_n = (X_1, Y_1), \ldots, (X_n, Y_n)$, which are n independent random copies of $(X, Y) \in \mathcal{X} \times \mathcal{Y}$. The feature X lives in some abstract space $\mathcal{X}$, which can be a Euclidean space, and Y is the label variable. In classification problems, Y takes on discrete values, while in regression problems, Y takes on continuous values. The goal is to find a function f that maps X to Y using the labeled examples in D_n.

Another naive approach is to minimise the empirical classification error $\mathbb{E}[\mathbb{1}_{\{-Yf(X)\geq 0\}}]$. To bypass the non convexity of $\mathbb{1}_{\mathbb{R}_+}$, we use convex risk minimization (CRM) (Boucheron et al. 2005). CRM defines a convex surrogate for the classification problem, called the cost function $\varphi : \mathbb{R} \mapsto \mathbb{R}_+$ convex, non-decreasing such that $\varphi(0) = 1$, hence $\varphi \geq \mathbb{1}_{\mathbb{R}_+}$.

The classification problem consists in minimising the expected φ-risk : $\mathbb{E}[\varphi(-Yf(X))]$. Typical loss functions are:

- logit loss: $\varphi(u) = \log_2(1 + e^u)$, leading to logistic regression methods and cross-entropy.
- square loss: $\varphi(u) = (1 + u)^2$ for $u \geq 0$ and $\varphi(u) = 1$ for $u \leq 0$.
- perplexity loss: $\varphi(u) = \log(e(1 + u)))$ for $u > -1$ and $\varphi(u) = -\infty$ for $u \leq -1$.
- hinge loss: $\varphi(u) = max(0, 1 + u)$, leading to SVM classification methods.
- exponential loss: $\varphi(u) = e^u$.

For supervised learning regression, typical losses are as follows:

- means square error: $\ell(Y, f(X)) = (Y - f(X))^2$ that leads to standard regression methods.
- mean absolute error: $\ell(Y, f(X)) = |Y - f(X)|$.
- cross entropy: $\ell(Y, f(X)) = -\frac{1+Y}{2} \log \frac{1+f(X)}{2}$ leading to logistic regression with the usual convention $0 \log 0 = 0$ justified by $\lim_{x \to 0^+} x \log x = 0$ (trivially obtained by L'Hopital's rule).
- Huber loss: $\ell(Y, f(X)) = \frac{1}{2}(Y - f(X))^2$ if $|Y - f(X)| \leq \delta$ and $\ell(Y, f(X)) = \delta(|Y - f(X)| - \frac{1}{2}\delta^2$

We see from above that for SL there are numerous loss functions and that cross-entropy is one criterion among others. We will see that this cross-entropy has a nice interpretation in RL.

6.3.2 Reinforcement Learning Background

RL is usually modelled by an agent that interacts with an environment $\mathcal{E}$ over several discrete time steps. At each time step t, the agent levies a state s_t and picks an action a_t from a set of possible actions $\mathcal{A}$. This choice is done according to its policy π, where π is a mapping from states s_t to actions a_t. Once the action is decided and executed, the agent levies the next state s_{t+1} and a scalar reward r_t. This goes on until the agent reaches a terminal state. The expected cumulative discounted return $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the sum of accumulated returns, where at each time step, future returns are discounted with the discount factor $\gamma \in (0, 1]$. At time t, a rational agent seeks to maximise its expected return given his current state s_t.

We traditionally define

- the value of state s under policy π is defined as $V^\pi(s) = \mathbb{E}[R_t | s_t = s]$ and is simply the expected return for following policy π from state s ((Watkins 1989)).

- the action value function under policy π $Q^{\pi}(s,a) = \mathbb{E}\left[R_t | s_t = s, a\right]$ is defined as the expected return for selecting action a in state s and following policy π ((Williams 1992)).

Both the optimal value function $Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$ and the optimal value of state $V^*(s) = \max_{\pi} V^{\pi}(s)$ satisfy Bellman equations.

Whenever states and actions are too large, we are forced to represent the action value function with a function approximator, such as a neural network. Denoting the parameters θ, the state action function writes $Q(s,a;\theta)$

The updates to θ can be derived from a variety of reinforcement learning algorithms. In particular, in value-based methods, policy-based model-free methods directly parameterize the policy $\pi(a|s;\theta)$ and update the parameters θ by performing, typically approximate, gradient ascent on $\mathbb{E}[R_t]$.

An illustration of such a method is REINFORCE due to (Williams 1992). Standard REINFORCE updates the policy parameters θ in the direction $R_t \nabla_\theta \log \pi(a_t|s_t;\theta)$, which is an unbiased estimate of $\nabla_\theta \mathbb{E}[R_t]$. It is possible to reduce the variance of this estimate while keeping it unbiased by subtracting a learned function of the state $b_t(s_t)$, known as a baseline (Williams 1992), from the return. The resulting gradient is $\nabla_\theta \log \pi(a_t|s_t;\theta)\left(R_t - b_t(s_t)\right)$. This approach can be viewed as an actor-critic architecture where the policy π is the actor and the baseline b_t is the critic (Sutton & Barto 2018, Degris et al. 2012).

To understand the link between supervised and reinforcement learning, it is worth putting side by side the various terms in Reinforce as already presented in this thesis, actor critic and supervised learning as in the table 6.1

Term	RL (REINFORCE)	RL (Actor Critic)	SL
true label	expected future rewards: $R(\tau)$	expected advantage: $A(s,a) = Q(s,a) - V(s)$	label: Y
log term	log of policy: $\log \pi(a_t \mid s_t)$	log of policy: $\log \pi(a_t \mid s_t)$	cross entropy term: $\log(\hat{Y})$
cross entropy	Monte Carlo expectation: $\frac{\sum_{i=1}^{N}\sum_{t=1}^{T} R_i(\tau)\log\pi(a_{i,t}\|s_{i,t})}{N}$	Monte Carlo expectation: $\frac{\sum_{i=1}^{N}\sum_{t=1}^{T} A(s_{i,t},a_{i,t})\log\pi(a_{i,t}\|s_{i,t})}{N}$	Monte Carlo expectation: $\frac{\sum_{i=1}^{N}\sum_{t=1}^{T} Y_{i,t}\log\hat{Y}_{i,t}}{N}$

Table 6.1: Comparing SL and RL for REINFORCE and A2C methods

6.3.3 Mathematical proof

We will now be able to state a proposition to make the connection between supervised learning (SL) and reinforcement learning (RL) more rigorously.

Proposition 6.1. *Under the assumptions that we are only looking at deterministic policies meaning that the mapping between state to action is a determinitic function $\pi(x_t)$, and that these actions $a_t = \pi(x_t)$ do not influence future states $x_{t+1,\dots}$, we can prove that doing a cross-entropy loss in SL with label equal to the optimal future cumulative reward leads to the same results are the REINFORCE and A2C policy gradient.*

Proof. Given the assumptions, the future cumulative reward denoted by R_t is a function of the current action $\pi(x_t)$, therefore equal to some function $f : x_t \mapsto f(\pi(x_t))$, plus some extra terms that do not depend on the current action $\pi(x_t)$ and can be written as a constant K. This effectively means that if we define a new function $g : x_t \mapsto f(\pi(x_t)) + K$, we can construct a function g that maps states x to future cumulative reward. The same applies to the expected advantage $A(s, a)$ in A2C.

Hence we can do the following:

1. Map the true label in SL, Y, to the expected future rewards $R(\tau)$ in REINFORCE and the expected advantage $A(s, a)$ in A2C:

$$Y \mapsto \begin{cases} R(\tau) & \text{for REINFORCE} \\ A(s,a) & \text{for A2C} \end{cases}. \tag{6.1}$$

2. Map the cross entropy term to the logarithm of the policy in REINFORCE and A2C:

$$\log(\hat{Y}) \mapsto \begin{cases} \log \pi(a_t \mid s_t) & \text{for REINFORCE} \\ \log \pi(a_t \mid s_t) & \text{for A2C} \end{cases}. \tag{6.2}$$

Using these mappings, the cross-entropy loss in SL maps to

$$\mathcal{L}_{SL} = \frac{\sum_{i=1}^{N} Y_i \log \hat{Y}_i}{N} \mapsto \begin{cases} \dfrac{\sum_{i=1}^{N} \sum_{t=1}^{T} R_i(\tau) \log \pi(a_{i,t} \mid s_{i,t})}{N} & \text{for REINFORCE} \\ \dfrac{\sum_{i=1}^{N} \sum_{t=1}^{T} A(s_{i,t}, a_{i,t}) \log \pi(a_{i,t} \mid s_{i,t})}{N} & \text{for A2C} \end{cases} \tag{6.3}$$

This shows that the cross-entropy loss in SL is indeed equal to the objective functions in REINFORCE and A2C under the given assumptions and mappings, which concludes the proof. □

6.3.4 Connection with Kullback Leibler divergence

Last but not least, recall that there is a connection between cross entropy and Kullback Leibler divergence. Recall that the cross entropy for the distributions p and q over a given set is defined as follows:

$$H(p, q) = \mathbb{E}_p[-\log q]. \tag{6.4}$$

There is a straightforward connection to the Kullback–Leibler divergence $D_{\mathrm{KL}}(p\|q)$ of q from p, sometimes referred to as the relative entropy of p with respect to q given by

$$H(p, q) = H(p) + D_{\mathrm{KL}}(p\|q). \tag{6.5}$$

where $H(p)$ is the entropy of p. As the entropy term $H(p)$ is constant given the true distribution p, minimising the cross entropy or the Kullback-Leibler divergence is equivalent.

6.3.5 Kullback Leibler divergence implications

We are now in the position to establish the connection between cross-entropy and Kullback-Leibler (KL) divergence in the context of policy gradient methods (PGMs) and the relationship between supervised learning (SL) and reinforcement learning (RL).

Proposition 6.2. *Adding an entropy regularization term to the objective function in policy gradient methods (PGMs) modifies the optimization problem, making it equivalent to minimizing a modified Kullback-Leibler divergence. The connection between SL and RL is preserved when the influence of actions on the environment does not take place in future states.*

Proof. Recall the relationship between cross-entropy, $H(p, q)$, and Kullback-Leibler divergence, $D_{\mathrm{KL}}(p\|q)$:

$$H(p, q) = H(p) + D_{\mathrm{KL}}(p\|q). \tag{6.6}$$

Minimizing cross-entropy with respect to q is equivalent to minimizing the KL divergence since the entropy term $H(p)$ is constant given the true distribution p. Now consider a policy gradient method (PGM) that minimizes cross-entropy as previously shown when proving the connection between SL and RL. The PGM can be cast as a cross-entropy minimization problem:

$$\min_\theta H(Y, \pi_\theta) = \min_\theta \mathbb{E}_p[-\log \pi_\theta]. \tag{6.7}$$

If we add an entropy regularization term to the objective function with regularization coefficient λ, the problem becomes:

$$\min_\theta \left(H(Y, \pi_\theta) - \lambda H(\pi_\theta)\right) = \min_\theta \left(\mathbb{E}_p[-\log \pi_\theta] - \lambda H(\pi_\theta)\right). \tag{6.8}$$

We can rewrite the modified objective function using the relationship between cross-entropy and KL divergence:

$$\min_\theta \left(H(Y, \pi_\theta) - \lambda H(\pi_\theta)\right) = \min_\theta \left(H(Y) + D_{\mathrm{KL}}(Y\|\pi_\theta) - \lambda H(\pi_\theta)\right). \tag{6.9}$$

Since $H(Y)$ is a constant, the modified objective function can be interpreted as minimizing a modified KL divergence:

$$\min_\theta \left(D_{\mathrm{KL}}(Y\|\pi_\theta) - \lambda H(\pi_\theta)\right). \tag{6.10}$$

The connection between SL and RL holds as long as the influence of actions on the environment does not happen in future state. In both SL and RL, minimizing cross-entropy or KL divergence aims to make the predicted distribution π_θ close to the true distribution Y. The entropy regularization term helps balance exploration and exploitation in RL, while in SL, it encourages more diverse predictions.

Thus, the proposition is proven. Adding an entropy regularization term to the objective function in policy gradient methods (PGMs) modifies the optimization problem, making it equivalent to minimizing a modified Kullback-Leibler divergence. The connection between SL and RL is preserved when the influence of actions on the environment is negligible. □

6.3.6 What happens in practice?

For reinforcement learning, the connections between cross-entropy, Kullback-Leibler (KL) divergence, and policy gradient methods (PGMs) provide an insightful interpretation. As previously demonstrated, PGMs can be formulated as cross-entropy minimization problems. Given that the difference between KL divergence and cross-entropy lies in the entropy term, it is reasonable to incorporate this entropy term into the gradient descent optimization.

In an ideal scenario, the entropy term should be multiplied by one. However, in practice, as the entropy term is estimated by empirical entropy, it makes sense to multiply the entropy term by a regularization coefficient λ. This modification results in a variation of PGMs that includes an entropy regularization term, as opposed to simply computing a gradient on cross-entropy, as in the REINFORCE algorithm. This change effectively transforms the minimization problem into one that minimizes a modified KL divergence. The cross-entropy term is computed based on future expected rewards. The analogy between supervised learning (SL) and reinforcement learning (RL) holds as actions do not influence the environment. In practice the latter assumption can be alleviated by stating that actions should little to no influence on the environment.

Remark 6.3. The close connection between RL and SL is both intuitive and logical. SL can be cast as an optimization problem, solvable by gradient descent, which is implemented through stochastic gradient descent (SGD) and backpropagation in neural networks. Similarly, value estimation and value optimization in RL can be formulated as optimization problems, solvable by gradient descent through TD-learning or Q-learning. Furthermore, policy optimization can be solved by gradient descent using policy gradient methods. The primary difference between the two learning approaches lies in the loss function being optimized.

6.4 Summary

This chapter explores the close relationship between Supervised Learning (SL) and Reinforcement Learning (RL). Specifically, we demonstrate that the vanilla Policy gradient method in RL can be viewed as cross-entropy minimization supervised learning problem, in which the true labels are the expected optimal future reward or advantage, while the log cross entropy term is the log policy term. The strong assumption to rigorously prove the analogy between SL and RL is in addition that action do not influence future states, which is the case of our deep reinforcement learning problem for portfolio allocation. Despite the fact that it is not possible to determine the optimal rewards in advance and hence the corresponding labels, the connection between the two learning methods is still a significant and meaningful one as it offers an original understanding of the fundamental principles behind these two methods.

This work has been published as a conference paper (Benhamou & Saltiel 2020), highlighting its significance in the field of RL research. This chapter concludes the thesis.

PART III

Conclusions & open problems

Conclusion and Future Works

This thesis focuses on exploring the potential of Deep Reinforcement Learning (DRL) for portfolio allocation, with the aim of providing a more efficient solution than traditional portfolio methods. We emphasize the following points regarding the benefits of DRL for portfolio allocation:

- **The addition of contextual information**: The inclusion of contextual information is critical in learning from noisy and rapidly changing financial environments. To this end, we propose a dual-entry network architecture that uses two sub-networks: one that receives direct observations (e.g., past prices and standard deviation) and another that receives contextual information (e.g., level of risk aversion, precursors of a future recession, corporate profits, to name a few.). Our experiments demonstrate that this dual-entry network performs better than a network that only uses price information as presented in Chapters 2, 3, or 4.

- **The forward motion procedure**: Given the non-stationary nature of time-dependent data, such as financial data, it is critical to test the stability of DRL models over time. We propose a methodology for evaluating DRL models, called "walk-forward analysis," which trains and tests the model iteratively on an extended data set. This approach can be viewed as the time-series equivalent of cross-validation and validates the effectiveness of the selected hyperparameters over time and the stability of the resulting models and is presented in more details in Chapter 2.

- **The usage of DRL with Quantitative models**: We also explore the feasibility of using a model-free DRL approach to select quantitative models, such as volatility targeting models. This nice combination of a quantitative approach to volatility targeting with a DRL selection turns out to be performing better than simple benchmarks and suggests that DRL has some capacity to adapt to changing regimes, provided it is using some contextual features as illustrated in Chapter 3.

- **One-day lag between price observation and action**: To ensure the results are consistent with reality, we assume that prices are observed at time t, but the action occurs at time $t + 1$. This lag allows for more realistic decision-making and, in turn, more accurate portfolio allocation but makes the problem more challenging.

- **Strong foundations from theoretical investigation**: When comparing the traditional approach to portfolio allocation, based on the work of Markowitz, where the aim is to find the portfolio with the minimum risk given a target expected return, we have proved that DRL is a natural and powerful extension to traditional financial methods. Indeed, traditional financial methods involve an optimization problem where the objective is to minimize the risk associated with a set of portfolio weights subject to the constraints of positive weights that sum to one and the expected return being above a certain target. However, these methods are only a one-period optimization problem. In contrast, the DRL approach generalizes these classical quantitative approaches of portfolio theory by extending the optimization problem to a multi-period optimal control problem and taking non-myopic view of present but also future rewards. More specifically the optimization problem in DRL is multi-period and naturally maps market states to portfolio allocation, making it more likely to adapt to a changing environment. Empirical evidence shows that incorporating contextual states improves the performance of the algorithm. Interestingly, the underlying optimization problem in reinforcement learning generalizes portfolio allocation methods in two ways as presented in Chapter 4. The generalisation of traditional financial methods is across multiple dimensions. The first way in which reinforcement learning generalizes portfolio allocation methods is that it is a multi-period optimization problem. In traditional portfolio allocation methods, weights are usually determined for a single period, such as the last year, two years, five years and so on. However, in deep RL, the optimization problem involves results at each time step, making it a multi-period optimization problem. This allows the agent to adapt to a changing environment and make decisions based on past experiences. Moreover, this approach can easily incorporate day lags, which are commonly used by market practitioners to account for the delay between the decision, action, and portfolio returns. The second way in which reinforcement learning generalizes traditional portfolio allocation methods is that it optimizes a function of the policy rather than simple weights. In traditional financial methods, we optimize weights that determine the allocation of assets in the portfolio. However, in deep RL, we optimize a function that maps the current state of the environment to an action to be taken by the agent. This function is usually represented by a deep neural network with millions of parameters. Although it is a more complex optimization problem than the traditional portfolio allocation methods, it can lead to more accurate results by capturing complex patterns and relationships between the variables. The third way in which reinforcement learning generalizes traditional portfolio allocation methods is that the reward is somehow forward-looking meaning that the objective is not an immediate myopic reward but rather the sum of all the future discounted rewards.

- **DRL methods are efficient in variance reduction**: In Chapter 5, we investigate the variance reduction properties of DRL methods and focus on the critical actor method, which we interpret as the solution to a Monte Carlo simulation problem. The estimation of the expectation terms is crucial in actor-critic methods, as they are used to compute the policy gradient. However, the estimation of these expectations may not be accurate, leading to large variability in the gradient update and slow convergence of the gradient policy method. To address this issue, we explore variance reduction techniques, which involve finding a modified expression for the policy gradient that has the same expected value but a smaller variance. This approach aims to produce a less noisy gradient, leading to more stable learning and faster convergence towards the optimal policy.

- **Last but not least, under extreme assumptions, DRL is indeed supervised learning**: In

Chapter 6, we establish a connection between gradient descent reinforcement learning and supervised learning methods. Specifically, we show that the gradient descent reinforcement learning method can be reformulated as a supervised learning method with a cross-entropy loss function and labels equal to the optimal rewards. Although this result is theoretical, as it is impossible to know the optimal rewards in advance, it provides insight into the underlying similarities between the two learning methods.

While this thesis has explored the potential of Deep Reinforcement Learning (DRL) for portfolio allocation and highlighted its benefits, there are still many important areas that warrant further research and development. These areas include :

- **Generalization to real-world financial markets**: The experiments conducted in this thesis provide valuable insights into the performance of DRL algorithms for portfolio allocation. However, further research is needed to evaluate their effectiveness in real-world financial markets, considering factors such as transaction costs, market impact, and liquidity constraints. Real-world implementation also requires addressing practical challenges, such as data availability, model interpretability, and scalability.

- **Risk management and concentration risk mitigation**: Indeed, concentration risk is a significant challenge in portfolio allocation, where a portfolio is heavily invested in a small number of assets, making it vulnerable to large losses if any of those assets perform poorly. To mitigate this risk, it is crucial to develop robust risk management techniques within DRL-based portfolio allocation to mitigate concentration risk. This can involve incorporating risk measures, diversification constraints, and dynamic portfolio rebalancing strategies to ensure the stability and resilience of portfolio allocations. Finding the right reward function that considers both risk and return is an ongoing research area, and addressing concentration risk in DRL algorithms is essential for stable and diversified portfolio allocations, especially in the context of selecting hedging strategies.

- **Interpretability and explainability**: DRL algorithms often operate as black boxes, making it challenging to interpret the decisions and actions taken by the models. Enhancing the interpretability and explainability of DRL-based portfolio allocation methods is crucial for practitioners and investors to trust and understand the reasoning behind the suggested portfolio allocations. Research efforts should focus on developing techniques to provide transparent explanations for DRL-based decisions, such as feature importance analysis, model visualization, and rule extraction methods.

- **Market regime detection and adaptation**: Financial markets exhibit different regimes over time, characterized by varying market conditions and dynamics. Adapting DRL-based portfolio allocation strategies to different market regimes can further improve their performance and robustness. Research is needed to develop methods for detecting market regimes and automatically adjusting the DRL models or selecting appropriate models tailored to each regime.

- **Integration with external data sources**: Incorporating additional sources of data, such as news sentiment, macroeconomic indicators, and alternative data, can enhance the predictive power and adaptability of DRL-based portfolio allocation models. Integrating external data sources into the DRL framework requires addressing challenges related to data quality, feature engineering, and model integration.

- **Combining multiple DRL approaches**: Exploring the potential benefits of combining multiple DRL approaches, such as value-based and policy-based methods, can lead to more robust and effective portfolio allocation models. Hybrid DRL architectures, ensemble methods, or meta-learning techniques can be investigated to leverage the strengths of different DRL algorithms and improve their performance in portfolio allocation.

- **Potential synergies with supervised learning**: Combining reinforcement and supervised learning can potentially enhance the performance of DRL algorithms and enable them to solve more complex problems. By leveraging the power of supervised learning to guide the exploration process of the reinforcement learning algorithm, we can improve the sample efficiency and the convergence rate of the algorithm. This combination allows for a more structured learning process, with the supervised component providing valuable guidance based on labeled data, while the reinforcement learning component explores and learns from interactions with the environment. Additionally, incorporating auxiliary tasks, such as predicting market trends or macroeconomic indicators, can further enhance the performance of the algorithm by providing additional context and information.

- **Transfer learning and meta-learning**: Leveraging transfer learning and meta-learning techniques can enable the transfer of knowledge and experiences gained from one portfolio allocation problem to another, accelerating the learning process and improving performance. Investigating transfer learning and meta-learning approaches specific to portfolio allocation can lead to more efficient and adaptive DRL models.

- **Robust scenario generation**: Generating more scenarios is critical to improving the performance of DRL algorithms. While current methods rely on historical data to generate scenarios, there is a need to find more efficient and effective ways to generate new scenarios that are consistent with past experience. One promising approach is to use generative models that simulate new scenarios resembling the actual market environment. These generative models can capture the dynamics and dependencies observed in historical data and generate realistic scenarios for training and testing DRL algorithms. Further research is necessary to explore the development of generative models that can accurately capture the relevant features and characteristics of financial markets and generate scenarios that reflect their dynamics and uncertainties.

- **Feature selection in DRL**: Feature selection plays a crucial role in DRL algorithms by reducing the dimensionality of the input space and improving the model's performance. However, selecting the right set of features that are relevant and predictive of the desired outcome is a challenging task. One approach to feature selection in DRL is to use feature importance methods to rank the importance of each feature and select the most critical features for the algorithm. These methods can provide insights into which features contribute the most to the decision-making process and guide the selection of relevant information. Another approach is to employ neural networks to automatically learn the feature representation from raw data. Deep learning architectures can capture complex patterns and relationships in the data, potentially eliminating the need for explicit feature selection. Further research is required to explore the effectiveness and efficiency of different feature selection approaches in DRL algorithms, considering the specific characteristics and requirements of portfolio allocation problems.

In conclusion, this PhD thesis highlights the significant potential of Deep Reinforcement Learning

(DRL) methods for portfolio allocation. While this thesis does not claim to have answered all the questions in this field, it has laid the groundwork for further exploration and generated ongoing interest in these innovative methods within the investment community. Many important points and extensions remain to be addressed, and their resolution will contribute to the continuous development and improvement of DRL algorithms for portfolio allocation in finance. These advancements will promote the wider adoption of DRL methods in portfolio allocation and enhance their effectiveness in terms of portfolio allocation decisions, risk management, and investment outcomes for practitioners and investors. In summary, we firmly believe that DRL algorithms hold immense promise for portfolio allocation, and ongoing research will ensure their smooth integration into real-world investment practices.

7 Bibliography

Aboussalah, A. M. (2020), *What is the value of the cross-sectional approach to deep reinforcement learning?*, Available at SSRN.

Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. (2019), 'Optuna: A next-generation hyperparameter optimization framework', *arXiv preprint arXiv:1907.10902* .

Almgren, R. & Chriss, N. (2005), 'Direct estimation of equity market impact', *Journal of Risk* **3**, 57–62.

Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L. & Zaremba, W. (2020), 'Learning dexterous in-hand manipulation', *The International Journal of Robotics Research* **39**(1), 3–20.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L. & Parikh, D. (2015), Vqa: Visual question answering, *in* 'IEEE Conference on Computer Vision and Pattern Recognition', pp. 2425–2433.

Arora, S., Hazan, E. & Kale, S. (2012), 'The multiplicative weights update method: a meta-algorithm and applications', *Theory Comput.* **8**(1), 121–164.

Astrom, K. (1969), 'Optimal control of markov processes with incomplete state-information ii. the convexity of the lossfunction', *Journal of Mathematical Analysis and Applications* **26**(2), 403–406.

Babaeizadeh, M., Frosio, I., Tyree, S., Clemons, J. & Kautz, J. (2017), Reinforcement learning thorugh asynchronous advantage actor-critic on a gpu, *in* 'ICLR'.

Barth-Maron, G., Hoffman, M. W., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N. & Lillicrap, T. (2018), Distributional policy gradients, *in* 'International Conference on Learning Representations'.

Barto, A. G. & Dietterich, T. G. (2004), *Reinforcement Learning and Its Relationship to Supervised Learning*, Wiley-IEEE Press, pp. 45–63.

Barto, A., Sutton, R. & Anderson, C. (1983), 'Neuronlike adaptive elements that can solve difficult learning control problems', *IEEE Transactions on Systems, Man, and Cybernetics* **5**, 135.

Beji, C., Benhamou, E., Bon, M., Yger, F. & Atifl, J. (2021), Estimating individual treatment effects through causal populations identification, *in* 'ESANN 2021'.

Bellman, R. (1957*a*), *Dynamic Programming*, 1 edn, Princeton University Press, Princeton, NJ, USA.

Bellman, R. (1957*b*), 'A Markovian decision process', *Journal of Mathematics and Mechanics:* **679**.

Bellman, R. (1968), *Some Vistas of Modern Mathematics: Dynamic Programming, Invariant Imbedding, and the Mathematical Biosciences*, University Press of Kentucky.

Bellman, R. & Kalaba, R. (1957), 'Dynamic programming and statistical communication theory', *National Academy of Sciences of the United States of America* **43**, 8.

Benhamou, E. & Guez, B. (2018), 'Incremental sharpe and other performance ratios', *Journal of Statistical and Econometric Methods* .

Benhamou, E., Guez, B., Ohana, J.-J., Saltiel, D., Laraki, R. & Atif, J. (2023), Comparing deep rl and traditional financial portfolio methods, *in* 'ECML PKDD 2023: MIDAS workshop'. **URL:** *http://midas.portici.enea.it/*

Benhamou, E. & Saltiel, D. (2020), Similarities between policy gradient methods in reinforcement and supervised learning, *in* 'ESANN proceedings'.

Benhamou, E., Saltiel, D., Guez, B., Atif, J. & Laraki, R. (2021), From forecast to decisions in graphical models: a natural gradient optimization approach, *in* 'DIMACS 2021: From forecast to decisions workshop'.

Benhamou, E., Saltiel, D., Guez, B. & Paris, N. (2019), 'Testing sharpe ratio: luck or skill?', *ArXiv* .

Benhamou, E., Saltiel, D., Ohana, J.-J. & Atif, J. (2021), Detecting and adapting to crisis pattern with context based deep reinforcement learning, *in* 'ICPR 2021 proceedings'.

Benhamou, E., Saltiel, D., Ohana, J. J., Atif, J. & Laraki, R. (2021), Deep reinforcement learning (drl) for portfolio allocation, *in* Y. Dong, G. Ifrim, D. Mladenić, C. Saunders & S. Van Hoecke, eds, 'ECML PKDD - Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track', Springer International Publishing, Cham, pp. 527–531.

Benhamou, E., Saltiel, D., Ohana, J. J., Laraki, R. & Atif, J. (2020), Drlps: Deep reinforcement learning for portfolio selection, *in* 'ECML PKDD'.

Benhamou, E., Saltiel, D., Tabachnik, S., Bourdeix, C. & Chareyron, F. (2021), Adaptive supervised learning for financial markets volatility targeting models, *in* 'MIDAS: best paper award'.

Benhamou, E., Saltiel, D., Tabachnik, S., Wong, S. K. & Chareyron, F. (2021), Adaptive learning for financial markets mixing model-based and model-free rl for volatility targeting, *in* 'AAAMAS: ALA', AAAI Press.

Benhamou, E., Saltiel, D., Ungari, S. & Abhishek Mukhopadhyay, Jamal Atif, R. L. (2021), Knowledge discovery with deep rl for selecting financial hedges, *in* 'AAAI: KDF', AAAI Press.

Benhamou, E., Saltiel, D., Ungari, S. & Mukhopadhyay, A. (2020*a*), Bridging the gap between markowitz planning and deep reinforcement learning, *in* 'ICAPS proceedings'.

Benhamou, E., Saltiel, D., Ungari, S. & Mukhopadhyay, A. (2020*b*), Time your hedge with deep reinforcement learning, *in* 'ICAPS proceedings'.

Benhamou, E., Saltiel, D. & Verel, S. (2020), Bayesian CMA-ES: a new approach, *in* 'GECCO proceedings'.

Bernstein, P. L. & Sharpe, M. L. (1990), 'Minimum-variance portfolio allocation with transaction costs', *Journal of Financial and Quantitative Analysis* **25**(2).

Bertsekas, D. (1997), 'Nonlinear programming', *Journal of the Operational Research Society* **48**(3), 334–334.

Bishop, C. M. (2007), *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer.

Black, F. & Litterman, R. (1992), *Global portfolio optimization*, Financial Analysts.

Bollerslev, T. (1986), 'Generalized autoregressive conditional heteroskedasticity', *Journal of econometrics* **31**(3), 307–327.

Bottou, L., Curtis, F. & Nocedal, J. (2018), 'Optimization methods for large-scale machine learning', *Siam Review* **60**(2), 223–311.

Boucheron, S., Bousquet, O. & Lugosi, G. (2005), 'Theory of classification: A survey of some recent advances', *ESAIM: Probability and Statistics* **9**, 323.

Brown, N., Bakhtin, A., Lerer, A. & Gong, Q. (2020), Combining deep reinforcement learning and search for imperfect-information games, *in* 'Proceedings of the 34th International Conference on Neural Information Processing Systems', NIPS'20, Curran Associates Inc., Red Hook, NY, USA.

Capponi, A., Olafsson, S. & Zariphopoulou, T. (2021), *Personalized robo-advising: Enhancing investment through client interaction*, Management Science.

Challet, D. (2017), 'Sharper asset ranking with total drawdown duration', *Applied Mathematical Finance* pp. 1–22.

Chan, J. (2013), 'Moving average stochastic volatility models with application to inflation forecast', *Journal of Econometrics* **176**(2), 162–172.

Chebotar, Y., Hausman, K., Zhang, M., Sukhatme, G., Schaal, S. & Levine, S. (2017), Combining model-based and model-free updates for trajectory-centric reinforcement learning, *in* D. Precup & Y. W. Teh, eds, 'PMLR', Vol. 70 of *Proceedings of Machine Learning Research*, PMLR, International Convention Centre, Sydney, Australia, pp. 703–711.

Chopra, V. K. & Ziemba, W. T. (1993), 'The effect of errors in means, variances, and covariances on optimal portfolio choice', *Journal of Portfolio Management* **19**(2), 6–11.

Choueifaty, Y. & Coignard, Y. (2008), 'Toward maximum diversification', *Journal of Portfolio Management* **35**(1), 40–51.

Choueifaty, Y., Froidure, T. & Reynier, J. (2012), 'Properties of the most diversified portfolio', *Journal of Investment Strategies* **2**(2), 49–70.

Christoffersen, P., Errunza, V., Jacobs, K. & Jin, X. (2010), Is the potential for international diversification disappearing? Working Paper.

Chung, W., Thomas, V., Machado, M. C. & Roux, N. L. (2020), 'Beyond variance reduction: Understanding the true impact of baselines on policy optimization', *arXiv preprint arXiv:2008.13773* .

Cogneau, P. & Hübner, G. (2009), 'The 101 ways to measure portfolio performance', *SSRN Electronic Journal* .

Cong, L. W., Tang, K., Wang, J. & Zhang, Y. (2021), 'Alphaportfolio: Direct construction through deep reinforcement learning and interpretable ai', *SSRN Electronic Journal* **10**, 2139.

Congreve, W. (1755), *Love for Love*, G. Hamilton & J. Balfour.

Corsi, F. (2009), 'A simple approximate long-memory model of realized volatility', *The Journal of Financial Econometrics* **7**(2), 174–196.

Cybenko, G. (1989), 'Approximation by superpositions of a sigmoidal function', *Mathematics of Control, Signals, and Systems* **2**(4), 303–314.

de Vries, H., Strub, F., Chandar, S., Pietquin, O., Larochelle, H. & Courville, A. (2017), Guesswhat visual object discovery through multi-modal dialogue, *in* 'IEEE Conference on Computer Vision and Pattern Recognition', pp. 5503–5512.

Degris, T., Pilarski, P. M. & Sutton, R. S. (2012), 'Model-free reinforcement learning with continuous action in practice', *IEEE In ACC* **2012**, 2177–2182.

Deng, Y., Bao, F., Kong, Y., Ren, Z. & Dai, Q. (2016), 'Deep direct reinforcement learning for financial signal representation and trading', *IEEE Transactions on Neural Networks and Learning Systems* **28**, 1–12.

Dimeas, A. & Hatziargyriou, N. (2007), 'Agent based control for microgrids', *In IEEE Power Engineering Society General Meeting, pp* pp. 1–5.

Dixon, M. & Halperin, I. (2020), G-learner and girl: Goal based wealth management with reinforcement learning, preprint, arXiv.

Dreyer, A. & Hubrich, S. (2017), 'Tail risk mitigation with managed volatility strategies'.

Du, J., Jin, M., Kolm, P. N., Ritter, G., Wang, Y. & Zhang, B. (2020), 'Deep reinforcement learning for option replication and hedging', *The Journal of Financial Data Science* **2**, 44–57.

Du, X., Zhai, J. & Lv, K. (2016), 'Algorithm trading using Q-learning and recurrent reinforcement learning', *Positions* **1**.

Erhan, D., Bengio, Y., Courville, A. & Vincent, P. (2009), 'Visualizing higher-layer features of a deep network', *University of Montreal* **1341**, 3.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S. & Kavukcuoglu, K. (2018), IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures, *in* J. Dy & A. Krause, eds, 'Proceedings of the 35th International Conference on Machine Learning', Vol. 80 of *Proceedings of Machine Learning Research*, PMLR, Stockholmsmässan, Stockholm Sweden, pp. 1407–1416.

Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E. & Levine, S. (2018), 'Model-based value estimation for efficient model-free reinforcement learning'.

François-Lavet, V., Taralla, D., Ernst, D. & Fonteneau, R. (2016), Deep reinforcement learning solutions for energy microgrids management, *in* 'Proceedings of the 33nd International Conference on Machine Learning', Springer, New York, pp. 19–24.

Freund, Y. & Schapire, R. E. (1997), 'A decision-theoretic generalization of on-line learning and an application to boosting', *J. Comput. Syst. Sci* **55**(1), 119–139.

Fukushima, K. (1980), 'Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position', *Biological cybernetics* **36**(4), 193–202.

Gandhi, D., Pinto, L. & Gupta, A. (2017), Learning to fly by crashing. preprint.

Glasserman, P. (2004), *Monte Carlo methods in financial engineering*, Springer.

Glosten, L. R., Jagannathan, R. & Runkle, D. E. (1993), 'On the relation between the expected value and the volatility of the nominal excess return on stocks', *Journal of Finance* **48**(5), 1779–1801.

Grathwohl, W., Choi, D., Wu, Y., Roeder, G. & Duvenaud, D. (2018), Backpropagation through the void: Optimizing control variates for black-box gradient estimation, *in* 'International Conference on Learning Representations'.

Greensmith, E., Bartlett, P. & Baxter, J. (2004), 'Variance reduction techniques for gradient estimates in reinforcement learning', *Journal of Machine Learning Research* **5**(Nov), 1471–1530.

Grondman, I., Busoniu, L., Lopes, G. A. D. & Babuska, R. (2012), 'A survey of actor-critic reinforcement learning: Standard and natural policy gradients', *Trans. Sys. Man Cyber Part C* **42**(6), 1291–1307.

Grzes, M. & Kudenko, D. (2009), 'Learning shaping rewards in model-based reinforcement learning', *In AAMAS* **2009**.

Guéant, O. & Manziuk, I. (2019), 'Deep reinforcement learning for market making in corporate bonds: beating the curse of dimensionality', *Applied Mathematical Finance* **26**, 387–452.

Haarnoja, T., Zhou, A., Abbeel, P. & (Oct, S. L. (2018), Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, *in* 'Proceedings of the 35th International Conference on Machine Learning', by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1861–1870.

Hambly, B., Xu, R. & Yang, H. (2021), 'Recent advances in reinforcement learning in finance', *CoRR, abs/* **2112**, 04553.

Hammersley, J. M., D. C. H. (1964), *Monte Carlo Methods*, John Wiley & Sons, New York.

Hastie, T., Tibshirani, R. & Friedman, J. H. (2009), *The elements of statistical learning: data mining, inference, and prediction, 2nd Edition*, Springer series in statistics, Springer.

Haugen, R. & Baker, N. (1991), 'The efficient market inefficiency of capitalization-weighted stock portfolios', *Journal of Portfolio Management* **17**, 35–40.

Heess, N., Tirumala, D., Sriram, S., Lemmon, J., Merel, J., GregWayne, Y. T., Erez, T., Wang, Z., Eslami, A., Riedmiller, M. & Silver, D. (2017), Emergence of locomotion behaviours in rich environments. arxiv. preprint.

Helmbold, D. P., Schapire, R. E., Singer, Y. & Warmuth, M. K. (1996), On-line portfolio selection using multiplicative updates, *in* 'Lorenza Saitta', itor, Proc. of ICML, pp. 243–251.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Sendonaris, A., Dulac-Arnold, G., Osband, I., Agapiou, J., Leibo, J. Z. & Gruslys, A. (2017), 'Learning from demonstrations for real world reinforcement learning', *CoRR* **abs/1704.03732**.

Hochreiter, S., Younger, A. S. & Conwell, P. R. (2001), Learning to learn using gradient descent, *in* 'International Conference on Artificial Neural Networks', Springer, 87–94.

Hocquard, A., Ng, S. & Papageorgiou, N. (2013), 'A constant-volatility framework for managing tail risk', *The Journal of Portfolio Management* **39**, 28–40.

Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., van Hasselt, H. & Silver, D. (2018), Distributed prioritized experience replay, *in* 'International Conference on Learning Representations'.

Hornik, K. (1991), 'Approximation capabilities of multilayer feedforward networks', *Neural Networks* **4**(2), 251–257.

Hyndman, R. J. & Athanasopoulos, G. (2018), *Forecasting: principles and practice, 2nd edition*, Vol. 7, OTexts.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D. & Kavukcuoglu, K. (2016), 'Reinforcement learning with unsupervised auxiliary tasks.', *CoRR* **abs/1611.05397**.

Janner, M., Fu, J., Zhang, M. & Levine, S. (2019), 'When to trust your model: Model-based policy optimization'.

Jerome, J., Sanchez-Betancourt, L., Savani, R. & Herdegen, M. (2022), Model-based gym environments for limit order book trading. To be presented at the Benchmarks for AI in Finance Workshop at ICAIF'22.

Jiang, Z. & Liang, J. (2016), 'Cryptocurrency Portfolio Management with Deep Reinforcement Learning'.

Jiang, Z., Xu, D. & Liang, J. (2017), 'A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem', *arXiv e-prints* .

Kahneman, D. (2011), *Thinking, Fast and Slow*, Farrar, Straus and Giroux, New York.

Kakade, S. (2002), 'A natural policy gradient', *In Advances in Neural Information Processing Systems, pp* pp. 1531–1538.

Kelley, H. (1960), 'Gradient theory of optimal flight paths', *Ars Journal* **30**(10), 947–954.

Kingma, D. P. & Ba, J. (2015), Adam: A method for stochastic optimization, *in* '3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings'.

Kober, J., Bagnell, A. & Peters, J. (2013), 'Reinforcement learning in robotics: A survey', *The International Journal of Robotics Research* **32**(11), 1238–1274.

Konda, V. R. & Tsitsiklis, J. N. (2000), 'Actor-critic algorithms', *in Advances in Neural Information Processing Systems pp* pp. 1008–1014.

Konda, V. R. & Tsitsiklis, J. N. (2003), 'On actor-critic algorithms', *SIAM J. Control Optim.* **42**(4), 1143–1166.

Krief, R., Benhamou, E., Guez, B., Ohana, J.-J., Saltiel, D., Laraki, R. & Atif, J. (2023), Fsda: Tackling tail-event analysis in imbalanced time series data with feature selection and data augmentation, *in* 'ECML PKDD 2023: LIDTA workshop'.

Kritzman, M. (2014), 'Six practical comments about asset allocation', *Practical Applications* **1**(3), 6–11.

Le Guin, U. (1969), *The Left Hand of Darkness*, Ace Books.

LeCun, Y., Bengio, Y. et al. (1995), 'Convolutional networks for images, speech, and time series', *The handbook of brain theory and neural networks* **3361**, 10.

Levent, T., Preux, P., le Pennec, E., Badosa, J., Henri, G. & Bonnassieux, Y. (2019), Energy management for microgrids: a reinforcement learning approach, *in* '2019 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe)', pp. 1–5.

Levine, S., Finn, C. & Darrell, T. (2015), 'End-to-end training of deep visuomotor policies', *Journal of Machine Learning Research* **17**.

Levine, S., Pastor, P., Krizhevsky, A. & Quillen, D. (2016), 'Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection', *The International Journal of Robotics Research* **17**(1).

Liang, Z., Chen, H., Zhu, J., Jiang, K. & Li, Y. (2018), Adversarial deep reinforcement learning in portfolio management, preprint, arXiv.

Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. & Wierstra, D. (2015), 'Continuous control with deep reinforcement learning', *CoRR* .

Liu, H., Feng, Y., Mao, Y., Zhou, D., Peng, J. & Liu, Q. (2018), Action-dependent control variates for policy optimization via stein identity, *in* 'International Conference on Learning Representations'.

Lobo, M., Fazel, M. & Boyd, S. (2007), 'Portfolio optimization with linear and fixed transaction costs', *Annals of Operations Research, special issue on financial optimization* **152(1)**, 341–365.

Lopez de Prado, M. (2016), 'Building diversified portfolios that outperform out of sample:', *The Journal of Portfolio Management* **42**, 59–69.

Lopez de Prado, M. (2018), *Advances in Financial Machine Learning*, Wiley.

Lopez de Prado, M. (2020), *Machine Learning for Asset Managers*, Elements in Quantitative Finance, Cambridge University Press.

Lu, Y. K. & Perron, P. (2010), 'Modeling and forecasting stock return volatility using a random level shift model', *Journal of Empirical Finance* **17**(1), 138–156.

Ma, C., Zhang, J., Li, Z. & Xu, S. (2023), 'Multi-agent deep reinforcement learning algorithm with trend consistency regularization for portfolio management', *Neural Computing and Applications* **35**(18), 6589–6601.

Maillard, S., Roncalli, T. & Teïletche, J. (2010), 'The properties of equally weighted risk contribution portfolios', *The Journal of Portfolio Management* **36**(4), 60–70.

Mannion, P., Duggan, J. & Howley, E. (2016), *An Experimental Review of Reinforcement Learning Algorithms for Adaptive Traffic Signal Control*, Springer, pp. 47–66.

Mannion, P., Mason, K., Devlin, S., Duggan, J. & Howley, E. (2016), Multi-objective dynamic dispatch optimisation using multi-agent reinforcement learning, *in* 'AAMAS proceedings'.

Mao, H., Alizadeh, M., Menache, I. & Kandula, S. (2016), 'Resource management with deep reinforcement learning', *In ACM Workshop on Hot Topics in Networks, pp* pp. 50–56.

Mao, W., Zhang, K., Zhu, R., Simchi-Levi, D. & Başar, T. (2020), Model-free non-stationary rl: Near-optimal regret and applications in multi-agent rl and inventory control, preprint, arXiv.

Markowitz, H. (1952), 'Portfolio selection', *The Journal of Finance* **7**(1), 77–91.

Mason, K., Mannion, P., Duggan, J. & Howley, E. (2016), Applying multi-agent reinforcement learning to watershed management, *in* 'AAMAS Proceedings'.

Maxwell, J. (1873), *A Treatise on Electricity and Magnetism*, number vol. 1 *in* 'A Treatise on Electricity and Magnetism', Clarendon Press.

Minksy, M. (1970), 'Acm turing lecturet', *Journal of the Association for Computing Machinery* **17**.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. & Kavukcuoglu, K. (2016), Asynchronous methods for deep reinforcement learning, *in* 'International Conference on Machine Learning', PMLR, pp. 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. & Riedmiller, M. (2013), 'Playing atari with deep reinforcement learning', *NIPS Deep Learning Workshop* .

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. & Hassabis, D. (2015), 'Human-level control through deep reinforcement learning', *Nature* **518**, 529–33.

Nachum, O., Norouzi, M., Xu, K. & Schuurmans, D. (2017), Bridging the gap between value and policy based reinforcement learning, *in* 'NIPS', Curran Associates, Inc., pp. 2775–2785.

Nelson, B. (1990), Control variate remedies, Technical report, Operations Research 38.6, pp. 974–992. Neu, Gergely, Anders Jonsson, andVicenç Gómez (2017).Aunified viewof entropy-regularized markov decision processes. arXiv preprint.

Nevmyvaka, Y., Feng, Y. & Kearns, M. (2006), Reinforcement learning for optimized trade execution, *in* 'Proc. of ICML', pp. 673–680.

Nocedal, J. & Wright, S. (2006), *Numerical optimization*, Springer, Science & Business Media.

Noureldin, D. & Shephard, N. (2012), 'Multivariate high-frequency-based volatility (heavy) models'.

O'Donoghue, B., Munos, R., Kavukcuoglu, K. & Mnih, V. (2016), 'PGQ: combining policy gradient and q-learning', *CoRR* **abs/1611.01626**.

Ohana, J.-J., Ohana, S., Benhamou, E., Saltiel, D. & Guez, B. (2021), Explainable ai (xai) models applied to the multi agents environment of financial markets, *in* 'AAAMAS: EXTRAAMAS', AAAI Press.

Olah, C., Mordvintsev, A. & Schubert, L. (2017), *Feature Visualization*, Distill.

Pardo, R. E. (1992), *Design, Testing, and Optimization of Trading Systems*, Wiley, Trader's Advantage. 17.

Park, H., Sim, M. K. & Choi, D. G. (2020), 'An intelligent financial portfolio trading strategy using deep Q-learning', *Expert Systems with Applications* **158**.

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

Perchet, R., Leote de Carvalho, R., Heckel, T. & Moulin, P. (2016), 'Predicting the success of volatility targeting strategies: Application to equities and other asset classes'.

Peters, J. & Schaal, S. (2006), Policy gradient methods for robotics, *in* '2006 IEEE/RSJ International Conference on Intelligent Robots and Systems', IEEE, pp. 2219–2225.

Peters, J. & Schaal, S. (2008), 'Natural actor-critic', *Neurocomputing* **71**(7-9), 1180–1190.

Pong, V., Gu, S., Dalal, M. & Levine, S. (2018), 'Temporal difference models: Model-free deep RL for model-based control'.

Pontryagin, L., Boltyanskii, V., Gamkrelidze, R. & Mishchenko, E. (1962), *The mathematical theory of optimal processes*, John Wiley & Sons, Inc.

Robbins, H. & Monro, S. (1951), 'A stochastic approximation method', *The Annals of Mathematical Statistics, pp* pp. 400–407.

Rogers, L. & Satchell, S. (1991), 'Estimating variance from high, low and closing prices', *Annals of Applied Probability* **1**, 504–512.

Roncalli, T. & Weisang, G. (2016), 'Risk parity portfolios with risk factors', *Quantitative Finance* **16**(3), 377–388.

Ross, S. (2002), *Simulation*, Academic Press.

Ross, S., Gordon, G. J. & Bagnell, D. (2011), 'A reduction of imitation learning and structured prediction to no-regret online learning', *In: AISTATS, JMLR.org, JMLR Proceedings* **15**, 627–635.

Rubinstein, R. Y. & Marcus, R. (1985), 'Efficiency of multivariate control variates in monte carlo simulation', *Operations Research* **33**(3), 661–677.

Rumelhart, D., Hinton, G. & RonaldWilliams (1985), 'Learning internal representations by error propagation', *Tech. rep* **150**.

Saltiel, D. & Benhamou, E. (2019), Trade Selection with Supervised Learning and OCA, *in* 'Risk forum'.

Schaal, S. (1996), Learning from demonstration, *in* 'NIPS, MIT Press', NIPS, MIT Press, pp. 1040–1046.

Schaefer, A., Bailey, M., Shechter, S. & Roberts, M. (2005), Modeling medical treatment using Markov decision processes, *in* 'Operations research and health care', Springer Verlag, pp. 593–612.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. & (July, P. M. (2015), Trust region policy optimization, *in* 'Proceedings of the 32nd International Conference on Machine Learning', PMLR, Lille, France, pp. 1889–1897.

Schulman, J., Moritz, P., Levine, S., Jordan, M. & Abbeel, P. (2015), 'High-dimensional continuous control using generalized advantage estimation', *ICLR* .

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. (2017), 'Proximal policy optimization algorithms', *CoRR* **abs/1704.06440**.

Shannon, C. E. (1948), 'A mathematical theory of communication', *The Bell System Technical Journal* **27**, 379–423, 623–656.

Sharpe, W. F. (1966), 'Mutual fund performance', *The Journal of Business* pp. 119–138.

Sharpe, W. F. (1975), 'Adjusting for risk in portfolio performance measurement', *Journal of Portfolio Management* pp. 29–34.

Sharpe, W. F. (1992), 'Asset allocation: Management style and performance measurement', *Journal of Portfolio Management* pp. 7–19.

Silver, D. (2015), Deep reinforcement learning, *in* 'ICLR 2015 keynote speach'.

Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T. & Hassabis, D. (2016), 'Mastering the game of go with deep neural networks and tree search', *Nature* **529**(7587), 484–489.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D. & Jun, M. R. .-. (2014), Deterministic policy gradient algorithms, *in* 'Proceedings of the 31st International Conference on Machine Learning', PMLR, Bejing, China, pp. 387–395.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T. & Hassabis, D. (2017), 'Mastering the game of go without human knowledge', *Nature* **550**, 354–.

Sortino, F. A. & Price, L. N. (1994), 'Performance measurement in a downside risk framework', *The Journal of Investing* **3**, 59–64.

Spooner, T., Fearnley, J., Savani, R. & Koukorinis, A. (2018), Market Making via Reinforcement Learning, *in* 'Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems', AAMAS '18, International Foundation for Autonomous Agents and Multiagent Systems, pp. 434–442.

Spooner, T. & Savani, R. (2020), Robust Market Making via Adversarial Reinforcement Learning, *in* 'Proc. of the 29th International Joint Conference on Artificial Intelligence, IJCAI-20', International Joint Conferences on Artificial Intelligence Organization, pp. 4590–4596.

Sun, W., Venkatraman, A., Gordon, G. J., Boots, B. & Bagnell, J. A. (2017), Deeply AggreVaTeD: Differentiable imitation learning for sequential prediction, *in* 'ICML', Vol. 70, PMLR, pp. 3309–3318.

Sutton, R. (1984), 'Temporal credit assignment in reinforcement learning. phd thesis', *University of Massachusetts Amherst* **152**.

Sutton, R., McAllester, D., Singh, S. & Mansour, Y. (2000), *Policy Gradient Methods for Reinforcement Learning with Function Approximation*, In Advances in Neural Information Processing Systems.

Sutton, R., Precup, D. & Singh, S. (1999), 'Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning', *Artificial Intelligence* **112**(1-2), 181–211.

Sutton, R. S. & Barto, A. G. (2018), *Reinforcement Learning: An Introduction (2nd Edition*, The MIT Press, The MIT Press.

Sutton, R. S., McAllester, D., Singh, S. & Mansour, Y. (1999), Policy gradient methods for reinforcement learning with function approximation, *in* 'NIPS', MIT Press, Cambridge, MA, USA, pp. 1057–1063.

Talpaert, V., Sobh, I., Kiran, B. R., Mannion, P., Yogamani, S., El-Sallab, A. & Perez, P. (2019), 'Exploring applications of deep reinforcement learning for real-world autonomous driving systems', *arXiv e-prints* .

Tesauro, G. (1992), 'Practical issues in temporal difference learning', *Machine Learning* **8**, 257–277.

Tesauro, G. (1994), 'TD-gammon, a self-teaching backgammon program, achieves master-level play', *Neural Computation* **6**(2), 215–219.

Tesla, N. (1915), *The Wonder World to Be Created by Electricity*, Manufacturer's Record.

Thorndike, E. (1911), *Animal intelligence: Experimental studies*, New York: Macmillan.

Ungari, S. & Benhamou, E. (2021), Deep reinforcement learning for portfolio allocation, *in* 'Global Quant Network'.

Ungari, S., Benhamou, E., Saltiel, D., Guez, B. & Mukhopadhyay, A. (2021), From markowitz to deep reinforcement learning for portfoio allocation, *in* 'Risk Forum 2021'.

van Hasselt, H., Hessel, M. & Aslanides, J. (2019), 'When to use parametric models in reinforcement learning?'.

Vinyals, O., Babuschkin, I., Czarnecki, W., Mathieu, M., Dudzik, A., Chung, J., Choi, D., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J., Jaderberg, M. & Silver, D. (2019), 'Grandmaster level in starcraft ii using multi-agent reinforcement learning', *Nature* **575**.

Wang, H. (2019), 'Large scale continuous-time mean-variance portfolio allocation via reinforcement learning', *Available at SSRN* **3428125**.

Wang, H. & Zhou, X. Y. (2020), 'Continuous-time mean–variance portfolio selection: A reinforcement learning framework', *Mathematical Finance* **30**(4), 1273–1308.

Wang, S., Jia, D. & Weng, X. (2018), 'Deep reinforcement learning for autonomous driving', *ArXiv* **abs/1811.11329**.

Watkins, C. J. C. H. (1989), Learning from delayed rewards, PhD thesis, University of Cambridge England.

Watkins, C. J. C. H. & Dayan, P. (1992), 'Q-learning', *Machine Learning* **8**(3), 279–292.

Weaver, L. & Tao, N. (2001), The optimal reward baseline for gradient-based reinforcement learning, *in* 'Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence', pp. 538–545.

Werbos, P. (1974), Beyond regression: Newtools for prediction and analysis in the behavioral sciences, PhD thesis, Harvard University.

Williams, R. J. (1992), *Simple statistical gradient-following algorithms for connectionist reinforcement learning*, Vol. 8, Springer.

Wu, C., Rajeswaran, A., Duan, Y., Kumar, V., Bayen, A., Kakade, S., Mordatch, I. & Abbeel, P. (2018), Variance reduction for policy gradient with actiondependent factorized baselines, *in* 'International Conference on Learning Representations'.

Xiong, Z., Liu, X.-Y., Zhong, S., Yang, H. & Walid, A. (2018), 'Practical deep reinforcement learning approach for stock trading', *arXiv preprint arXiv:1811.07522* .

Ye, Z., Deng, W., Zhou, S., Xu, Y. & Guan, J. (2020), *Optimal trade execution based on deep deterministic policy gradient*, in Database Systems for Advanced Applications, Springer International Publishing pp. 638–654.

You, Y., Pan, X., Wang, Z. & Lu., C. (2017), Virtual to real reinforcement learning for autonomous driving. preprint.

Yu, P., Lee, J. S., Kulyatin, I., Shi, Z. & Dasgupta, S. (2019), Model-based deep reinforcement learning for dynamic portfolio optimization, preprint, arXiv.

Zhang, G. & Chen, Y. (2020), 'Reinforcement learning for optimal market making with the presence of rebate', *Available at SSRN* **3646753**.

Zhang, Z., Zohren, S. & Roberts, S. (2019*a*), 'Deeplob: Deep convolutional neural networks for limit order books', *IEEE Transactions on Signal Processing* **67**(11), 3001–3012.

Zhang, Z., Zohren, S. & Roberts, S. J. (2019*b*), Deep reinforcement learning for trading, *in* 'The Journal of Financial Data Science'.

Zhao, T., Hachiya, H., Niu, G. & Sugiyama, M. (2011), Analysis and improvement of policy gradient estimation, *in* 'Advances in Neural Information Processing Systems', pp. 262–270.

Zhengyao et al. (2017), 'Reinforcement learning framework for the financial portfolio management problem'.

Zoph, B., Vasudevan, V., Shlens, J. & Le, Q. V. (2017), 'Learning transferable architectures for scalable image recognition', *CoRR* **abs/1707.07012**.

Zumbach, G. (2007), 'The riskmetrics 2006 methodology', *Econometrics: Applied Econometrics & Modeling eJournal* .

PART IV

Appendix

A Résumé en français de la thèse

A.1 Questions abordées dans cette thèse

Cette thèse examine les questions suivantes:

1. Existe-t-il un cadre mathématique permettant de décider des pondérations des actifs de notre portefeuille à chaque instant en connaissant non seulement l'évolution dynamique des actifs du portefeuille mais aussi certaines données supplémentaires (dites données contextuelles)? C'est le sujet du chapitre 2.

2. Pouvons nous aussi utiliser le DRL pour sélectionner des modèles quantitatifs? C'est le sujet du chapitre 3.

3. Si nous sommes capables de créer un formalisme pour les premières questions grâce à l'apprentissage par renforcement profond (DRL), comment cela se compare-t-il aux méthodes traditionnelles d'allocation de portefeuille ? C'est le sujet du chapitre 4.

4. Comme les méthodes DRL essaient de simuler de multiples exécutions et de calculer une espérance, peut-on comprendre pourquoi ces méthodes sont efficaces dans l'échantillonnage et la résolution du calcul de l'espérance. C'est le sujet du chapitre 5.

5. En termes d'apprentissage, comment se comparent l'apprentissage par renforcement profond (DRL) et l'apprentissage supervisé (SL)? C'est le sujet du chapitre 6.

Le problème de l'allocation ou de la construction de portefeuille est une question financière traditionnelle. Il peut être formulé comme suit. Supposons qu'un gestionnaire d'actifs puisse investir dans N actifs financiers dont les valeurs financières au temps t sont désignées par $(P_i^t)_{i=1,\dots,N}$ et dont les rendements au temps t sont désignés par $(r_i^t)_{i=1,\dots,N}$ et définis comme la variation en pourcentage des valeurs des actifs financiers $r_i^t = P_i^t/P_i^{t-1} - 1$. A chaque pas de temps, le gestionnaire d'actifs doit décider du pourcentage investi dans chaque actif, que nous appelons poids du portefeuille $w_t = (w_t^1, w_t^2, \dots, w_t^N)$. Il s'agit techniquement de N-uplets. La décision du gestionnaire d'actifs est motivée par une certaine fonction ou objectif risque-récompense ou fonction de perte (loss en anglais), que nous désignons par $l(w)$.

Definition A.1. *Pour un portefeuille donné dont les rendements financiers journaliers sont notés r_t et la valeur de marché journalière P_t à l'instant t, en supposant un taux sans risque de zéro pour rester simple et en considérant un horizon temporel de t à T, le ratio de Sharpe (Sharpe (1966)) est défini comme le rapport du rendement financier annuel $R_{t,T}$ sur l'écart type annuel pour la période considérée $\sigma_{t,T}$.*

$$SR = \frac{R_{t,T}}{\sigma_{t,T}}$$

Le rendement annuel est calculé comme suit : $R_{t,T} = \frac{P_T}{P_t}^{1/YF(t,T)} - 1$ et l'écart-type du rendement annuel $\sigma_{t,T} = StdDev(\{r_t, \ldots, r_T\}) * \sqrt{252}$ avec $YF(t,T)$ la fraction d'année de la période de t à T et stdDev l'opérateur calculant l'écart type traditionnel d'une série temporelle. Le facteur $\sqrt{252}$ indique que nous supposons 252 jours par an.

Une fois les pondérations choisies, et en prenant pour convention que l'allocation initiale w_{-1} (pour l'indice temporel $t = -1$) est remplie de zéros (ce qui signifie simplement qu'avant de prendre des décisions d'allocation de portefeuille, nous n'avons pris aucune allocation précédente), la valeur du portefeuille en supposant des coûts de transaction $(b^i)_{i=1,\ldots,N}$ proportionnels à la différence absolue des pondérations peut facilement être calculée comme suit (Lobo et al. (2007))

Definition A.2. *L'évaluation du portefeuille avec les coûts de transaction est donnée comme les produits cumulés des rendements réalisés moins les coûts de transaction:*

$$P_u = P_0 \prod_{k=0}^{u-1} \left(1 + \sum_{i=1}^{N} w_k^i \times r_{k+1}^i - \sum_{i=1}^{N} b^i \mid w_{k-1}^i - w_k^i \mid \right) \tag{A.1}$$

Ceci peut être reformulé sous forme vectorielle comme suit:

$$P_u = P_0 \prod_{k=0}^{u-1} (1 + w_k \cdot r_{k+1} - b \cdot \mid w_{k-1} - w_k \mid) \tag{A.2}$$

Il convient de noter qu'il existe un décalage d'un jour entre les pondérations et les rendements. Ce fait très simple est souvent ignoré dans les articles universitaires Markowitz (1952) mais fait une énorme différence. Il s'agit de tenir compte du fait que les rendements du portefeuille se produisent au cours de la période suivant notre décision d'allocation de portefeuille.

Cette thèse examine la question centrale de savoir si l'apprentissage par renforcement peut résoudre la question de l'allocation de portefeuille. Rappelons que l'apprentissage par renforcement adresse le problème de décider, à partir de l'expérience, de la séquence d'actions à effectuer dans un environnement incertain afin d'atteindre certains objectifs. Inspiré par la psychologie comportementale (voir par exemple Sutton & Barto (2018)), l'apprentissage par renforcement (RL) propose un cadre formel à ce problème.

Plus généralement, l'apprentissage par renforcement (RL) est une discipline de l'apprentissage automatique qui s'intéresse à l'apprentissage de manière à prendre une séquence de décisions qui maximise une fonction de score, traditionnellement appelée fonction de récompense. L'apprentissage par renforcement est l'un des trois paradigmes fondamentaux et traditionnels de l'apprentissage automatique, avec l'apprentissage supervisé et l'apprentissage non supervisé.

D'un point de vue pratique, les logiciels informatiques peuvent utiliser cette technique pour trouver la meilleure stratégie possible afin de résoudre tout problème qui pourrait être formulé comme un problème d'apprentissage par renforcement. L'intérêt est manifeste sur des problèmes difficiles dont on ne connaît pas la solution optimale. C'est typiquement le cas de l'allocation d'actifs ou l'allocation de portefeuille. Il peut s'agir de tâches et de domaines multiples et très différents, comme jouer à des jeux Atari Mnih et al. (2013), à d'autres jeux vidéo comme Starcraft II Vinyals et al. (2019), maîtriser des jeux comme le Go Silver et al. (2017), ou le Poker Brown et al. (2020), entraîner des robots (voir Kober et al. (2013) pour une étude, Heess et al. (2017) et Andrychowicz et al. (2020) pour discuter de l'émergence des comportements de locomotion en environnement riche, Levine et al. (2016) pour l'apprentissage de la coordination œil main pour la préhension robotique, ou Gandhi et al. (2017) pour apprendre à un véhicule aérien à éviter les obstacles), l'entraînement de voitures autonomes You et al. (2017), faire de la finance Deng et al. (2016), optimiser des réseaux intelligents François-Lavet et al. (2016), résoudre des problèmes de santé Schaefer et al. (2005), répondre à des questions visuelles générales sur des scènes complexes en visualisant des objets Antol et al. (2015) ou en devinant des objets de Vries et al. (2017), résoudre des problèmes de gestion de l'énergie pour les micro-réseaux avec un contrôle basé par un agent Dimeas & Hatziargyriou (2007) ou par un apprentissage par renforcement plus traditionnel Levent et al. (2019), résoudre des problèmes d'ordonnancement des tâches dans des systèmes informatiques à haute performance avec un apprentissage par renforcement standard Mao et al. (2016) ou par des récompenses sophistiquées Grzes & Kudenko (2009) pour n'en citer que quelques-uns, car la liste des réalisations de l'apprentissage par renforcement s'allonge à un rythme sans précédent.

Traditionnellement, un problème de RL est formalisé comme un processus de contrôle stochastique à temps discret où nous avons un agent. Cet agent représente le décideur (un robot, un logiciel ou plus généralement tout système qui doit prendre une décision). L'agent fait face à un environnement entièrement décrit à chaque instant t par des états s_t dont les valeurs possibles sont un ensemble noté $\mathcal{S}$. Cependant, l'agent peut ou non avoir accès à l'état réel. C'est pourquoi nous faisons la distinction entre l'état et l'observation, qui est la donnée à laquelle l'agent a accès, et qui vit dans un ensemble noté par $\mathcal{O}$. Par exemple, pour une voiture, l'état d'une voiture à un pas de temps donné peut être la vitesse angulaire, l'accélération et la masse de la voiture. En finance, il peut s'agir de la valeur de différents actifs et de variables financières. L'agent interagit avec son environnement de la manière suivante: l'agent commence dans un état donné $s_0 \in \mathcal{S}$ et recueille une observation initiale $o_0 \in \mathcal{O}$. À chaque pas de temps t, l'agent doit effectuer une action a_t dont les valeurs sont décrites par un ensemble $\mathcal{A}$. Cette action peut être discrète ou continue et représente la décision prise par l'agent au temps t compte tenu de toutes les informations auxquelles l'agent a accès. Une fois que l'agent a pris sa décision, c'est-à-dire effectué son action, il y a trois conséquences:

- l'agent obtient une récompense r_t dont les valeurs appartiennent à un ensemble $\mathcal{R}$.
- l'état transite vers un nouvel état $s_{t+1} \in \mathcal{S}$,
- l'agent obtient une nouvelle observation $o_{t+1} \in \mathcal{O}$.

Ce cadre de contrôle formel a été proposé pour la première fois par Bellman & Kalaba (1957) et étendu plus tard à l'apprentissage automatique par Barto et al. (1983). Il est résumé par la figure A.1 qui souligne l'aspect dynamique de notre problème de décisions séquentielles.

Cette succession d'état action peut être aussi représenté par un modèle graphique suivant la figure

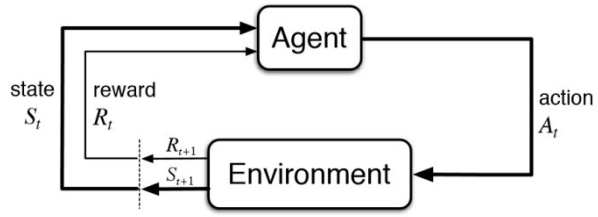


Figure A.1: Diagramme de l'apprentissage par renforcement résumant l'ordre chronologique des états, des actions et des récompenses.

A.2

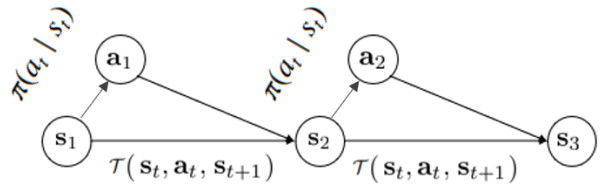


Figure A.2: Modèle graphique correspondant à l'apprentissage par renforcement. Les nœuds s_1, s_2 représentent les nœuds d'état, tandis que ceux dénotés par a_1, a_2 les nœuds d'action. Une fois que l'agent a effectué une action, l'environnement passe à l'état suivant. La probabilité de transition désignée par $p(s_{t+1}|s_t, a_t)$ satisfait la propriété de Markov car elle ne dépend que de l'état et de l'action précédents.

Pour formaliser le concept, nous définissons un processus de décision de Markov (PDM ou MDP en anglais) (Bellman (1957*b*)) comme un processus de contrôle stochastique en temps discret défini comme suit:

Definition A.3. ***MDP*** *: un MDP est un 5-tuple* $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$ *où:*

- $\mathcal{S}$ *est l'espace d'état,*
- $\mathcal{A}$ *est l'espace d'action,*
- $\mathcal{T} : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \mapsto [0, 1]$ *est la fonction de transition: ensemble des probabilités conditionnelles de transition entre les paires états, actions vers les états suivants,*
- $R : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{R}$ *est la fonction de récompense (immédiate), où* $\mathcal{R}$ *est un ensemble continu de récompenses possibles continu de récompenses possibles, généralement supposé borné compact, donc de la forme* $[0, R_{max}]$*, avec* $R_{max} \in \mathcal{R}^+$*.*
- $\gamma \in [0, 1)$ *est le facteur d'actualisation.*

Dans un MDP, la description de la dynamique du processus d'interaction entre un agent et un environnement produit une séquence d'états et d'actions. Plus loin dans cette thèse, nous appellerons cette séquence une *trajectoire*, ou un *épisode*. Nous appellerons également la paire état-action du pas de temps t le 2-uplet $(s_t, a_t) \in \mathcal{S} \times \mathcal{A}$. Nous désignons aussi la trajectoire vécue

par un agent par $\tau = (s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$ et plus généralement pour une trajectoire commençant en t, $\tau_t = (s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \ldots)$.

Une politique π définit la manière dont un agent choisit ses actions. Les politiques peuvent être classées sous le critère d'être soit stationnaires, soit non stationnaires. Un processus stationnaire est un processus stochastique dont la distribution de probabilité conjointe inconditionnelle ne change pas lorsqu'elle est décalée dans le temps. Par conséquent, les paramètres tels que la moyenne et la variance ne changent pas non plus dans le temps. En revanche, une politique non stationnaire dépend du pas de temps t et n'est utile que dans un contexte d'horizon fini où les récompenses cumulées que l'agent cherche à optimiser sont limitées à un nombre fini de pas de temps futurs. Dans notre thèse, nous ne considérerons que les politiques stationnaires car nous supposerons un horizon infini.

L'objectif de l'apprentissage par renforcement est de trouver une politique non nécessairement unique qui maximise l'espérance des récompenses futures. Cette espérance se comprend comme l'espérance suivant les différentes trajectoires pondérées de leur probabilité d'occurrence. En d'autres termes, en désignant par Π l'ensemble des politiques admissibles, et en utilisant la notation $\tau \sim \pi$ pour indiquer que la distribution de la trajectoire correspond à la probabilité d'effectuer l'action suivant la politique π, nous essayons de résoudre le programme d'optimisation suivant

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} \mathbb{E}_{\tau \sim \pi} [R(\tau)] \tag{A.3}$$

où $R(\tau)$ représente la somme cumulée et actualisée des récompenses suivant la trajectoire τ_t commençant en t ou plus simplement τ (également appelé récompense cumulative à partir de t) suivant l'équation suivante:

$$R(\tau) = \sum_{k=0}^{T} \gamma^k r_{t+k} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots \tag{A.4}$$

où le terme de récompense unique est raccourci pour garder une notation simple et est égal à $r_t = R(s_t, a_t)$.

La motivation de l'apprentissage profond est de représenter toute fonction arbitraire par une fonction $f : \mathcal{X} \mapsto \mathcal{Y}$ paramétrée avec un grand ensemble de paramètres souvent désignés par $\theta \in \mathbb{R}^{n_\theta}$ comme suit :

$$y = f(x; \theta) \tag{A.5}$$

Une caractéristique des réseaux neuronaux profonds est qu'ils s'appuient sur plusieurs couches de traitement pour traiter n'importe quelle fonction. Dans un réseau profond, chaque couche est constituée de transformations non linéaires. Une transformation non linéaire traditionnelle est la transformation Relu dont la définition est la suivante:

Definition A.4. *La fonction d'activation du redresseur ReLU (Rectified Linear Unit) est définie comme suit:* $f(x) = (ax + b)^+ = \max(0, ax + b)$.

Il est intéressant de noter que des séquences de transformations non linéaires permettent ainsi de représenter des fonctions complexes et ainsi d'avoir des apprentissages de niveaux d'abstraction différents (Erhan et al. (2009), Olah et al. (2017)).

L'apprentissage par renforcement profond est donc la combinaison de l'apprentissage par renforcement avec l'apprentissage profond.

A.2 Améliore-t-on les méthodes de portefeuilles en finance avec le DRL?

Dans le chapitre 2, nous nous posons la question de savoir si un robot peut apprendre efficacement dans un environnement bruyant et auto-adaptatif avec des observations séquentielles, non-stationnaires et non-homogènes? Par bruyant et non homogène, nous entendons que les données ont des propriétés statistiques différentes dans le temps. Par observations séquentielles, nous entendons que l'ordre chronologique compte et que les observations sont complètement modifiées si nous changeons leur ordre. Pour répondre à cette question, nous utilisons des robots de trading qui offrent un parfait exemple de données bruitées, d'observations fortement séquentielles sujettes à un changement de régime. Notre objectif est de créer un robot de gestion d'actifs augmenté pour le secteur de la gestion d'actifs.

Nous montrons notamment les trois points suivants:

- **L'ajout d'informations contextuelles.** L'utilisation des seules informations passées n'est pas suffisante pour l'apprentissage des robots dans un environnement bruyant et en évolution rapide. L'ajout d'informations contextuelles améliore considérablement les résultats. Techniquement, nous créons deux sous-réseaux : l'un alimenté par des observations directes (prix passés et écart-type) et l'autre par des informations contextuelles (niveau d'aversion au risque sur les marchés financiers, indicateurs précurseurs d'une future récession, bénéfices des entreprises...). Ce réseau à double entrée possède une performance supérieure à celle d'un réseau utilisant simplement les informations de prix.
- **Décalage d'un jour entre l'observation des prix et l'action.** Nous supposons que les prix sont observés au temps t mais que l'action ne se produit qu'au temps $t + 1$, pour être cohérent avec la réalité. Ce décalage d'un jour rend le problème beaucoup plus réaliste mais aussi plus difficile.
- **La procédure de marche avant.** En raison de la nature non stationnaire des données dépendantes du temps et en particulier des données financières, il est crucial de tester la stabilité des modèles DRL. Nous présentons une nouvelle méthodologie d'évaluation des modèles DRL, appelée analyse walk-forward, qui entraîne et teste itérativement le modèle sur un ensemble de données étendu. Cela peut être considéré comme l'analogie de la validation croisée pour les séries temporelles. Cela permet de valider que les hyper paramètres sélectionnés fonctionnent bien dans le temps et que les modèles résultants sont stables dans le temps.

Nous comparons les performances de 5 modèles: Modèle DRL basé sur des réseaux convolutifs avec des états contextuels (indicateur de sentiment, corrélation à 6 mois entre actions et obligations et indice principal de crédit), modèle DRL sans états contextuels, modèle qui suit le gagnant,

modèle qui suit le perdant et modèle suivant le portefeuille de Markowitz revu tous les trois mois. Les graphiques résultants sont affichés dans la figure A.3 avec la position d'actif risqué seule en bleu et les modèles en orange. Sur ces 5 modèles, seuls le modèle de DRL et le modèle qui suit le gagnant sont capables de fournir une augmentation significative de la performance nette grâce à une stratégie de couverture efficace sur la période 2007 à 2020. Le modèle DRL est en outre capable de mieux s'adapter à la crise du Covid et d'avoir une meilleure efficacité en termes de rendement net mais aussi de ratios de Sharpe et de Sortino sur 3 et 5 ans comme le montre le tableau A.1.

Table A.1: Comparaison des modèles sur 3 et 5 ans

	3 Years			
	return	Sortino	Sharpe	max DD
Risky asset	10.27%	0.34	0.38	- 0.34
DRL	**22.45%**	**1.18**	**1.17**	-0.27
Winner	13.19%	0.66	0.72	-0.35
Loser	9.30%	0.89	0.89	**-0.15**
DRL no context	8.11%	0.42	0.47	-0.34
Markowitz	-0.31%	-0.01	-0.01	-0.41
	5 Years			
	return	Sortino	Sharpe	max DD
Risky asset	9.16%	0.54	0.57	- 0.34
DRL	**16.42%**	**0.98**	**0.96**	-0.27
Winner	10.84%	0.65	0.68	-0.35
Loser	7.04%	0.78	0.76	**-0.15**
DRL no context	6.87%	0.44	0.47	-0.34
Markowitz	-0.07%	-0.00	-0.00	-0.41

Dans le tableau A.2, nous fournissons une liste de 32 modèles basés sur les choix suivants : architecture de réseau (LSTM ou CNN), entraînement contradictoire avec bruit dans les données ou non, utilisation d'états contextuels, et fonction de récompense (bénéfice net et Sortino), utilisation du décalage d'un jour entre les observations et les actions. Nous constatons que le meilleur modèle DRL avec la contrainte de rotation du décalage journalier est celui qui utilise les réseaux convolutifs, l'apprentissage adversarial, les états contextuels et la fonction de récompense net profit. Ces 4 paramètres sont significatifs pour notre modèle DRL et modifient considérablement les performances du modèle comme le montre le tableau. Nous comparons également le même modèle avec et sans état contextuel et nous voyons dans le tableau A.3 que l'utilisation de l'état contextuel améliore considérablement les performances du modèle. Ceci est assez intuitif car nous fournissons des données plus significatives au modèle.

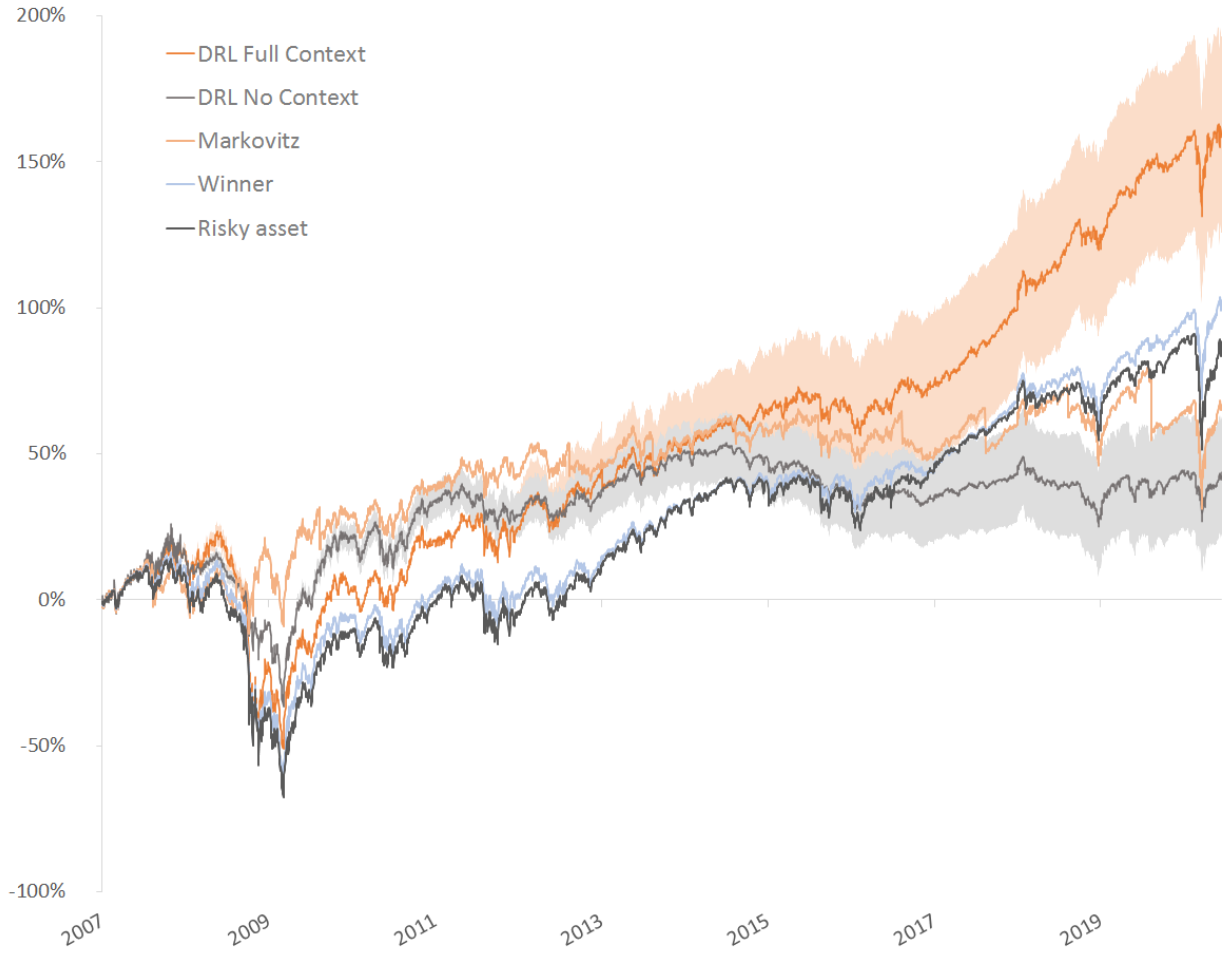


Figure A.3: Comparaison des différents modèles (DRL avec et sans contexte, Follow the winner et Markowitz versus l'actif risqué). Comme ces stratégies sont des stratégies de couverture qui devraient améliorer l'actif risqué, elles devraient mieux performer que l'actif risqué. Parmi tous ces modèles, le meilleur modèle est le modèle DRL avec des fonctionnalités contextuelles. La plupart des modèles ne sont pas en mesure de s'adapter en permanence aux changements de régime et, par conseéquently under-perform compared to the standalone risky asset position on a long, sous-performent par rapport à la position d'actifs risqués autonome sur une longue période comme 2007 à 2020. Ils fonctionnent initialement bien dans la peériod like initiale de 2007 toà 202015 mais ne sont pas capables de s'adapter après 2015.

Table A.2: Comparaison de modèles basée sur la fonction de récompense, le réseau (unités CNN ou LSTM), la formation contradictoire (bruit dans les données) et l'utilisation de l'état contextuel

reward	network	adversarial training	contextual states	performance with 1 day lag	performance with 0 day lag
Net_Profit	CNN	Yes	Yes	81.8%	123.8%
Net_Profit	CNN	No	Yes	75.2%	112.3%
Net_Profit	LSTM	Yes	Yes	65.9%	98.8%
Net_Profit	LSTM	No	Yes	64.5%	98.5%
Sortino	LSTM	No	Yes	61.8%	87.4%
Net_Profit	LSTM	No	No	56.6%	59.8%
Sortino	LSTM	No	No	48.5%	51.4%
Net_Profit	LSTM	Yes	No	47.5%	50.8%
Sortino	LSTM	Yes	Yes	29.6%	47.6%
Sortino	LSTM	Yes	No	28.4%	47.0%
Sortino	CNN	No	Yes	26.5%	45.3%
Sortino	CNN	Yes	Yes	26.3%	29.3%
Sortino	CNN	Yes	No	-16.7%	16.9%
Net_Profit	CNN	Yes	No	-29.5%	13.9%
Sortino	CNN	No	No	-45.0%	10.6%
Net_Profit	CNN	No	No	-47.7%	8.6%

Table A.3: Impact of contextual state

reward	network	adversarial training	contextual states impact
Net_Profit	CNN	Yes	111.4%
Net_Profit	CNN	No	122.9%
Net_Profit	LSTM	Yes	18.5%
Net_Profit	LSTM	No	7.9%
Sortino	LSTM	No	13.3%
Sortino	LSTM	Yes	1.2%
Sortino	CNN	No	71.5%
Sortino	CNN	Yes	43.0%

A.3 Peut on mieux anticiper des crises par DRL?

Dans le chapitre précédent 2, nous avons expliqué comment appliquer le DRL à l'allocation de portefeuille. Dans ce chapitre, nous allons étudier si le DRL peut s'adapter à un environnement changeant. Être capable d'adapter l'allocation de portefeuille à un environnement de crise comme la crise du Covid est une préoccupation majeure pour l'industrie financière.

L'approche standard de l'allocation de portefeuille, qui sert de base à notre recherche, repose sur la détermination des pondérations de portefeuille en fonction d'un critère de rendement du risque. La méthode dite de Markowitz (1952) trouve l'allocation optimale en déterminant le portefeuille avec une variance minimale compte tenu d'un rendement cible ou, de manière équivalente, le portefeuille avec un rendement maximal compte tenu d'un niveau de variance cible (l'optimisation duale). Cependant, cette approche souffre d'un défaut majeur en raison des estimations de risque peu fiables des excès de rendement et des covariances des stratégies de portefeuille. Cela conduit non seulement à des allocations instables, mais aussi à des réactions lentes face à des environnements changeants Black & Litterman (1992). Si nous voulons trouver une méthode d'allocation plus dynamique, l'apprentissage par renforcement profond est une méthode séduisante. Il reformule le problème d'optimisation du portefeuille comme un programme de contrôle continu avec des récompenses différées. Par rapport aux résultats déjà trouvés dans le chapitre précédent, nous soulignons les points suivants dans la méthode DRL:

- La fonction de récompense est essentielle. La récompense du ratio de Sharpe conduit à des résultats différents par rapport à une fonction de récompense directe de la performance nette finale.
- Les réseaux CNN sont plus performants que les réseaux à base d'architecture LSTM et capturent des caractéristiques implicites de changement de régime.
- L'utilisation d'ajout de bruit sur les scénarios d'apprentissage améliore légèrement le modèle.
- la dépendance aux allocations précédentes n'améliore pas le modèle.

- Enfin, l'approche DRL se distingue des approches traditionnelles en privilégiant la concentration par rapport à la diversification. Ceci est cohérent avec le fait que la politique en DRL vise à trouver au sein d'un portefeuille la meilleure allocation. C'est un caractère différenciant fort du DRL par rapport aux approches financières quantitatives traditionnelles. Si on souhaite réduire cet aspect de concentration, on devra mettre en place une reward favorisant la diversification.

A.4 Faut-il combiner de l'apprentissage par renforcement sans et avec modèles?

Dans le chapitres précédent, nous avons utilisé une approche DRL sans modèle. Dans le chapitre 3, nous examinons la question de pouvoir combiner l'approche sans et avec modèle. L'apprentissage par renforcement (RL) vise à l'acquisition automatique de compétences ou plus généralement d'une forme d'intelligence, afin de se comporter de manière appropriée et judicieuse dans des situations potentiellement inédites. Lorsqu'il s'agit de situations réelles, deux défis se posent : disposer d'une méthode d'apprentissage efficace en termes de données et être capable de gérer des systèmes dynamiques complexes et inconnus qui peuvent être difficiles à modéliser. Parce que la nature dynamique de l'environnement peut être difficile à apprendre, un premier courant de méthodes a consisté à modéliser l'environnement avec un modèle. Cela s'appelle traditionnellement le RL à base de modèle. Ces méthodes ont tendance à bien fonctionner dans des environnements complexes et changeants. A l'inverse, des méthodes dites sans modèle ont été développées pour éviter des biais cognitifs de modélisation.

Nous montrons dans ce chapitres les points suivants:

- **L'utilisation du RL sans modèle pour sélectionner divers modèles fonctionne**. Dans un environnement bruyant et de régime changeant comme les séries chronologiques financières, Nous utilisons une approche sans modèle pour choisir le modèle de ciblage de volatilité optimale. Sans surprise par rapport aux travaux des deux premiers chapitres, nous trouvons que l'utilisation d'informations contextuelles améliore le modèle.

- **Procédure de sensibilité des caractéristiques.** Inspiré par le concept d'importance des caractéristiques dans les méthodes de boosting de gradient, nous examinons l'importance de variables ou caractéristiques de notre modèle de RL profond basée par le calcul de sa sensibilité aux variables initiales. Cela nous permet de classer chaque caractéristique à chaque date afin de fournir des explications sur les raisons pour lesquelles notre agent DRL choisit une action particulière.

- **Statistique pour tester la stabilité du modèle.** Nous introduisons le concept de différence statistique afin de valider que le modèle résultant est statistiquement différent des résultats du modèle de référence

Le chapitre 3 conclut la première partie de la thèse.

La deuxième partie de la thèse vise à analyser l'intérêt et les justifications derrière le DRL.

A.5 En quoi le DRL généralise-t-il Markowitz?

Le chapitre 4 adresse la question de la comparaison des approches classiques de théorie du portefeuille et de l'approche par DRL. La théorie du portefeuille de Markowitz est de pouvoir comparer différents actifs en tenant compte du rendement et du risque. Dans la théorie moderne du portefeuille (MPT), le risque est représenté par la variance des rendements des actifs. Si nous prenons différents actifs financiers en nombre N et que nous représentons leurs rendements et leurs risques, nous pouvons trouver une frontière efficiente. Mathématiquement, si l'on désigne par $w = (w^1, ..., w^N)$ les poids d'allocation avec $1 \geq w^i \geq 0$ pour $i = 0...N$, résumés par $1 \geq w \geq 0$, avec les contraintes supplémentaires que ces poids s'additionnent à 1 : $\sum_{i=1}^{N} w^i = 1$, on peut montrer que cette question d'allocation de portefeuille se résume à une optimisation.

Soit $\mu = (\mu^1, ..., \mu^N)^T$ les rendements attendus de nos stratégies en nombre N et Σ la matrice des covariances des rendements des stratégies. Soit r_{min} le rendement attendu minimum. Le problème d'optimisation de Markowitz à résoudre est de minimiser le risque compte tenu d'un objectif de rendement espéré minimum comme suit :

$$\begin{aligned} &\underset{w}{\text{Minimise}} \quad w^T \Sigma w \\ &\text{subject to} \quad \mu^T w \geq r_{min}, \sum_{i=1...N} w^i = 1, 1 \geq w^i \geq 0 \ \ \forall i \end{aligned} \tag{A.6}$$

On le résout par programmation quadratique standard. Grâce à la dualité, il existe une maximisation équivalente avec un risque maximal donné σ_{max} pour laquelle le problème s'écrit comme suit :

$$\begin{aligned} &\underset{w}{\text{Maximise}} \quad \mu^T w \\ &\text{subject to} \quad w^T \Sigma w \leq \sigma_{max}, \sum_{i=1...N} w^i = 1, 1 \geq w^i \geq 0 \ \forall i \end{aligned} \tag{A.7}$$

On comprend dès lors que les modèles d'allocation classique de portefeuille sont en fait une simple optimisation à une période où les poids visent à maximiser un critère de récompense. Dans le cas de Markowitz, il s'agit du ratio de Sharpe. L'approche par DRL généralise ces approches quantitatives classiques de la théorie du portefeuille en étendant le problème d'optimisation à un problème de contrôle optimal multi-période puisque le problème à résoudre s'écrit maintenant de la façon suivante:

$$\begin{aligned} &\underset{\pi(.)}{\text{Maximise}} \quad \mathbb{E}[R_T] \\ &\text{subject to} \quad a_t = \pi(s_t) \end{aligned} \tag{A.8}$$

où π est la politique de l'agent qui consiste étant donné des états à fournir des allocations a_t (ou dans le langage de l'apprentissage par renforcement des actions) de façon à maximiser

un critère de récompense R_T. Comme nous évoluons dans un environnement stochastique, nous devons calculer l'espérance de la récompense cumulée soit $\mathbb{E}[R_T]$. Cette récompense cumulée peut revêtir différentes formes comme un ratio de Sharpe sur toute la vie du portefeuille ou encore la performance nette finale du portefeuille. Si nous prenons par exemple comme récompense la performance nette finale du portefeuille, et que nous écrivons P_t le prix au temps t de notre portefeuille, et son rendement au temps t : r_t^P et le vecteur de rendement des actifs du portefeuille au temps t : $\vec{r}_t$, notre problème d'optimisation est maintenant multi période et plus élaboré que celui donné dans le cadre de Markowitz. La performance nette finale s'écrit comme $P_T/P_0 - 1 = \prod_{t=1}^{T}(1 + r_t^P) - 1$. Le rendement r_t^P est une fonction de notre action de planification a_t comme suit : $(1 + r_t^P) = 1 + \langle \vec{a}_t, \vec{r}_t \rangle$ où $\langle \cdot, \cdot \rangle$ est le produit interne standard de deux vecteurs. En outre, si nous nous rappelons que la politique est paramétrée par certains paramètres du réseau profond, θ : $a_t = \pi_\theta(s_t)$, nous pouvons rendre notre problème d'optimisation légèrement plus détaillé comme suit :

$$\begin{aligned} \underset{\theta}{\text{Maximise}} \quad & \mathbb{E}\left[\prod_{t=1}^{T}(1 + \langle \vec{a}_t, \vec{r}_t \rangle)\right] \\ \text{subject to} \quad & a_t = \pi_\theta(s_t). \end{aligned} \tag{A.9}$$

Il convient de noter que par rapport aux méthodes d'allocation classique, le problème d'optimisation sous-jacent en apprentissage par renforcement généralise de deux manières les méthodes d'allocations de portefeuille:

- Premièrement, nous optimisons une fonction π et non de simples poids $w = (w^i)_{i=1..N}$. Bien que cette fonction à la fin soit représentée par un réseau neuronal profond et donc des paramètres, c'est conceptuellement très différent puisque nous effectuons une optimisation dans l'espace des fonctions $\pi : \mathcal{S} \to \mathcal{A}$, qui est un espace beaucoup plus grand que celui de simple paramètres $\mathbb{R}^l$.
- Deuxièmement, il s'agit d'une optimisation multi-période car elle implique des résultats à chaque pas de temps $t = 1$ à $t = T$, ce qui la rend également plus complexe, mais aussi plus réaliste.

A.6 Les méthodes de DRL sont-elles des méthodes de réduction de variance?

Le chapitre précédent a montré que le DRL est une extension des méthodes financières traditionnelles. Le DRL vise à maximiser l'espérance des récompenses cumulées. Le calcul de l'espérance des récompenses cumulées n'est pas trivial. Dans le chapitre 5, nous examinerons comment les méthodes DRL sont efficaces pour calculer cette espérance et comment elles peuvent être liées aux techniques de réduction de la variance.

Lorsque les états et/ou les actions opèrent dans un espace de grande dimension, nous représentons notre politique par un réseau profond dont les paramètres sont notés θ. Typiquement ces paramètres représentent les différentes couches de notre réseau profond ainsi que les paramètres de la/les fonctions d'activation. L'intuition des méthodes d'acteur critiques (ACM) est d'exploiter une

politique de descente de gradient avec une variance réduite. L'objectif de ce chapitre est de le montrer de façon précise. Rappelons que le gradient de la politique est donné par le gradient logarithmique de la politique pondéré par la somme des récompenses futures actualisées (voir Williams (1992) et Sutton, McAllester, Singh & Mansour (1999))

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_\tau\Bigg[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t|\ s_t)\underbrace{\Bigg(\sum_{t'=t+1}^{T}\gamma^{t'-t}r_{t'}\Bigg)}_{R_t}\Bigg] \\ &= \mathbb{E}_\tau\Bigg[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(a_t \mid s_t)R_t\Bigg] \end{aligned} \tag{A.10}$$

où τ représente une trajectoire, R_t la somme des récompenses futures actualisées, a_t les actions de notre modèle de RL, π_θ la politique, s_t les états, et $\gamma \in [0, 1)$ le facteur d'actualisation. Par conséquent, l'équation (A.10) montre que nous mettons à jour les paramètres du réseau profond de la politique à travers des mises à jour de simulation de Monte Carlo visant à estimer les termes d'espérance. Le calcul de cette espérance est une partie critique de l'algorithme et mérite que nous nous y arrêtions. Si l'estimation de cette espérance n'est pas très précise en raison d'une grande variance de notre estimateur fourni par la moyenne empirique standard, nous aurons une grande variabilité dans notre mise à jour du gradient, d'où une méthode de politique de gradient à convergence lente. Il est donc très judicieux de voir si nous pouvons trouver une autre expression de notre gradient de politique avec une variance plus faible. Cette approche consistant à trouver une expression modifiée à l'intérieur de l'espérance qui a la même valeur espérée mais une variance plus faible est appelée réduction de variance Hammersley (1964). Une méthode typique de réduction de la variance consiste à utiliser une ou plusieurs variables de contrôle (voir par exemple Ross (2002)). Une variance plus faible dans le gradient produira un gradient moins bruyant et provoquera un apprentissage moins instable conduisant à une distribution de la politique s'orientant plus rapidement vers la direction optimale.

Nous montrons le résultat suivant:

Proposition A.5. *Méthodes d'acteur critique - Les estimateurs traditionnels de gradient de politique sont sans biais et sont en fait des estimateurs avec variable de contrôle de l'estimateur initial de gradient de politique REINFORCE donné par* $\mathbb{E}_{\pi_\theta}\Big[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(s,a)R_t\Big]$*:*

- $\mathbb{E}_{\pi_\theta}\Bigg[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(s,a)Q(s,a)\Bigg]$ *(Q-AC)*
- $\mathbb{E}_{\pi_\theta}\Bigg[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(s,a)A(s,a)\Bigg]$ *(A-AC)*
- $\mathbb{E}_{\pi_\theta}\Bigg[\sum_{t=0}^{T-1}\nabla_\theta \log \pi_\theta(s,a)TD(s)\Bigg]$ *(TD-AC)*

où la fonction d'avantage (A) est définie comme

$$A(s,a) = \mathbb{E}_{\pi_\theta}[r_{t+1} + \gamma V(s_{t+1}) \mid s_t = s, a_t = a] - V(s_t)$$

et la fonction de différence temporelle (TD) comme

$$TD(s_t) = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

Cette proposition justifie pleinement les méthodes d'acteur critique. Sa preuve est relativement simple et s'écrit de la façon suivante:

Preuve A.6. Abordons un par un les différents estimateurs donnés dans la proposition A.5.

L'estimateur **Q Actor Critic (Q-AC)** est donné par $\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s,a) Q(s,a)\right]$. Elle s'écrit aussi sous la forme

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s,a)(\underbrace{R(s)}_{\hat{m}} - \underbrace{(R(s) - Q(s,a))}_{\hat{t}})\right]$$

ce qui montre qu'il s'agit d'un estimateur à variable de contrôle à condition de prouver que

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s,a)(Q(s,a) - R(s))\right] = 0 \tag{A.11}$$

L'équation (A.11) est trivialement vérifiée car l'une des définitions de la fonction de valeur de l'action d'état $Q(s,a)$ (souvent appelée fonction 'Q') est la suivante :

$$Q(s,a) = \mathbb{E}_{\pi_\theta}[R(s) \mid s_t = s, a_t = a]$$

La loi de l'espérance totale stipule que si X est une variable aléatoire et Y une variable aléatoire quelconque sur le même espace de probabilité, alors $\mathbb{E}[\mathbb{E}[X \mid Y]] = \mathbb{E}[X]$ ou en d'autres termes $\mathbb{E}[\mathbb{E}[X \mid Y] - X] = 0$, ce qui conclut la preuve de la méthode Q-Actor Critic (Q-AC) avec $X = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s,a) R(s)$ et $Y = (s_t = s, a_t = a)$.

Quant à la méthode **Acteur Critique par fonction Advantage (ACA)**, le même raisonnement montre qu'il suffit de prouver que

$$\mathbb{E}_{\pi_\theta}\,!\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s,a)(A(s,a) - R(s))\right] = 0$$

pour prouver qu'il s'agit également d'une méthode à variante de contrôle. Rappelons que la fonction d'avantage est donnée par $A(s,a) = Q(s,a) - V(s)$. En utilisant le résultat précédemment prouvé pour la méthode Q AC (équation (A.11)), il suffit de prouver que

$$\mathbb{E}_{\pi_\theta}\,!\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s,a) V(s)\right] = 0 \tag{A.12}$$

Enfin, prouvons que la méthode **TD Actor Critic (TD-AC)** est également une variante de contrôle. Rappelons que le terme de différence temporelle est donné par

$$TD(s_t) = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

En utilisant l'équation (A.12), il suffit de prouver que

$$\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s,a)(r_{t+1} + \gamma V(s_{t+1}) - R(s_t))\right] = 0 \tag{A.13}$$

Il s'agit à nouveau d'une application directe de la loi de l'espérance totale comme suit

$$V(s_{t+1}) = \mathbb{E}_{\pi_\theta}[\sum_{s=t+2}^{T} \gamma^{s-(t+2)} r_s \mid s_{t+1}]$$

alors que

$$R(s_t) = \sum_{s=t+1}^{T} \gamma^{s-(t+1)} r_s$$

Par conséquent, nous pouvons appliquer la loi de l'espérance totale avec
$X = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(s,a) \sum_{s=t+2}^{T} \gamma^{s-(t+1)} r_s$ et $Y = s_{t+1}$. Ceci conclut la preuve.

A.7 Y a t -il des similitudes entre apprentissage par renforcement et supervisé?

Dans le chapitre précédent, nous avons montré que les méthodes d'acteur critique sont en fait des méthodes avec variable de contrôle. Nous nous posons dans ce chapitre la question de savoir s'il existe des similitudes entre l'apprentissage par renforcement et l'apprentissage supervisé.

Dans l'apprentissage par renforcement (AR), les méthodes de gradient de politique (PGM) sont fréquemment utilisées Williams (1992), Sutton & Barto (2018), Silver et al. (2014), Lillicrap et al. (2015). Les MGP sont des techniques de RL qui reposent sur l'optimisation des paramètres des politiques par rapport à la récompense cumulative attendue en utilisant l'optimisation par descente de gradient. Elles sont traditionnellement opposées aux méthodes d'apprentissage de la valeur (ou itérations de la valeur) Watkins & Dayan (1992) qui soit continuent à améliorer la fonction de valeur à chaque itération jusqu'à ce que la fonction de valeur converge, soit optimisent les paramètres de la fonction de valeur. La fonction de valeur est définie comme la valeur attendue de la récompense cumulée conditionnelle à un état initial et à une action. Le principe de la PGM est très simple. Améliorer progressivement la politique par descente de gradient. La MGP comporte deux concepts importants. Il s'agit d'une méthode de politique, comme son nom l'indique. Il s'agit également d'une méthode de descente de gradient. La politique signifie que nous observons et agissons. Les méthodes de descente de gradient utilisent le fait que le meilleur déplacement local se fait le long du gradient. Comme le déplacement est au premier ordre, un taux d'apprentissage doit garantir que l'amélioration de la politique n'est pas trop importante à chaque étape. Les MGP ont été popularisées dans REINFORCE Williams (1992) et dans Sutton, Precup & Singh (1999) et ont reçu une plus grande attention avec les méthodes Actor Critic Konda & Tsitsiklis (2003), Peters & Schaal (2008) en particulier lors de l'utilisation des MGP profondes Mnih et al. (2016) qui combinent les méthodes de politique et de valeur. En outre, on a récemment découvert qu'un

terme de régularisation de l'entropie peut accélérer la convergence O'Donoghue et al. (2016), Nachum et al. (2017), Schulman et al. (2017).

En regardant en détail l'algorithme REINFORCE Williams (1992), nous pouvons remarquer que le terme de gradient par rapport à la politique peut en effet être interprété comme le terme logarithmique de l'entropie croisée en apprentissage supervisé. Si, en outre, nous faisons le lien entre RL et SL, en soulignant que le problème RL peut être reformulé comme un problème SL où les étiquettes vraies sont remplacées par les récompenses futures actualisées attendues, et les probabilités estimées par les probabilités de la politique, le lien entre RL et SL devient évident. De plus, en exploitant la relation étroite entre l'entropie croisée et la divergence de Kullback Leibler, nous pouvons interpréter les termes de régularisation de l'entropie de manière très naturelle. C'est précisément l'objectif de ce court chapitre : attirer l'attention sur le lien étroit entre RL et SL pour donner des justifications théoriques à certaines des techniques utilisées dans les MGP.

Effectivement, on peut rappeler que l'algorithme REINFORCE calcule la quantité suivante:

Proposition A.7. *La descente du gradient dans REINFORCE peut également être calculée en minimisant la quantité suivante*

$$\widetilde{J}(\theta) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} R_i(\tau) \log \pi_\theta(a_{i,t} \mid s_{i,t}) \tag{A.14}$$

Pour la méthode Advantage Actor Critic, la descente du gradient peut également être calculée en minimisant la quantité suivante :

$$\widetilde{J}(\theta) = \frac{\sum_{i=1}^{N} \sum_{t=1}^{T} A(s_{i,t}, a_{i,t}) \log \pi_\theta(a_{i,t} \mid s_{i,t})}{N} \tag{A.15}$$

La preuve n'est pas très longue et est la suivante.

Preuve A.8. Désignons par $r(s_t, a_t)$ la récompense pour un état s_t et une action a_t. Supposons que nous ayons un certain horizon temporel (qui peut être fini ou infini). Désignons par $\tau = (s_1, a_1, \ldots, s_T, a_T)$ une trajectoire générée par notre approximateur de politique gouverné par un paramètre θ. En utilisant la propriété de Markov de notre processus MDP, la probabilité d'une trajectoire donnée $\mathbb{P}(\tau|\theta)$ peut être décomposée en un produit de probabilités conditionnelles comme suit :

$$\mathbb{P}(\tau|\theta) = \mathbb{P}(s_1) \prod_{t=1}^{T} \pi_\theta(a_t \mid s_t) \mathbb{P}(s_{t+1} \mid s_t, a_t) \tag{A.16}$$

En prenant le logarithme de l'équation ci-dessus (A.16), en utilisant le fait que le logarithme d'un produit est la somme des logarithmes, nous obtenons :

$$\log \mathbb{P}(\tau|\theta) \quad = \quad \log \mathbb{P}(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t \mid s_t) + \log \mathbb{P}(s_{t+1} \mid s_t, a_t) \tag{A.17}$$

En différenciant par rapport au thêta, on obtient (puisque la plupart des termes ne dépendent pas de θ):

$$\nabla_\theta \log \mathbb{P}(\tau|\theta) \quad = \quad \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t \mid s_t) \tag{A.18}$$

Rappelons que nous voulons minimiser le rendement attendu, $J(\theta)$, défini comme suit

$$J(\theta) = \mathbb{E}_{\tau\sim\mathbb{P}(\tau|\theta)}\left[\sum_{t=1}^{T} r(s_t, a_t)\right] = \int_{\tau} R(\tau)\mathbb{P}(\tau|\theta)d\tau \tag{A.19}$$

La notation ci-dessus $\tau \sim \mathbb{P}(\tau|\theta)$ indique que nous échantillonnons des trajectoires τ à partir de la distribution de probabilité de notre approximateur de politique gouverné par θ et $R(\tau)$ est la somme de toutes les récompenses futures (actualisées).

Pour trouver le paramètre θ optimal, nous effectuons une descente de gradient et devons donc calculer

$$\nabla_\theta J(\theta) \quad = \quad \nabla_\theta \int_{\tau} R(\tau)\mathbb{P}(\tau|\theta)d\tau$$

En utilisant le fait que nous pouvons interchanger l'intégrale et l'espérance (A.20), en supposant des fonctions lisses et une convergence dominée de Lebesgue pour justifier que nous pouvons amener le gradient sous l'intégrale, nous obtenons :

$$\nabla_\theta J(\theta) \quad = \quad \int_{\tau} R(\tau)\nabla_\theta\mathbb{P}(\tau|\theta)d\tau \tag{A.20}$$

Comme le logarithme du gradient d'une fonction est le quotient du gradient et de la fonction, on a :

$$\nabla_\theta J(\theta) \quad = \quad \int_{\tau} R(\tau)\nabla_\theta \log \mathbb{P}(\tau|\theta)\mathbb{P}(\tau|\theta)d\tau \tag{A.21}$$

En exprimant l'intégrale sous forme d'espérance, on conclut :

$$\nabla_\theta J(\theta) \quad = \quad \mathbb{E}\left[R(\tau)\nabla_\theta \log \mathbb{P}(\tau|\theta)\right] \tag{A.22}$$

où dans les équations ci-dessus, nous avons d'abord utilisé le fait que nous pouvons interchanger l'intégrale et l'espérance (A.20), en supposant une fonction lisse et une convergence dominée de Lebesgue pour justifier que nous pouvons ramener le gradient sous l'intégrale, puis dans (A.21), nous avons utilisé le fait que le log gradient d'une fonction est le quotient du gradient et de la fonction, et enfin nous avons exprimé l'intégrale comme une espérance. Enfin, en utilisant le fait que le gradient de la probabilité logarithmique de la trajectoire est la somme du gradient des probabilités logarithmiques de la politique (A.18), nous obtenons l'expression finale :

$$\nabla_\theta J(\theta) = \mathbb{E}\left[R(\tau)\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right] \tag{A.23}$$

Pour rendre cela plus facile, il suffit de l'exprimer sous la forme d'une somme de Monte Carlo comme suit :

$$\nabla_\theta J(\theta) = \lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} R_i(\tau)\nabla_\theta \log \pi_\theta(a_{i,t} \mid s_{i,t}) \tag{A.24}$$

Comme le terme de l'équation de droite ne dépend de θ que dans le terme $\log \pi_\theta(a_{i,t} \mid s_{i,t})$, minimiser $J(\theta)$ revient à minimiser $\widetilde{J}(\theta)$ donné par :

$$\widetilde{J}(\theta) = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} R_i(\tau) \log \pi_\theta(a_{i,t} \mid s_{i,t}) \tag{A.25}$$

Comme pour Advantage Actor Critic, la preuve ci-dessus est la même et conduit au résultat.

On comprend dès lors que l'apprentissage par renforcement par descente de gradient est en fait un apprentissage supervisé avec une fonction de perte d'entropie croisée et des étiquettes égales aux récompenses optimales. Ce résultat, bien que théorique en raison de l'impossibilité de connaître à l'avance les récompenses optimales, établit un lien profond entre les deux méthodes d'apprentissage et montre que les méthodes d'apprentissage supervisé dans le cas particulier d'une connaissance des labels sont plus performantes que celle de RL car il n'y a pas d'incertitudes sur les labels.

RÉSUMÉ

La promesse de l'apprentissage par renforcement profond (DRL) est de ne faire aucun choix à priori en termes de décision ou de règles et de laisser la machine trouver la meilleure solution. Dans cette thèse, nous montrons que ce type d'apprentissage apporte une solution nouvelle pour l'allocation de portefeuille.
Dans une première partie, nous présentons comment appliquer le DRL à l'allocation de portefeuille et tenir compte de la nature spécifique des séries temporelles financières. Nous introduisons le concept de variable contextuelle permettant un meilleur apprentissage. Nous modifions la validation croisée à une approche progressive afin de garantir que les données d'apprentissage ne contiennent aucun point futur par rapport aux données de validation et de test. Nous montrons empiriquement que le DRL permet d'aller au-delà de l'état de l'art des méthodes d'allocations de portefeuille en trouvant des portefeuilles mieux adaptés aux conditions de marché, grâce à des couches de convolutions permettant de capter la dépendance entre les données de marché et les décisions d'allocation. Nous appliquons cela à la détection de crise pendant le Covid. Nous concluons cette partie par une approche à base de modèles où le DRL sélectionne des modèles de ciblage de volatilité. Dans une seconde partie, nous présentons des résultats théoriques justifiant l'approche DRL. Nous montrons comment l'approche DRL généralise les théories classiques du portefeuille. Nous exposons comment les méthodes de DRL réalisent des réductions de variance. Nous trouvons des similitudes entre l'apprentissage par renforcement et l'apprentissage supervisé.

MOTS CLÉS

Apprentissage par renforcement profond, allocation de portefeuille, données contextuelles

ABSTRACT

The promise of deep reinforcement learning (DRL) is to make no initial assumptions in terms of decisions or rules and let the machine find the best solution. In this thesis, we show that this type of machine learning method provides a new solution for portfolio allocation. In the first part, we present how to apply DRL to portfolio allocation and take into account the specific nature of financial time series. We introduce the concept of contextual variables allowing better learning. We change the cross-validation to a stepwise approach to ensure that the training data does not contain any future points compared to the validation and test data. We empirically show that DRL makes it possible to go beyond the state of the art of portfolio allocation methods by finding portfolios better adapted to market conditions, thanks to layers of convolutions allowing us to capture the dependence between market data and allocation decisions. We apply this to crisis detection during Covid. We conclude this part with a model-based approach where DRL selects volatility-targeting models. In the second part, we present theoretical results justifying the DRL approach. We show how the DRL approach generalises classical portfolio theories. We study how DRL methods achieve variance reduction. We find similarities between reinforcement learning and supervised learning.

KEYWORDS

Deep reinforcement learning, portfolio allocation, contextual data