MASARYK UNIVERSITY

FACULTY OF ECONOMICS AND ADMINISTRATION

Master's Thesis

Brno 2024

Bc. Lukáš Galeta

MASARYK UNIVERSITY

FACULTY OF ECONOMICS AND ADMINISTRATION

Asset allocation with reinforcement learning

Master's Thesis

BC. LUKÁŠ GALETA

Advisor: prof. Ing. Štefan Lyócsa, PhD.

Department of Economics

Degree Programme: Mathematical and Statistical Methods in Economics

Brno, Spring 2024

MUNI
ECON

Bibliographic record

Author:	Bc. Lukáš Galeta Faculty of Economics and Administration Masaryk University Department of Economics
Title of Thesis:	Asset allocation with reinforcement learning
Degree Programme:	Mathematical and Statistical Methods in Economics
Field of Study:	Mathematical and Statistical Methods in Economics
Supervisor:	prof. Ing. Štefan Lyócsa, PhD.
Academic Year:	2023/2024
Number of Pages:	101
Keywords:	reinforcement learning, Q-learning, asset allocation

Annotation

Quantitative algorithmic trading is a flourishing field due to the capability of machine learning models to find hidden patterns in data, which may lead to the generation of excess returns. One of the most promising areas of machine learning in quantitative algorithmic trading is reinforcement learning.

This diploma thesis aims to introduce and evaluate a simple reinforcement learning framework for asset allocation. Asset allocation problem is modelled as a partially observable Markov decision process, where the reinforcement agent cannot fully observe the environment. This diploma thesis utilises Q-learning and the Gaussian mixture model with K-means clustering. The agent's objective function is defined as the maximisation of the discounted portfolio return of 5 following trading days. The proposed reinforcement learning framework outperformed the S&P 500 benchmark index regarding average return, maximal drawdown, and Sharpe ratio.

The most robust model, a stacked model of stacked models, consistently overperforms the benchmark S&P 500 buy and hold strategy from 2015 to 2024. The model was able to achieve persistently higher returns and similar maximal drawdown. This model achieves a Sharpe ratio of 2.2 compared to 1.5 on test data. The most powerful stacked model, which consisted of a leveraged agent utilising VIX regime switcher, was able to achieve a return of 231% in contrast to 100%, the Sharpe ratio of 2.6 in comparison to 1.5 with a similar maximal drawdown of 41% as buy and hold strategy.

Results also show that a state representation system as the solution to partial observability of the trading environment should be improved. The key findings are the following. Firstly, Q-learning with extended leverage action can overperform the buy and hold strategy with a similar maximal drawdown. Secondly, market regime switch is a powerful technique to combine reinforcement learning agent with.

Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Brno, 9. May 2024

Author's signature

MUNI
ECON

MASARYKOVA UNIVERZITA
EKONOMICKO-SPRÁVNÍ FAKULTA
LIPOVÁ 41A, 602 00 BRNO
IČ: 00216224
DIČ: CZ00216224

ZADÁNÍ DIPLOMOVÉ PRÁCE

Akademický rok: 2023/2024

Student:	Bc. Lukáš Galeta
Program:	Matematické a statistické metody v ekonomii
Název práce:	Alokace aktiv pomocí reinforcement learning
Název práce anglicky:	Asset allocation with reinforcement learning
Cíl práce, postup a použité metody:	Cíl práce: Cílem této práce je navrhnout a vyhodnotit obchodní strategie založené na alokaci aktiv pomocí metod reinforcement learning. Autor práce by měl navrhnout vlastní strategie a porovnat jejich výkonnost se standardními strategiemi. Postup psaní diplomové práce: Vytvoření přehledu alokace aktiv pomocí metod strojního učení a metod vyhodnocení úspěšnosti investičního portfolia. Přehled existujících výzkumů s využitím různých strategií. Po vlastním návrhu investiční strategie bude následovat vyhodnocení navržených investičních strategií pomocí vhodných ukazatelů. Potom navrhneme několik doporučení pro různé investory. Použité metody: Analýza výkonnosti portfolia, vnitřní výnosové procento, drawdown, rizikově očištěný výnos, reinforcement learning. Práce v jazyku R, Python nebo jiný.
Rozsah grafických prací:	Podle pokynů vedoucího práce
Rozsah práce bez příloh:	60 – 80 stran
Literatura:	YANG, Bing, Ting LIANG, Jian XIONG a Chong ZHONG. Deep reinforcement learning based on transformer and U-Net framework for stock trading. *Knowledge-Based Systems*. AMSTERDAM: Elsevier, 2023, roč. 262, 10 s. ISSN 0950-7051. Dostupné z: https://dx.doi.org/10.1016/j.knosys.2022.110211. JANG, Junkyu a NohYoon SEONG. Deep reinforcement learning for stock portfolio optimization by connecting with modern portfolio theory. *Expert Systems with Applications*. OXFORD: Elsevier, 2023, roč. 218, 12 s. ISSN 0957-4174. Dostupné z: https://dx.doi.org/10.1016/j.eswa.2023.119556. NGO, Vu Minh, Huan Huu NGUYEN a Van Nguyen PHUC. Does reinforcement learning outperform deep learning and traditional portfolio optimization models in frontier and developed financial markets? *Research in International Business and Finance*. AMSTERDAM: ELSEVIER, 2023, roč. 65, 27 s. ISSN 0275-5319. Dostupné z: https://dx.doi.org/10.1016/j.ribaf.2023.101936. R, Harvey C a Y LIU. Evaluating Trading Strategies. *The Journal of Portfolio Management*. 2014, roč. 40, č. 5, s. 108-118. Dostupné z: https://dx.doi.org/10.3905/jpm.2014.40.5.108. BERNARDI, M a Catania L AMP. The model confidence set package for R. *International Journal of Computational Economics and Econometrics*. 2018, roč. 8, č. 2, s. 144-158. Dostupné z: https://dx.doi.org/10.1504/IJCEE.2018.091037.
Vedoucí práce:	prof. Ing. Štefan Lyócsa, PhD.

Pracoviště vedoucího práce:	Katedra financí
Konzultant:	doc. Ing. Tomáš Výrost, PhD.

Datum zadání práce: 13. 7. 2023

Termín odevzdání diplomové práce a vložení do IS je uveden v platném harmonogramu akademického roku.

V Brně dne: 9. 5. 2024

Acknowledgements

I would like to express gratitude to my diploma thesis supervisor, prof. Ing. Štefan Lyócsa PhD. and to my consultant doc. Ing. Tomáš Výrost, PhD. I would like to thank them for their professional guidance and patience.

I would like to also thank my family, partner and closest friends for supporting me in my efforts.

Contents

Introduction

Jesse Livermore, pioneer of stock trading, once said, “I don’t believe in luck, I believe in probability.” (Lefèvre et al., 2010). It is not sure if Jesse Livermore, at the turn of the 19th and 20th centuries, ever thought about machines which would predict the stock price based on historical data, machines which would buy and sell in a portion of seconds, and machines which would trade consistently. The approach today is known as quantitative algorithmic trading.

This diploma thesis is about a machine, an agent, who acts like a human trader. This diploma thesis presents a reinforcement learning framework for the portfolio allocation problem.

There are several reasons why quantitative algorithmic trading is a powerful approach for investing and trading. Firstly, quantitative algorithmic trading is consistent. Algorithms strictly follow their internal decision process heuristics. This behaviour mitigates the effect of inconsistency in trading, which is one of the most common challenges for retail traders.

Secondly, algorithms can be trained on historical data and evaluated on unseen test data. Correctly executed backtesting may precisely estimate future characteristics of a trading strategy. This utilisation of probability and optimisation techniques on historical data makes quantitative algorithmic trading a well recognised approach. Lastly, quantitative algorithmic trading incorporating various machine learning techniques can find hidden patterns in data that are not observable by humans. For these reasons, it is worth continuing in this field.

The foundations of quantitative trading were probably laid by Louis Bachelier and his doctoral thesis "Theory of Speculation" in 1900 (El Karoui and Parent, 2022). Bachelier provided a model for option pricing under a normal distribution.

Back then, most of the models focused on option pricing. Harry Markowitz (1952) proposed the mean-variance model, the key concept of Modern Portfolio Theory. A mean-variance portfolio uses expected return and variance to evaluate the performance of a portfolio. This leads to a portfolio of assets such that the expected return is maximised for a given level of risk. A mean-variance portfolio is well recognised, and it is commonly utilised as a benchmark with the buy and hold strategy. Edward Thorp (1975) introduced the usage of the Kelly criterion in portfolio selection and investment strategy optimisation.

Thorp also utilises probability theory in blackjack, introducing a method known as counting (Patterson, 2010).

The development of computer science in the 1990s gave rise to a new branch in quantitative trading known as quantitative algorithmic trading and the utilisation of biology inspired algorithms in quantitative algorithmic trading.

Trippi and Desieno (1992) were among the first papers to introduce trading with neural networks. Genetic algorithms with feature discretisation neural networks in trading were adopted by Kim et al. (2000). Yao et al. (2000) focused on option price forecasting using neural networks. Chen et al. (2003) published a paper on forecasting an emerging financial market with neural networks. Hassand and Nath (2005) introduced market forecasting using hidden Markov model. Hau (2001) investigated the impact of distance on high frequency and latent trading. These were some of the first research studies that created quantitative algorithmic trading, which is known today.

Stock returns prediction with long short-term memory recurrent neural networks was discussed by Chen et al. (2015), and trading with recurrent neural networks was discussed by Fabbri and Moro (2018). Qiu et al. (2020) discussed long short-term memory with attention mechanism. More recent advantages in graphical pattern recognition for trading were introduced by Chen and Tsai (2020) and the expansion of the multi-modality graph neural network by Cheng et al. (2022). WU et al. (2020) focused on adaptive stock trading strategies based on deep reinforcement learning. Yang et al. (2023) combine deep reinforcement learning with transformer. Jang and Seong (2023) discussed deep reinforcement learning and connection to modern portfolio theory. All presented methods leverage statistics to create models leading to higher returns and lower maximal drawdowns. Investors utilising these systems may not be exposed to such significant losses as those relying on different methods.

This diploma thesis aims to introduce and evaluate a simple reinforcement learning framework for asset allocation. The whole research focuses on the research question: How can reinforcement learning framework be effectively applied to optimise portfolio allocation in financial markets. The main hypothesis in this diploma thesis is the following. Portfolio allocation based on reinforcement learning framework overperforms buy and hold allocation strategy. The second hypothesis is as follows. A reinforcement learning agent combined with a market regime switch decreases the maximal drawdown of the portfolio compared to an agent without a market regime switch.

Reinforcement learning is a machine learning model where an agent observes the environment and communicates with the environment through a set of actions. This diploma thesis utilises a Q-learning agent with fixed discrete actions. Q-learning algorithm assigns values to state action pairs based on iterative calculations of their Q value. The reinforcement learning framework consists of training agent on S&P 500 data from 1996 to 2015 and evaluating performance on test unseen data from 2015 to 2024.

It is shown that agents can observe only close, high and low prices, which are insufficient for informed decision making. As a result, the portfolio allocation problem is formulated as a partially observable Markov decision process. A new state representation system has been introduced to represent states. It is shown that a newly introduced system has to be able to represent similar states in the same way and distinguish between different states.

States are generalised into clusters in order to deal with partial observability of the trading environment. States are clustered into states, and values of actions are averaged according to states in the same cluster. Values of actions are assigned to unseen test states corresponding to an average of clusters.

The paper is organised as follows. The first chapter familiarises the reader with research in reinforcement learning based portfolio optimisation and trading techniques. The second chapter consists of the theory behind problem modelling and reinforcement learning. The following chapter introduces the reinforcement learning framework, describes state action design, and introduces a state representation system. The fourth chapter presents the results. Each simulation consists of a description, figures, and tables for clarity. In chapter 5, the results are confronted with their limitations and compared with those of other authors. Challenges and future improvements are also presented here with a simple estimation of the expense ratio. The conclusion is the last chapter of this diploma thesis.

1 Literary review

One of the first researchers to highlight the potential of reinforcement learning in trading and portfolio management tasks is Moody et al. (1998). In his paper, Moody distinguishes between supervised learning and reinforcement learning in trading and portfolio optimisation. He shows that supervised learning techniques are used to forecast price movements, and forecasts are used as input for the trading module afterwards. The challenge comes with data labelling since the portfolio optimisation problem is problematic to formulate as a supervised machine learning problem.

In contrast, reinforcement learning algorithms can approximate solutions to stochastic dynamic programming problems, just as trading environments do. Moody focused on a recurrent reinforcement learning agent with three actions, trading S&P 500 and Treasury Bills. Recurrent reinforcement learning agent optimises trading systems directly by utilising objective function. Differential Sharpe ratio was selected as the objective function to maximise. It is shown that agents with differential Sharpe ratio task achieve better risk-adjusted returns than agents trained to maximise profit. They show in the testing period from 1970 to 1994 that they could overcome buy and hold strategy. Moving average of annualized Sharpe ratio of their strategy is persistently higher, reaching up to 2.5. Moreover, their annualized Sharpe ratio reaching difference up to 1, compared to moving average of annualized Sharpe ratio of buy and hold strategy.

Moody continued his work in the following paper (2001), extending his current research with Q-learning and other methods. Various researchers continued in this field, such as Gao and Chan (2000), Dempster et al. (2006), Lee et al. (2007), and Du et al. (2016).

In recent years, two main approaches have been used to model the trading environment and train reinforcement learning agents: discrete action space modelling and continuous action space modelling. Both methods provide various advantages and disadvantages, some of which are introduced.

1.1 Discrete action models research

The discrete action space modelling approach is more straightforward and less complex than continuous action space modelling. Discrete action models allow only discrete actions with specific amounts, such as buy 50 shares, sell 20 shares, etc. These algorithms, such as Q-learning which learns the value of an action *a* in a particular state *s* by visiting the state *s* and taking action *a* (Watkins, 1989). In contrast, the extension of Q-learning and deep Q-learning using a neural network approximates the Q-value function for ample state space (Mnih, 2013).

These discrete action space architectures are straightforward and uncomplicated to implement, which is their advantage. On the other hand, discrete action space reinforcement learning algorithms have limited capabilities caused by limited possible actions. Hence, there could be as good or better actions in continuous action space than in discrete action space.

Discrete action space algorithms are well-known and have been commonly employed for algorithmic trading, as Chakole et al. (2021), Théate and Ernst (2021), Jeong and Kim (2019), Shi et al. (2021), and Pendharkar and Cusatis (2018) show.

Chakole et al. (2021) focused on Q-learning with two different state representation systems. The first state representation system is based on clustering similar states with K-means clustering algorithm with 3, 6 and 9 clusters, where the Commodity Channel Index, Relative Strength Index, and (%R) Williams Percent Range are used to represent states. The second state representation system is based on their OHLC candlestick data calculation. Results are compared with buy and hold strategy and decision tree predictions. Unfortunately, in their testing period from 2006 to 2018, their proposed models did poorly regarding the Sharpe ratio, although returns were higher than benchmark models.

More specifically, at NASDAQ, the decision tree and buy and hold strategy achieved Sharpe ratio of 1.24 and 1.90, with Average Annual Return of 6.6% and 14.8%. Two proposed models achieved a Sharpe ratio of 0.40 and 0.52, with an Average Annual Return of 19% and 23.5%. At SENSEX, the Decision tree and buy and hold strategy achieved a Sharpe ratio of 1.99 and 1.82, with an Average Annual Return of 13.7% and 13.5%. Two proposed models achieved a Sharpe ratio of 0.26 and -0.13, with an Average Annual Return of 32.4% and 16.5%.

Pendharkar and Cusatis (2018) focused on on-policy SARSA (State-action-reward-state-action algorithm) and off-policy Q-learning agents. Both models were tested with two different objectives. The first objective was return maximisation, and the second objective was differential Sharpe ratios. Agent portfolio consists of two assets, the S&P 500 and a 10-year U.S. Treasury note. Different agents are tested on different test samples and frequencies. Unlike other researchers, Pendharkar and Cusatis focus on quarterly, semi-annual and annual rebalancing periods. Their results show that most of their agents accumulated approximately 98% of their capital in the S&P 500 and the rest in bonds. They also point out that most agents considered in their research cannot beat both indices consistently. They also highlight that agents that maximalise differential Sharpe ratios have slightly better performance than agents' with return maximisation objective and that annual trading frequency gives better results than quarterly or semi-annual trading frequency.

Théate and Ernst (2021) adopt a different approach. Instead of rebalancing portfolio in a predefined period, they let their agents trade on higher frequency data. Agents dispose with only two actions buy and sell. The agent's objective is to maximalise total discounted reward. To solve this problem, they formalise algorithmic trading decision-making into a reinforcement learning problem. They introduce Trading Deep Q-Network algorithm based on Deep Q-Network architecture. Buy and hold together with trend following and mean reversion strategies are used as benchmarks. The proposed Trading Deep Q-Network algorithm is tested on various stocks and indices from 2018 to the end of 2019. Results show that the proposed agent has slightly better results than buy and hold strategy measured by the Sharpe ratio. Other strategies have significantly worse performance.

Trading S&P 500, buy and hold achieves the Sharpe ratio of 0.83 and proposed agent 0.83. Results on Google shares are worse with buy and hold strategy Sharpe ratio 0.57 and proposed agent 0.23. On the other hand, results on Apple shares are in favour of the proposed agent, which achieved a Sharpe ratio of 1.42 while buy and hold strategy was 1.24. Unlike other authors, Théate and Ernst implement trading and other related trading costs as spread market impact and delays in orders, significantly impacting trading strategy performance.

Lastly, there is a paper by Jeong and Kim (2019). They combine deep Q-network with deep neural network. As a result, they propose a trading system that can determine the number of shares with a deep neural network and combine it with the action selected by deep Q-network. The number of shares to trade is predicted regardless of the trading action. Jeong and Kim also identify the uncertainty of financial markets as a crucial component of any algorithmic system. Thus, the proposed system is switching between trained and predefined actions. In practice, default action is taken instead of trained action if an uncertainty indicator exceeds a predetermined threshold. They also introduce transfer learning to handle the overfitting problem caused by noisy and volatile financial time series. As a result, their proposed systems with all three components significantly increase profits trading S&P 500 and EuroStoxx50, which overcomes the classical reinforcement learning approach.

All models and approaches introduced in the papers above employ discrete action space, which could lead to sub-optimal actions in the trading environment. For instance, an agent with discrete action space selects between actions to buy shares worth of: 10% of capital, 50% of capital, 100% of capital, etc. The agent cannot select optimal action, such as buying shares worth 78% of capital, because the action is simply not available. The last paper from Jeong and Kim (2019) addresses the issue of a sub-optimal amount of shares to buy or sell by predicting the optimal amount of shares, where the predicted amount of shares comes from a continuous distribution.

As a result, deep Q-learning agent designed to interact with the environment through discrete actions interact with the environment with continuous actions. Although this framework is able to overcome various challenges, this framework is also facing new challenges. For instance, the agent's actions are weighted with the error between learned and optimal policy, as well as by the prediction error of the number of shares to trade, which could lead to worse performance than the traditional discrete action space model.

However, action space could be defined as buying shares worth 1% of capital, 2%, etc. This effectively transfers discrete action space into semi-continuous action space. The problem is computational complexity caused by large action spaces for traditional Q-learning and Deep Q-learning based agents as Van de Wiele (2020) shows.

1.2 Continuous action models research

To overcome the described problems, agents with continuous action space are introduced. These agents are Deterministic Policy Gradient, Deep Deterministic Policy Gradient and Twin Delayed Deep Deterministic Policy Gradient.

- Deterministic Policy Gradient (DPG) directly learns the policy, which maps states to actions using gradient ascent techniques. This algorithm uses an actor-critic approach, where the actor updates the policy based on suggestions from the critic, which evaluates the policy (Silver et al., 2013).
- Deep Deterministic Policy Gradient (DDPG) extends Deterministic Policy Gradient to work with high-dimensional state spaces using deep learning (Lillicrap, 2015).

- Twin Delayed Deep Deterministic Policy Gradient (TD3) extends the Deep Deterministic Policy Gradient with twin critics and delayed policy updates to deal with overestimation and improve algorithm stability (Fujimoto, 2013).

Policy-based algorithms are more advanced and complex than Q-learning and Deep Q-learning. They are beneficial for multi-asset portfolio optimisation, where the state and action space are too large. There are two reasons for that. Firstly, these methods optimise the policy directly without explicitly using a value function. This approach is efficient in complex environments where value-based methods may struggle due to the difficulty of maximising over the action-value functions (Sutton, 1999). Secondly, policy-based methods can converge to a stable policy by continuously improving it in small steps. This is advantageous in unstable and volatile financial time series (Nachum, 2017).

Introduced algorithms are capable of interacting with continuous state space environments. This feature is particularly useful in environments with higher level of complexity, for example, robotics and autonomous self-driving cars. Transferring the robotics environment from the real world into a reasonable discrete state space environment is challenging to overcome. On the other hand, in the trading environment, continuous states could be helpful, although not necessary. While trading tick data, continuous state space representation is obligatory. In contrast, discrete state space could be used in trading, day, week or month data.

Wu et al. (2020), Kabbani and Duman (2022), Xiong et al. (2018), etc., focused on the utilisation of policy-based algorithms in trading and portfolio management.

Wu et al. (2020) discuss the complexity and dynamical property of stock markets, where changing conditions are key challenge. They propose adaptive stock trading strategies with deep reinforcement learning methods. The key concept is to use Gated recurrent unit to extract important features from all features, which consist of Exponential moving averages, moving average convergence divergence and other indicators. Gated recurrent unit extract different features based on market conditions. Wu et al. (2020) show that these dynamically extracted features represent the intrinsic characteristics of the stock market and could be used more effectively with followed reinforcement learning models than fixed features.

Consequently, they propose two trading strategies with reinforcement learning models. The first is the Gated Deep Q-learning trading strategy, which combines a state representation control mechanism with deep Q-learning. It uses dynamically selected features as described above. The second model is the Gated Deterministic Policy Gradient trading strategy, where the gate unit has the same function as in the previous model.

Proposed models were trained from 2008 to the beginning of 2016 and tested from 2016 to the end of 2018. They are compared to state-of-the-art direct reinforcement learning models. Results on Apple stock show that the Gated Deep Q-learning agent achieved a Sortino ratio of 1.02, and the Gated Deterministic Policy Gradient agent achieved a Sortino ratio of 1.30. In contrast, the direct reinforcement learning agent achieved a Sortino ratio of only 0.25.

Kabbani and Duman (2022) formulate the trading problem as a partially observed Markov decision process. They use two types of data to represent states. Firstly, they use time series description statistics such as Relative Strength Index, Simple Moving Average, Stochastic Oscillator, etc. Secondly, they use sentiment data from the BERT model to represent states with sentiment scores for each asset in the portfolio. They work with 10 asset portfolio where major stocks like Apple, Microsoft, and Goldman Sachs are available. The formulated trading problem is solved using the Twin Delayed Deep Deterministic Policy Gradient. On test data from 2016 to the end of 2017, they reported a Sharpe ratio of 2.68, which could be considered as "good" by investors.

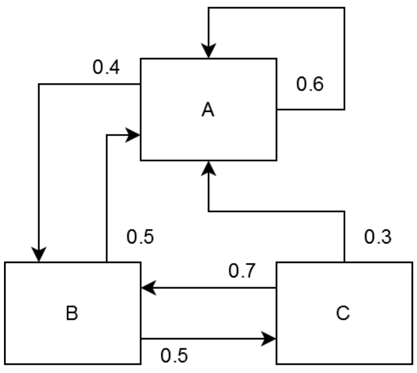
Xiong et al. (2018) focused on a Deep Deterministic Policy Gradient applied to a set of 30 stocks in which the agent's reward function is portfolio Return Maximization. The proposed model and its portfolio achieved a Sharpe ratio of 1.79, while the Min-Variance portfolio achieved 1.45 and the Dow Jones Industrial Average index 1.27 from 2016 to 2018 on test data. Their results demonstrate that the Deep Deterministic Policy Gradient strategy can outperform the benchmark Dow Jones Industrial Average and the minimum variance portfolio.

2 Theory behind problem modelling

This chapter introduces the theoretical background behind modelling the trading environment and finding solution with reinforcement learning. The chapter starts with the Markov chain and Markov decision process, followed by policy and Bellman equations. The last part consists of Q-learning.

2.1 Markov chain

A mathematical framework, the Markov chain and its extension, the Markov Decision Process, is needed to model the reinforcement learning problem. Andrey Markov developed Markov chains in 1906 (Gagniuc, Paul, 2017). A Markov chain is a stochastic model describing a sequence of states, where the sequence is stochastic and given by a transition matrix (Markov, 1913 republished 2006).



Source: own processing

Figure 1: Example of a Markov chain

Figure 1 above describes the Markov chain. This Markov chain has three states: state A, B, and state C, where state A is the initial state. Transition probabilities from state A to B are 0.4, and from A to A are 0.6. The agent can potentially stay in the A to A loop. The agent cannot move from state A to state C directly. It has to visit state B in order to get to C. In contrast, the agent can visit state A from state C directly with a probability of 0.3.

A Markov chain is a stochastic process in which all future states are determined by current states. Thus, a Markov chain satisfies the Markov property, formally written as equation 2.1. The probability of transitioning to the next state S_{t+1}, depends only on state S_t. Thus, the Markov property indicates that predictions based on current states should be as precise as predictions made with knowledge of previous states. Consequently, previous and future states are independent (Markov, 1913 republished 2006).

$$P(s_{t+1} \mid s_t) = P(s_{t+1} \mid s_1, \ldots, s_t) \tag{2.1}$$

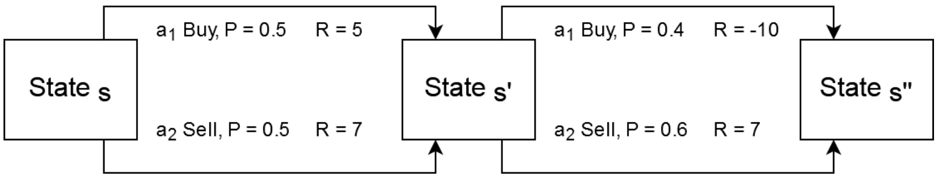
Markov chain could be described in discrete or continuous time and state space. From continuous time and state space comes the Wiener process introduced by Wiener (1923). The Wiener process is crucial to financial time series modelling and financial derivatives valuation. Markov chain is recognised in many areas, such as Mathematics, Physics, Chemistry, and Informatics.

2.2 Markov decision process

Markov decision process (MDP) was introduced by Bellman (1957) in the 1950s and formally describes an environment for reinforcement learning. MDP can describe most of the reinforcement learning problems.

MDP is defined as the tuple (S, A, P, R), where:

- S is a finite set of states with an initial state s_0,
- A is a finite set of actions available to the agent,
- P corresponds to transition probabilities from state s to state s', given action a, denoted by $P_a(s, s') = P(S_{t+1} = s' \mid S_t = s, A_t = a)$,
- R is a reward function $R_a(s, s')$.



Source: own processing

Figure 2: Example of trading MDP

Figure 2 shows an example trading MDP environment. Please note that the trading environment can be modelled as MDP differently. Actions a_1 and a_2, corresponding to

Buy and Sell. This environment is deterministic, sequential, and single agent based. The agent selects an action based on stochastic policy described with probability P. States represent trading days.

R denotes rewards from selecting particular actions in states. These rewards are not known before taking action. In the middle state, it is shown that rewards from opposite actions could be asymmetrical. It depends solely on the reward design of the environment. With different reward functions, different behaviours can be learned.

Note that the agent will transit to the same next state regardless of the selected action. In this particular case, the agent traverses state space with a given state order. The agent's objective is to traverse from the initial to the terminal state and accumulate the maximal possible reward during traversing.

2.2.1 Reward process

The agent's objective is to maximise the long-term reward over all visited states from the initial state to the terminal state. The agent's example total reward could be formulated as equation 2.2 (Bellman, 2015).

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.2}$$

The return G_t is the total discounted reward from time-step t, and R_t are future rewards calculated retroactively. γ is a discount factor where $\gamma \in [0, 1]$, which discounts future rewards to present value rewards (Bellman, 1953).

The purpose of discounting is to highlight immediate rewards over delayed rewards. A discount factor close to 0 leads to short-sighted evaluation, and delayed rewards have almost no impact. On the other hand, a discount factor close to 1 leads to far-sighted evaluation.

2.2.2 Discounting

It is common to discount rewards in MDPs for various reasons. Firstly, it is mathematically convenient to discount rewards since the agent can avoid infinite returns in the case of the cyclic Markov process. An example of the cyclic Markov process is a process where the state s has a transition probability to states *s* equal to 1, while the state s is not a terminal state as Foster (1953) shows.

Secondly, financial rewards are subject to financial interest. Immediate rewards can earn more interest than delayed rewards in future as it is incorporated into standard financial mathematics equations. Another reason is that humans and animals tend to prefer immediate rewards over delayed rewards. Consequently, reward discounting is a way to mimic animal and human behaviour more precisely (Russell, Norwig, 2016).

Lastly, discounting future rewards is a way to incorporate uncertainty into the decision process. Rewards significantly distant should not have the same probability weight as close rewards whose probability is close to certain.

2.3 Partially observable Markov decision processes

The partially observable Markov Decision process (MDP) can model problems where the environment is fully observable. The problem which arises is partial observability of the trading environment. Åström (1965) describes MDP as a process where information about the environment is incomplete. The agent observes only the open, low, high, close price and traded volume. This information is not sufficient to represent states properly. The agent is not able to observe the whole trading environment, which is extremely complex since stock prices reflect the whole economy.

For example, a company sells products or services to customers based on its business model. Sales could or could not depend on the product's economic cycle. A company's current profits and, more importantly, future profits determine its share price.

Let's assume that there is a cluster of companies. Now, agents participating in financial markets decide which share to buy and which share to sell. Share price is determined by supply and demand. Moreover, states are basically created by the actions of other agents who participate in financial markets.

More aspects should be mentioned, such as interest rates, relationships between particular tradable assets, latention trading, and trading based on exclusive or insider information.

All these aspects are crucial for fully observing the trading environment. For these reasons, the trading environment is created as an outcome of other partially observable to almost non-observable environments.

In the case of a partially observable environment, the Agent must construct its own state representation (Russell, Norwig, 2016). For this reason, the agent is creating new explanatory variables to describe states. The agent uses only open, low, high, and close price and volume time series. Consequently, with better state representation, more informed decisions can be made even if the agent is not able to observe the environment completely. Agents transfer partially observable environments into more observable environments with their own state representation systems (Russell, Norwig, 2016). Different steps are taken during the simulation to transfer trading POMDP to MDP. As a result, the trading environment is modelled and considered as a fully observable environment for the needs of this diploma thesis.

2.4 Policies

The agent's objective is to maximise total reward in MDP. The way to select the best actions is policy. It is common to denote policy as π. A policy π is a distribution over actions given states (Puterman, 2014).

$$\pi(a, s) = P(A_t = a \mid S_t = s) \tag{2.3}$$

Policy specifies which action the agent should take for any state that the agent reaches. Hence, a policy fully defines the behaviour of an agent. MDP policies do not depend on the previous state, MDP policies depend only on the current state. As a result, the policy is stationary (Russell, Norwig, 2016).

There are deterministic policies.

$$\pi \colon S \to A \tag{2.4}$$

In this type of policy, the agent selects actions with certainty, given the current state. In other words, given a state, policy always maps the state to the same action.

And there are stochastic policies.

$$\pi \colon S \times A \to [0, 1] \tag{2.5}$$

In this type of policy, the agent selects actions with a certain probability distribution over the action space, given the current state. This means that the policy may select different actions with different probabilities for a given state.

The problem is how to determine the right policy. Assume a fully stochastic policy with a uniform distribution. This policy will probably not lead to the maximum possible rewards. Hence, a policy's quality could be determined by measuring how good the results of selected actions are given the policy.

An optimal policy maximises the total potential reward, denoted as π^*. Given π^*, the agent executes the action $\pi^*(s)$, as Puterman (2014) shows. It is necessary to measure how valuable it is to be in a particular state or how profitable it is to take action in order to determine the optimal policy.

2.4.1 Value functions

Bellman (1957) introduces two common value functions, all functions and key concepts below were introduced in his work. The state-value function represents the expected cumulative reward G_t that an agent can achieve starting from state s and following policy π.

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] \tag{2.6}$$

It considers all possible future rewards that the agent might receive while following the policy. In other words, the state value function says how good it is to be in a particular state.

The action-value function represents the expected cumulative reward G_t that an agent can achieve starting from state s, taking action a, and then following policy π.

$$Q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \tag{2.7}$$

It considers all possible future rewards that the agent might receive while following the policy. In other words, the action value function says how good it is to take action in a particular state s following policy π.

Furthermore, optimal Value Functions can be expressed.

The optimal state value function, $V^*(s)$, represents the maximum expected return that an agent can achieve starting from state s over all policies.

$$V^*(s) = \max_{\pi} V_{\pi}(s) \tag{2.8}$$

Similarly, the optimal action-value function, denoted $Q^*(s, a)$, represents the maximum expected return that an agent can achieve starting from state s and selecting action a over all policies.

$$Q^*(s, a) = \max_{\pi} Q_{\pi}(s, a) \tag{2.9}$$

Optimal value functions represent the best possible performance an agent can achieve in MDP. An MDP is considered "solved" when $Q^*(s, a)$ is known. Once it is known, the agent knows the best action to take in each state to maximise its expected return.

2.4.2 Optimality theorem

As explained above, the optimal value function represents the maximum expected return that an agent can achieve starting from state s and selecting action a over all policies. Policy is mapping states to actions.

The question is which policy leads to the optimal value function. The optimal policy describes how an agent should behave optimally to maximize the expected return. Partly ordering over policies determines which policy is better than another policy (Bellman, 1957).

$$\pi \geq \pi' \text{ if } V_{\pi}(s) \geq V_{\pi'}(s), \forall s \tag{2.10}$$

This equation is comparing two policies π and π'. π is at least as good as or better than π' if the value function of π is at least as good as or better than π' for all given states.

This equation implies that policy π can be recognised as better than π' if the value function of π is better or at least as good as π' in all states with at least a value function from one state better. The policy can not be better in 80% of states[1] and worse in 20% of states and then be recognised as better.

Consequently, Bellman's (1957) optimality theorem for MDP was introduced. For any MDP, there exists at least one optimal policy π^* that is better than or equal to all other policies.

$$\pi \geq \pi^*, \forall \pi.$$

"All optimal policies achieve the same optimal value function $V^*(s) = V_{\pi^*}(s)$ for any state s."

This means that the value function of the optimal policy is equal to the optimal value function for all states.

"All optimal policies achieve the same optimal action-value function $Q^*(s,a) = Q_{\pi^*}(s,a)$."

This means that the action-value function of the optimal policy is equal to the optimal action-value function for all state-action pairs.

"An optimal policy can be found by maximizing over the optimal action-value function $Q^*(s,a)$."

$$\pi^*(a,s) = \begin{cases} 1 & \text{if } a =_{a' \in A} Q^*(s,a') \\ 0 & \text{otherwise} \end{cases}$$

The optimal policy selects an action with probability 1 that maximizes $q^*(s,a)$. This action will provide maximal reward. There is always a deterministic optimal policy for any MDP.

2.5 Bellman equations

One approach to solving MDPs is through the Bellman equations, which Bellman (1957) introduced. All the following equations come from his work Dynamic Programming. The Bellman equations describe the relationship between the value of a state or state-action pair and the value of its successor states. These equations are an important part of dynamic programming and reinforcement learning algorithms for finding optimal decision-making in sequential decision processes.

1. Assuming a finite number of states, this would typically be interpreted as a majority of states.

The state-value function under policy π is given by:

$$V_\pi(s) = \sum_a \pi(a,s) \sum_{s',r} P(s',r|s,a)[r + \gamma V_\pi(s')] \tag{2.11}$$

$V_\pi(s)$ represents the value of state s under policy π, $\pi(a,s)$ denotes the probability of taking action a in state s, and $P(s',r|s,a)$ is the transition probability to state s' from state s, with reward r when action a is taken and γ is the discount factor.

The Bellman Optimality Equation for state-value captures the relationship between the value of a state and the value of its successor states under the optimal policy.

$$V^*(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V^*(s')] \tag{2.12}$$

$V^*(s)$ represents the optimal value of state s, given by selecting the best possible action a in each state, which maximises the sum of the reward and the discounted value of the next state.

The term $\max_a$ denotes to selecting the best action a over all possible actions in state s, $\sum_{s',r} P(s',r|s,a)$ denotes the sum over all possible next states s' and rewards r, weighted by their transition probabilities under the action a taken in state s, $V^*(s')$ represents the optimal value of state s' and γ is the discount factor.

The Bellman Optimality Equation for the action-value function is the function representing the maximum expected return starting from state s, taking action a, and then following the optimal policy.

$$Q^*(s,a) = \sum_{s',r} P(s',r|s,a)[r + \gamma \max_{a'} Q^*(s',a')] \tag{2.13}$$

$Q^*(s,a)$ represents the optimal value of taking action a in state s. $\sum_{s',r} P(s',r|s,a)$ is the sum over all possible next states s' and rewards r, weighted by their transition probabilities under the action a taken in state s. The last term represents the discounted maximum from the optimal value of taking action a' in state s'.

Analytical solutions to the Bellman equations can be derived in simple MDPs. However, due to the high dimensionality of the state and action spaces, analytical solutions are not feasible for complex problems. Howard (1960) introduced a numerical iterative solution.

First among these methodologies is value iteration. This method iteratively applies the Bellman Optimality Equation for the value function until convergence to the optimal values. At each iteration, the value function is updated by taking the maximum over all possible actions in each state (Howard, 1960).

The second method is the policy iteration algorithm. This method alternates between policy evaluation and policy improvement stages. In the policy evaluation step, the value function for the current policy is computed using the Bellman Expectation Equation. During the policy improvement step, the policy is updated to maximise the current value function (Howard, 1960).

While these methods are effective, they require a model of the environment. A model-based approach leverages the explicit model to refine optimal policies or value functions iteratively. However, this approach requires a priori knowledge of the environment's transition probabilities and rewards, which may not be available (Russell, Norwig, 2016).

Russel and Norwig also describe that, unlike the model-based approach, model-free reinforcement learning algorithms, such as Q-learning (Watkins, 1989) and SARSA (Rummery, Niranjan, 1994), do not require a model of the environment to be known for the algorithm. Model-free algorithms can navigate environments with unknown dynamics.

More specifically, these algorithms learn directly from observed experiences, iteratively refining action-value estimates towards optimality through empirical feedback from the environment in the form of rewards.

2.6 Q-learning

Watkins (1989) introduces Q-learning as an iterative algorithm to solve Bellman equations. Q-learning learns an action-value function $Q(s, a)$, which estimates the expected cumulative reward of taking action in a state s while following an optimal policy.

The objective of Q-learning is to find the optimal action-value function $Q^*(s, a)$, which gives the maximum expected cumulative reward while selecting the action with the highest Q value. In other words, the $Q(s, a)$ function provides expected "utility" for any given action and state.

Q values from the $Q(s,a)$ function are estimated in an iterative process, where the initialisation state of Q values are commonly zeros. Each interaction between the agent and environment updates Q-values. Through this process, the agent gradually improves its estimates of the optimal action-value function, converging towards $Q^*(s,a)$.

Iterative estimation of $Q(s,a)$ function can be observed below.

$$Q^{new}(s,a) \leftarrow (1-\alpha) \cdot Q^{old}(s,a) + \alpha \cdot [r + \gamma \cdot \max_{a'} Q(s',a')] \tag{2.14}$$

α represents the learning rate $0 \leq \alpha \leq 1$, higher learning rate determines the weight of newly obtained value over known knowledge, r denotes the immediate reward received upon taking action in a state s. The term $\max_{a'} Q(s',a')$ stands for the maximum Q value over all possible actions in the next state s', and γ is the discount factor, $0 \leq \gamma \leq 1$.

2.6.1 Exploit explore trade off

As it was mentioned before, an agent is learning through experience which is gained by interacting with the environment. One of the most significant benefits of Q-learning and similar agents is the ability to exploit or explore the environment.

At each state, an agent with probability $1-\epsilon$, where $0 \leq \epsilon \leq 1$, selects exploit, which stands for selecting the action with the highest Q-value, or explore, an alternative actions probability ϵ, which stands for selecting a random action. This strategy ensures that the agent continues to explore the environment, even as its knowledge grows, thereby preventing convergence to suboptimal solutions, as Sutton and Barto (2018) show.

In other words, the agent tries different random approaches to maximise reward while traversing state space. A novel found solution could be exploited with an exploitation approach, so a new solution is adopted. Moreover, it is common to use decay schedules for ϵ and focus more on exploitation with an increasing number of lapsed episodes.

It should be mentioned that the number of lapsed episodes affects the exploration rate which is balanced with the need for stability, efficiency and moving from a suboptimal solution to a more optimal one. Moreover, the conditions to explore or to exploit, selection and approach should grow in sophistication with an increasing number of lapsed episodes.

Consequently, if the needed magnitude of episodes is provided, the agent is able to exploit any weakness in the environment architecture based on the complexity of the state and action space.

2.6.2 Initialising the Q-values with non-zero values

Complex state and action space could be a great challenge for the Q-learning model to overcome (Russell, 2016). The initial matrix of Q-values is filled with zeros. Thus, the agent selects actions randomly, since all the actions have the same zero value. It takes a number of epochs to start reinforcement learning agent converging properly, based on the complexity of state and action space.

One possible solution is to initialise the matrix of action values to be more informative with a prior values of actions (Sutton, Barto 2018). This could be problematic in various fields, although in finance time series, prior values of actions could be roughly estimated. Financial markets have been trending upwards long-term in recent decades. This finding could be transferred into the Q value matrix. For example, actions focused on buying assets, namely speculation on an upward trend, could have higher initial Q-values than actions focused on selling assets.

Sutton and Barto (2018) discussed a different approach: optimistic initialisation. Q-values are initialised to values higher than the agent is expected to receive. Then, taking action with the highest value, results in decreasing value of selected action and another action with higher values is selected in the next epoch. This effectively forces the agent to explore environment since all state-action pairs are attractive.

2.6.3 Reward functions and Q-values

The reward function maps each state-action pair to a scalar value, representing the immediate feedback received by the agent upon executing a particular action in a specific state. Rewards could be positive, rewarding the agent for desirable actions or negative, punishing the agent for non-desirable actions.

There could be sparse or dense rewards. Sparse rewards reward agents for achieving great progress in the environment, such as achieving an objective terminal state or traversing through some kind of checkpoints (Russell, 2016). In contrast, Dense Rewards reward agents often and for smaller steps.

Sparse and dense rewards are closely connected to the Credit Assignment Problem. Sparse rewards are considered as delayed, it can be challenging for the agent to determine which actions were responsible for the received rewards, complicating the learning process, as Watkins (1989) shows.

Another challenge is balancing short-term and long-term rewards. Short-term rewards motivate agents to be short-sighted, and long-term rewards to be long-sighted. It depends on the desirable agent's behaviour and environment architecture how to balance rewards (Sutton, Barto, 2018).

In addition, as the Q-learning equation above shows, the Q value is not determined solely by reward. Moreover, $\max Q(s', a')$ is also added to the Q value. In reality, this could lead to interesting situations. For example, immediate rewards based on transitioning from state s to s' could be negative. The agent had to overcome some obstacle, and the agent could be damaged. On the other hand, the agent could transit to a high-value state. Thus, the value from the next state given by the term $\max Q(s', a')$ will outweigh the negative reward, and high value is assigned to the action taken.

More examples and different implications and effects of different reward functions in relation to investing are discussed later.

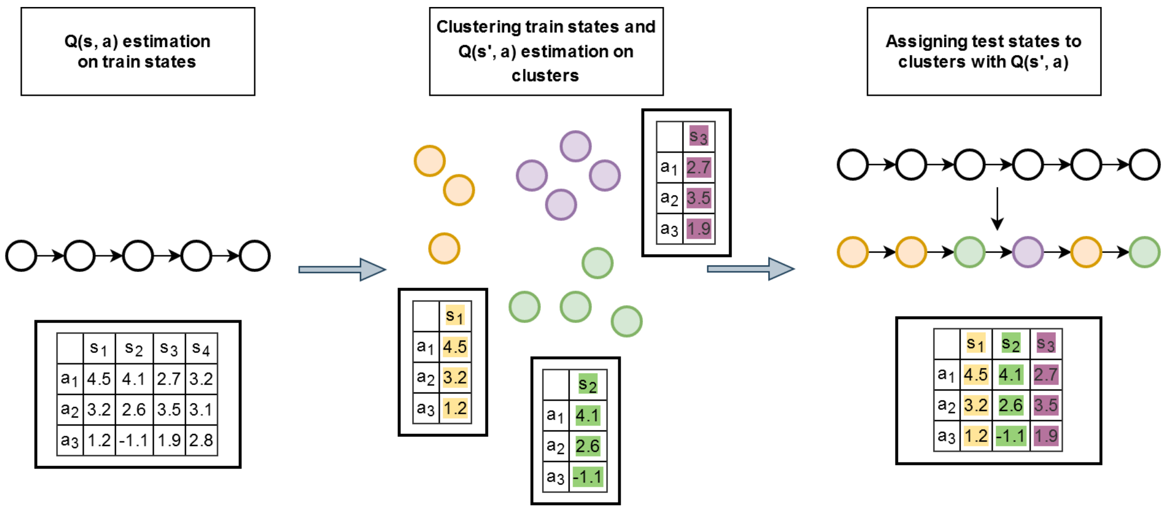
3 Simulation methodology

The previous chapter focuses on the theoretical explanation of the decision process behind the portfolio allocation problem and its solution, which is a reinforcement learning agent, specifically a Q-learning agent.

This chapter confronts the described theoretical methods with challenges arising from their application to real portfolio allocation problems. As a result, the reinforcement simulation framework, along with its vital components and evaluation techniques, is introduced.

3.1 Framework introduction

The simulation process, starting with Q-table initialisation and ending with backtesting on unseen data, is divided into three stages, observable in the Figure 3 below. These three stages are: $Q(s, a)$ estimation, clustering and $Q(s', a)$ construction and backtesting. The simulation approach is similar to Chakole's et al. (2021).


Q(s, a) estimation on train states
Clustering train states and Q(s', a) estimation on clusters
Assigning test states to clusters with Q(s', a)

	s_1	s_2	s_3	s_4
a_1	4.5	4.1	2.7	3.2
a_2	3.2	2.6	3.5	3.1
a_3	1.2	-1.1	1.9	2.8

	s_1
a_1	4.5
a_2	3.2
a_3	1.2

	s_2
a_1	4.1
a_2	2.6
a_3	-1.1

	s_3
a_1	2.7
a_2	3.5
a_3	1.9

	s_1	s_2	s_3
a_1	4.5	4.1	2.7
a_2	3.2	2.6	3.5
a_3	1.2	-1.1	1.9

Source: own processing

Figure 3: Simulation diagram

3.1.1 $Q(s, a)$ estimation

The objective of Q-learning is to estimate the $Q(s, a)$ function on training data. Q-values are initialised with zeros, and the Q-learning agent starts traversing from the initial state to the terminal state. Action is selected randomly or with a greedy policy as part of the exploration process. The Q-value is updated with the reward, and the agent moves to the next state. When the terminal state is reached, the agent returns to the initial state with updated Q-values from the last episode. There are n episodes. $Q(s,a)$ is estimated to have desirable accuracy with the last episode.

3.1.2 Clustering and $Q(s',a)$ construction

Training states are clustered into clusters with one of the clustering algorithms. Clustered states construct $Q(s',a)$, where s' stands for cluster, and the value of each action is the average of the same action over all states in the cluster. Newly constructed $Q(s',a)$ provides value for action a when cluster s' is given. Clustering does not change the environment or position of states in the environment.

Afterwards, the initialised clustering algorithm from previous steps assigns unseen test states to clusters. The clustering algorithm appears in this stage as a classification algorithm that classifies into clusters that were created by the same model. Generalising whole state space into a few cluster states with averaged action values significantly reduces state space and inevitably leads to loss of learned information.

In the first trial, the k-nearest neighbours algorithm (Cover, Hart, 1967) was used. The algorithm was able to assign the closest neighbours mathematically properly, although the states were not the same even if the values of explanatory variables were similar. In other words, similarity represented states have significantly different action values for the same actions. This is again an issue due to the fact that the trading environment is POMDP. As a result, clustering is used with the following benefits.

Firstly, with clustered states, states in particular clusters share common attributes and patterns described by explanatory variables. For example, with 3 clusters, it could be said that cluster 1 stands for the overpriced market, cluster 2 for the fairly priced market, and cluster 3 for the underpriced market.

Secondly, clusters consist of hundreds of states. Even if some states should be considered as a part of different clusters because their action values do not correspond to other states in the cluster, averaging of actions mitigates these effects. Consequently, when the unseen state is assigned into a cluster, action is assigned with the paradigm that historically, if the state is in cluster s, action a is the best action. Even if there are states where action a does not lead to good results, statistically, action a is still the best action. For these reasons, $Q(s', a)$suppresses the negative consequences of POMDP.

3.1.3 Backtesting

The last step is to backtest the strategy. There are two backtests. The first backtest is focused on training states, which are divided into clusters. Consequently, instead of $Q(s, a)$, $Q(s', a)$ is used and a greedy policy is applied. It is important to apply clustering also to an already seen states. The aim is to backtest the whole framework, not only certain parts of the framework. The second backtest focuses on unseen test states, which are also assigned to clusters and $Q(s, a)$ values for actions together with greedy policy is applied.

Both backtests start in the initial state. The action with the highest $Q(s', a)$ value is selected. The financial return is saved, and the agent moves to the next state. This process is repeated until the terminal state is achieved.

3.2 State action space design

Two approaches were considered for modelling portfolio allocation problem with MDP. The first approach is an environment with a two-dimensional state space design. States contain information about the market and agents' portfolios, such as how many stocks of which asset is held, account balance, etc. As a result, various states are linked to one trading day, and the same set of states is connected to the following trading day, etc. There is no firmly established sequence of states.

Consequently, when the agent selects and executes the action 'Buy', the agent receives a reward and moves to the state where the bought asset is part of the agent portfolio. The intuition behind this environment architecture is that the agent will receive a reward based on portfolio return and $\max_{a'} Q(s', a')$ while following optimal policy π. These values could be interpreted as how good it is to have some kind of portfolio in some type of market situation, corresponding to how good it is to have a particular stock in that specific market situation.

The problem arises from the extreme number of states needed to capture different properties. This will inevitably lead to the enormous number of iterations of Q-learning starting to converge to $Q^*(s, a)$. Although this approach seems reasonable and agents can make better informed decisions, state action space dimensionality causes computational complexity, making this approach almost unusable for Q-learning.

In the first approach, only one state is assigned to one trading day, resulting in a single-dimension state space design. States contain only information about the market. There is no information about the agent or held portfolio.

Since the states do not contain information about portfolio, the way agent trades have to be revised. The agent has to be able to select an action without memory. As a result, the agent selects between actions describing how much capital will be invested before transitioning into the next state. This leads to the framework, where the agent selects an action corresponding to how much capital will be assigned to which asset. For example, the agent selects between buying asset with 0% of capital, 50% of capital and 100% of capital. When an agent goes from 100% to 50% of invested capital, the agent effectively lowers its position in the portfolio, not knowing its actual position.

When an agent selects an action, the agent is transited to a new state and rewarded for taking action. In contrast to the first approach, new states do not depend on selected actions. Selected actions do not determine the next state. There is a firmly established sequence of states.

This single-dimension state action space design benefits from non-complexity. However, there are also some drawbacks. Selected actions will receive $\max_{a'} Q(s', a')$, which increases with lapsed episodes. Consequently, the reward of taking action will be higher with elapsed episodes. However, all actions from state s result in transition into state $s^{'}$. There is no way to transition from state s to state $s^{''}$ or $s^{'''}$ etc. Thus, learning will be biased. As a result, Q-learning equation 2.14 is modified into equation 3.1.

$$Q^{new}(s, a) \leftarrow (1 - \alpha) \cdot Q^{old}(s, a) + \alpha \cdot r \tag{3.1}$$

Modified Q-learning determines the value of action without the value of the next state. Action value is determined solely by reward r, corresponding to the reward function, which is calculated as portfolio return caused by the investment decision selected by action a. Particular reward functions are presented in the chapter devoted to simulation results.

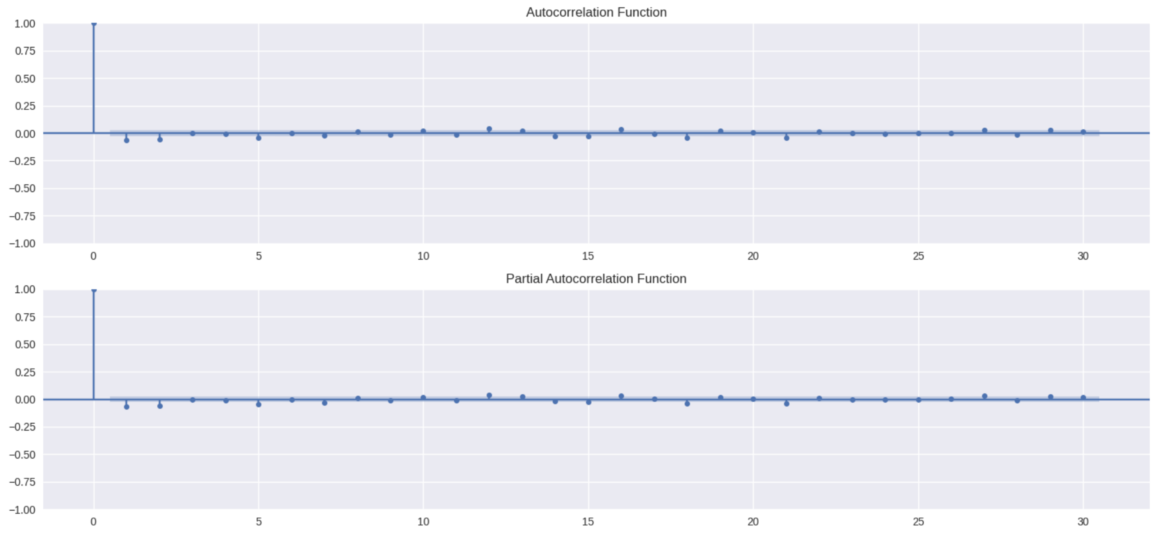
3.3 State representation system

As shown above, the introduced reinforcement learning framework strongly relies on state representation system. Only reliable state representation allows values to be assigned correctly to actions in unseen test states.

3.3.1 Challenges of state representation

The problem which arises from POMDPs is how to represent states with available information. States have to be represented in a way that clustering algorithms are able to detect common patterns and cluster similar states together. Moreover, state representation has to describe similar states in similar way and has to be powerful enough to distinguish between different states.

For example, an autoregressive process (Yule, 1927) constructed based on the Box-Jenkins methodology (1976) is a popular econometric tool utilising only differences in close prices. Figure 4 shows the autocorrelation function and partial autocorrelation function. It will probably be challenging to use an autoregressive model with this time series since there is almost no significant usable autocorrelation. Moreover, differences seem to be some kind of random process.



Source: own processing

Figure 4: ACF PACF functions

Trying to obtain the data generation process of time series, as the autoregressive process does, is a completely different task than clustering states. On the other hand, it is somehow intuitive that representing states only with differences that follow a random process is not sufficient. Moreover, the difference between 'open' and' close' is probably not enough to distinguish between different states with different market situations.

Also, the state representation system should not be strongly distinguishing and "noise catching". Overpowered state representation systems could lead to overfitting and the creation of a significant amount of outlier states. Moreover, it is important to generalise since financial market time series are encumbered with a significant amount of noise. Although powerful state representation could lead to excess returns of strategy.

The challenge of POMDP is becoming obvious now, and own state representation system is needed.

3.3.2 New explanatory variables

To represent states, open, low, high, close, and volume are used to construct new explanatory variables as moving averages, deviations, and range-based oscillators are also constructed. Although this approach can be considered obsolete compared with those discussed in the literature overview, it also has several strong advantages.

Firstly, the introduced functions are completely deterministic and can be calculated at any given time. If a time series is considered continuous, statistics are also continuous. For example, this is often not the case for alternative data.

Secondly, this approach does not have to deal with challenges connected to data synchronisation. For example, it could be problematic to synchronise data from social media or news announcements for the purpose of model training. This could lead to a possible forward-looking bias, where the model is aware of information before it is released to the public.

The Moving Average (MA) is used to smooth out price data by creating a continuously updated average price. The MA can be simple or exponential:

$$\mathrm{SMA}_t = \frac{Price_1 + Price_2 + \ldots + Price_n}{n} \tag{3.2}$$

where $Price_1, Price_2, \ldots, Price_n$ are the prices over n periods.

Exponential Moving Average (EMA).

$$\mathrm{EMA_t} = \left(\frac{Price_t - EMA_{t-1}}{SF} \right) + EMA_{t-1} \tag{3.3}$$

where SF is the Smoothing Factor, typically $\frac{2}{n+1}$, and n is the number of periods.

The RSI is a momentum oscillator that measures the speed and magnitude of price movements. It oscillates between zero and 100. RSI is used to identify overbought or oversold conditions in a market.

$$\mathrm{RSI} = 100 - \frac{100}{1 + RS} \tag{3.4}$$

$$RS = \frac{\text{Average Gain over } n \text{ periods}}{\text{Average Loss over } n \text{ periods}} \tag{3.5}$$

3.3.3 Data preprocessing

As shown before, proposed clustering algorithms cluster observations based on their position in multi dimensional space. It is crucial to ensure that all dimensions have the same scale. Standardisation is required. If scales are different, different features have different weights. The larger the scale, the more significant impact a particular feature could have, generally speaking, as Kaufman and Rousseeuw show (1990). Consequently, different scales lead to different clusters, which will probably be biased.

Multicollinearity is another problem that has to be solved. High correlation effectively means that there is no excess information gained from a variable, which effectively increases weight in the clustering of one explanatory variable. As a result, two highly correlated explanatory variables cause the clustering algorithm to consider almost the same variable twice. This could lead to problems with generalising unseen data, as Kaufman and Rousseeuw show (1990).

Variables are eliminated in an iterative process. In each step, a correlation matrix is calculated, and the variable with the highest correlation coefficient is eliminated. This process is repeated until the desired level of multicollinearity is achieved.

3.4 Clustering algorithms

Gaussian Mixture Model (GMM) and K-means clustering algorithms are used for state clustering.

3.4.1 Gaussian Mixture Model

GMM is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters (Rasmussen, 1992).

A GMM represents the probability distribution of a dataset $X = \{x_1, x_2, \ldots, x_n\}$ as a weighted sum of K Gaussian densities. Each point x_i in the dataset is assumed to come from a Gaussian distribution chosen randomly according to the mixing weights. The probability density function of a GMM is given by equation 3.6.

$$f(x) = \sum_{k=1}^{K} z_k \mathcal{N}(x|\mu_k, \Sigma_k) \tag{3.6}$$

Here, z_k are the mixing weights, with $\sum_{k=1}^{K} z_k = 1$ and $\pi_k \geq 0$, $\mathcal{N}(x|\mu_k, \Sigma_k)$ is the Gaussian density function with mean μ_k and covariance matrix Σ_k, K is the number of Gaussian distributions in the mixture.

The parameters of a GMM as mixing weights, means and covariances are typically estimated using the Expectation-Maximization algorithm. This algorithm iteratively adjusts the parameters to maximize the likelihood of the data given the model.

3.4.2 K-means

The K-means model assigns data points to clusters in the way that the sum of the distances between each point and the centroid of its cluster is minimized introduced by Hartigan and Wong (1979). Centroids are initialized randomly or by a heuristic approach at the beginning. Datapoints are assigned to the cluster with the closest centroid. Distances, such as Euclidean or Manhattan, can be used with different performance as Suwanda et al. (2020) show.

The position of centroids is recalculated based on data points assigned to the cluster, described in equation 3.7.

$$c_k = \frac{1}{|n_k|} \sum_{x_j \in S_k} x_j \tag{3.7}$$

Here, c_k represents the new centroid for the k-th cluster, $|N_k|$ is the number of data points in the i-th cluster, and x_j represents the data points that belong to cluster k. This process is repeated until there are no changes in the positions of centroids.

3.5 Train and test split

Splitting datasets into training and test sub-datasets is a key model evaluation technique. The common splitting ratio is around eighty percent of observations for the training period (Hastie, 2009). It is important not to violate the time dependency of observations. For instance, when training a model on 2020 data, testing data from 2021 should be used.

The ability of the current machine learning models to analyse non-linear datasets and find even small, almost negligible patterns is why machine learning is popular in all fields where some form of forecast is needed. It could be problematic to use 30 years of historical data in case of short-term financial time series predictions of up to 1 hour. Since the model would probably learn obvious noise movements and patterns that are not present in the current trading environment.

This could happen mainly because of gradual and consistent research advances in the machine learning field. Specifically, using a machine learning model before everyone knows that model might be a great way to generate significant excess returns over the other traders and investors. Consequently, this leads to the situation, where models outperform the market during the train period and fail the test period, simply because the known patterns are not involved anymore. There are around 20 years of history data in the followed simulations, the main reasons for this approach are the following.

Firstly, the objective of this diploma thesis is to create a model that identifies lasting and persistent patterns. These patterns occur stable over time on the entire observed dataset. The idea behind focusing on stable patterns is straightforward. Patterns in the last twenty years are more likely to occur in the following months and years. Consequently, forecast performance should not differ significantly on train and test datasets.

Secondly, 20 years should be enough to capture different market regimes and outlier observations, such as sudden high volatility periods and market crises. As discussed before, the trading environment is a partially observable Markov decision model in which each agent is forced to create its own state representation system. With this new system, the environment is more observable. Although this approach is efficient, it is not perfect.

The agent is not able to observe the news announcement. Significantly different announced news than anticipated could cause discontinuous jumps alongside continuous fluctuations. The more observations, the better understanding and the larger the statistical sample. When the model faces more discontinuous jumps, it can learn how to react, and more importantly, sudden jumps will be included in the model history performance. Thus, sudden jumps should not create unseen model results and could help construct more aware confidence intervals.

3.6 Evaluation metrics

To measure the performance of different investment strategies, different metrics are used to inform about the different properties of strategies. All presented metrics summarize complex strategies and their different equity curves into one scalar number. This compression is inevitably followed by some loss of information.

All of these metrics should be assessed together with the whole equity curve inspection since the equity curve can be used to explain metrics in a completely different way. A strategy with maximal drawdowns strongly correlated with the market significantly differs from a strategy with more stochastic uncorrelated drawdowns with the market. For example, a market-correlated drawdown during Coronavirus 2020 is more acceptable than a drawdown during a strong bull market.

3.6.1 Sharpe ratio

The Sharpe ratio is the most popular strategy or portfolio performance metric, developed by William F. Sharpe (1996) in 1996. It measures excess returns related to volatility. Thus, the Sharpe ratio is calculated.

$$\text{Sharpe ratio} = \frac{E_p - R_f}{\sigma_p} \tag{3.8}$$

E_p is the expected return of the investment or portfolio. R_f is the risk-free rate of return, in the US, government bonds are often used as R_f. σ_p is the standard deviation of the returns.

A higher Sharpe ratio indicates better risk-adjusted performance, which corresponds to a higher return for the same level of risk. The key concept of the Sharpe ratio indicates that total return does not reflect the amount of risk taken. The Sharpe ratio emphasises equity smooth curves with positive slope. Strategies with a better Sharpe ratio can always take advantage of lower volatility and leverage their positions to achieve similar or better returns than strategies with a lower Sharpe ratio (Harvey, Liu, 2014).

The crucial drawback of the Sharpe ratio is that it considers the standard deviation created by positive returns in the same way as negative returns. Cumulative returns similar to jump processes or staircase functions are penalised heavily by the Sharpe ratio, even with a strongly positive slope.

3.6.2 Sortino ratio

The Sortino ratio is a modified Sharpe ratio. Unlike the Sharpe ratio, the Sortino ratio does not penalize strategy for deviation generated by positive returns (Sortino and Price, 1994).

$$\text{Sortino ratio} = \frac{E_p - R_f}{\sigma_d} \tag{3.9}$$

E_p is the expected return, R_f is the risk-free rate, and σ_d is the downside deviation, which measures the volatility of negative returns. Further, downside deviation σ_d is calculated as the sum of squared negative returns given by threshold, divided by the number of returns, effectively creating an average negative squared return.

$$\sigma_d = \sqrt{\frac{\sum_{t=1}^{n} (\min\{0, R_t - U\}^2)}{n}} \tag{3.10}$$

R_t denotes t-th return, U is the threshold which adjusts the "aggressivity" of Sortino ratio, n stands for the total number of observations below the threshold.

Since the Sortino ratio does not see positive volatility as a negative attribute of investment strategy, the Sortino ratio evaluation could lead to situations where a strategy considered by the Sharpe ratio as underperforming is overperforming in terms of the Sortino ratio. An example could be provided of strategies that consist of many low-return trades and a few high positive return trades.

Despite its obvious advantages, the Sortino ratio is not as well established as the Sharpe ratio. This could be because the calculation of downside deviation is unclear for common investors.

3.6.3 Maximal drawdown

Maximal drawdown is a metric that informs about the maximal peak-to-trough decline in the value of a portfolio over the entire time period (Chekhlov, 2005).

$$MDD = \max_{t\in[0,T]} \left\{ \max_{s\in[0,t]} \{P_s - P_t\} \right\} \tag{3.11}$$

There are several reasons and implications for maximal drawdown.

Firstly, maximal drawdown provides a clear measure of the downside risk associated with a strategy. Maximal drawdown could be recognized as the "red flag" performance metric of strategy. It could be an all-around performance strategy. However, a significant maximal drawdown could make the investment strategy non-investable.

Secondly, maximal drawdown also provides rough estimates of the point where the strategy is no longer working. Approaching or exceeding the maximum historical drawdown should be an impulse for strategy inspection.

Lastly, maximal drawdown indicates the psychological demandingness of investment strategy (Thaler, 2005). Since all drawdowns in "decent" proportion to maximal drawdown do not necessarily mean that strategy should be inspected, large drawdowns could be simply the property of strategy. Thus, psychological pressure from high portfolio volatility could be present.

3.6.4 Information ratio

Information ratio is the metric that asses strategy portfolio capability to produce surplus returns over selected benchmark consistently (Goodwin, 1998). While the Sharpe ratio evaluates a strategy based on risk-adjusted excess returns compared to the risk-free rate, the Information ratio evaluates a strategy based on risk-adjusted excess returns over the benchmark.

$$\text{Information ratio} = \frac{E_p - R_{bench}}{\sigma_e} \tag{3.12}$$

Where E_p is portfolio return, R_{bench} corresponds to return of selected benchmark and σ_e measures standard deviation of excess returns.

4 Results

This section contains the results of all simulations. It starts with simple experiments with one asset and simple actions up to more complex experiments, with multiple assets and complex reward functions.

The agent is allowed to trade daily. The data consists of the daily returns of the SPDR S&P 500 ETF (SPY). ETF is used in order to let agent trade on the real market data, not a theoretical value of the S&P 500 index.

The simulation parameters are provided before every experiment. Cumulative returns are plotted into figures, the first figure represents results on the train dataset, and the second figure represents results on the test dataset. Each simulation also contains table with metrics to compare different strategies statistically.

It should be mentioned that results could be slightly different in the case of replication due to the stochastic environment exploration and occasional convergence issues of clustering algorithms. Random seeds are used and set. On the other hand, even if the replicated results could be slightly different, the key concepts and impacts of experiments should be the same.

4.1 Simulation 1 - Limited actions

The first simulation is the simplest one. The parameters of the simulation follow Table 1 below.

Table 1: Simulation 1 - Hyperparameters

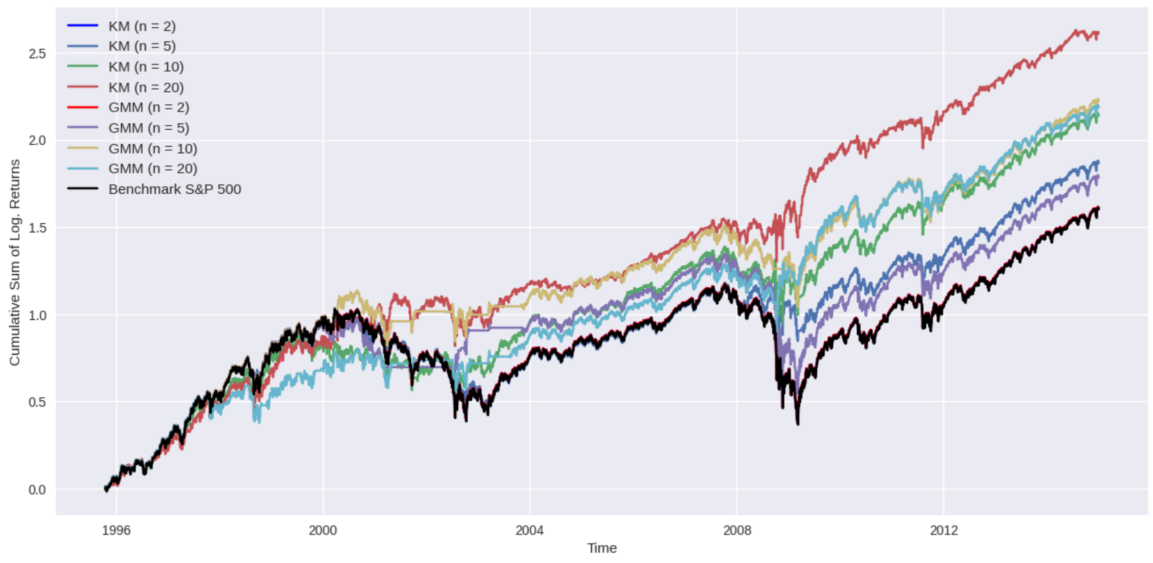
Parameter	Value
Learning Rate	0.1
Exploration Rate	0.3
Epochs	2 000
Actions	0, 0.25, 0.5, 0.75, 1
Training period	16.10.1995 - 1.1.2015
Testing period	1.1.2015 - 1.1.2024
Assets	S&P 500 (SPY)

Source: own processing

In this simulation, the agent can select from 5 actions that determine allocation size into the S&P 500 index. Agent reward is given by equation below.

$$R(s,a) = 0.9 \cdot r_{t+1} + 0.7 \cdot r_{t+2} + 0.5 \cdot r_{t+3} + 0.2 \cdot r_{t+4} \tag{4.1}$$

In this equation, r_t corresponds to portfolio return generated in time t, which corresponds to trading days. Return is discounted in order to balance between closer portfolio returns than distant portfolio returns. Discounting vector balances between short and long-sighted agent.

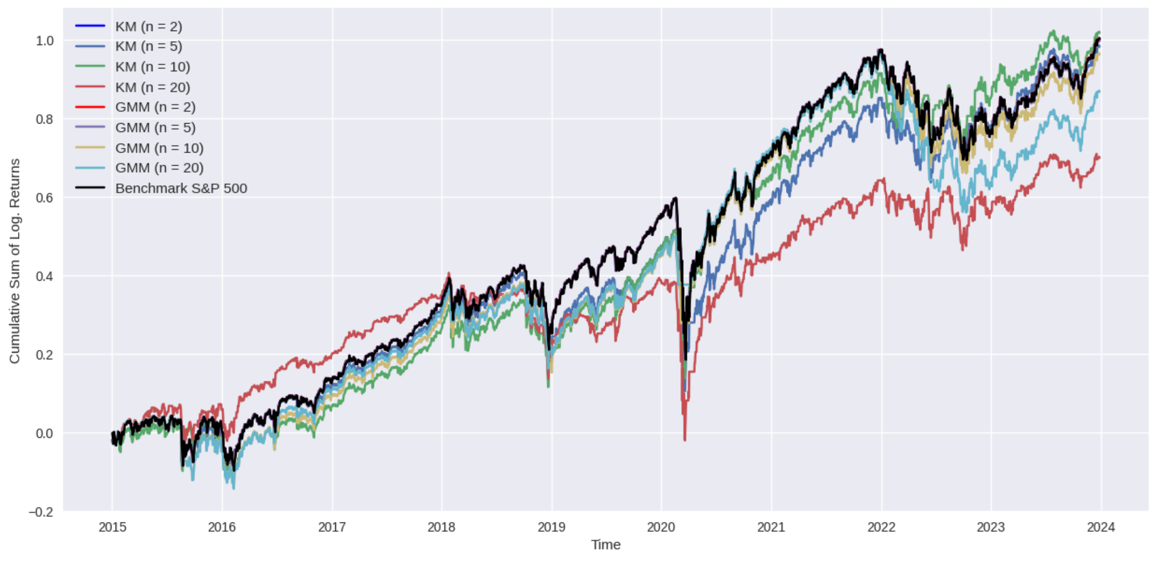


Source: own processing

Figure 5: Simulation 1 - Cumulative log. returns on train data

Figure 5 displays the cumulative logarithmic returns of strategies on the train dataset. Almost all strategies achieve remarkable results. The best strategy is a K-means strategy with 20 clusters. Interestingly, many of the strategies were able to avoid or "soft-land" the 2008 crisis.

In contrast, Figure 6 shows that the overperforming K-means strategy with 20 clusters is the worst strategy on test data. Some of the presented strategies were able to achieve results similar to those of the benchmark. However, all these strategies failed compared to the training data results, which should have partially determined the test results.



Source: own processing

Figure 6: Simulation 1 - Cumulative log. returns on test data

Table 2: Simulation 1 - Results

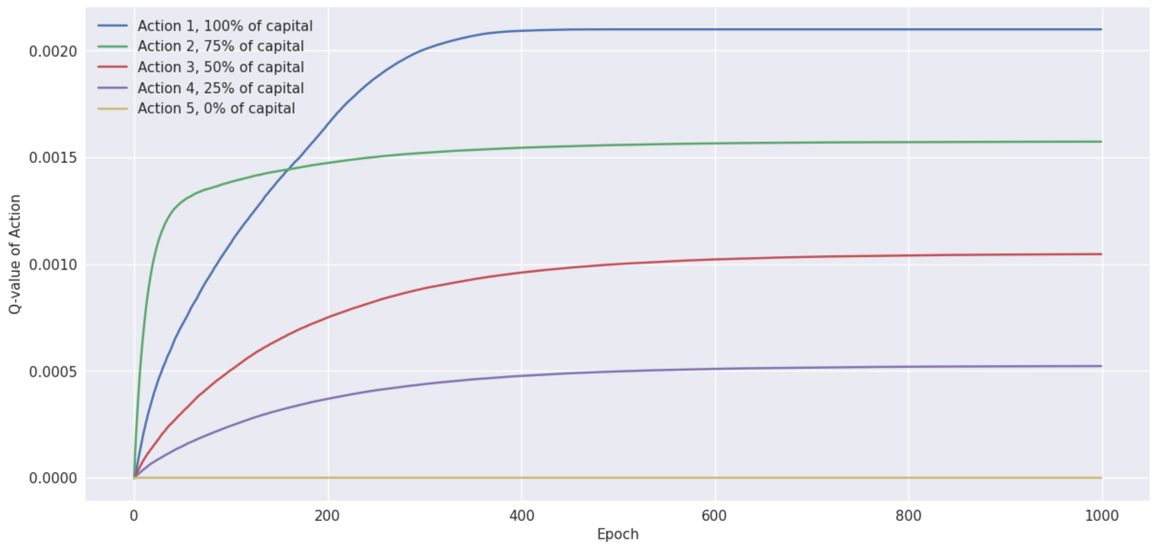
Model	Tot. ret.(%)	Avg. y. ret.(%)	Std.(%)	Max. DD(%)	SR Anl.	SR	SoR Anl.	SoR	IR	MW p-val
Training Data										
S&P 500	160.558	8.367	20.028	80.273	0.318	1.393	0.444	1.947	-	-
KM(2)	160.042	8.340	20.029	80.273	0.317	1.387	0.442	1.938	-0.004	0.997
KM(5)	186.515	9.719	18.506	53.618	0.417	1.827	0.590	2.586	0.211	0.740
KM(10)	213.692	11.135	17.863	43.753	0.511	2.240	0.729	3.194	0.437	0.597
KM(20)	260.798	13.590	17.283	29.998	0.671	2.938	0.983	4.308	0.840	0.237
GMM(2)	160.042	8.340	20.029	80.273	0.317	1.387	0.442	1.938	-0.004	0.997
GMM(5)	178.302	9.291	18.114	80.273	0.403	1.763	0.567	2.483	0.145	0.426
GMM(10)	222.032	11.570	17.162	47.064	0.558	2.443	0.789	3.458	0.520	0.741
GMM(20)	218.531	11.387	17.287	38.956	0.543	2.379	0.783	3.430	0.493	0.522
Testing Data										
S&P 500	100.265	11.165	18.159	41.124	0.505	1.512	0.689	2.065	-	-
KM(2)	100.029	11.139	18.159	41.124	0.503	1.508	0.687	2.059	-0.003	0.994
KM(5)	98.140	10.929	17.199	41.124	0.519	1.556	0.709	2.125	-0.027	0.747
KM(10)	101.664	11.321	16.604	37.743	0.561	1.682	0.774	2.319	0.018	0.536
KM(20)	69.854	7.779	16.042	42.620	0.360	1.080	0.492	1.474	-0.393	0.112
GMM(2)	100.029	11.139	18.159	41.124	0.503	1.508	0.687	2.059	-0.003	0.994
GMM(5)	99.562	11.087	18.159	41.124	0.500	1.500	0.683	2.048	-0.009	0.984
GMM(10)	96.165	10.709	15.860	30.557	0.549	1.645	0.750	2.248	-0.056	0.876
GMM(20)	86.665	9.651	15.032	40.450	0.509	1.525	0.695	2.083	-0.192	0.706

Source: own processing

Table 2 shows that the best strategies achieve almost double the Sharpe ratio of the benchmark S&P 500. In addition, some strategies were able to overcome the S&P 500 in Sharpe ratio and minimal maximal Drawdown. Results on unseen test data are significantly different, as Table 2 shows.

This simple setup shows that it is not possible to overcome the benchmark index with current settings. The simulation consists of two models: an reinforcement learning Q-learning model and a clustering model using K-means and GMM. The performance of this simulation pipeline should be investigated.

Intuitively, the Q-learning agent´s parameters correspond to a standard Q-learning setup. 1000 iterations should be enough for 5 actions to find the optimal policy, as there are many attempts to investigate every single action. The Q-learning agent should converge adequately. However, empirical evidence is needed.



Source: own processing

Figure 7: Simulation 1 - Convergence of action values

Figure 7 shows that there are no problems with convergence, and optimal policy was found. Figure 7 represents a randomly selected state that is observed in every epoch. Q-table values are saved. This sub-simulation runs 1,000 times, and lines represent the averages of these runs. In the observed state, the market appreciates. Thus, it is clear (based on objective function formalised as Equation 4.1) that an action with 100% of capital has double the value of an action with only 50% of capital, and so on. Interestingly, ϵ-greedy policy preferred action 2 until epoch 170.

Q-learning converges correctly to the optimal policy, but underperformance on test data is probably caused by clustering. The impact of different numbers of clusters can be observed in Figures 5 and 6 . More clusters, such as 10 and 20, overperform other strategies in the train dataset since 10 or 20 clusters are enough to cluster specific market states and distinguish even between small nuances. This sensitivity also has drawbacks. Market states tend to be classified incorrectly on test data, and some noise probably caused that state was assigned to the "wrong" cluster. Moreover, performance on train data and underperformance on test data indicate a high probability of an overfitted strategy.

On the other hand, 2 or 5 clusters are more conservative, decreasing the probability of misculstering. The limitation arising from a small number of clusters is that it over-generalises state space. As a result, the strategy is not able to exploit different market conditions because there are only two generalised states. The results show that the right number of clusters is probably between 5 and 10. Interestingly, this is precisely the same number of clusters that Pakest et al. 2020 incorporate.

This simulation also reveals one disturbing conclusion. As Table 2 obviously shows, the Sharpe ratio, the Sortino ratio and the Information ratio do not indicate strategy performance on unseen test data. Moreover, there is a negative correlation between the train and test data results. In other words, it is possible to maximise objective function on train data. However, it does not imply that a set of rules which leads to high values on the train dataset will also lead to high values on the test dataset. More importantly, results say very little about selecting the right metrics that could be used as an objective for a Q-learning agent. This finding is the most significant finding of this simulation.

The question is, is there any way how to improve state generalisation without complete overwork of the whole simulation framework. More complex algorithms such as DPG, DDPG, and TD3, which incorporate different approaches, could be used, or the current system could be modified. Firstly, the state representation system could be changed. On the other hand, the trading environment is POMDP, which is well known for its complexity and noise. Secondly, clustering algorithms could be changed or modified, for example, by using different distance measure systems. Lastly, as discussed before, finding a reliable representation system to represent states and a cluster model to assign states "right" values of actions will probably be difficult.

Nevertheless, Q-learning is working exceptionally well. In the current simulation, the agent was restricted to 5 conservative actions. The agent was not able to generate return on asset depreciation or exploit and leverage asset appreciation. Extending the agent's actions to the capability of short selling and long trading with leverage could outweigh the drawbacks of the current state representation system and value-to-action assignment.

In summary, the results of the first simulation show:

- good results on the training dataset do not imply good results on the test dataset,
- reasonable number of clusters is probably between 5 and 10, the same number of clusters which Pakesh et al. 2020 incorporate,
- Q-learning agents converge properly, and optimal policy can be found without difficulties.

4.2 Simulation 2 - Extended actions

The second simulation, which directly follows the first simulation, has two main differences. Firstly, an agent can select from 7 actions, including short selling and leveraging long positions. Secondly, the number of clusters is changed following the results from the first simulation. All other parameters of the simulations are the same as the previous simulation.

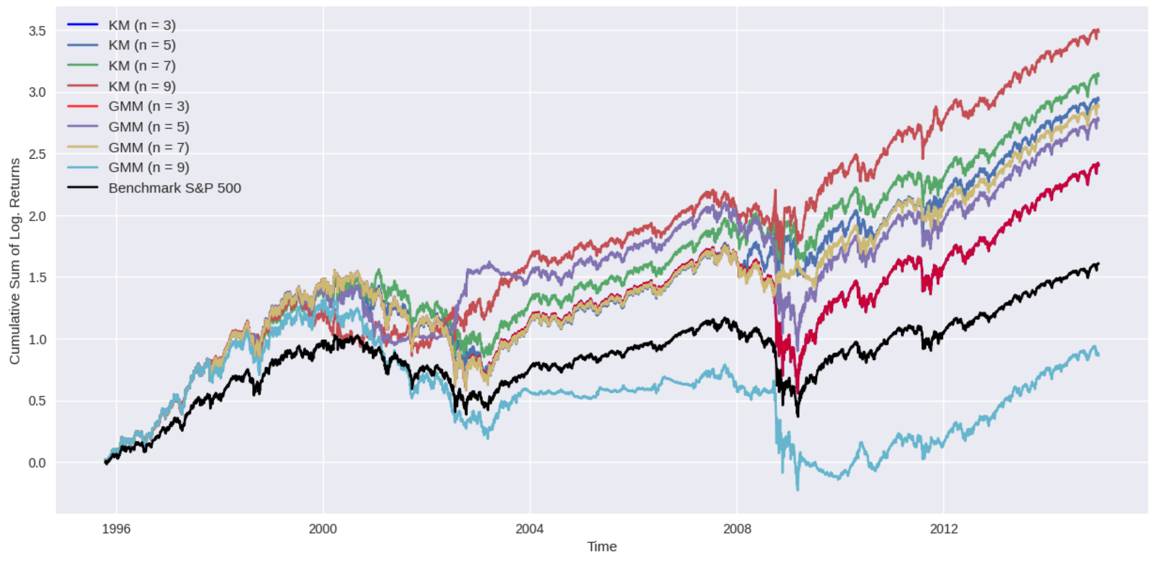
Table 3: Simulation 2 - Hyperparameters

Parameter	Value
Learning Rate	0.1
Exploration Rate	0.3
Epochs	1 000
Actions	-0.5, -0.25, 0, 0.25, 0.5, 0.75, 1, 1.25, 1.5
Training period	15.10.1995 - 1.1.2015
Testing period	1.1.2015 - 1.1.2024
Assets	S&P 500 (SPY)

Source: own processing

Agent reward is given by equation below, where r_t corresponds to portfolio return generated in time t, which corresponds to trading days.

$$R(s,a) = 0.9 \cdot r_{t+1} + 0.7 \cdot r_{t+2} + 0.5 \cdot r_{t+3} + 0.2 \cdot r_{t+4} \tag{4.2}$$

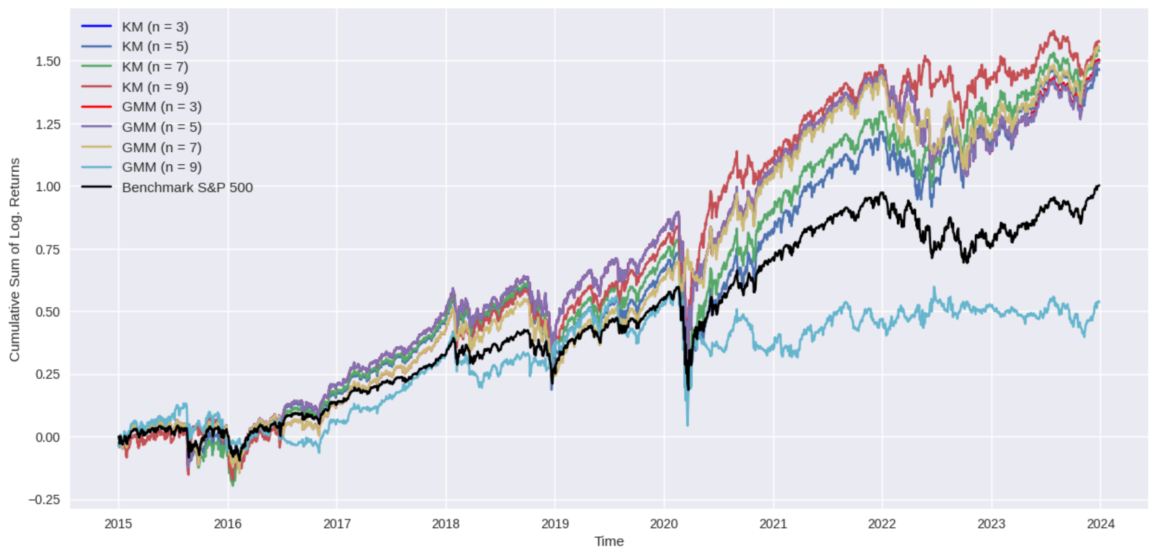


Source: own processing

Figure 8: Simulation 2 - Cumulative log. returns on train data

Figure 8 shows trading strategies on train data. Similarly to the first simulation, strategies overcome the benchmark S&P 500 index. Strategies achieve higher returns for the price of higher volatility of returns.

In the first simulation, strategies were not able to overcome the benchmark index. In contrast, Figure 9 shows that almost all strategies are able to overcome the benchmark index on test data. GMM with 9 clusters was not able to overcome the index on train data as well as on test data. An issue with cluster convergence is probably present.



Source: own processing

Figure 9: Simulation 2 - Cumulative log. returns on test data

Table 4: Simulation 2 - Results

Model	Tot. ret.(%)	Avg. y. ret.(%)	Std.(%)	Max. DD(%)	SR Anl.	SR	SoR Anl.	SoR	IR	MW p-val
Training Data										
S&P 500	160.558	8.367	20.028	80.273	0.318	1.393	0.444	1.947	-	-
KM(3)	240.063	12.509	30.043	120.409	0.350	1.532	0.490	2.145	0.488	0.054
KM(5)	293.008	15.268	28.020	77.746	0.474	2.074	0.672	2.944	0.854	0.070
KM(7)	312.690	16.294	28.168	70.958	0.507	2.223	0.720	3.154	0.980	0.034
KM(9)	348.292	18.149	27.029	58.419	0.597	2.617	0.852	3.734	1.237	0.051
GM(3)	240.063	12.509	30.043	120.409	0.350	1.532	0.490	2.145	0.488	0.054
GM(5)	276.583	14.413	27.504	120.409	0.451	1.977	0.637	2.790	0.754	0.144
GM(7)	287.536	14.983	26.052	96.699	0.498	2.183	0.702	3.076	0.873	0.051
GM(9)	86.127	4.488	25.489	153.964	0.098	0.428	0.135	0.591	-0.512	0.414
Testing Data										
S&P 500	100.265	11.165	18.159	41.124	0.505	1.512	0.689	2.065	-	-
KM(3)	150.043	16.708	27.239	61.686	0.540	1.618	0.738	2.213	0.480	0.177
KM(5)	146.265	16.288	25.963	61.686	0.550	1.649	0.753	2.257	0.459	0.333
KM(7)	153.888	17.137	26.033	61.686	0.581	1.742	0.796	2.386	0.534	0.241
KM(9)	157.484	17.537	25.132	54.925	0.618	1.853	0.855	2.562	0.585	0.485
GM(3)	150.043	16.708	27.239	61.686	0.540	1.618	0.738	2.213	0.480	0.177
GM(5)	149.111	16.604	27.238	61.686	0.536	1.607	0.733	2.197	0.472	0.185
GM(7)	155.261	17.289	22.127	37.149	0.691	2.071	0.961	2.880	0.659	0.290
GM(9)	53.577	5.966	22.365	55.532	0.177	0.531	0.239	0.716	-0.507	0.252

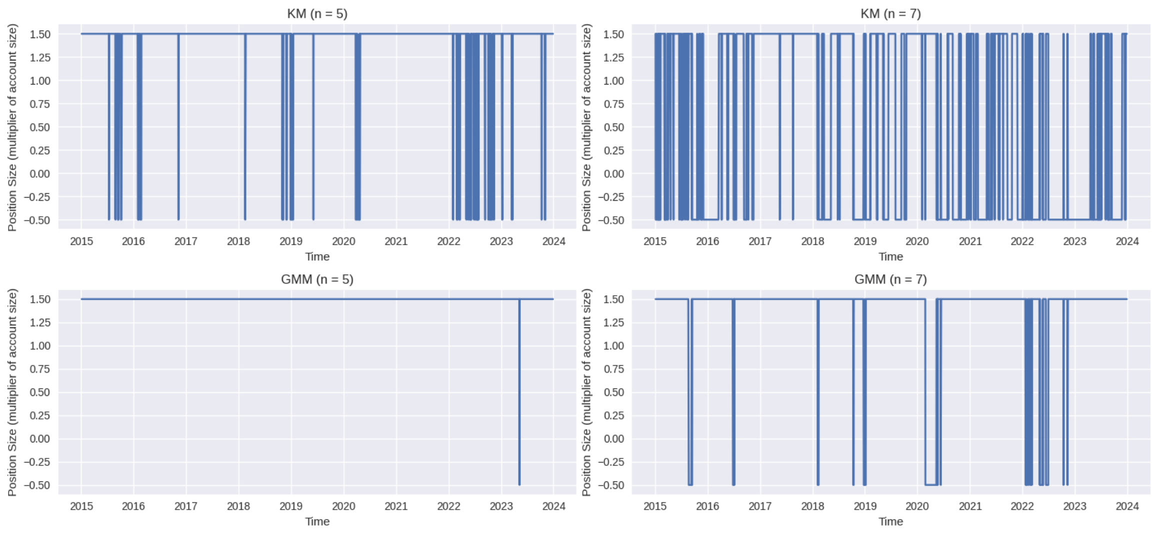
Source: own processing

Results on train data show that some strategies were able to double the total return compared to the S&P 500. Please note that the maximum drawdown in size of 120% and 153% is theoretical, measuring the maximal loss connecting the local maximum followed by the local minimum, indicating the worst-case scenario. In comparison, the S&P 500 achieved the same situation loss of 80%. These over 100% drawdowns are caused by 1.5 leverage, which was incorrectly applied in the depreciation of the market. This is the problem of insufficient state space generalisation into 3 and 5 clusters. Additional market regime switching methodology should be added. As discussed, the problem arises from POMDP. In 2008 crisis, price was probably determined primarily by fundamental analysis.

On the other hand, some strategies were able to overcome the benchmark index, namely strategies such as KM with 7 and 9 clusters. These strategies were also able to achieve significant Sharpe ratios and Sortino ratios.

Results on test data show that all (without GMM 9 clusters) strategies overperform the benchmark index. The Sharpe ratio and Sortino ratio favour developed strategies. Strategies exploit the potential of leverage and short selling, which leads to better Sharpe ratios and Sortino than the benchmark index.

However, standard deviation and maximal drawdown are not in favour of developed strategies. The state representation system is probably insufficient to contain information about approaching market depreciation. Some kind of default action in case of high uncertainty should be added in the same way as Jeong and Kim (2019) did.



Source: own processing

Figure 10: Simulation 2 - Position sizing

Figure 10 shows the agent´s actions. Agents exploit newly added actions, and agents following the epsilon greedy policy focus on edge actions such as short selling -0.50 and leverage long position 1.5 since. These actions lead to maximal rewards, which maximises the mentioned actions. The K-means model switches its actions often. In contrast, GMM models tend to be less reactive to changing market conditions. The figure 10 shows that strategies do not tend to short selling in 2020. nevertheless, for example, the year 2023 was by K-means with 7 clusters recognised as states for short selling.

In summary, the results of the second simulation show:

- agents capable of short selling and leverage can outperform the benchmark index in total return as well as in Sharpe ratio,
- agents are capable of exploiting any opportunity as leverage in the environment,
- agents with leverage may or may not face a higher drawdown.

4.3 Simulation 3 - Sharpe ratio reward function

As was shown in the simulation before, the S&P 500 index can be overperformed with the use of agents capable of short selling and entering a long position on leverage. This agent achieves a higher average return and a higher Sharpe and Sortino ratio. The problem that arises is a higher standard deviation of returns and a higher maximal drawdown. To overcome these challenges, the Sharpe ratio reward function is introduced.

Table 5: Simulation 3 - Hyperparameters

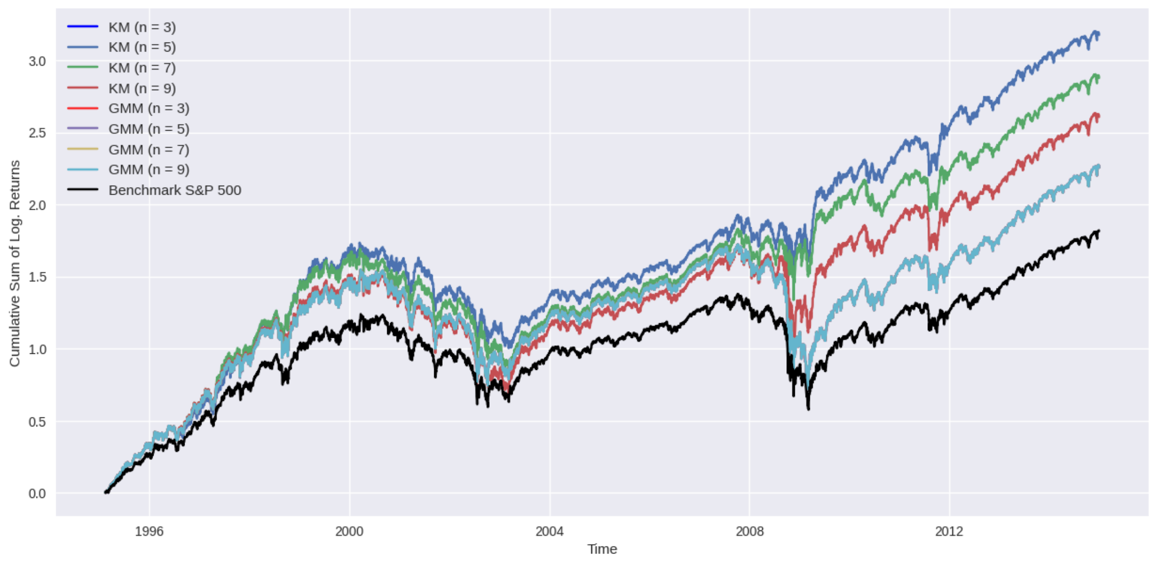
Parameter	Value
Learning Rate	0.1
Exploration Rate	0.3
Epochs	2 000
Actions	-0.5, -0.25, 0, 0.25, 0.5, 0.75, 1, 1.25, 1.5
Training period	1.2.1995 - 1.1.2015
Testing period	1.1.2015 - 1.1.2024
Assets	S&P 500 (SPY)

Source: own processing

Agent reward is given by the equation below, which calculates the annualized Sharpe ratio.

$$R(s,a) = \frac{\frac{1}{5}\sum_{t=1}^{T=5} r_t - \frac{R_f}{252}}{\sqrt{\frac{1}{T-1}\sum_{t=1}^{T=5}(r_{t+i} - \bar{r})^2} \cdot \sqrt{252}} \tag{4.3}$$

r_t corresponds to portfolio return generated in time t with action selected in current state, t corresponds to trading days, R_f is risk free rate. Five trading days are selected in order to balance between short and long-sighted agent.

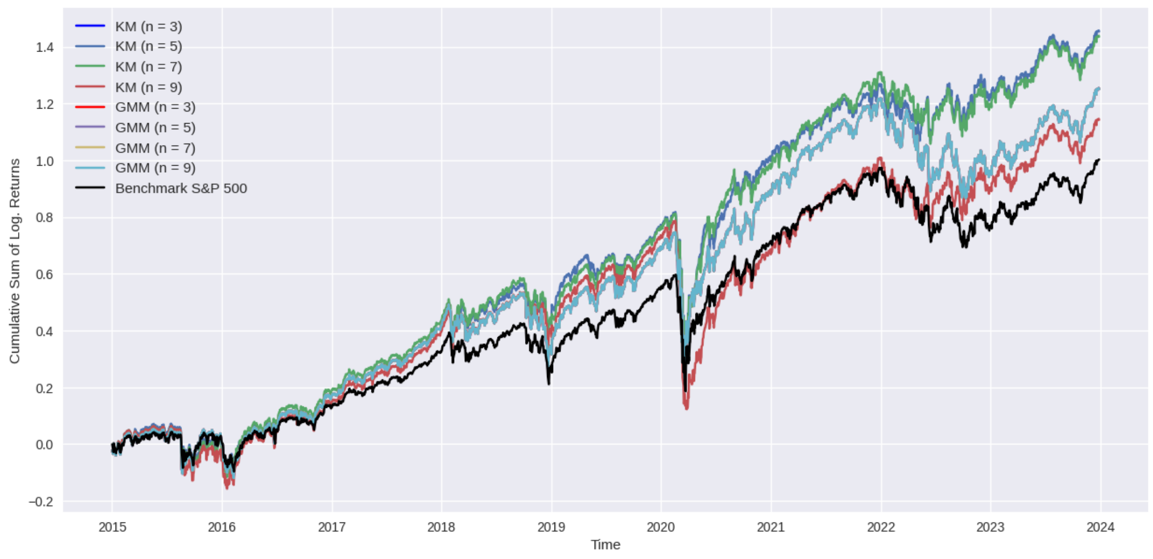


Source: own processing

Figure 11: Simulation 3 - Cumulative log. returns on train data

Figure 11 shows that strategies are similar, and their equity curves only shift. Some of the curves are hidden, which means that their trajectories are the same as the trajectory of another strategy. Thus, these strategies are hidden in the plot behind another strategy. Interestingly, some strategies were able to minimise the impact of the 2008 crisis.

Figure 12 shows that the Sharpe ratio objective function did not help strategies avoid a sharp decline in 2020. Unfortunately, even strategies that nearly avoided the 2008 crisis were not able to react before 2020.



Source: own processing

Figure 12: Simulation 3 - Cumulative log. returns on test data

Table 6: Simulation 3 - Results

Model	Tot. ret.(%)	Avg. y. ret.(%)	Std.(%)	Max. DD(%)	SR Anl.	SR	SoR Anl.	SoR	IR	MW p-val
Training Data										
S&P 500	181.466	9.135	19.747	80.273	0.361	1.610	0.506	2.254	-	-
KM(3)	225.546	11.354	24.686	100.341	0.379	1.689	0.531	2.366	0.303	0.268
KM(5)	317.543	15.985	23.543	73.557	0.594	2.648	0.852	3.797	0.970	0.182
KM(7)	287.697	14.483	23.754	84.259	0.526	2.342	0.751	3.346	0.755	0.252
KM(9)	260.676	13.122	23.596	84.347	0.471	2.101	0.659	2.939	0.569	0.223
GMM(3)	225.546	11.354	24.686	100.341	0.379	1.689	0.531	2.366	0.303	0.268
GMM(5)	225.546	11.354	24.686	100.341	0.379	1.689	0.531	2.366	0.303	0.268
GMM(7)	225.546	11.354	24.686	100.341	0.379	1.689	0.531	2.366	0.303	0.268
GMM(9)	225.546	11.354	24.686	100.341	0.379	1.689	0.531	2.366	0.303	0.268
Testing Data										
S&P 500	100.265	11.165	18.159	41.124	0.505	1.512	0.689	2.065	-	-
KM(3)	125.036	13.924	22.699	51.405	0.525	1.574	0.718	2.151	0.268	0.451
KM(5)	145.345	16.185	21.646	51.405	0.655	1.964	0.902	2.702	0.505	0.482
KM(7)	143.519	15.982	21.875	51.405	0.639	1.915	0.876	2.624	0.481	0.368
KM(9)	114.230	12.720	20.751	66.407	0.517	1.548	0.691	2.069	0.166	0.418
GMM(3)	125.036	13.924	22.699	51.405	0.525	1.574	0.718	2.151	0.268	0.451
GMM(5)	125.036	13.924	22.699	51.405	0.525	1.574	0.718	2.151	0.268	0.451
GMM(7)	125.036	13.924	22.699	51.405	0.525	1.574	0.718	2.151	0.268	0.451
GMM(9)	125.036	13.924	22.699	51.405	0.525	1.574	0.718	2.151	0.268	0.451

Source: own processing

On train data, strategies based on K-means achieve great performance. Moreover, strategy K-means with 5 clusters was able to overperform benchmark S&P 500 in total returns, average yearly returns, lower maximal drawdown, and significantly higher Sharpe ratio. In contrast, GMM strategies were not able to converge, and the initial estimate provided by K-means was not changed. Interestingly, maximising total return earns a higher Sharpe ratio than optimising the Sharpe ratio.

On test data, Table 6 shows that the agent optimising the Sharpe ratio significantly reduces the maximal drawdown for the acceptable decrease in total return. The best performance was provided by K-means with 5 clusters. It is a good sign that the best strategy from train data is also the best strategy on test data. It may indicate that the strategy is not overfitted or underfitted. In fact, other K-means strategies are also consistent in their performance, and they are supreme over GMM. Overall, strategies have smaller drawdown than strategies from simulation 2, and a decrease in total return and average return is acceptable. More importantly, the best strategies decreased maximal drawdown and average annual return unproportionally. There is a decrease of 10% in the case of maximal drawdown and only around 1% in the case of average yearly return.

In summary, the results of the third simulation show:

- agents maximising the Sharpe ratio may achieve lower maximal drawdown than agents maximising total return,
- GMM may face convergence issues.

4.4 Simulation 4 - VIX market regime switcher

As results of previous simulations show, agents can overcome benchmarks with extended actions that allow them to exploit the environment. These strategies, introduced in the Simulation 2 section, doubled returns and increased the Sharpe and Sortino ratio. The problem that arises is a higher maximal drawdown. Some of the strategies exceed 100% of the maximal drawdown.

In Simulation 3, an agent maximising the Sharpe ratio over the following five returns is introduced to fight maximal drawdown and retain returns, overcoming the benchmark S&P 500 index. This agent lowered maximal drawdown and slightly decreased returns. Nevertheless, the agent was not able to avoid the sharpest depreciation in asset value, for example, the decline in 2020.

The most problematic challenge is the unpredictability of sharp declines with the current framework. The framework is focused on state representation and selecting the optimal action. It is not focused on price movement prediction, although these two approaches have a strong relationship.

Results from all simulations lead to this conclusion. Simply explained, the challenge arises in making a decision between the appreciation phase, market correction, sharp decline, and depreciation.

For instance, agents learn that leverage maximises objective function in the market appreciation phase, known as the uptrend. Agents also learn that the best action in negative oscillation around a trend known as market correction is leverage since significant appreciation commonly follows market correction. In case of persistent asset depreciation, agents select short-sell action or decrease position size.

The most problematic challenge comes from sharp declines. Agent expects corrections. Thus, the agent selects leverage, and instead, the decline continues, and the agent´s loss is greater than the loss on the traded asset. Results show that agents tend to start short selling almost at the point where the decline stops. Consequently, when the market begins to appreciate, the agent is conservative and misses significant returns, which are needed to compensate for the initial loss. The second possible scenario is that the agent always stays with leverage. As a result, maximal drawdown significantly exceeds the benchmark index and may not fully recover as a buy and hold strategy.

On the other hand, this approach statistically makes sense and works well most of the time, but not in times of crisis. This challenge also arises from the POMDP of the trading environment. The current state representation system is not able to capture the approaching crisis. This finding supports commonly known and discussed topics that simple quantitative trading systems often face difficulties when market conditions change dramatically and unexpectedly.

A new agent is introduced to this simulation to overcome the mentioned challenges. The most significant change in this simulation and in the agent´s decision process is the VIX regime switch with default action. The VIX regime switcher with default action aims to avoid sharp declines. The regime switcher is based on the moving average of VIX compared with the predefined threshold. A predefined threshold should be balanced between oversensitivity, which leads to frequent unnecessary signals, and undersensitivity, which leads to delayed reaction to major events. When this threshold is exceeded, the market is considered as uncertain, and the default action is selected.

Default action consists of 50% of capital accumulated in trading asset. This framework follows Jeong and Kim (2019). VIX switcher includes some challenges as well. For instance, some other states are falsely recognised as states with high level of uncertainty. In order to mitigate the negative consequences of short selling and to mitigate the loss of opportunity, default action consisting of 50% of capital accumulated in the trading asset is introduced.

The current state representation system is not able to differentiate between states that are followed by sharp declines and states that are not followed by sharp declines. The benefit of using the VIX regime switch is that VIX overtakes the development of financial markets. VIX is calculated as the difference in premiums of options with different expiration. Additional aspects, such as implied volatility and consensual expectation of investors, are priced in options. VIX effectively brings new, additional information to the state representation system, which cannot be observed by the current state representation system. As a result, states can be generalised and differentiated more effectively.

To summarise, an agent with a VIX regime switch selects the default action of 50% of capital accumulated in trading asset in states where the moving average of VIX exceeds the threshold. Agents usually act in states where the moving average of VIX does not exceed the threshold.

Table 7: Simulation 4 - Hyperparameters

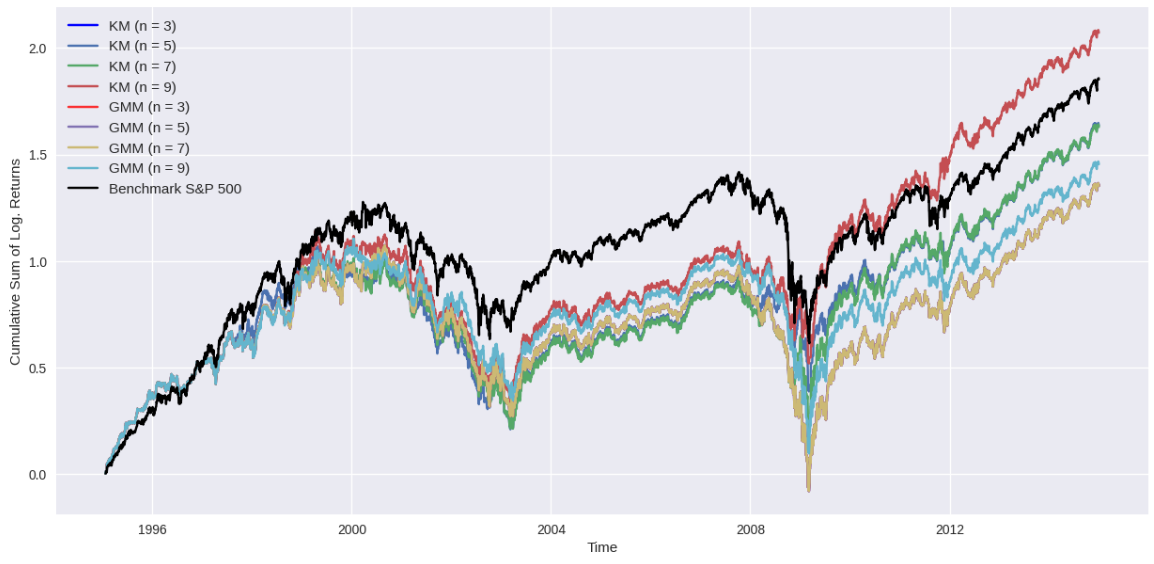
Parameter	Value
Learning Rate	0.1
Exploration Rate	0.3
Epochs	1 000
Actions	0, 0.5, 1, 1.5
Training period	1.2.1995 - 1.1.2015
Testing period	1.1.2015 - 1.1.2024
Assets	S&P 500 (SPY)

Source: own processing

Agent reward is given by equation below, where r_t corresponds to portfolio return generated in time t, which corresponds to trading days.

$$R(s,a) = 0.9 \cdot r_{t+1} + 0.7 \cdot r_{t+2} + 0.5 \cdot r_{t+3} + 0.2 \cdot r_{t+4} \tag{4.4}$$

There are also some minor changes. Firstly, the state representation system is adjusted. VIX switch limits agents to trade primarily in "certain" markets, where market appreciation and market corrections are more common than market depreciation. As a result, the state representation system has been changed to focus on state differentiation in market appreciation and market corrections such as solid market correction, weak market correction, etc. Secondly, the number of actions is decreased, and short selling is restricted. A modified state representation system could lead to false signals to short selling. The effects of false state assignment and representation described at the beginning of this section could be negatively affected. Thus, agents could generate more losses with short-sell actions than without them. Moreover, the agent can select the action not to trade, which should prevent losses in case of market depreciation. The other parameters of the simulation are the same.

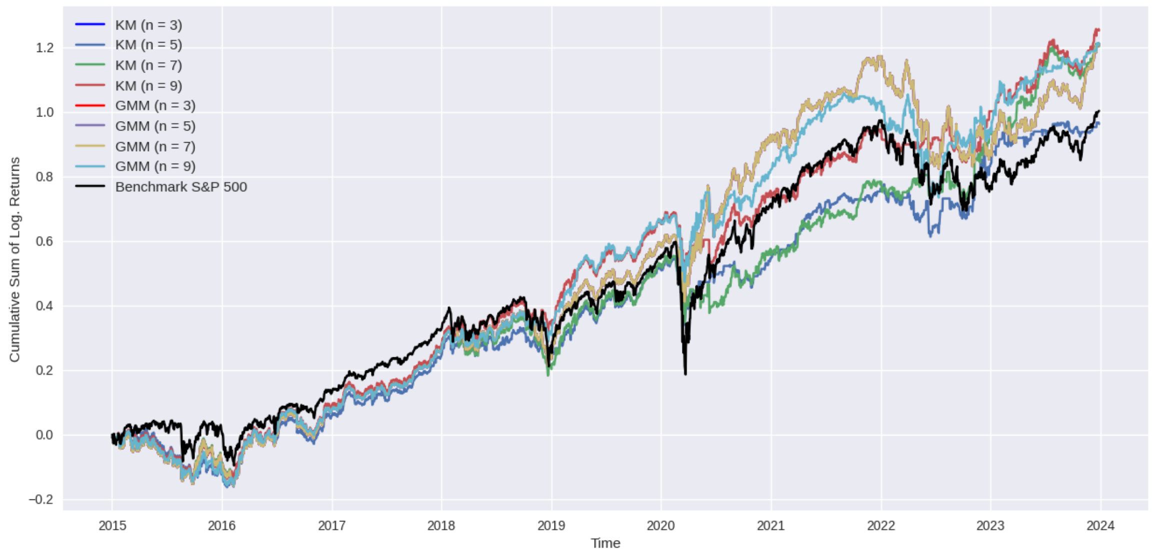


Source: own processing

Figure 13: Simulation 4 - Cumulative log. returns on train data

Figure 13 shows the cumulative returns of strategies on train data. As the Figure 13 shows, strategies underperform the benchmark index on the training dataset. This gap in returns was created around the year 2000 when agents stepped back and decreased leverage before the peak in 2000 and selected leverage actions too early. In addition, VIX levels were low during the 2008 crisis and high after the crisis. Thus, agents did not use default actions to decrease losses and did not exploit market appreciation with leverage after 2008. This is a significant setback of the introduced VIX regime switcher.

Figure 14 shows that agents struggled in the first months, and the benchmark index S&P 500, achieved better performance. The problem of agents is as follows: the market



Source: own processing

Figure 14: Simulation 4 - Cumulative log. returns on test data

goes sideways, and agents generate loss by switching between actions and default action. For example, the market depreciates with 100% of the agent's capital, and the market appreciates with 50% of the agent's capital, etc. Nevertheless, when the market starts trending, agents are able to exploit leveraged actions and achieve similar results as the benchmark index. More importantly, strategies do not experience the same loss in the 2020 crisis as the S&P 500. Also, agents are able to manage situations after the year 2022 with more ease. All of this is possible because of the VIX regime switcher.

Table 8: Simulation 4 - Results

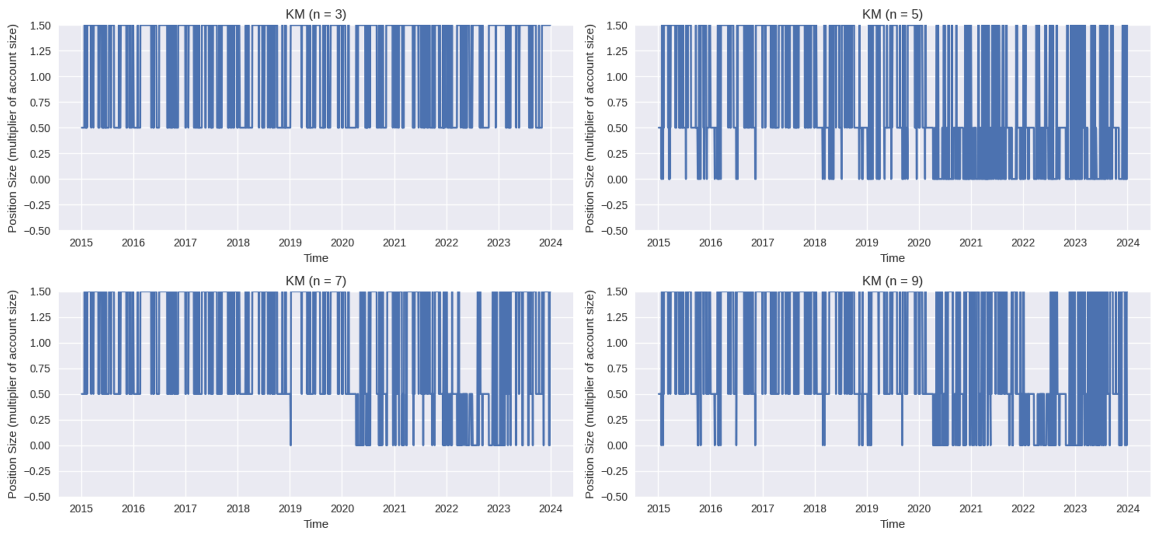
Model	Tot. ret.(%)	Avg. y. ret.(%)	Std.(%)	Max. DD(%)	SR Anl.	SR	SoR Anl.	SoR	IR	MW p-val
Training Data										
S&P 500	185.286	9.299	19.722	80.273	0.370	1.652	0.518	2.313	-	-
KM(3)	135.246	6.788	19.616	115.399	0.244	1.090	0.335	1.497	-0.395	0.682
KM(5)	163.533	8.208	17.068	85.893	0.364	1.623	0.505	2.256	-0.184	0.424
KM(7)	162.867	8.174	18.861	82.521	0.327	1.461	0.456	2.034	-0.181	0.700
KM(9)	207.160	10.397	18.087	80.375	0.464	2.072	0.655	2.924	0.181	0.746
GMM(3)	135.246	6.788	19.616	115.399	0.244	1.090	0.335	1.497	-0.395	0.682
GMM(5)	135.246	6.788	19.616	115.399	0.244	1.090	0.335	1.497	-0.395	0.682
GMM(7)	135.246	6.788	19.616	115.399	0.244	1.090	0.335	1.497	-0.395	0.682
GMM(9)	145.419	7.298	19.101	100.751	0.277	1.238	0.381	1.700	-0.319	0.633
Testing Data										
S&P 500	100.265	11.165	18.159	41.124	0.505	1.512	0.689	2.065	-	-
KM(3)	120.682	13.439	17.147	34.854	0.667	1.999	0.931	2.790	0.263	0.927
KM(5)	96.200	10.712	12.814	20.711	0.680	2.038	0.987	2.957	-0.059	0.133
KM(7)	120.262	13.392	14.066	20.776	0.810	2.427	1.156	3.463	0.280	0.483
KM(9)	125.155	13.937	13.511	20.776	0.884	2.648	1.250	3.745	0.353	0.529
GMM(3)	120.682	13.439	17.147	34.854	0.667	1.999	0.931	2.790	0.263	0.927
GMM(5)	120.682	13.439	17.147	34.854	0.667	1.999	0.931	2.790	0.263	0.927
GMM(7)	120.682	13.439	17.147	34.854	0.667	1.999	0.931	2.790	0.263	0.927
GMM(9)	120.830	13.455	15.382	32.289	0.745	2.232	1.059	3.173	0.277	0.449

Source: own processing

Table (8) shows that strategies struggle with training data. Only K-means with 9 clusters achieve better results than the benchmark index. This is mainly because of changes in the state representation system to deploy the VIX regime switcher, which was not functioning as proposed.

More importantly, as Table (8) shows, the performance of strategies on test data is similar to that of strategies from previous simulations. The most significant advantage of newly introduced strategies is exceptionally low maximal drawdowns with retained high returns that exceed the return of the benchmark index S&P 500.

Returns of 120% and maximal drawdown of 20.7% instead of 41.1% of the benchmark are great results. Sharpe and Sortino, ranging from 2 to 2.6 and from 3 to 3.7, respectively, approve the domination of proposed strategies over the benchmark S&P 500 index. These findings indicate the successful deployment of the VIX regime switcher on test data.



Source: own processing

Figure 15: Simulation 4 - Position sizing

Figure (15) shows the history of selected actions. Compared with Figure 10from Simulation 2, agents are more reactive and actively exploit the environment. Agents switch between three actions: buy leverage 1.5, default action 0.5, determined by the VIX regime switcher, and 0, corresponding to the most conservative action. Short selling is restricted in this simulation. As the figure shows, agents tend to be more conservative after the year 2020, and significant market appreciation from 2020 and peak in 2022 are traded with caution.

In summary, the results of the fourth simulation show:

- VIX regime may significantly decrease maximal drawdown and retain high returns,
- estimating the VIX threshold may be problematic. The proposed threshold was not able to recognise the 2008 crisis, but 2020 was recognised.

4.5 Simulation 5 - Stacked models

Challenges that arise from all simulations are that minor changes in the number of clusters significantly impact strategy performance. Thus, the performance of different approaches can not be compared since the results are not representative. Moreover, some strategies are established in various market conditions. Slightly different test periods or parametrisation may lead to significantly different performance results.

To overcome the mentioned challenges, agents from the same simulation are stacked together, and a simple average is used as a meta-model. In practice, agents propose actions describing the amount of capital to accumulate. Proposes are averaged, and the simple average of the proposed capital is accumulated. As a result, stacked agents are more representative and could be compared more confidently.

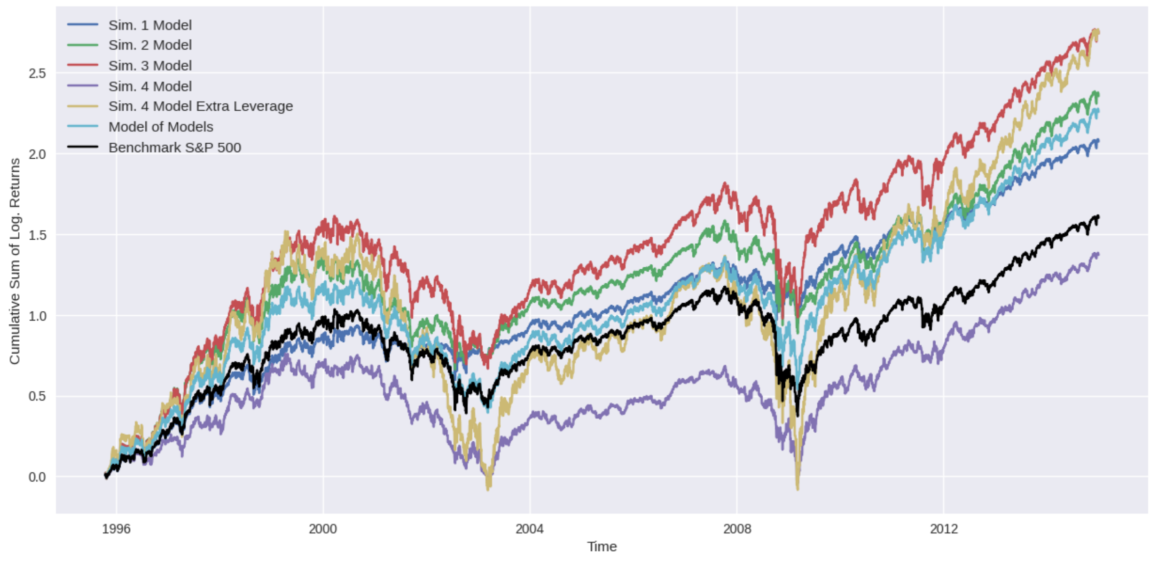
Models are stacked according to Table 9.

Table 9: Simulation 5 - Hyperparameters

Stacked Model	Simple Average of
Sim. 1 Model	KM(2,5,10,20), GMM(2,5,10,20)
Sim. 2 Model	KM(3,5,7,9), GMM(3,5,7,9)
Sim. 3 Model	KM(3,5,7,9)
Sim. 4 Model	KM(3,5,7,9)
Sim. 4 Model E. L.	KM(3,5,7,9), doubled leverage
Model of Models	Sim. (1-4 Models)

Source: own processing

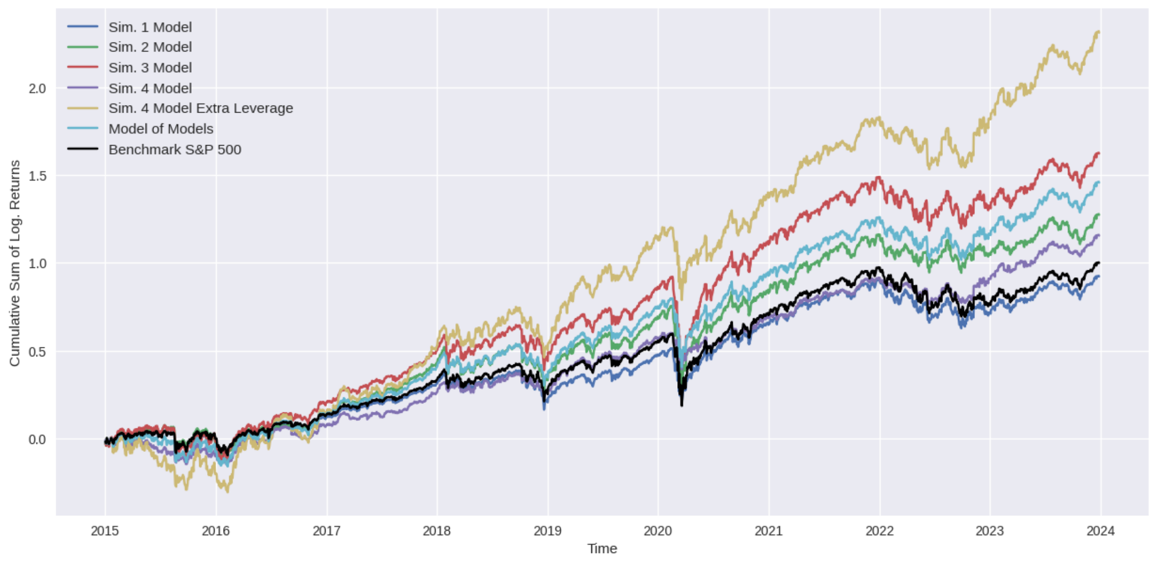
Figure 16 shows the cumulative returns of stacked agent strategies on train data. Even stacked agents cannot avoid market depreciation after 2000 and 2008, although some strategies are not affected as significantly as others.



Source: own processing

Figure 16: Simulation 5 - Cumulative log. returns on train data

Figure 17 shows the cumulative returns of stacked agent strategies on test data. In contrast to the train dataset, Model 4 with extended leverage significantly outperforms other strategies while achieving the same maximal drawdown.



Source: own processing

Figure 17: Simulation 5 - Cumulative log. returns on test data

Table 10: Simulation 5 - Results

Model	Tot. ret.(%)	Avg. y. ret.(%)	Std.(%)	Max. DD(%)	SR Anl.	SR	SoR Anl.	SoR	IR	MW p-val
Training Data										
S&P 500	185.286	9.299	19.722	80.273	0.370	1.652	0.518	2.313	-	-
Sim. 1 Mod.	207.306	10.803	16.630	39.338	0.529	2.319	0.762	3.337	1.906	<0.0
Sim. 2 Mod.	235.398	12.266	23.557	71.133	0.436	1.909	0.615	2.693	2.264	0.625
Sim. 3 Mod.	274.158	14.286	27.324	94.542	0.450	1.970	0.630	2.759	2.536	0.349
Sim. 4 Mod.	137.113	7.145	18.130	80.243	0.284	1.243	0.395	1.729	-0.536	0.078
Sim. 4 Mod. E. L.	274.227	14.290	36.260	160.486	0.339	1.485	0.472	2.069	1.233	0.636
Mod. of Models	225.640	11.758	23.378	82.895	0.417	1.829	0.586	2.567	2.112	0.006
Testing Data										
S&P 500	100.265	11.165	18.159	41.124	0.505	1.512	0.689	2.065	-	-
Sim. 1 Mod.	92.343	10.283	15.455	31.014	0.536	1.606	0.733	2.196	-0.563	<0.0
Sim. 2 Mod.	127.401	14.187	21.341	50.996	0.571	1.711	0.776	2.325	1.431	0.771
Sim. 2 Mod.	162.338	18.077	23.292	52.347	0.690	2.068	0.949	2.845	2.291	0.429
Sim. 4 Mod.	115.575	12.870	13.321	20.736	0.816	2.445	1.165	3.490	0.586	0.200
Sim. 4 Mod. E. L.	231.149	25.740	26.643	41.472	0.891	2.670	1.276	3.823	3.242	0.544
Mod. of Models	145.761	16.231	19.327	39.208	0.736	2.207	1.023	3.065	2.804	0.084

Source: own processing

Table 10 shows the benefits of model stacking. Most importantly, different strategies have their maximal drawdown in various states. Consequently, stacking models and averaging their returns do not lead to an average of their statistics. Thus, the maximal drawdown of stacked strategies is lower than the simple average of these strategies' statistics.

The stacked agent from the first simulation achieved the lowest maximal drawdown on train data, mainly because the agent was restricted to using leverage. Leverage restriction and the inability to exploit market appreciation lead to lower returns on test data. However, the results of the stacked model are better than those of particular strategies in the first simulation.

Interestingly, the model optimising Sharpe ratio has a higher maximal drawdown and higher return on the train and test dataset than the model optimising return. On the other hand, the model optimising the Sharpe ratio has higher Sharpe and Sortino ratios on both datasets.

Model 4 struggles on train data, as it was described in the previous section. In contrast, results on test data show that the stacked model achieves an exceptionally low maximal drawdown of 20.7% and a higher return than the S&P 500 benchmark index. In order to exploit the capabilities of this agent, the same agent with higher leverage is introduced. This agent can achieve a return of 231% compared to 100% of the S&P 500 with the same maximal drawdown of 41%. These results are exceptionally competitive. The VIX regime switcher causes the supremacy of this model. The only drawback of this model is the exceptionally high maximal drawdown on train data, which is caused by changes in the state representation system, higher leverage and challenging calibration of the VIX regime switcher.

Overall, the most stable results come from the Model of Models. Some agents underperform on test data and overperform on train data and vice versa. This is not the case with the Model of Models. This stacked model shows stability in time, a crucial component of any model. It seems that this model is not strongly affected by changing market conditions. This is because the different models with different attributes and abilities are stacked together.

In summary, the results show:

- stacked models effectively achieve lower maximal drawdown than particular strategies while retaining high returns,

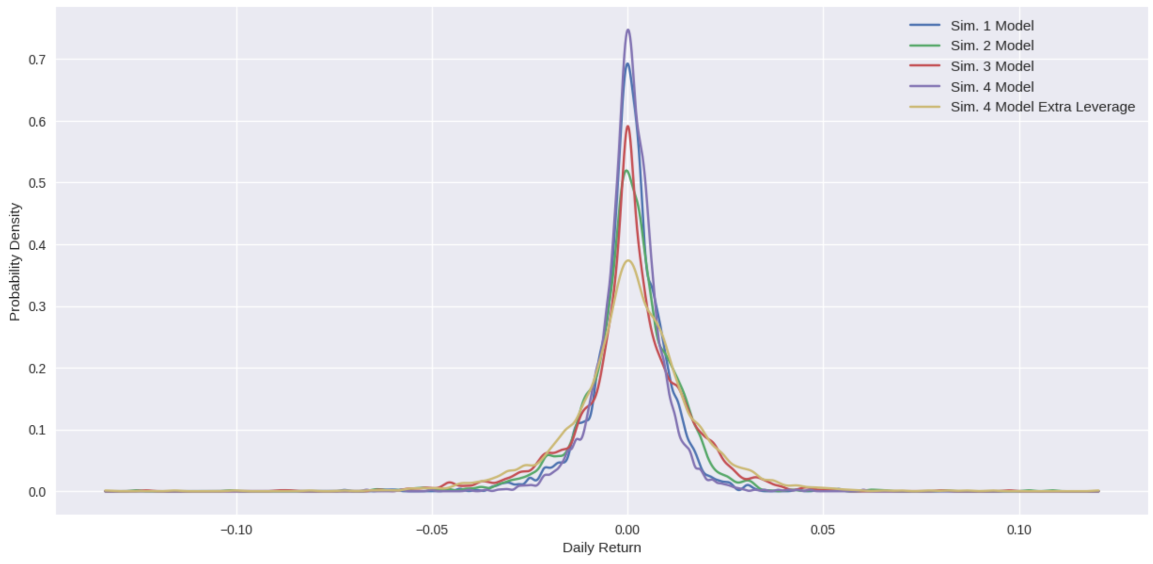
- Model of Models, which combines different approaches and different agents, has the most stable performance in time and can face changing market conditions,

- stacking models with exceptionally low maximal drawdown can lead to an additional decrease in maximal drawdown. This model may use additional leverage, leading to exceptionally high returns while achieving the same maximal drawdown as the benchmark.

4.6 Monte Carlo simulation

A Monte Carlo simulation calculates the expected cumulative returns of introduced strategies within the confidence interval. As Detemple, Garcia, and Rindisbacher (2003) show, it is a standard method for modelling the probability of different outcomes in stochastic process.

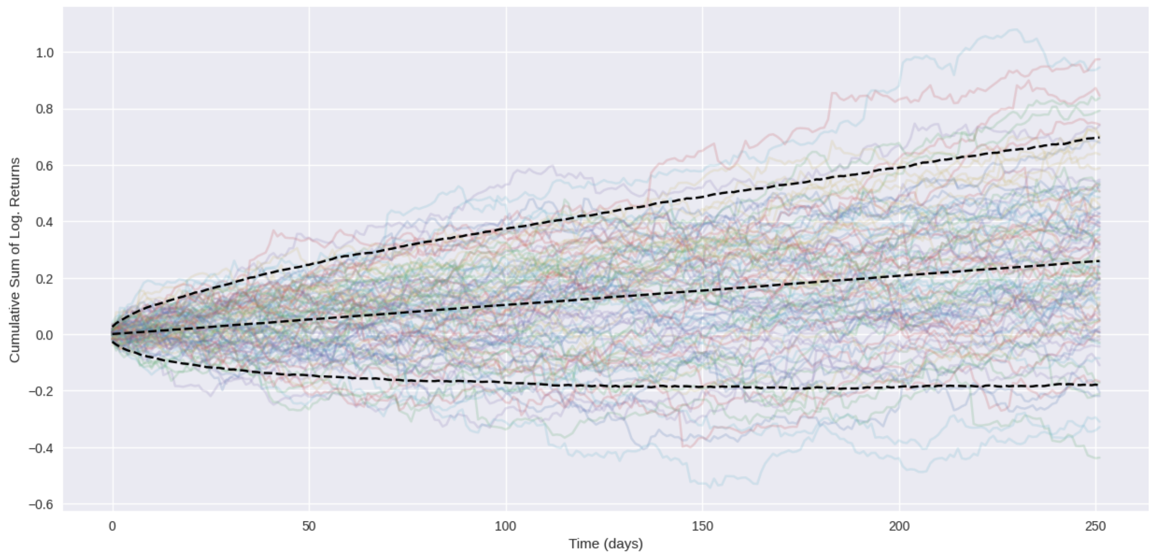
Monte Carlo simulation generates potential trajectories of strategies, where daily returns of strategy are randomly selected from the probability density function of historical returns of strategy on test data. Non-parametric kernel density estimation is used to estimate the probability density function of returns (O'Brien et al., 2016). Mersenne Twister pseudo-random number generator generates trajectories with as few skew as possible (Matsumoto and Nishimura, 1998).



Source: own processing

Figure 18: Estimated PDFs

Figure 18 shows the probability density functions of strategies calculated on test data. All probability density functions are centred around 0, distributions are slightly left-skewed, and a few long tails can be observed.



Source: own processing

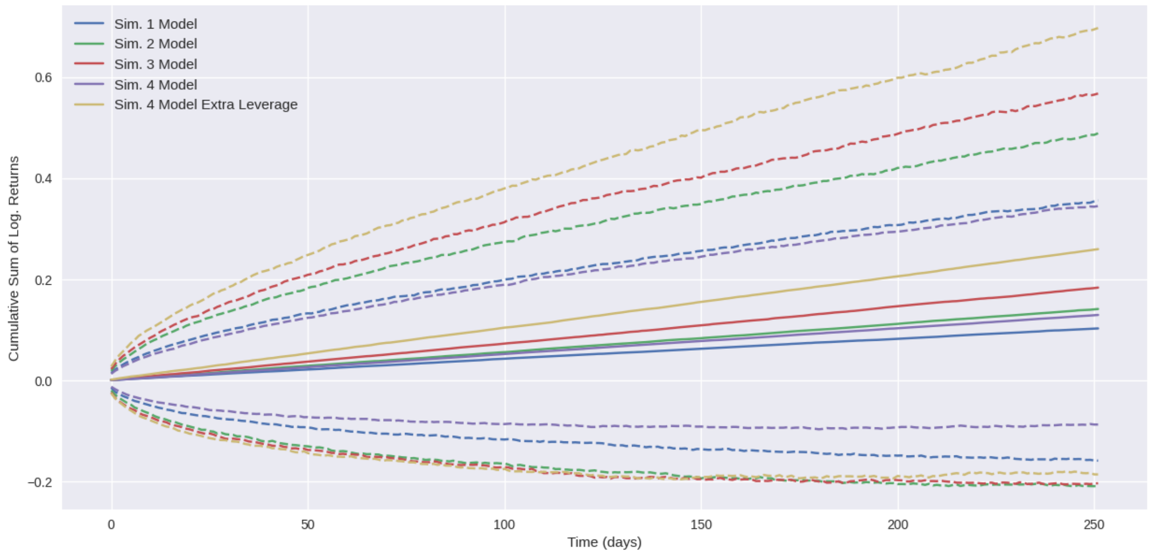
Figure 19: Monte Carlo trajectories simulation

Figure (reffig:Sim6Traj shows possible trajectories of Sim. 4 Model with extended actions, generated with Monte Carlo simulation. The majority of trajectories oscillate around the mean. However, some of the strategies diverge significantly. These trajectories represent the probability that even strategies with well-simulated and known probability density functions may experience significant losses.

Figure 20 shows a simulated expected return with 5% and 95% confidence intervals. Simulation has 10,000 trajectories per model and 252 steps to simulate one year of trading. This Figure shows the supremacy of Model 4 with extra leverage, with the maximum expected loss being approximately 20%, similar to the other models. In contrast, the average and maximal expected returns are significantly higher. Moreover, confidential intervals show that maximal loss stops at approximately 20%. In addition, Model 1 has the same upside potential as Model 4 with base leverage. However, Model 4 has a significantly smaller probability of losses than Model 1.

In summary, the results show:

- different models have almost the same probability of achieving a loss of 20% in one year, while the likelihood of achieving a higher return differs significantly,



Source: own processing

Figure 20: Estimated confidence intervals

- simulated possible trajectories indicate that models may experience significant losses.

5 Discussion

The previous chapter showed the capabilities of the Q-learning agents and clustering algorithms framework. Some of the models faced challenges in convergence or unacceptable high maximum drawdowns. In the end, however, it was shown that the proposed framework trained on data from 1996 to 2015 and tested from 2015 to 2024 is competitive. Results are discussed from different points of view in this chapter.

5.1 Results comparison

One of the most interesting findings is the introduced framework's ability to deal with noisy and complex financial time series data. Q-learning is an entry-level reinforcement learning algorithm. However, as was shown, with correct parametrisation and action selection, Q-learning finds optimal policy smoothly. Q-learning, together with K-means and Gaussian mixture model clustering, is capable of results similar to significantly more complex frameworks. With benefits such as simplicity and explainability, this framework is highly competitive.

In contrast, Théate and Ernst (2021) could not overcome the buy and hold strategy trading S&P 500, even with the Deep Q-learning agent, which is a significantly more complex model. Their approach also relied on active trading rather than weight rebalancing. On the other hand, their test period is from 2018 to 2019. The short, unfavourable test period probably caused non-competitive results.

The framework presented in this diploma thesis is similar to the framework proposed by Chakole et al. (2021). Both frameworks consist of Q-learning agents and clustering algorithms. Different numbers of clusters were tested, and simulation results show that strategies from 3 to 7 clusters achieve the best results. This finding corresponds to Chakole et al. (2021), which focuses on the same range of clusters and the same K-means clustering algorithm.

Their results did not prove the efficiency of the introduced approach, which does not correspond to the findings in this diploma thesis. Although tested assets are different, the NASDAQ and S&P 500 are comparable. Agents in this diploma thesis were able to overcome the buy and hold strategy while relying on a similar state representation system with K-means and show that clustering may be utilised successfully.

The major difference between Chakole et al. (2021) and this diploma thesis is the utilisation of short selling and leverage. Agents capable of buying assets with leverage can exploit the environment effectively, leading to higher results. However, leverage may result in a higher maximal drawdown.

On the other hand, Pendharkar and Cusatis (2018) showed that their agents accumulate most of their capital into traded assets in most states. These findings correspond to the results of the diploma thesis, where the agent selected the maximal possible accumulation most of the time. Thus, agents tend to exploit the environment and probably should have options to exploit the environment properly. Moreover, agents maximising the Sharpe ratio achieved higher returns, corresponding to Pendharkar and Cusatis's (2018) findings.

It was shown that a market regime switch is an effective way to deal with high maximum drawdowns. Agents implemented a market regime switch with a default action consisting of an accumulation of 50% of capital, were able to face volatile markets, as results from Simulation 4 showed. Moreover, regime switches lead to the introduction of agents with higher returns and only half maximal drawdown in contrast to the S&P 500 buy and hold strategy.

Jeong and Kim (2019) highlight the power of market regime switches. This paper inspired the use of market regime switches with default action in this diploma thesis. Moreover, using the market regime switch may lead to higher returns and a lower maximal drawdown, which also corresponds to Jeong and Kim (2019), who declare that using the switch increases profits.

Another factor distinguishing this diploma thesis from other papers is the stacking of agents. Stacked agents proved to be an exceptional solution since agents experience lower maximal drawdowns. More importantly, stacked agents are more robust since they incorporate various agents with differently generalised states. Thus, the results of these models are more persistent over time. Stacked model of stacked models persistently overperform buy and hold strategy.

5.2 Current challenges and their possible improvements

Results based on the framework presented in this diploma thesis outline future directions discussed in this section. Possible improvements include a better state representation system, adopting complex algorithms, and developing market regime switch algorithms. These possible improvements are also part of potential future research.

5.2.1 State representation system

The greatest challenge for the current simulation framework is the state representation system. Time series descriptive statistics with various parameters are used to represent different states. States are clustered to create generalised states.

The challenge lies in creating a system recognising the differences between market correction and market depreciation states. The trading environment was correctly labelled as POMDP, similar to Kabbani and Duman (2022), where agents had to create their own state representation system. Results showed that the created explanatory variables do not contain information that allows agents to observe the trading environment fully. However, the introduced state representation system is better than the state representation system based on directly observable prices.

Clustering algorithms correctly determine the cluster to which a state should be assigned in the majority of cases. On the other hand, a smaller portion of falsely determined states caused the most significant complications, such as exceptionally high maximal drawdowns in market crises.

The real challenge is finding the balance between over-sensitive and under-sensitive systems. For example, Simulation 4 tested a slightly different state representation system, which achieved superb results compared to previous simulations on test data. Similar states were clustered together, and differences between distant states were found. However, results from the train data showed that these results could be more persistent and that the dynamics of the trading environment have probably changed. This corresponds to Wu et al. (2020) findings. Wu et al. (2020) incorporate the same explanatory variables and suggest that explanatory variables should change based on market conditions. The results of this diploma thesis indicate the same conclusion.

This challenge is closely related to the second challenge, which is selecting the correct number of clusters and convergence issues. Two algorithms were used. While K-means was converging properly, the Gaussian Mixture Model faced some difficulties. The Gaussian Mixture Model was not able to find patterns in data that would follow a mix of Gaussian distributions.

The more clusters lead to a more sensitive system. These systems may recognise different states, leading to higher returns and lower maximal drawdowns. Results showed that high numbers of clusters might lead to situations where clusters are determined based on noise in data. Results also showed that models with fewer clusters are more robust. Nevertheless, they do not have the discriminating ability to overcome benchmark S&P 500 performance with such a difference as models with more clusters.

A solution for both challenges could be the introduction of new explanatory variables, consisting of new time series based variables such as wavelet transform (Burrus et al., 1998) and Hilbert-Huang transformation with IMFs decomposition as Huang et al.(2003) show. Additional explanatory power could be created by using supervised machine learning models such as XGBoost (Chen et al., 2015) or LSTM (Hochreiter and Schmidhuber, 1997) to generate input for the reinforcement learning framework.

Moreover, results from Simulation 4 showed that a significant amount of explanatory power could be extracted from explanatory variables that are not directly connected to time series data, such as the VIX index. These results show that it could be helpful to integrate different sources of explanatory variables. These findings correspond to Kabbani and Duman (2022), who calculate sentiment scores with the BERT model.

For example, macroeconomic data such as consumption and production index, inflation, and household sentiment. The most significant drawback of macroeconomic data is their frequency, as they are often released quarterly. In addition, data from social media and news announcements may be included. These data may be challenging to process, although additional information would be inevitably extracted. Importantly, these sources could be highly efficient in volatile and uncertain market conditions.

5.2.2 Multi-asset portfolio and advanced algorithms

A single asset portfolio provides benefits from overall simplicity. In contrast, a multi-asset portfolio with properly chosen assets may lead to lower unsystematic risk. This is not exactly a problem of the index S&P 500 used in this thesis because S&P is already portfolio of 500 companies. Consequently, a portfolio with carefully selected companies from the S&P 500 index may lead to higher returns and lower maximal drawdowns.

The current framework is not able to handle multi-asset portfolios for two reasons. Firstly, as mentioned previously, the state representation system should be improved. The state representation system capable of capturing relationships between assets should be developed to handle multiple assets. Modern portfolio theory uses a covariance matrix to capture relationships between assets.

Secondly, due to computational complexity, Q-learning cannot be used for a higher number of assets. Models able to handle state action space with higher dimensionality would be required. Moreover, it could be helpful to implement continuous actions and continuous states to achieve optimal solutions, which may lead to higher returns.

This model could be Deep Q-learning, Deterministic Policy Gradient, Deep Deterministic Policy Gradient, or Twin-Delayed Deep Deterministic Policy Gradient. In addition to being able to handle the ample state action space of multi-asset portfolios, these more complex agents optimise the policy directly without explicitly using a value function.

In practice, this may lead to a framework where clustering into states is unnecessary. All simulations and results showed that the most susceptible part of the framework is clustering into states. Avoiding this problem may lead to better, more consistent results.

Even advanced and current state-of-the-art algorithms cannot discover information in data if there is simply no information indicating upcoming high volatility market conditions. On the other hand, clustering algorithms based on their parameters as distance calculation equations mostly assign the same weights to all of the explanatory variables. Neural networks may assign weights to explanatory variables according to their explanatory power. Moreover, neural networks are capable of operating with data without linear dependencies.

As a result, neural networks may learn to predict in two regimes, certain and uncertain markets. For example, a neural network may determine one or two explanatory variables or their combination, which describes current or coming uncertainty markets and changes predicted actions accordingly. The current framework based on clustering is not capable of internal regime switching. This would be a major benefit of more advanced complex frameworks similar to Kabbani and Duman (2022), who also consider the trading environment as POMDP and successfully deploy Twin Delayed Deep Deterministic Policy Gradient.

5.2.3 Regime switch

Even complex models would face challenges arising from the trading environment's POMDP. First, as discussed above, even the state-of-the-art model cannot discover information in data if there is simply no information. Second, due to POMDP, the same explanatory variables may represent significantly different states.

Due to these closely related reasons, additional regime switchers with additional data sources may be needed. Regime switchers may focus on one thing only: distinguishing between different market regimes. This approach has the following benefits, which may benefit the whole framework.

Firstly, training a model that focuses on multiple objectives could be challenging. As shown in the results chapter, the introduced framework may avoid sharp declines and crises, although statistically, it was better to experience significant losses and exploit significant market appreciation.

Secondly, as shown in Simulation 4, additional sources of information can be smoothly incorporated into regime switchers. These new explanatory variables could be carefully selected as variables that may have a higher probability of estimating the certainty of markets. These explanatory variables may also be incorporated into the reinforcement learning agent. However, additional explanatory variables may lead to further problems as a model may focus on these added explanatory variables. Consequently, reinforcement learning agents may use these explanatory variables to describe states that may not be appropriate for these variables.

As a result, external components utilising different information sources in the form of regime switcher, which was introduced in Simulation 4 in simple form, may lead to a significant decrease in maximal drawdown. The introduced regime switcher optimised only maximal drawdown. This framework experienced significantly better results compared to Simulation 3, where maximal drawdown was optimised together with return in the form of Sharpe ratio maximisation.

5.3 Practical aspects

This section consists of practical aspects of introduced strategies such as expense ratio estimation and recommendations for investors.

5.3.1 Expense ratio estimation

Introduced strategies rebalance daily, although the amount of capital allocated does not change daily. S&P 500 index may be traded via various ETFs. The extra costs of borrowing money arising from leverage are avoidable, and leveraged ETFs can provide leverage.

Table 11: Expense ratio analysis

Parameter	Value
Capital	1 mil. USD
Average Number of Traded Sh.	6600
Number of Rebalances	252
Spread per Sh.	0.01
Fees per Sh.	0.002
Market Aspect per Sh.	0.02

Source: own processing

As leveraged ETFs are commonly priced around 100 USD, 5,000 to 7,500 shares are needed to simulate the desired leverage with the provided capital. The market aspect consists of slippage, timing costs, and tracking errors.

The following equation is used for the rough estimation of the expense ratio.

$$\text{Expens. ratio} = \frac{\text{Num. Shares} \cdot (\text{Spread} + \text{Fees} + \text{Market Aspects}) \cdot \text{Rebalances}}{\text{Capital}} \tag{5.1}$$

The roughly estimated cost of this strategy ranges from 0.5% to 6% p.a. plus the expense ratio of the selected ETF, where a 6% expense ratio corresponds to rebalancing every day, where agents short-sells one day and the following day, agent change weights to maximal leverage and vice versa, for one year. This corresponds to the maximal presented trading costs. Agents trading less frequently may have an expense ratio ranging from 1% to 2% in the low or mid-volatile market, plus the expense ratio of selected ETFs.

It should be mentioned that the strategy is not optimised in terms of cost-effectiveness. Some of the agents trade frequently, and some do not. Some agents have not traded for many months. Some actions could be avoided, and the number of trades could be decreased while retaining competitive characteristics.

One solution could be to decrease trading frequency to weekly rebalancing. A more convenient solution to reduce trading costs could be to utilise various ETFs and other instruments and divide rebalancing orders into smaller orders, for example, a 5-15 minute time window.

5.3.2 Recommendations for investors

Results from the simulation have direct implications for investors. Recommendations are as follows.

- The most critical recommendation is consistency. Strategies outperform benchmark indexes mainly because they follow "rules" that are statistically backtested.
- Reinforcement learning may be deployed to solve various portfolio allocation problems. However, compared to standard modern portfolio theory, the complexity of the whole framework significantly increases with the number of assets.
- Descriptional statistics of time series can be used for state representation. However, the trading environment is POMDP.
- Market regime switchers operating with additional explanatory variables may significantly lower maximal drawdown.
- Stacking agents with more or less statistical independence returns may lead to higher returns and lower maximal drawdowns.

Conclusion

This diploma thesis aims to introduce and evaluate a simple reinforcement learning framework for asset allocation. This diploma thesis theoretically described and introduced the reinforcement learning framework for stock trading, which was subsequently tested and evaluated. The proposed reinforcement learning framework outperformed the S&P 500 benchmark index in terms of average return, maximal drawdown, and Sharpe ratio.

Q-learning off-policy discrete action space algorithm is utilised to find an optimal policy through the iteration process. Optimal policy maps state to an optimal action which maximises cumulative reward. The agent's objective function is defined as the maximisation of the discounted portfolio return of 5 following trading days.

The whole portfolio allocation optimisation problem is modelled as MDP. However, it is explained that the trading environment is POMDP, and the agent's own state representation system is needed. States are represented by time series descriptive statistics such as lagged returns, exponential moving average, RSI, and others.

To overcome challenges arising from noisy and POMDP trading environment, assigning state-action values to unseen states by classifying states into clusters is incorporated. K-means and Gaussian mixture models with 3, 5, 7, and 9 clusters are deployed. Assigning unseen test states into generalised clusters of states helped to determine the optimal action for test states.

The S&P 500 index, which consists of 500 US companies, was selected to test the reinforcement learning framework. Time serie captures the S&P 500 from 1995 to 2024. Time serie is divided into two parts corresponding to the train and test datasets, following standard time series machine learning forecasting methodology. The train dataset consists of observations from 1995 to 2015, and the test dataset consists of observations from 2015 to 2024. Total return, average yearly return, maximal drawdown, Sharpe ratio, and Sortino ratio are used to evaluate portfolio performance.

The Introduced reinforcement learning framework is empirically tested in five simulations. The first simulation consists of agents with limited actions. Results showed that the problem of model selection was based on past performance. High performing models on the train dataset were overcome by low performing models on the test dataset.

The first simulation also showed that the Q-values of actions converge properly. Thus, the optimal policy can be found.

The second simulation showed that agents capable of short selling and trading with leverage can easily overcome benchmark indexes. These agents overperform the returns of the S&P 500 on train and test data. The best model achieved returns over 340% compared to 160% on train data and 157% compared to 100% on test data while retaining the Sharpe ratio of 2.6 and 1.8. However, most of the agents increased maximal drawdown simultaneously with returns.

The third simulation aimed to retain returns from simulation 2 and lower maximal drawdowns of agents. Agent maximising Sharpe ratio was utilised. This approach led to a decrease in the maximal drawdown of some agents. However, this approach was not as effective as expected.

The fourth simulation utilised the VIX market regime switch with default action calculated as the moving average of the VIX index and predefined threshold. Switch consisted of two regimes corresponding to certain and uncertain markets. Agents from Simulation 4 avoided the 2020 crisis and achieved unexpectedly low maximal drawdowns while returns were retained. However, the results on train data show that the VIX market regime switcher did not recognise the 2008 crisis, which led to false signals.

These findings showed that patterns and relationships change, and the method's effectiveness is subject to time decay. Agents from Simulation 4 achieved a highly competitive maximal drawdown of 20.7% compared to the 41% maximal drawdown of S&P 500 on test data and achieved a total return higher by 20% simultaneously. These results showed that reinforcement learning agents supported by market regime switch lead to unexpectedly low maximal drawdowns while returns were retained.

The last simulation stacked models from the same Simulations together, which led to the creation of models with time-persistent results. Simulation 4 showed that stacking models with even partially independently distributed returns may significantly increase returns and lower maximal drawdown.

The most robust model, a stacked model of stacked models, consistently overperforms the benchmark S&P 500 buy and hold strategy. The model was able to achieve persistently higher returns and similar maximal drawdown. This model achieves a Sharpe ratio of 1.8 compared to 1.4 on train data and 2.2 compared to 1.5 on test data. The most powerful stacked model, which consisted of a leveraged agent utilising VIX regime switcher, was able to achieve a return of 231% in contrast to 100%, the Sharpe ratio of 2.6 compared to 1.5 with a similar maximal drawdown of 41% as buy and hold strategy. These results show the high competitiveness of the whole introduced reinforcement learning framework.

Based on the evidence provided in Simulations 1, 2, 3, 4, and 5, where benchmark buy and hold strategy was overcome, the hypothesis that portfolio allocation based on a reinforcement learning framework overperforms a buy and hold allocation strategy is accepted. Based on the provided evidence from Simulation 4 and 5, where VIX regime switch with default action is introduced, the second hypothesis that a reinforcement learning agent combined with a market regime switch decreases the maximal drawdown of the portfolio compared to an agent without a market regime switch is accepted.

Stacked agents are further investigated using the Monte Carlo simulation framework. It shows that models should not exceed losses higher than 20% in one year with a 95% probability. Generated trajectories also indicated that introduced models might face significant unexpected losses.

Obtained results are compared with those of other authors in the chapter devoted to discussing results. Compared to other authors, results from the simple reinforcement learning framework indicate the proposed framework's high competitiveness. Moreover, proposed agents were more competitive than agents of different authors. This is mainly due to the ability of agents to exploit the environment by utilising short selling and leverage. State representation system drawbacks, number of used clusters, and importance of market regime switches correspond to other researchers.

Various limitations of the current framework such as a state representation system, inability to simulate multi-asset portfolios and continuous actions, are presented. It is shown that the expense ratio for introduced agents may range from 1% to 6% p.a.

Future research may focus on sophisticated state representation systems and more advanced algorithms, such as twin-delayed deep deterministic policy gradient.

Bibliography

[1] ÅSTRÖM, Karl Johan. Optimal control of Markov processes with incomplete state information I. Journal of mathematical analysis and applications, 1965, 10: 174-205.

[2] BELLMAN, Richard E.; DREYFUS, Stuart E. Applied dynamic programming. Princeton university press, 2015.

[3] BELLMAN, Richard. A Markovian decision process. Journal of mathematics and mechanics, 1957, 679-684.

[4] BOX, George EP; JENKINS, Gwilym M. Time series analysis. Forecasting and control. Holden-Day Series in Time Series Analysis, 1976.

[5] BURRUS, C. Sidney; GOPINATH, Ramesh A.; GUO, Haitao. Wavelets and wavelet transforms. rice university, houston edition, 1998, 98.

[6] COVER, Thomas; HART, Peter. Nearest neighbor pattern classification. IEEE transactions on information theory, 1967, 13.1: 21-27.

[7] DEMPSTER, Michael AH; LEEMANS, Vasco. An automated FX trading system using adaptive reinforcement learning. Expert systems with applications, 2006, 30.3: 543-552.

[8] DETEMPLE, Jerome B.; GARCIA, Ren; RINDISBACHER, Marcel. A Monte Carlo method for optimal portfolios. The journal of Finance, 2003, 58.1: 401-446.

[9] DU, Xin; ZHAI, Jinjian; LV, Koupin. Algorithm trading using q-learning and recurrent reinforcement learning. positions, 2016, 1.1: 1-7.

[10] EL KAROUI, Nicole; PARENT, Antoine; PRADIER, Pierre-Charles. Louis Bachelier's Théorie de la spéculation: The missing piece in Walras' general equilibrium. Documents de travail du Centre d'Économie de la Sorbonne, 2022.

[11] FABBRI, Mirco; MORO, Gianluca. Dow Jones Trading with Deep Learning: The Unreasonable Effectiveness of Recurrent Neural Networks. In: Data. 2018. p. 142-153.

[12] FOSTER, Frederic G. On the stochastic matrices associated with certain queuing processes. The Annals of Mathematical Statistics, 1953, 24.3: 355-360.

[13] FUJIMOTO, Scott; HOOF, Herke; MEGER, David. Addressing function approximation error in actor-critic methods. In: International conference on machine learning. PMLR, 2018. p. 1587-1596.

[14] GAO, Xiu; CHAN, Laiwan. An algorithm for trading and portfolio management using q-learning and sharpe ratio maximization. In: Proceedings of the international conference on neural information processing. 2000. p. 832-837.

[15] GAUGNIUC, Paul A. (2017). Markov Chains: From Theory to Implementation and Experimentation. USA, NJ: John Wiley & Sons. pp. 2–8. ISBN 978-1-119-38755-8.

[16] GOODWIN, Thomas H. The information ratio. Financial Analysts Journal, 1998, 54.4: 34-43.

[17] HARTIGAN, John A.; WONG, Manchek A. Algorithm AS 136: A k-means clustering algorithm. Journal of the royal statistical society. series c (applied statistics), 1979, 28.1: 100-108.

[18] HARVEY, Campbell R.; LIU, Yan. Evaluating Trading Strategies. The Journal of Portfolio Management. 2014, 40, č. 5, s. 108-118.

[19] HASSAN, Md Rafiul; NATH, Baikunth. Stock market forecasting using hidden Markov model: a new approach. In: 5th international conference on intelligent systems design and applications (ISDA'05). IEEE, 2005. p. 192-196.

[20] HASTIE, Trevor, et al. The elements of statistical learning: data mining, inference, and prediction. New York: springer, 2009.

[21] HAU, Harald. Location matters: An examination of trading profits. The Journal of Finance, 2001, 56.5: 1959-1983.

[22] HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long short-term memory. Neural computation, 1997, 9.8: 1735-1780.

[23] HOWARD, Ronald A. Dynamic programming and markov processes. 1960.

[24] HUANG, Norden E., et al. Applications of Hilbert–Huang transform to non-stationary financial time series analysis. Applied stochastic models in business and industry, 2003, 19.3: 245-268.

[25] CHAKOLE, Jagdish Bhagwan, et al. A Q-learning agent for automated trading in equity stock markets. Expert Systems with Applications, 2021, 163: 113761.

[26] CHEKHLOV, Alexei; URYASEV, Stanislav; ZABARANKIN, Michael. Drawdown measure in portfolio optimization. International Journal of Theoretical and Applied Finance, 2005, 8.01: 13-58.

[27] CHEN, An-Sing; LEUNG, Mark T.; DAOUK, Hazem. Application of neural networks to an emerging financial market: forecasting and trading the Taiwan Stock Index. Computers & Operations Research, 2003, 30.6: 901-923.

[28] CHEN, Jun-Hao; TSAI, Yun-Cheng. Encoding candlesticks as images for pattern classification using convolutional neural networks. Financial Innovation, 2020, 6.1: 26.

[29] CHEN, Kai; ZHOU, Yi; DAI, Fangyan. A LSTM-based method for stock returns prediction: A case study of China stock market. In: 2015 IEEE international conference on big data (big data). IEEE, 2015. p. 2823-2824.

[30] CHEN, Tianqi, et al. Xgboost: extreme gradient boosting. R package version 0.4-2, 2015, 1.4: 1-4.

[31] CHENG, Dawei, et al. Financial time series forecasting with multi-modality graph neural network. Pattern Recognition, 2022, 121: 108218.

[32] JANG, Junkyu and NohYoon SEONG. Deep reinforcement learning for stock portfolio optimization by connecting with modern portfolio theory. Expert Systems with Applications. OXFORD: Elsevier, 2023, roč. 218, 12 s. ISSN 0957-4174.

[33] JEONG, Gyeeun; KIM, Ha Young. Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. Expert Systems with Applications, 2019, 117: 125-138.

[34] KABBANI, Taylan; DUMAN, Ekrem. Deep reinforcement learning approach for trading automation in the stock market. IEEE Access, 2022, 10: 93564-93574.

[35] KAUFMAN, Leonard; ROUSSEEUW, Peter J. Finding groups in data. an introduction to cluster analysis. Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics, 1990.

[36] KIM, Kyoung-jae; HAN, Ingoo. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. Expert systems with Applications, 2000, 19.2: 125-132.

[37] LEE, Jae Won, et al. A multiagent approach to Q-learning for daily stock trading. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2007, 37.6: 864-877.

[38] LEFÈVRE, Edwin, Jon D. MARKMAN and Paul Tudor JONES. Reminiscences of a Stock Operator: With New Commentary and Insights on the Life and Times of Jesse Livermore, Annotated Edition. [s.l.]: John Wiley & Sons, 2010. ISBN 978-0-470-48159-2.

[39] LILLICRAP, Timothy P., et al. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.

[40] MARKOV, Andreĭ Andreevich. An example of statistical investigation of the text Eugene Onegin concerning the connection of samples in chains. Science in Context, 2006, 19.4: 591-600.

[41] MARKOWITZ, Harry. The utility of wealth. Journal of political Economy, 1952, 60.2: 151-158.

[42] MATSUMOTO, Makoto; NISHIMURA, Takuji. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation (TOMACS), 1998, 8.1: 3-30.

[43] MNIH, Volodymyr, et al. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

[44] MOODY, John, et al. Performance functions and reinforcement learning for trading systems and portfolios. Journal of forecasting, 1998, 17.5-6: 441-470.

[45] MOODY, John; SAFFELL, Matthew. Learning to trade via direct reinforcement. IEEE transactions on neural Networks, 2001, 12.4: 875-889.

[46] NACHUM, Ofir, et al. Bridging the gap between value and policy based reinforcement learning. Advances in neural information processing systems, 2017, 30.

[47] O'BRIEN, Travis A., et al. A fast and objective multidimensional kernel density estimation method: fastKDE. Computational Statistics & Data Analysis, 2016, 101: 148-160.

[48] Patterson, Scott (2010-02-02). The Quants: How a New Breed of Math Whizzes Conquered Wall Street and Nearly Destroyed It. Crown. ISBN 978-0-307-45339-6.

[49] PENDHARKAR, Parag C.; CUSATIS, Patrick. Trading financial indices with reinforcement learning agents. Expert Systems with Applications, 2018, 103: 1-13.

[50] PUTERMAN, Martin L. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014.

[51] QIU, Jiayu; WANG, Bin; ZHOU, Changjun. Forecasting stock prices with long-short term memory neural network based on attention mechanism. PloS one, 2020, 15.1: e0227222.¨

[52] RASMUSSEN, Carl. The infinite Gaussian mixture model. Advances in neural information processing systems, 1999, 12.

[53] RUMMERY, Gavin A.; NIRANJAN, Mahesan. On-line Q-learning using connectionist systems. Cambridge, UK: University of Cambridge, Department of Engineering, 1994.

[54] RUSSELL, Stuart J.; NORVIG, Peter. Artificial intelligence: a modern approach. Pearson, 2016.

[55] SHARPE, William F. The Sharpe Ratio. Journal of Portfolio Management, 1994, 21.1: 49.

[56] SHI, Yong, et al. Stock trading rule discovery with double deep Q-network. Applied Soft Computing, 2021, 107: 107320.

[57] SILVER, David, et al. Deterministic policy gradient algorithms. In: International conference on machine learning. Pmlr, 2014. p. 387-395.

[58] SORTINO, Frank A.; PRICE, Lee N. Performance measurement in a downside risk framework. the Journal of Investing, 1994, 3.3: 59-64.

[59] SUTTON, Richard S., et al. Policy gradient methods for reinforcement learning with function approximation. Advances in neural information processing systems, 1999, 12.

[60] SUTTON, Richard S.; BARTO, Andrew G. Reinforcement learning: An introduction. MIT press, 2018.

[61] SUWANDA, R.; SYAHPUTRA, Zulfahmi; ZAMZAMI, Elvi M. Analysis of euclidean distance and manhattan distance in the K-means algorithm for variations number of centroid K. In: Journal of Physics: Conference Series. IOP Publishing, 2020. p. 012058.

[62] THALER, Richard H. (ed.). Advances in behavioral finance, Volume II. Princeton University Press, 2005.

[63] THÉATE, Thibaut; ERNST, Damien. An application of deep reinforcement learning to algorithmic trading. Expert Systems with Applications, 2021, 173: 114632.

[64] THORP, Edward O. Portfolio choice and the Kelly criterion. In: Stochastic optimization models in finance. Academic Press, 1975. p. 599-619.

[65] TRIPPI, Robert R.; DESIENO, Duane. Trading equity index futures with a neural network. Journal of Portfolio management, 1992, 19: 27-27.

[66] VAN DE WIELE, Tom, et al. Q-learning in enormous action spaces via amortized approximate maximization. arXiv preprint arXiv:2001.08116, 2020.

[67] WATKINS, Christopher John Cornish Hellaby. Learning from delayed rewards. 1989.

[68] WIENER, Norbert. Differential-space. Journal of Mathematics and Physics, 1923, 2.1-4: 131-174.

[69] WU, Xing, et al. Adaptive stock trading strategies with deep reinforcement learning methods. Information Sciences, 2020, 538: 142-158.

[70] XIONG, Zhuoran, et al. Practical deep reinforcement learning approach for stock trading. arXiv preprint arXiv:1811.07522, 2018, 1-7.

[71] YANG, Bing, Ting LIANG, Jian XIONG and Chong ZHONG. Deep reinforcement learning based on transformer and U-Net framework for stock trading. Knowledge-Based Systems. AMSTERDAM: Elsevier, 2023, yr. 262, 10 s. ISSN 0950-7051.

[72] YAO, Jingtao; LI, Yili; TAN, Chew Lim. Option price forecasting using neural networks. Omega, 2000, 28.4: 455-466.

[73] YULE, George Udny. VII. On a method of investigating periodicities disturbed series, with special reference to Wolfer's sunspot numbers. Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character, 1927, 226.636-646: 267-298.

List of Tables

List of Figures