BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION COMPUTER SCIENCE IN ENGLISH

DIPLOMA THESIS

Advancing Market Integrity: A Proximal Policy Optimization Approach to Detect Spoofing and Layering in Algorithmic Trading

Supervisor
Lecturer Professor, PhD Diana-Lucia Miholca

Author
Iulia-Diana Groza

2024

ABSTRACT

Contents

Chapter 1

Introduction

Chapter 2

Algorithmic Trading and Market Integrity

The objective of this chapter is to present the terminology required for a better understanding of the main topic of the thesis, specifically providing a brief theoretical insight on algorithmic trading. Furthermore, we explore how *spoofing* and *layering* are performed, and how such manipulative tactics can deeply impact financial markets.

2.1 Introduction to Algorithmic Trading

Undoubtedly, the landscape of modern financial markets has undergone major transformations in recent decades due to the widespread adoption of algorithmic strategies. In general, *algorithmic trading* (or widely referred to as *automated trading* or *black-box trading*) represents the execution of trades at precise moments by leveraging the use of computer programming. Put another way, algorithmic trading improves the *liquidity* of markets by ruling out the involvement of human emotions and execution delays specific to *traditional market-making*. In their seminal work, Hendershott & Riordan (2013) provide a comprehensive examination of algorithmic trading and its implications for market liquidity, by studying the impact of algorithmic trading strategies on market dynamics [HR13].

In the context of market manipulation, we aim to focus specifically on *high-frequency trading* (HFT), which is characterised by speedy execution, when it comes to buying or selling securities. HFT is applied not only in stock markets, but in exchanging stock options and futures as well [Dur10].

Thanks to recent advancements in hardware development and the impact of *Field Programmable Gate Arrays* (FPGAs) on ultra-low latency, even a couple of *nanoseconds* could make the difference between *profit* and *loss*.

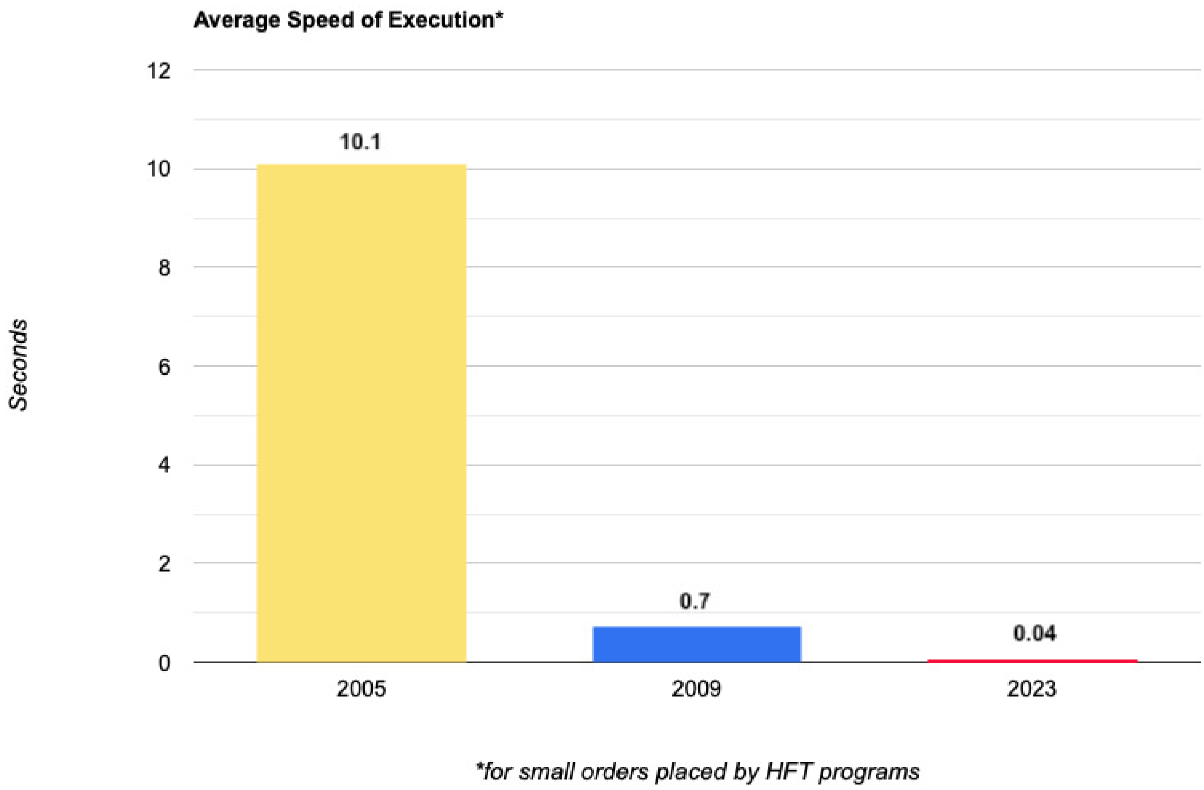

Figure 2.1: The average speed of execution for small orders in HFT has come to the realm of nanoseconds thanks to the development of ultra-low latency systems.

Given how ultra-low latency adversely affects market dynamics, we now introduce the fundamental concept in HFT: the *order*. Put succinctly, a trade cannot occur without a previously placed order to initiate it.

In his article ”Limit Order Books”, Martin D. offers an ample and insightful definition for orders, as well as for fundamental terms that are frequently used in the financial literature [MDGH13]:

Definition 2.1.1 (Order) *An order $x = (p_x, \omega_x, t_x)$, submitted at time t_x, with price p_x, and size $w_x > 0$ (respectively, $w_x < 0$), is a commitment to sell (respectively, buy) up to $|\omega_x|$ units of the traded asset at a price no less than (respectively, no greater than) p_x.*

Definition 2.1.2 (Bid Price) *The bid price at time t is the highest stated price among active buy orders at time t,*

$$b(t) := \max_{x \in B(t)} p_x. \tag{2.1}$$

Definition 2.1.3 (Ask Price) *The ask price at time t is the lowest stated price among active sell orders at time t,*

$$a(t) := \min_{x \in A(t)} p_x. \tag{2.2}$$

Definition 2.1.4 (Mid Price) *The mid price at time t is*

$$m(t) := [a(t) + b(t)]/2. \quad (2.3)$$

Definition 2.1.5 (Bid-Ask Spread) *The bid-ask spread at time t is*

$$s(t) := a(t) - b(t). \quad (2.4)$$

It is important for an investor to know how to leverage the placement of two major order types in stock trading: *market orders* and *limit orders*.

A *market order* involves *buying* or *selling* a security immediately, the price of the transaction being strongly linked to the time of its execution (different from submission). This implies that the price at submission time might (at most times) deviate from the price at execution time; price remains unchanged only when the *bid/ask* price is exactly at the last traded price. Therefore, with immediate execution, market orders are more aggressive - volatility increases drastically, especially for investments with fewer shares on the market or smaller trade volumes.

Pushing the market in the opposite direction, *limit orders* (sometimes known as *pending orders*) are characterised by total price control, coming with a specific set of instructions on the execution of the trade. Investors specify the maximum *bid* price or the minimum *ask* price for their stocks. The brokerage will execute the order only if the price of the financial instrument aligns with the specified bounds; otherwise, the order will be left *unexecuted*. Interestingly, the conditions imposed in limit orders can even lead to partial orders (only a part of the shares will be traded), as the price of the investment can modify *mid-order*. In the next sub-chapter, we will explore how the less-aggressive nature of limit orders is exploited to produce "artificial" shifts and influence the stock markets.

Since market manipulation most commonly occurs via limit orders, as previously indicated, our subsequent discussion will elaborate on basic order dynamics: how limit orders are stored, processed and amended. Most exchanges make use of *Central Limit Order Books* (CLOBs), or simply known as *Limit Order Books* (LOBs), to execute limit orders. LOBs are transparent, real-time, anonymous, and low-cost-in-execution systems that utilise *order books* and *matching engines* to map bid/ask orders of investors based on *price-time* priority. Considering a financial instrument which investors place orders for, the best market consists of mapping the highest bid offer to the lowest ask offer.

Martin D. offers the following definition for an LOB [MDGH13]:

Definition 2.1.6 (Limit Order Book) *An LOB $L(t)$ is the set of all active orders in a market at time t.*

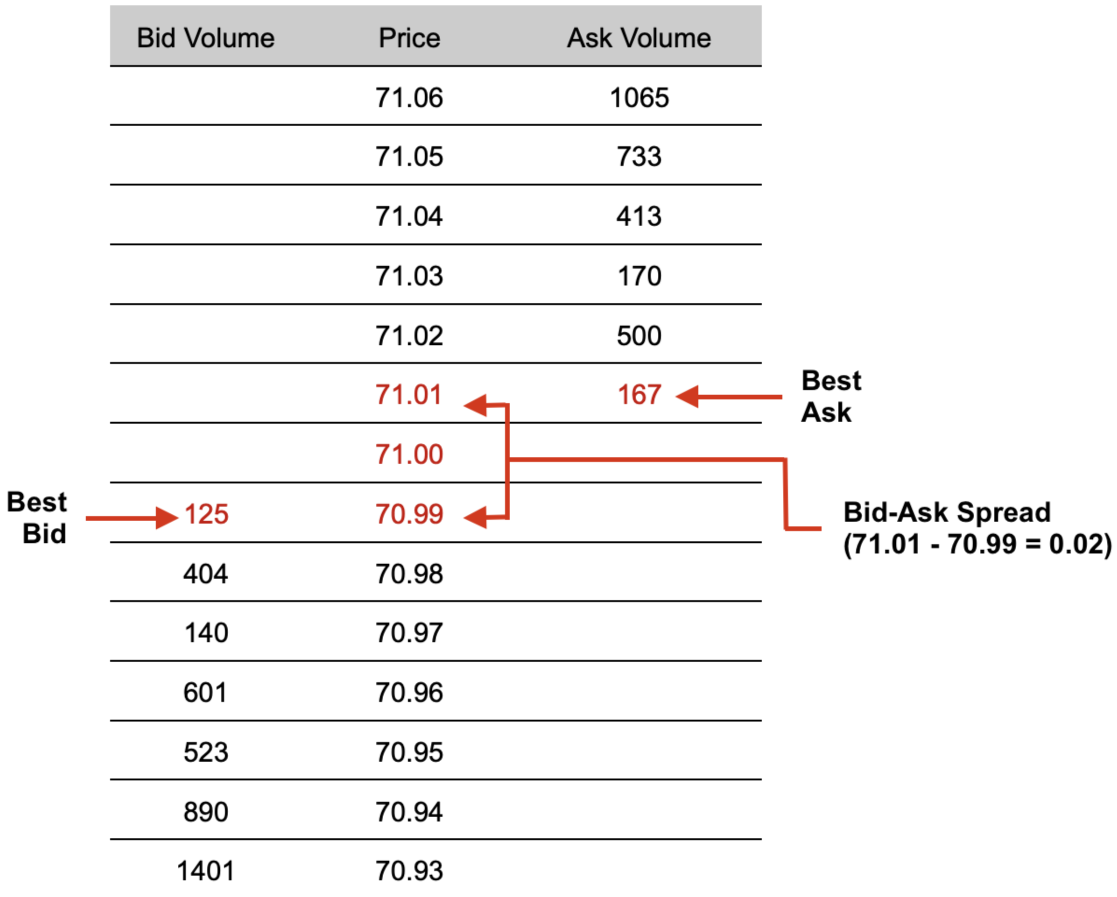

Figure 2.2: Graphical example of a 15-level deep LOB on a particular financial instrument. It depicts how traders judge liquidity and how they arrive at the bid-ask spread.

We continue with defining three crucial concepts for understanding CLOBs: *liquidity*, *price discovery*, and *depth of market*.

Liquidity refers to the degree to which a security can be easily traded in the market at a price reflecting its intrinsic value. Put another way, liquidity is the efficiency of converting an asset into ready cash without affecting its market price - the most liquid security is cash itself.

The most standard way of measuring liquidity is using the *current ratio* formula:

Definition 2.1.7 (Current Ratio) *The current ratio is a liquidity ratio measuring the capability of an investor of having enough resources to meet their short-term obligations:*

$$current\_ratio = \frac{current\_assets}{current\_liabilities}. \tag{2.5}$$

Price discovery denotes the (explicit or deduced) process in which buyers and sellers establish the fair price of a financial instrument in the market, at a given time.

It implies conducting market research analysis, by evaluating supply and demand, environmental, geopolitical and socioeconomic factors.

Depth of Market (DOM) refers to the volume of orders pending to be transacted for a particular security at different price levels - it is the overall level (or breadth) of open orders.

A large order can significantly impact DOM and liquidity. If a large buy order enters the market, it can exhaust the available sell orders at lower prices, increasing the price as it fulfills higher-priced sell orders. Conversely, a large sell order can fulfill all the buy orders at higher prices, driving the price down as it starts matching with lower-priced buy orders. This impact on price through the supply and demand balance is a direct outcome of the market's liquidity and depth.

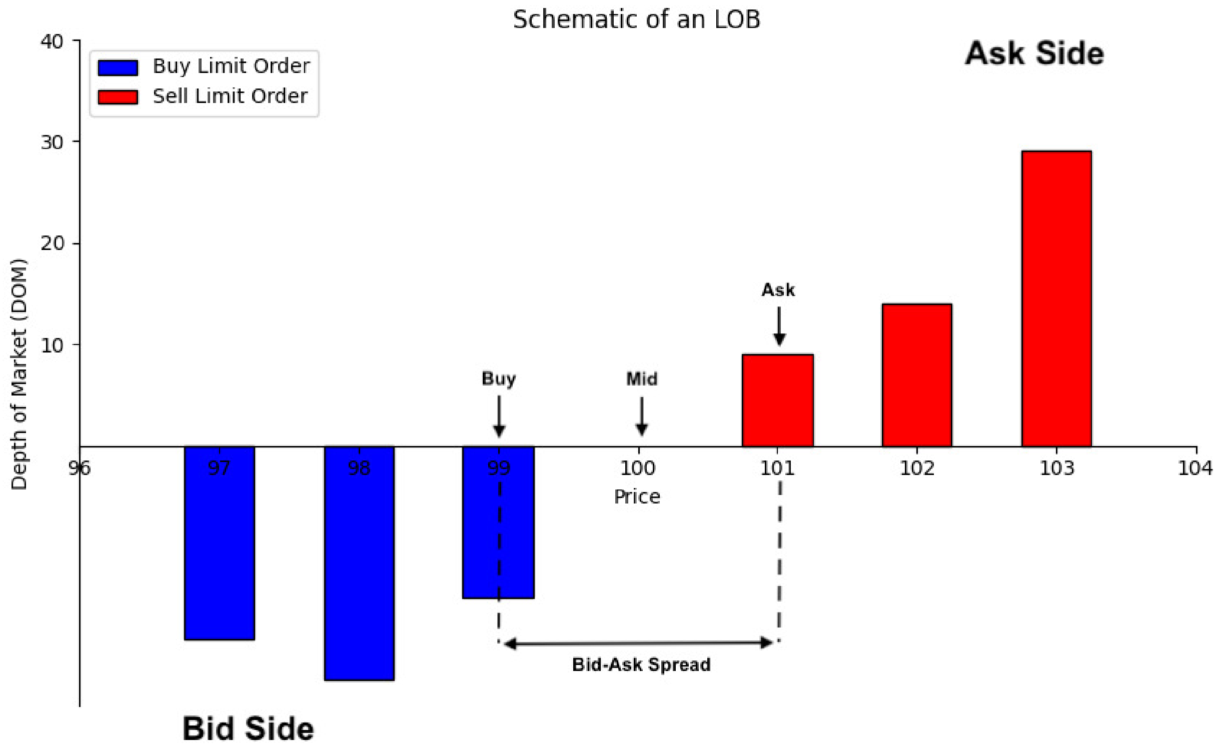

Figure 2.3: Schematic of an LOB

While we already covered the fundamentals of order placement, it is essential to acknowledge the mechanisms for order modification and cancellation in an LOB. In terms of *modification*, traders are generally allowed to update the limit price or the quantity of an existing order via the trading platform of the exchange. Modifications are generally permitted until the execution has begun, depending on the trading platform or exchange. For instance, if a buy limit order is partially filled, the quantity or price may be adjusted for the remaining unfilled portion. As a consequence, after an order has been modified, the investor should be aware about the impact of such operation on the LOB - the priority of the order in the LOB might be affected. For example, if the investor increases (respectively, decreases) the buy (respectively, ask) price of the security, the order might shift its position in the order book, as it becomes more competitive. Changing the quantity of an order could also impact both the priority in the LOB and the likelihood of execution.

Order *cancellation* in trading can occur for various strategic reasons. Cancellation for manipulative intents will be covered in a later chapter; however, traders might *legitimately* cancel limit orders because of shifting market conditions (e.g. news events affecting stock prices, market moving against their strategy). The impact of cancellations on LOBs includes the change in available liquidity, bid-ask spread and perceived market depth.

LOBs can be classified into different *levels*, based on the amount of information they provide:

1. **Level 1** - It encapsulates basic market data, such as:
   - *Bid price*;
   - *Bid size*;
   - *Ask price*;
   - *Last price*;
   - *Last size*.
2. **Level 2** - Additionally to Level 1 data, it doesn't provide just the highest bid and lowest offer, but also bids and offers at other prices:
   - *Highest bid prices*;
   - *Bid sizes*;
   - *Lowest ask prices*;
   - *Ask sizes*.
3. **Level 3** - It provides even deeper information than Level 2 data. Level 3 data refers to non-aggregated bids and asks placed by individual market makers. A Level 3 data feed would include every individual bid and ask, including *time series*.

LOB snapshots can be procured from various stock & crypto exchanges or trading platforms, via *WebSocket feeds* (for real-time data), or *REST APIs* (for historical data). The implementation proposed in this thesis will make use of *Level 3 LOB data*, to take full advantage of temporal information.

2.2 Challenges in Market Integrity: Spoofing and Layering

With the rise of algorithmic trading and the automation of financial markets, there is no doubt that the market becomes more and more exposed to major risks of fraud and exploitation. Affecting market integrity takes multiple forms, including market manipulation, insider trading, and short selling, all of them causing ample market destabilization due to their complex and ever-evolving nature. This thesis targets the challenges brought by market manipulation, particularly by two of its widely spread forms: spoofing and layering.

Market manipulation is defined as a set of ”actions intended to cause an artificial movement in the market price, so as to make a profit or avoid a loss” [AAD+16]. Therefore, it is easy to justify why such cases of misconduct erode the confidence of investors and challenge the perception of market stability.

The Dodd-Frank Wall Street Reform and Consumer Protection Act of 2010 (“Dodd-Frank Act”) defines *spoofing* in the realm of algorithmic trading as ”bidding or offering with the intent to cancel the bid or offer before execution” [Dod10]. A trader places multiple large limit orders on a security with no intention of executing them. This is performed with the aim of artificially decreasing (respectively, increasing) the price of an asset, with the later intent of placing the actual buy (respectively, sell) order after cancellation, giving the false impression to other traders that the security is ”on-demand”.

Layering is a particular case of spoofing, which consists of placing multiple *non-bona fide* orders at *multiple price tiers*, in contrast to spoofing, where orders are entered only at the top of the order book.

Spoofing and layering often cause extreme volatility and rapid price movements. In such cases, they largely contribute to flash crashes. A *flash crash* is an event in electronic markets, where prices of a financial instrument *drop* dramatically, but then they immediately *rebound*. Flash crashes are *sudden* and short-lived: at the end of the day, it appears as if the crash had never happened.

While some methods are relatively straightforward (e.g. spreading *fake news*), spoofing and layering, take more subtle forms, and the methods used by malicious investors are constantly evolving. Additionally, in some cases, it is difficult to decide whether an investor has the intention of causing harm (e.g. a company buying its own shares to rise their price). This leads to the complexity of developing an efficient system that detects spoofing and layering in all of its shapes, and not misclassify peculiar cases of legal practices.

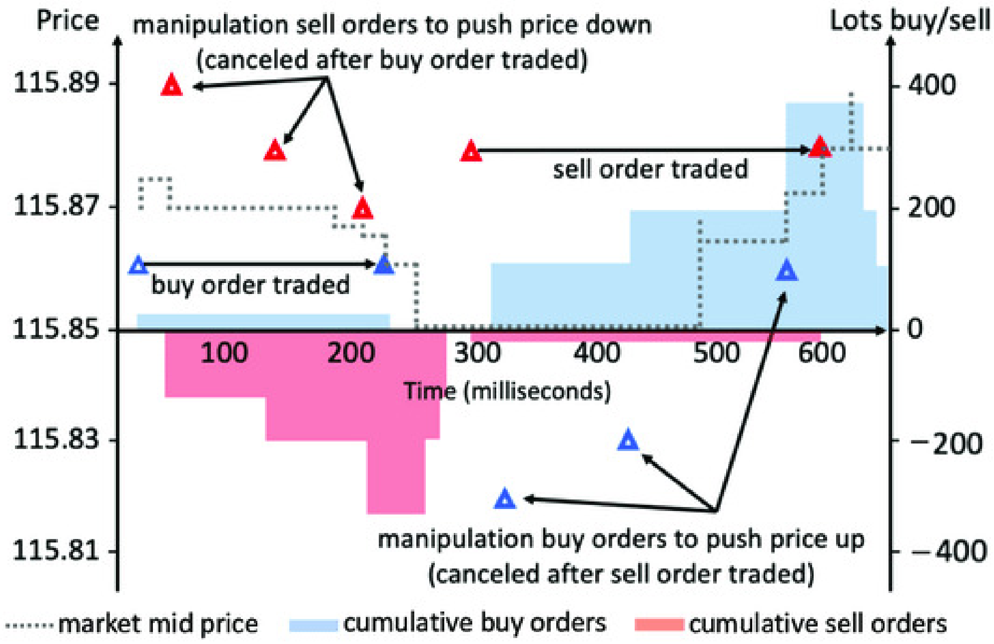

Figure 2.4: Illustrating an example of spoofing. Source: UK Financial Conduct Authority Final Notice 2013 [WHVW21]. "0.6 s, demonstrating how quickly and effectively such manipulation behavior can affect the market and profit from spoofed belief."

2.3 Case Studies of Notable Market Events

In his seminal work, *Spoofing, Market Manipulation, and the Limit-Order Book*, J. Montgomery highlights two notorious cases involving spoofing and layering allegations [Mon16], with those that pleaded guilty for the criminal actions being sentenced to up to *five years of imprisonment* and paying forfeits of as much as *US$300.000*.

One pivotal event revolves around Michael Coscia, who in November 2015 became the first person convicted of spoofing, under the *Dodd-Frank Act*. M. Coscia placed large "bait" orders for futures contracts on various commodities, through exchanges from the United States and the United Kingdom. This lead to a net profit for Mr. Coscia of *US$1.4 million* over a span of ten weeks as part of six attempts of spoofing.

In the most notable case of layering, Aleksandr Milrud lead a complex scheme for trading US securities. Mr. Milrud recruited and shared profits with online traders from China and Korea, who placed and then quickly cancelled HFT buy and sell orders at multiple price tiers, yielding net profits of as much as *US$600,000* in a day. This accumulated in net monthly profits between *US$1 million* and *US$50 millions*.

As part of our study, we aim to analyse LOB data from flash crashes, due to the high frequency of spoofing and layering attempts during such events. Specifically, we conduct our research on the flash crash of *LUNA*, from May 2022. Over a span of just three days, *Terra*, the third-largest cryptocurrency ecosystem following Bitcoin and Ethereum, experienced a dramatic collapse, erasing *US$50 billion* in market valuation [LMS23].

2.4 Ethical and Legal Considerations

As highlighted in the legal cases presented in the previous chapter 2.3, the realm of financial trading is not only governed by economic principles, but also operates within a framework of ethical and legal standards. It is essential to implement and continually revise regulations to diminish the systematic losses produced by information asymmetry sourced from market manipulation.

In *"Principles of Financial Regulation"*, John Armour covers multiple aspects regarding the regulation of market manipulation, both in the European Union and the United States [AAD+16]. The integrity of financial markets hinges on the equitable treatment of all market participants. Unethical practices, such as market manipulation through spoofing and layering, undermine this fairness and can lead to significant distortions in market dynamics. Moreover, the lack of transparency in certain trading activities poses a threat to the trust essential in financial markets. This is particularly critical for retail investors, who might lack the resources to identify and navigate deceptive market practices effectively.

As mentioned in subchapter 2.2, legal frameworks play a vital role in regulating market activities and deterring unethical behavior. The legislation in both EU and US is designed to prevent market manipulation and promote a transparent trading environment. As noted in a report by Steel Eye [Steon], besides the *Dodd-Frank Act* [Dod10], market manipulation in US rules under multiple laws, including:

- *Commodity Exchange Act (CEA) (Section 4c(a)(5)(C))* [CEA13];
- *Securities Exchange Act of 1934 10(b)* [SEA34];
- *FINRA Rule 2020* [FIN08].

Regarding the UK and EU legislation, the *Market Abuse Regulation* (MAR) [MAR14] encompasses regulations related to insider trading, the illegal disclosure of privileged information, and the manipulation of markets.

Chapter 3

Reinforcement Learning and Optimization Methods

This chapter aims to cover fundamental *Reinforcement Learning* (RL) concepts that can be applied in high-frequency trading, specifically in market manipulation detection. We will dive deeper into how *Proximal Policy Optimization* (PPO) is applied to potentially solve market integrity issues.

3.1 Reinforcement Learning Mechanisms: Markov Decision Processes, Policy Gradients and Actor-Critic Methods

It is no secret that in recent years, the interest in *Machine Learning* (ML) renewed thanks to exponential advancements and its extension to a broad range of fields. In less than a decade, ML has become an integrating part of almost every aspect of our lives: education, business & marketing automation, our social and personal life, and ML will only increase its potency in the future, especially with the recent rise of *Generative AI* (GenAI) and the race for developing the first instance of *Artificial General Intelligence* (AGI). As of the time of writing, creating AGI is the primary objective of pioneering AI research companies such as *OpenAI* [Ope24], *Google DeepMind*, and *Anthropic*.

The three main ML *paradigms* are: *Supervised Learning* (SL), *Unsupervised Learning* (UL), and *Reinforcement Learning* (RL). This chapter solely focuses on fundamental concepts of RL, as the only methods that we will use in the development of our detector are based on reinforcement. We will now proceed with covering the most essential terms and concepts in RL.

The term *"reinforcement"* originates from research conducted on animal behaviour, in the context of adaptive learning and experimental psychology - in the occurrence of an event and the proper relation to a response, the goal is to increase the probability of that specific response occurring again in the same situation [Kim61]. The same principles are transposed in *Reinforcement Learning* (RL), an area of Machine Learning, described as "the science of decision making" through *optimal control*: an agent must take a suitable action to maximize cumulative rewards for achieving the desired result. In contrast to supervised learning, where the key result is given by labeling training data, the reinforcement agent is bound to adapt and learn from its experience, in the absence of a training dataset.

When it comes to how the behaviour of the agent is constructed through rewards, there are two types of reinforcement: *positive* and *negative*. In *positive reinforcement*, the occurrence of an event, being performed due to a specific behaviour, increases the frequency of that particular behaviour. Reinforcement with a positive effect allows maximised performance, and sustains long-term changes. An overuse of positive reinforcement produces an overload of states, thus diminishing the results.

In contrast, *negative reinforcement* highlights the idea of refining the behaviour of an agent through stopping or avoiding negative conditions. Reinforcement with negative effect builds up the bare minimum structure for an agent to work, providing defiance to a minimum standard of performance.

The environment in which an agent is placed is usually modeled as a *Markov decision process* (MDP), due to the use of *dynamic programming* (DP) techniques [vOW12]. The main difference between DP and RL is that the latter does not assume information about the mathematical model used in the MDP, but they rather target large MDPs where exact models become infeasible [EL22].

Definition 3.1.1 (Markov Decision Process) *An MDP is a discrete-time stochastic control process for decision making in systems that are partly random, and partly under the control of the decision maker.*

We can now translate the definition of an MDP in the context of an RL model. It is generally represented as a *4-tuple* (S, A, P_a, R_a), where:

- S is a set of *environment* and *agent states*;
- A is a set of actions of the agent, called the *action space* (alternatively, A_s is the set of actions available from state s);
- $P_a(s, s') = Pr(s_{t+1} = s' \mid s_t = s, a_t = a)$ is the probability that action a in state s at time t will lead to state s' at time $t + 1$;
- $R_a(s, s')$ is the immediate reward (or expected immediate reward) received after transitioning from state s to state s', due to action a.

The state and action spaces may be finite or infinite. Processes with countably infinite state and action spaces can be simplified into ones with finite state and action spaces [Wro84].

We are now able to introduce the crux in RL, when it comes to the strategy utilised by the agent in pursuit of its goals: the *policy*.

Definition 3.1.2 (Policy) *A policy function π is a (potentially probabilistic) mapping from the perceived state space (S) of the environment to its action space (A):*

$$\pi : A \times S \to [0, 1]$$
$$\pi(a, s) = \Pr(A_t = a \mid S_t = s)$$

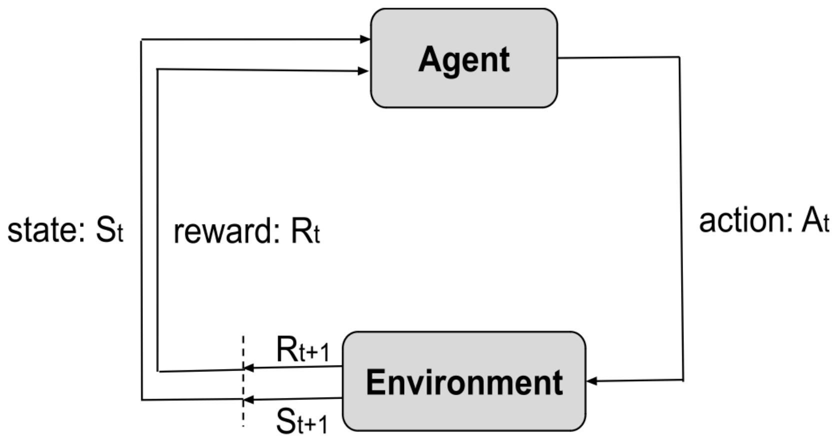

Figure 3.1: The Canonical Agent-Environment Feedback Loop. Basic algorithm for an RL approach: at each time t, the agent receives the current state S_t and reward R_t. It then picks and sends to the environment an action A_t from the set of available actions. The environment moves to a new state S_{t+1} and the associated reward R_{t+1} is computed. The aim of the agent is to learn a policy that maximizes the expected cumulative reward.

Consequently, the network that transforms input frames into output actions is called *"policy network"*. This allows us to finally consolidate the theoretical basis that underlies the methodology of our thesis: one of the simplest ways to train a policy network is given by control algorithms entitled *"policy gradients"*. All MDPs have at least one *optimal* policy.

Definition 3.1.3 (Parameterised Policy Objective) *The objective is to maximise the expected reward, following the imposed parameterised policy:*

$$J(\theta) = \mathbb{E}_\pi[r(\tau)] \tag{3.1}$$

A solution method, targeting the maximization problem, that is widely recognized in the ML literature is *Gradient Ascent* (respectively, *Descent).* In gradient ascent, an update rule is used for iterating through the parameters:

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t) \tag{3.2}$$

The challenge brought by policy gradients is how we determine the gradient of the objective defined at equation 3.1. It is known that integrals are slightly inefficient in computational setting, so a workaround needs to be defined. This consideration lies at the base of the *Policy Gradient Theorem*.

Definition 3.1.4 (Policy Gradient Theorem) *The derivative of the expected reward is given by the expectation of the product between the reward and the gradient of the log of the policy π_θ:*

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[r(\tau)] = \mathbb{E}_{\pi_\theta}[r(\tau)\nabla_\theta \log \pi_\theta(\tau)] \tag{3.3}$$

One of the biggest challenges in the realm of reinforcement learning has been given by the *Exploration vs. Exploitation Dilemma*.

Exploration refers to the agent's action of trying new strategies that lead to better long-term rewards. It's akin to venturing into the unknown; the agent experiments with different actions and observes their outcomes. Exploration is crucial in the early stages of learning or in dynamically changing environments where previous knowledge may become outdated.

Exploitation, on the other hand, involves leveraging the knowledge the agent has already acquired to make decisions that yield the highest immediate reward. When exploiting, the agent selects the best-known action based on existing information, favoring short-term gains.

The trade-off comes into play, since both actions cannot be performed simultaneously and need to be balanced. If an agent explores too much, it may miss out on known rewards. Conversely, if it exploits too often, it may overlook better options that it hasn't discovered yet.

Balanced strategies have been developed in an attempt to solve the dilemma, one of the most known being the *ϵ-greedy method*, where the agent explores randomly with probability ϵ and exploits with probability $1 - \epsilon$. As learning progresses, the value of ϵ can be reduced, shifting the balance from exploration to exploitation as the agent gains more knowledge.

With this in mind, we are now able to contour a couple of general RL tenets that are pivotal to Proximal Policy Optimization (PPO), which will be explained thoroughly in the next subchapter.

Temporal Difference (TD) Learning is an essential concept in reinforcement learning that combines ideas from *Monte Carlo* methods and *Dynamic Programming* [vOW12].

It allows an agent to learn directly from raw experience without a model of the environment's dynamics. As an agent interacts with the environment, TD Learning updates the value of states in a way that the expected future rewards are estimated more accurately over time.

Moreover, *Q-learning*, a widely known TD Learning algorithm, is particularly notable for its ability to compare the expected utility of the available actions without requiring a model of the environment [WD92]. This value-based method has been instrumental in developing Proximal Policy Optimization (PPO), which further refines the policy optimization process. At the heart of TD learning and Q-learning stand *value functions*.

Considering the Canonical Agent-Environment Feedback Loop scheme from figure 3.1, we can define *value functions* as a measure of the expected long-term reward attainable in certain conditions defined by the states and action space. There are two types of value functions in RL: *state-value* and *action-value*. By the end of this subchapter, we will explore the relation between the two and how they lead to informed and optimal decision-making, required by PPO.

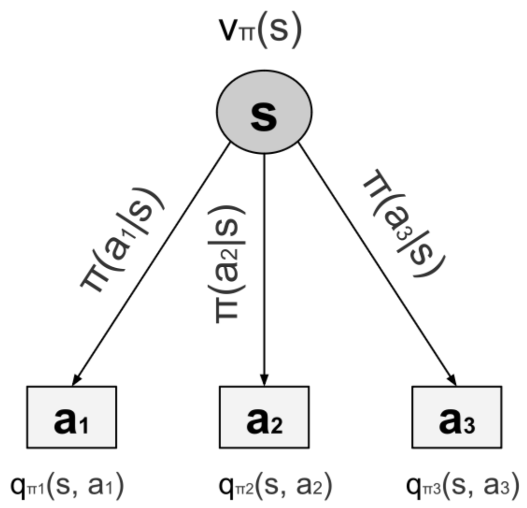

Figure 3.2: Value functions in the context of the action space $A_s = \{a_1, a_2, a_3\}$, generated by state s. Each action a_i is taken with probability $\pi(a_i|s)$.

Definition 3.1.5 (State-Value Function) *The state-value function denotes the expected cumulative reward, starting from state s, following policy π:*

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t r_t \mid s_t = s \right] \tag{3.4}$$

where γ is the discount factor that determines how long the return depends on future rewards.

Alternatively, the total cumulative reward at timestep t can be written using the goal G as shown below [Gor17]:

$$V^{\pi}(s) = E_{\pi}\{G_t \mid s_t = s\} \tag{3.5}$$

Definition 3.1.6 (Action-Value Function) *The action-value function denotes the expected cumulative reward, starting from state s, following policy π, taking action a:*

$$Q_{\pi}(s,a) = E_{\pi}\left[\sum_{t=0}^{T-1} \gamma^t r_t \mid s_t = s, a_t = a\right] \tag{3.6}$$

In terms of goal G, the action-value function becomes [Gor17]:

$$Q^{\pi}(s,a) = E_{\pi}\{G_t \mid s_t = s, a_t = a\} \tag{3.7}$$

The relationships between $V_{\pi}(s)$ and $Q_{\pi}(s,a)$ (in terms of each other), in a stochastic policy π, are given by the following Bellman equations [Gor17]:

$$V^{\pi}(s) = \sum_{a \in A} \pi(a \mid s) * Q^{\pi}(s,a)$$
$$Q^{\pi}(s,a) = \sum_{s' \in S} P(s' \mid s,a)\left[R(s,a,s') + \gamma V^{\pi}(s')\right]$$

Definition 3.1.7 (Advantage Function) *The advantage function, $A^{\pi}(s,a)$, quantifies how much better it is to take a specific action a in state s over randomly selecting an action according to the policy's probability distribution. Mathematically, it's defined as $A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$.*

The important role the advantage function plays in PPO is that it provides a relative measure of the value of actions, guiding the policy update process.

Another staple in modern RL frameworks is given by *Actor-Critic Methods*. These are, in fact, *policy gradients* applied in *TD learning*. Konda & Tsitsiklis describe the *"actor"* component of the model as responsible for selecting actions based on a policy that is directly parameterized, while the *"critic"* assesses the actions taken by the actor by computing a value function [KT00].

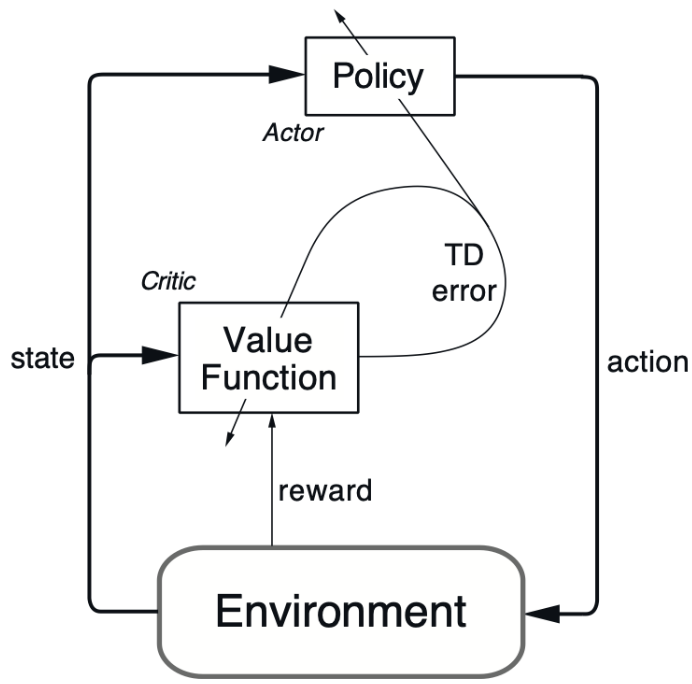

Figure 3.3: The actor-critic architecture. Source: *Reinforcement Learning: An Introduction (Sutton & Barto, 2018)* [SB18].

Having established the foundational concepts of Reinforcement Learning, such as value functions and Temporal Difference Learning, we are now well-positioned to transition to a more focused examination of *Proximal Policy Optimization* (PPO) in the upcoming subchapter.

3.2 Proximal Policy Optimization

Proven successful on a wide variety of tasks from robotic control to surpassing grandmasters at multiple strategy games, *Proximal Policy Optimization* (PPO) is a *deep reinforcement learning* algorithm designed by OpenAI in 2017. Since then, it became the default RL algorithm used by the AI research company, due to its simplicity and outstanding performance [Ope17].

The road to success in RL is often marked by major challenges. One potential issue arising is that training data is itself dependent on the current policy, since agents generate it by interacting with the environment, rather than relying on a static dataset, as it is the case for supervised learning. This implies that data distributions of observations and rewards are constantly updating as the agent learns, leading to major instability in the whole training process. Another problem frequently encountered is that RL approaches suffer from a very high sensitivity to *hyperparameter tuning*.

To address these issues, the OpenAI team developed the Proximal Policy Optimization algorithm. The core purpose behind PPO was to strike a balance between ease of implementation, sample efficiency and ease of tuning.

PPO is a *policy gradient method*. Therefore, unlike Deep-Q Network, it does *not* rely on *experience replay* (where transition experiences are stored in a *replay buffer* [RMT17]); instead the agent learns *online*. We now introduce the *Policy Gradient Loss*, (strongly related to theorem 3.3), as defined by Schulman et al. in their seminal work [SWD+17].

Definition 3.2.1 (Policy Gradient Loss) *The Policy Gradient Loss is an expectation that maximizes the log-probability of beneficial actions weighted by their advantage estimates, thereby reinforcing effective behaviors:*

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t \mid s_t) \hat{A}_t \right], \tag{3.8}$$

where $\hat{A}_t$ is the noisy estimate for the advantage function at timestep t, and π_θ a stochastic policy.

One potential issue arises when gradient descent is repeatedly applied to the same batch of collected experience: hyperparameters may become suboptimally tuned, extending well beyond the range appropriate for data collection, resulting in inaccurate estimates of $\hat{A}_t$. This issue can be circumvented by ensuring policy updates do not significantly deviate from the previous policy.

This idea was widely introduced by Schulman et al. two years earlier [SLA+15], in the development of *Trust Region Policy Optimization* (TRPO). TRPO serves as the foundational algorithm upon which PPO is constructed.

As a result, in TRPO, a *KL constraint* is added to the *"surrogate" objective*, which will block any major deviation in policy updates [SWD+17]:

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{old}}(a_t \mid s_t)} \hat{A}_t \right] \tag{3.9}$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t \left[\mathrm{KL}[\pi_{\theta_{old}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)] \right] \leq \delta. \tag{3.10}$$

We will refer to the surrogate objective as $L^{\mathrm{CPI}}(\theta)$, *CPI* standing for the *conservative policy iteration*. Additionally, we denote the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}$. For instance, an action more likely to occur in the current policy than in the old one will have $r_t(\theta) > 1$. L^{CPI} is defined as:

$$L^{\mathrm{CPI}}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)} \hat{A}_t \right] = \hat{\mathbb{E}}_t \left[r_t(\theta) \hat{A}_t \right]. \tag{3.11}$$

The KL constraint guarantees monotonic improvement and an efficient optimization of control policies in TRPO [SLA+15]. However, the constraint may cause additional overhead in the optimization process, potentially leading to undesirable training behavior.

Fixing this issue is exactly the essential target PPO managed to achieve. The main goal of the first-order algorithm is to maintain the monotonic improvement of TRPO, while replacing the hard constraint over the surrogate objective (formula 3.10) with a penalty [SWD+17]. This leads to the main objective of PPO, the *Clipped Surrogate Objective*:

$$L^{\mathrm{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right], \tag{3.12}$$

where ϵ is a hyperparameter, the first parameter of the *minimum* function is $L^{\mathrm{CPI}}(\theta)$, while the second term aims to adjust the surrogate objective by clipping the probability ratio.

This minimalist, yet efficient approach ensures that r_t remains bounded in the interval $[1-\epsilon, 1+\epsilon]$. The value of the advantage estimate may be positive or negative, thus $\hat{A}_t$ affecting the effect of the main operator:

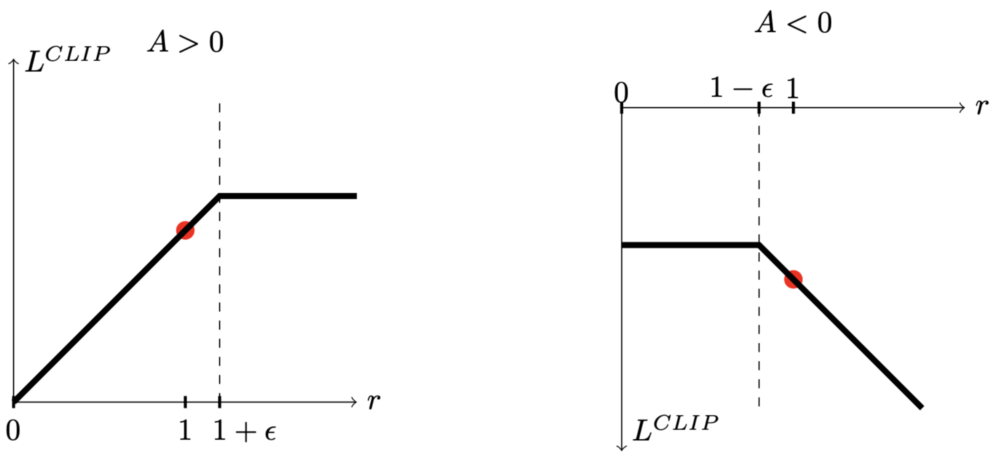

Figure 3.4: One timestep of L^{CLIP} with an advantage estimate of positive (left) and negative (right) value. The first plot showcases a "good" action, more probable after the gradient step. The latter displays the behavior of the agent in the case of undoing the last update policy due to a "bad" action. Source: *Proximal Policy Optimization Algorithms, Schulman et al., 2017* [SWD+17].

We lastly introduce the final training objective in PPO:

$$L_t^{\mathrm{CLIP+VF+S}}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{\mathrm{CLIP}}(\theta) - c_1 L_t^{\mathrm{VF}}(\theta) + c_2 S[\pi_\theta](s_t) \right], \tag{3.13}$$

where c_1 and c_2 are coefficients, S denotes an entropy bonus, and L_t^{VF} is a squared-error loss.

The final objective formula lies at the foundation of the algorithm implementation proposed by Schuman et al. The PPO algorithm makes use of fixed-length trajectory segments, and optimizes the surrogate loss using *minibatch SGD* (or for a better performance, *Adam*) [SWD+17]:

Algorithm 1 PPO, Actor-Critic Style

for iteration = 1, 2, ... **do**
    for actor = 1, 2, ..., N **do**
        Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
        Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
    end for
    Optimize surrogate L with respect to θ, with K epochs and minibatch size $M \leq NT$
    $\theta_{\text{old}} \leftarrow \theta$
end for

Source: *Proximal Policy Optimization Algorithms, Schulman et al., 2017* [SWD+17]

3.3 Employing Proximal Policy Optimization Techniques in Uncovering Market Manipulation

In 2022, Chip Huyen presented in one of her most acclaimed pieces of work, *Designing Machine Learning Systems*, a 2020 survey analysing the large landscape of use cases of enterprise ML (in both internal and external scopes) [Huy22]. We notice that *27%* of enterprise ML applications focus on *"Detecting fraud"*. This demonstrates the feasibility of utilizing ML techniques in our work in detecting forms of financial market manipulation, such as *spoofing & layering*.

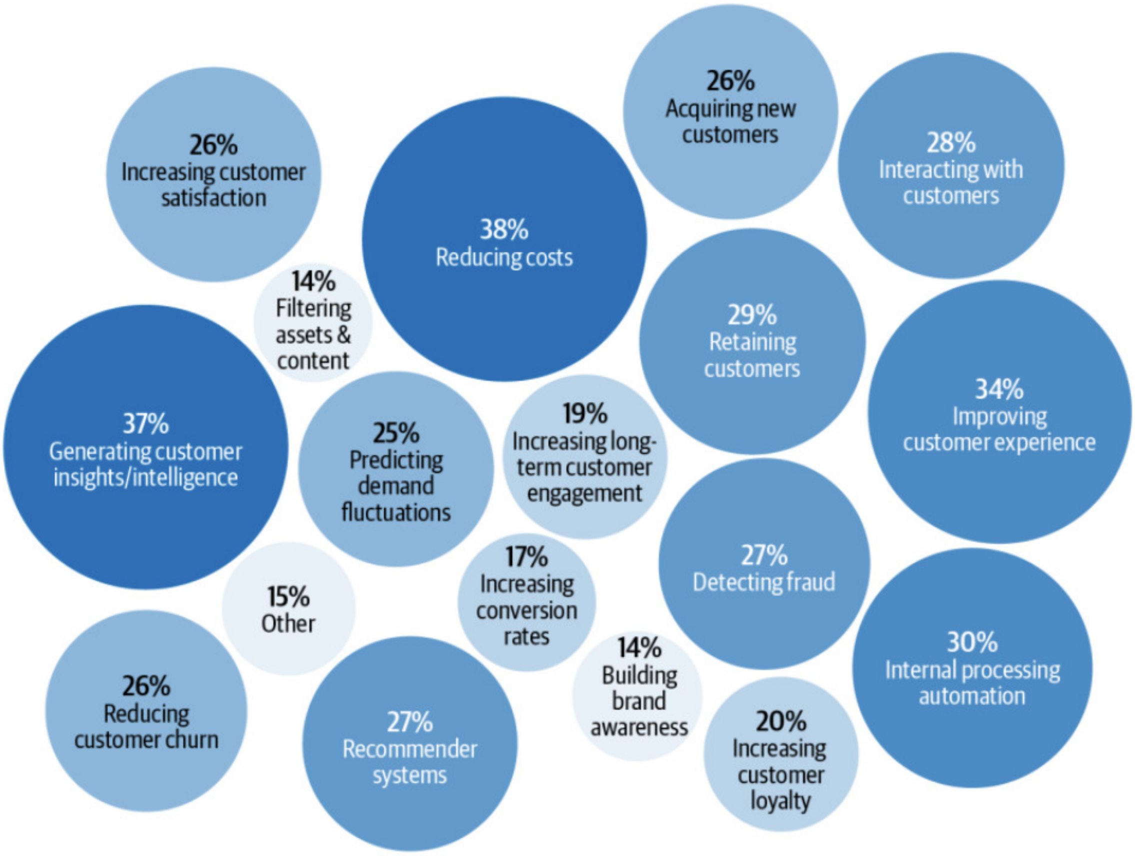

Figure 3.5: 2020 state of enterprise ML. Source: *Designing Machine Learning Systems, C. Huyen* [Huy22].

RL is applied in multiple domains, such as game theory, simulation-based optimization, swarm intelligence, statistics, and multi-agent systems. One obvious characteristic that gives reinforcement learning a major advantage over supervised learning is that, by definition, supervised learning cannot produce an output that is better than human output, since data labelled by humans is involved in the process. This makes RL one of the best candidates in the realm of sports, games, autonomous vehicles, and, most relevant to our thesis, finance. Spoofing and layering are incredibly sophisticated market manipulation tactics, and investors engaging in such practices constantly improve their tactics to avoid detection, making the instances even harder to be captured by the human eye.

PPO, with its strategic advantage in learning complex policy representations and against the variance in market data, stands out as a promising candidate for improving market surveillance. The algorithm's adaptability and efficiency in policy updates empower it to discern subtle patterns of fraudulent behavior, offering a substantial leap forward in the ongoing effort to uphold market integrity. In the upcoming chapters, we will explore how PPO principles can be applied to maximize efficiency in detecting financial market abuse through spoofing and layering, during the development of *spoof.io*, the application designed as part of this thesis.

Chapter 4

Literature Survey on Spoofing Detection

This chapter aims to take a deep dive into related work on detecting spoofing and layering attempts, or other forms of market manipulation. Before presenting our approach and results, we will analyse and compare solutions based on *Gated Recurrent Units* (GRU), *Feedforward Neural Networks*, *latent multivariate autoregressive* processes, *Adaptive Hidden Markov Models* with anomaly states, and even approaches that model order limit order books using *Statistical Physics* tools.

4.1 A Supervised Learning Approach

In the realm of preventing and detecting market manipulation, supervised learning has been a favoured approach, largely due to its capability to discern patterns from labeled instances of illicit behaviour. Therefore, trading activities can be simply classified as legitimate or manipulative. We will now present in detail two significant contributions in the surveillance of financial markets, that address slightly different problems in market integrity, and utilise labeled data in two different manners. What is more important, is that these approaches will allow us to uncover the trade-offs associated with selecting LOB data of different granularities: Level 1 and Level 2, respectively.

In 2016, Leangarun et al. presented in their seminal work, ”Stock price manipulation detection using a computational neural network model”, *supervised deep learning* detection methods that focus both on spoofing and *pump-and-dump* attempts [LTT16]. In pump-and-dump, investors create a buying frenzy by misinformation to ”pump” the price of the stock, and then eventually ”dump” their shares, by selling them at the artificially inflated price.

Their initial approach involved labeled Level 1 LOB data (which is clearly more accessible to investors) procured from NASDAQ companies, such as Intel, Microsoft, and Amazon. While this decision was proved to be quite successful regarding pump-and-dumping (model achieved 88.25% accuracy [LTT16]), the model failed to detect spoofing effectively. This lead to constructing a model that involved 1-minute LOB snapshots from Level 2 data. One major difference between L1 and L2 data is that Level 2 LOBs contain information about *order cancellations*. We previously discussed in chapter 2 how the volume and time of cancelling previously placed limit orders is a major indicator of spoofing instances. Thus, the results in the case of indicating spoofing positions from L2 data had a much higher accuracy than those from the L1-data model.

The model was based on a *feedforward neural network* (FNN) consisting of 25 nodes in the input layer, 3 nodes on the hidden layer, and one node in the output layer.

Regarding the input, the L2 LOB snapshots were processed to be separated into the OHLCV (open, high, low, close and volume) data - L1 information - of five time steps, and the manipulated class information - additional L2 data. The input layer nodes of the feedforward neural network stored the OHLCV information, while the manipulated class played a key role in analysing the circumstances of cancellation or deletion orders. This information was eventually fed into the price manipulation model. The forecast output was a binary variable that denoted whether the interval represented a spoofing instance or not.

There were three conditions imposed by the FNN in the process of searching for spoofing instances:

1. The price of the cancellation order is close enough to the bid or to the ask price.
2. The value of the cancellation volume is high enough.
3. The value of the last buy order volume is high enough.

For all three conditions, *threshold values* have been defined for calculating the absolute difference between the two terms of the inequalities.

When it comes to evaluating and experimenting with the model developed on L1 data, the testing phase implied *leave-one-out cross validation* (LOOCV). The reason for selecting this evaluation method is that spoofing attempts do not occur as often in real-life situations as in a simulated environment. For reasonable classification results, training data should generally have a 1:1 ratio between the number of points classified as spoofing attempts, and non-spoofing points. Therefore, as opposed to test data, training sets are usually built on data obtained from exceptional events, such as flash crashes, which do not occur on a daily basis.

The detection model fed with L1 data obtained the following results:

Table 4.1: Results of the FNN model with OHLCV data as input. Source: [LTT16]

Average mean square error	1.3992
False-positive error	1.3479
False-negative error	1.4505

It was observed that price volatility is not a key factor in detecting spoofing attempts (due to the inability of comparing *Average True Range* (ATR) values) [LTT16].

In *"Protecting Retail Investors from Order Book Spoofing using a GRU-based Detection Model"* (2021), Tuccella, Nadler and Serban developed an *early detection* model for spoofing, based on *Gated Recurrent Units* (GRU), using Level 2 data from several cryptocurrency exchanges [TNS21]. The outstanding contribution of their work is that detection is performed 2 seconds prior to the end of the manipulation attempt, thus rather acting as a prevention tool. This is a reasonable time span for investors to take action prematurely and avoid trades on a manipulated market.

The Level 2 LOB data was extracted from exchanges like Bitfinex and Kraken, on multiple pairs of cryptocurrencies, over the time span of several months (from November 2019 to May 2020), totaling to over 335 GB of data. For computational efficiency, the gathered raw data was compressed, and the LOB depth was restricted to 25 levels.

Regarding the criteria for classifying points as part of spoofing attempts, Tuccella et al. make use of the same three conditions that were imposed by Leangarun et al. in their work (previously described in this chapter 4.1). Process of price manipulation is finished once all three criteria are met. Let T_0 be that time of flagging a point as a spoofing instance. The time series $T = (T_0 - 12, T_0 - 11, ..., T_0 - 2)$ is used in training the classifier early detection model.

The advantage of opting for a GRU-based approach is that GRU is similar in terms of performance to a *Long Short-Term Memory* (LSTM) strategy, but faster in training. The time dependencies handled by the GRU are then fed into a *feedforward neural network*, using an *Adam* optimiser and hyperparameters tuned with a random search approach. The output is a scalar denoting the probability of the time series representing a price manipulation instance.

Since the focus of the tool is on *early detection*, in the evaluation process, both the accuracy of the model and a fast "response time" during the price manipulation attempt are used as performance indicators. The average accuracy of the model is *75%*, proving that reasonable results can be achieved even in real-time detection.

4.2 Leveraging Unsupervised Anomaly Detection Methods

In the context of supervised learning, labeling market manipulation instances is not very feasible and simple, as it requires manual validation. Moreover, price manipulation practices and strategies are constantly evolving and getting more sophisticated. This leads to unsupervised methods based on anomaly detection becoming a more reasonable option in the context of real-time fraud detection. This chapter explores the solution described in "Adaptive Hidden Markov Model With Anomaly States for Price Manipulation Detection" (2015), and developed by Cao et al. [CLC+15].

Their approach centers around an *Adaptive Hidden Markov Model with Anomaly States* (AHMMAS), which outperformed the previously developed solutions in terms of recognition of various intraday price manipulation methods. As input, Level 2 stock data was procured from NASDAQ and the London Stock Exchange. The model was tested with both simulated, and real market data. The most valuable aspect of analysing the contents of this paper is that this will allow us to have an ensemble perspective over other standard algorithms used in the literature, such as: *Gaussian Mixture Models* (GMM), *k-Nearest Neighbors* (kNN), and *One-Class Support Vector Machines* (OCSVM).

In the context of applying anomaly detection in financial market problems, the goal is to detect *unusual patterns* in the bid-ask price time series. This is where the temporal information contained by L2 data becomes helpful. It was observed that price manipulation is mostly associated with short, high-frequency oscillations around standard equity price levels. Keeping track of such anomalous movements along with the original bid-ask values is the key to building an effective detector [CLC+15].

The feature extraction model works under three patterns of *waveforms*, described in figure 4.1: *Sawtooth*, *Square*, and *Pulse*. As part of the signal decomposition process in the extraction of features, *wavelet transform* was applied. The feature extractor contains a *wavelet filter* and a *gradient calculator*. The four features extracted from the set of bid-ask prices are:

- The original price: P_t;
- The short-term oscillation: $\hat{P}_t$;
- The first-order derivatives of the original price: dP_t/dt;
- The first-order derivatives of the short-term oscillation: $d\hat{P}_t/dt$.

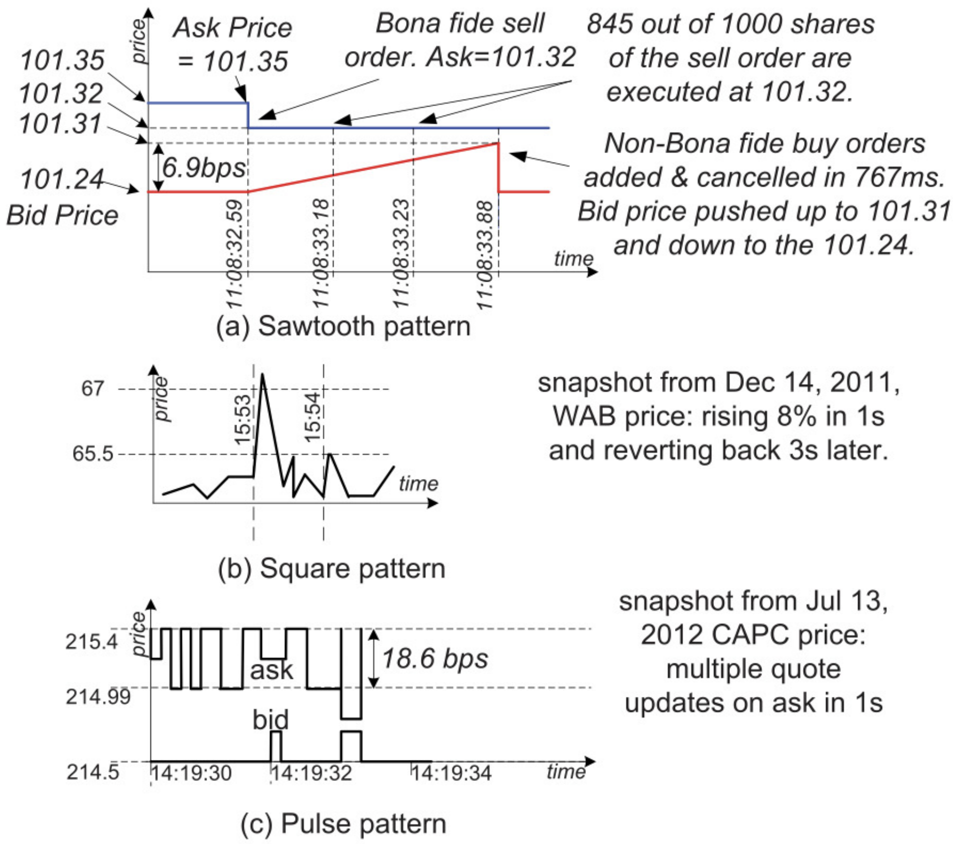

Figure 4.1: Feature extraction module: (a) *Sawtooth* pattern. (b) *Square* pattern. (c) *Pulse* pattern. Source: [CLC+15].

Three issues are highlighted about applying a traditional HMM in detecting manipulation in financial markets. The first one revolves around the complexity of using HMM with multiple features, since it is usually built around one-dimensional sequential data. Additionally, HMMs are usually unable to detect the anomaly type as well, thus making it impossible to differ from the three manipulation patterns. Lastly, the *probability density functions* (pdfs) of equity prices are not constant due to the dynamic nature of the time series.

Consequently, the authors introduced an improved HMM with anomaly states. The pdf of each of the four features is learned independently by the *Gaussian Mixture Model* (GMM). Building the hidden anomaly states starts by setting *anomaly thresholds* for each pdf, similarly to solutions based on *one-class support vector machines* (OCSVM). The data for which the pdf is below 99%, is considered abnormal, this condition allowing to capture the most 0.5%, and the least 0.5% frequent values for each feature.

The thresholds allow partitioning the data into multiple parts that serve as the foundation of the hidden states used in the HMM. The hidden states of all four features are fused into the 32 hidden states of the new adaptive HMMAS. Finally, the 32 states are further simplified as three final manipulation states. Depending on the features that prove to be anomalous, each of the hidden states falls into one of these three categories.

The adaptive nature of the HMMAS is given by the training being performed in a previous time range, constructed as a *sliding window* with length equivalent to data from one day (detection is done on intraday market data). Therefore, the model becomes an AHMMAS. The window is constantly slid forward to adapt to the non-stationary time series and to mantain the closest points as reference for training. The deviation between the current data sequence and the previous training dataset is identified by *t-test* [CLC+15]. Computational efficiency and performance are balanced using a t-test module with significance level of 1%.

For evaluating the results of the AHMMAS, synthetically generated data was injected into real market data, due to the lack of frequent market manipulation instances in real data. Normal statistical features (such as mean, variance, and volatility) were maintained. The performance of the model is measured using the *receiver operating characteristic* (ROC). The following table contains values for the *Area Under the ROC Curve* (AUC) for the AHMMAS run on 7 different stocks, compared with OCSVM, kNN, and GMM. The AHMMAS clearly outperformed all benchmark models.

Table 4.2: AUC of Four Detection Models on Seven Real Stock Data Sets. Source: [CLC+15]

AUC	AHMMAS	OCSVM	kNN	GMM
AAPL	0.8142	0.6603	0.7926	0.6695
ARM	0.8270	0.5830	0.7982	0.7918
BARC	0.8710	0.6125	0.7627	0.6466
GOOG	0.8025	0.6593	0.5612	0.6163
INTC	0.8971	0.6970	0.6280	0.5200
MSFT	0.7336	0.6419	0.6250	0.6802
VOD	0.8775	0.7044	0.7278	0.7495

4.3 Modeling the Order Book Dynamics Using Statistical Physics

We will now explore the distinctive, yet powerful applications of statistical physics in uncovering price manipulation, through modeling the dynamics of order books. The methodology adapts principles from *econophysics* (including *Brownian motion*, *fluctuation-dissipation* relations, and *statistical aggregation*) to analyse the microstructural behavior of financial markets, providing a novel perspective on surveillance and regulatory measures. The current subchapter reviews several studies that have contributed to this field, presenting their approaches and findings.

In "The key role of liquidity fluctuations in determining large price changes" (2005), Fabrizio Lillo and J. Doyne Farmer examine the significant role of *liquidity fluctuations* in explaining large price changes in financial markets, focusing on the London Stock Exchange (LSE) [LDF05].

Liquidity gaps, quantified as blocks of adjacent price levels lacking quotes, significantly contribute to price volatility. These gaps, particularly the first three observed in the dataset of 16 high-volume stocks, display a probability distribution characterized by *"fat tails"*, suggesting that extreme values are more common than a Gaussian distribution would predict. It was observed that these gaps follow *power-law distributions* with indices that measure the tail's heaviness. Such distributions reflect the substantial variability in liquidity, which can result in large price swings.

Lillo & Farmer further prove that liquidity gaps are persistent over time through *long-memory* processes. This persistence is quantified using *autocorrelation functions* and *Detrended Fluctuation Analysis* (DFA), revealing that gaps exhibit a memory effect where past states influence future states beyond a short-term scope. The study confirms that the size of the gaps is not only a momentary snapshot of market conditions, but part of a consistent pattern that affects future liquidity and price changes.

Moreover, the gaps are interconnected across different levels of the order book and between the buy and sell sides. The synchronous behavior of gaps indicates that liquidity issues on one side of the market can simultaneously affect the other.

In "Financial Brownian Particle in the Layered Order-Book Fluid and Fluctuation-Dissipation Relations" (2014), Yura et al. introduce an innovative model to describe financial market dynamics, drawing a parallel with the physical theory of *Brownian motion*. This approach conceptualizes the financial markets dynamics as a *colloidal particle* (representing the price) moving through a fluid composed of smaller particles (buy and sell orders). Comprehensive market data was utilised in validating the theoretical model, focusing on correlations within the order book of the foreign exchange market [YTST14].

Empirically, Yura et al. found that the inner layer of the order book, which is closer to the current market price, exhibits a stronger and short-term memory correlation with the price changes. In contrast, the outer layer shows a weaker but longer-lasting memory effect, as noted in [LDF05] as well. This differentiation between the inner and outer layers once again proves the influence of liquidity dynamics on price changes across different timescales.

The study also explores the *fluctuation-dissipation relation* (FDR) in the context of this financial Brownian motion. Under certain conditions, the classic FDR, which links the fluctuations in a system to its response to perturbations, can be applied to financial markets.

One of the key results involves the statistical analysis of the *"drag"* within the market, caused by the interaction between different order layers. This drag impacts the market's responsiveness to new information, thereby affecting price volatility. The analysis confirms that the market dynamics exhibit characteristics similar to those of particles in a fluid, where the "temperature" of the fluid could be analogous to market volatility.

Lastly, we examine the contribution of Li, Polukarov and Ventre in their seminal work, "Detecting Financial Market Manipulation with Statistical Physics Tools" (2023). Their solution for identifying spoofing & layering attempts during the LUNA flash crash from May 2022, focuses on modeling the dynamics within the LOB by treating market orders as particles. This analogy allows the computation of a momentum measure, $m = s \cdot v$, where s is the size of the order and v is its velocity, which is defined as the rate of price change per unit of time. The key contribution of the paper is that they leverage modeled *Level 3* LOB data in performing the detection. Their approach proves to be effective, outperforming the standard anomaly detection *Z-score* model [LPV23].

The core of their model is the aggregation of this momentum over a defined period to capture the systemic behavior of the market. They define the velocity of the midprice $v_M(t) = \frac{z_M(t) - z_M(t-\Delta t)}{\Delta t}$, where $z_M(t)$ is the market midprice at time t, and Δt is the sampling interval.

This model is then employed to analyse the order flow and detect anomalies in real trading data from the LUNA/USD market, especially during its flash crash. It focuses on the "active area" of the LOB, where most trading activities occur, and extends to a "passive area" to detect less obvious manipulative activities.

In terms of evaluation, the method is compared against a conventional Z-score-based anomaly detection method, across data of the LUNA and Bitcoin cryptocurrencies. The model successfully identifies manipulative trading behaviors by tracking the cumulative sum of net momentum, calculated as:

$$M = \sum_{t \in (T-\Delta t, T]} \sum_{\gamma = b_M - \alpha}^{a_M + \alpha} N_\gamma(t) \sum m \tag{4.1}$$

where $N_\gamma(t)$ is the number of limit orders at depth γ at time t, and α defines the depth of the active area around the bid-ask spread [LPV23].

During the LUNA flash crash, the model detected significant spoofing activities by observing the deviations in the momentum measures, which were not captured by the Z-score model. Limit orders played a crucial role in driving the prices down, where the attempt of market orders to drive prices up was unsuccessful, indicating potential manipulative activities. The comparison in accuracy between their model and the Z-score method shows a clearer detection of anomalous events with the former, particularly in identifying the exact times and types of orders involved in spoofing.

4.4 Comparative Analysis of Detection Methods

The final section of this chapter aims to perform a thorough comparative analysis of the price manipulation detection methods discussed in our literature survey. We will explore multiple trade-offs, including computational efficiency, performance, flexibility, and granularity of market data. Both the advantages and disadvantages of each method will be highlighted. This will serve as a solid reasoning foundation during the development of our novel approach, which utilises Proximal Policy Optimization.

Before diving into the analysis, we will establish the criteria for evaluating each discussed method, which are of high interest for the development of our tool:

- **Granularity of input data**: The complexity of the order book information that was used during the development and the testing phases (Level 1, Level 2, respectively Level 3).
- **Accuracy**: The ability of the method to correctly identify instances of market manipulation without generating false positives. Detection of more sophisticated methods of market manipulation should obtain a reasonable accuracy, regardless of the granularity of the input data.
- **Computational Efficiency**: The amount of computational resources required and the speed of the detection process, which are crucial for real-time analysis.
- **Flexibility and Reliability**: The method's ability to perform under different market conditions and across various types of assets.
- **Ease of Implementation**: The complexity involved in implementing and maintaining the method in operational environments.

Supervised learning models, particularly those employing deep learning architectures like GRUs and FNNs, have shown high accuracy in detecting known patterns of market manipulation. The GRU-based model developed by Tuccella et al. [TNS21] achieved an accuracy of 75%, which is commendable for real-time detection. Their work proved the advantage of GRU over LSTM, the former being faster during the training phase. However, these models require extensive labeled datasets for training, which can be a significant limitation in markets where manipulation tactics evolve rapidly.

On the other hand, the FNN model discussed by Leangarun et al. [LTT16] demonstrated the impact of data granularity by showing improved results with Level 2 data over Level 1 data, highlighting the importance of detailed order book information (especially on cancellation orders and their depth) in enhancing accuracy of manipulation detection. Lastly, both papers define solid criteria for labeling spoofing & layering instances (presented during the first subchapter 4.1), that will serve as a good framework for defining the surrogate objective, and the cumulative reward of our agent.

The Adaptive Hidden Markov Model with Anomaly States (AHMMAS) introduced by Cao et al. [CLC^{+}15] represents a significant advancement in unsupervised detection methods. It outperformed traditional models by effectively recognizing anomalous trading patterns without prior labeling of L2 snapshots. Although this model offers powerful detection capabilities and adapts to new data through its sliding window mechanism, its complexity and computational demands could pose challenges for real-time market surveillance.

The works of Lillo & Farmer [LDF05], and Yura et al. [YTST14] provide foundational insights into how liquidity fluctuations and price movements can be analyzed using statistical physics. Their methods are particularly adept at capturing complex interactions within the order book that traditional models might overlook. This demonstrates the complex interplay between market orders, limit orders, and cancellations, a remark that is essential in our study conducted on spoofing & layering.

Li, Polukarov, and Ventre's approach [LPV23] successfully applied these principles to detect spoofing & layering during the LUNA cryptocurrency flash crash. Their model, which considers orders as particles whose interactions can be quantified and analyzed, showcased superior performance in identifying manipulative behaviors that were not detected by more traditional Z-score-based anomaly detection methods. Their work proved that the use of more granular data, of Level 3, can be both successful and computationally efficient at the same time. Thanks to their exceptional results and insightful statistics, our solution will make use of the exact same L3 real market data on the LUNA/USD pair, gathered during the LUNA flash crash from May 2022 (11/05/2022, 16:00-20:00).

When comparing all methods, it's evident that while supervised and unsupervised learning strategies provide accurate tools for detecting known and evolving manipulative patterns, statistical physics approaches bring a novel and profound analytical depth. An ideal solution would combine the simplicity and computational power of supervised learning in real-time detection, the absence of a requirement for labeled data in unsupervised learning strategies, and the accuracy given by state-of-the-art econophysics methods. We consider that a PPO-based approach will effectively harness most of these advantages.

Chapter 5

spoof.io: A RL-driven Web-App Tool for Spoofing Detection in LUNA

5.1 Experimental Results

5.2 Application Development

5.2.1 Analysis and Architecture

5.2.2 Design and Functionalities

5.2.3 Implementation

5.2.4 User Guide

5.3 Future Considerations

Chapter 6

Conclusions

Bibliography

[AAD+16] John Armour, Dan Awrey, Paul Davies, Luca Enriques, Jeffrey N. Gordon, Colin Mayer, and Jennifer Payne. 181 Trading and Market Integrity. In *Principles of Financial Regulation*. Oxford University Press, 07 2016.

[CEA13] Commodity exchange act (cea). Section 4c(a)(5)(C), 2013. Accessed: 2024-03-26.

[CLC+15] Yi Cao, Yuhua Li, Sonya Coleman, Ammar Belatreche, and Thomas Martin McGinnity. Adaptive hidden markov model with anomaly states for price manipulation detection. *IEEE Transactions on Neural Networks and Learning Systems*, 26(2):318–330, 2015.

[Dod10] Dodd-frank wall street reform and consumer protection act. Pub. L. No. 111-203, 124 Stat. 1376, 2010. Available at U.S. Government Publishing Office.

[Dur10] Michael Durbin. *All About High-Frequency Trading*. McGraw-Hill, US, 2010.

[EL22] Shengbo Eben Li. *Reinforcement Learning for Sequential Decision and Optimal Control*. Springer, 2022.

[FIN08] Finra rule 2020, 2008. Accessed: 2024-03-26.

[Gor17] Matthew Gormley. Reinforcement learning introduction. Lecture Notes for 10-601: Machine Learning, Carnegie Mellon University, 2017. Accessed: 2024-04-01.

[HR13] Terrence Hendershott and Ryan Riordan. Algorithmic trading and the market for liquidity. *Journal of Financial and Quantitative Analysis*, 48(4):1001–1024, 2013.

[Huy22] Chip Huyen. *Designing machine learning systems*. " O'Reilly Media, Inc.", 2022.

[Kim61] Gregory A Kimble. Hilgard and marquis'" conditioning and learning.". 1961.

[KT00] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*, volume 12, 2000.

[LDF05] FABRIZIO LILLO and J. DOYNE FARMER. The key role of liquidity fluctuations in determining large price changes. *Fluctuation and Noise Letters*, 05(02):L209–L216, 2005.

[LMS23] Jiageng Liu, Igor Makarov, and Antoinette Schoar. Anatomy of a run: The terra luna crash. Working Paper 31160, National Bureau of Economic Research, April 2023.

[LPV23] Haochen Li, Maria Polukarova, and Carmine Ventre. Detecting financial market manipulation with statistical physics tools, 2023.

[LTT16] Teema Leangarun, Poj Tangamchit, and Suttipong Thajchayapong. Stock price manipulation detection using a computational neural network model. In *2016 Eighth International Conference on Advanced Computational Intelligence (ICACI)*, pages 337–341, 2016.

[MAR14] Market abuse regulation (mar). European Union Regulation, 2014. Accessed: 2024-03-26.

[MDGH13] Stacy Williams Mark McDonald Daniel J. Fenn Martin D. Gould, Mason A. Porter and Sam D. Howison. Limit order books. *Quantitative Finance*, 13(11):1709–1742, 2013.

[Mon16] John D. Montgomery. Spoofing, market manipulation, and the limit-order book. May 2016. Available at SSRN: `https://ssrn.com/abstract=2780579` or `http://dx.doi.org/10.2139/ssrn.2780579`.

[Ope17] Openai baselines: Ppo, 2017. Accessed: 2024-04-01.

[Ope24] OpenAI. Openai charter, 2024. Accessed: 2024-03-26.

[RMT17] Melrose Roderick, James MacGlashan, and Stefanie Tellex. Implementing the deep q-network, 2017.

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[SEA34] Securities exchange act of 1934. Section 10(b), 1934. Accessed: 2024-03-26.

[SLA+15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR.

[Steon] Steel Eye. Spoofing: A growing market manipulation risk and focus for regulators, Year of Publication. Accessed: 2024-03-10.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[TNS21] Jean-Noel Tuccella, Philip Nadler, and Ovidiu Serban. Protecting retail investors from order book spoofing using a gru-based detection model, 2021.

[vOW12] Martijn van Otterlo and Marco Wiering. *Reinforcement Learning and Markov Decision Processes*, pages 3–42. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[WD92] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[WHVW21] Xintong Wang, Christopher Hoang, Yevgeniy Vorobeychik, and Michael Wellman. Spoofing the limit order book: A strategic agent-based analysis. *Games*, 12:46, 05 2021.

[Wro84] Andrew Wrobel. On markovian decision models with a finite skeleton. *Zeitschrift für Operations Research*, 28:17–27, 1984.

[YTST14] Yoshihiro Yura, Hideki Takayasu, Didier Sornette, and Misako Takayasu. Financial brownian particle in the layered order-book fluid and fluctuation-dissipation relations. *Phys. Rev. Lett.*, 112:098703, Mar 2014.