Reinforcement Learning in the Market with Adverse Selection

by

Zihao Xu

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2020

Author . . . .
Department of Electrical Engineering and Computer Science
May 15, 2020

Certified by . . . .
Leonid Kogan
Nippon Telegraph and Telephone Professor of Management and
Professor of Finance
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science Chair,
Department Committee on Graduate Students

Reinforcement Learning in the Market with Adverse Selection

by

Zihao Xu

Submitted to the Department of Electrical Engineering and Computer Science
on May 15, 2020, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

The study of the market with adverse selection risks is an appealing topic in the field of market microstructure. Multiple theoretical models have been proposed to address this issue over the past few years, such as the Kyle model(1985), the Glosten-Milgrom model(1985) and so on. The main goal of these theoretical models is to provide an optimal pricing strategy based on the market condition they used. However, the market is a competitive but not always efficient environment. The optimal pricing strategy cannot provide enough insights in the markets with multiple interacting agents. Also, the theoretical models cannot be easily extended to other complex market. In our work, we apply the deep reinforcement learning techniques to train neural agents in our designed multiagent environment. The result shows that the neural agents could learn the best strategy conditioned on the pricing behaviors of other competitors. It suggests a new approach to study the price formation process in the complex market.

Thesis Supervisor: Leonid Kogan
Title: Nippon Telegraph and Telephone Professor of Management and Professor of Finance

Acknowledgments

I would like to thank my parents for their support.

I would like to thank professor Leonid Kogan for his kindness. The thesis is only possible with his guidence in providing ideas and getting results.

I would like to thank professor Tomaso Poggio, professor Gabriel Kreiman and other friends at CBMM for their patience.

I would like to thank my collegues and classmates at ORC and CSAIL.

I would like to thank the friends that I met at MIT Warehouse and 504 Beacon street.

The thesis ends my life as a student here at MIT. The only regret I have is that I could not have the chance to physically attend the ceremony with all my other graduating friends due to COVID-19, but nothing more. It's been a great time. I will cherish the days here in Cambridge. Hope everyone stays safe and stays healthy.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Recent advances in deep reinforcement learning have led to huge successes in several real-world applications, including traditional board games [15], multiplayer competitive games and self-driving cars. The key idea of the reinforcement learning is based on the interactive virtual environment, in which the deep reinforcement learning agent tried to learn the strategy or policy that maximizes the rewards it could obtain from the environment. In our work, we focus on the application of deep reinforcement learning in the financial market, specifically on the market microstructure modeling.

This chapter outlines the general overview of the market making processes and the price formation procedures and how deep reinforcement learning techniques could help study the price equilibria in the market. In section 1.1, we make a review on market microstructure and different types of market making agents. In section 1.2, based on the basic market microstructure knowledge, we proposed our market setting and applied deep reinforcement learning agents to interactively learn the market structure and compete with other players in the market.

1.1 Market Microstructure

The core idea in the market microstructure theory [12] [2] states that the price of certain assets does not need to be exactly the same as the expectation of value due to various types of frictions and risks. The phrase "Market Microstructure" is first termed in 1976 by Mark Garman to describe a series of behaviors in market making. Since then, it's been a term referred to the study of micro trading behaviors inside the market. Specifically, these trading behaviors involve several components including market rules, price dynamics, asymmetric information flows, market makers and investors. In our study, we only concentrate on the price dynamics with the presence of asymmetric information flows and the competitions among different market makers.

Market makers are a group of liquidity providers by quoting bid(sell) and ask(buy) prices that they want to trade with other investors. The issue of market making has been extensively studied in the previous literature, mainly focusing on the issue of inventory risks [8] [7] and adverse selection risks [5] [10]. The inventory risks refer to the situation in which market makers make more quotes than the market can digest. Therefore, this leaves extra quotes in the market makers' inventories, which result in the potential risks of losing the value. The other commonly studied risk is the adverse selection risk. It is termed to describe the case where there are a group of informed traders inside the market, so that there is some risk that the market makers are at disadvantageous positions. In other words, adverse selection risk reflects the existence of asymmetric information flows in the market.

In the thesis, we are looking at the adverse selection risk problem. There are extensive theoretical studies on the adverse selection risk problems. A typical model is called bid-ask model or Glosten-Milgrom model. This model is trying to address the issue of adverse selection risks by providing the solutions on how to quote optimal bid or ask prices. In the meanwhile, the model also gives a way to study the trading behaviors and price dynamics when the adverse selection risk is present in the market.

1.2 Learning and Market Making

Due to the microscopic nature, the micro-market possesses tons of time-series data to describe the market behaviors. This provides opportunities for machine learning models. Machine learning techniques could efficiently process millions of data points and returns suggestive results on the current market trends or even trading signals. However, researchers in the field are mostly interested in data obtained from the real market condition and less interested in the simulated data from the simulated market. Previous works on the intersection of market making and reinforcement learning mostly work on the market with inventory risks [4]. Not much work has focused on the asymmetric information and the adverse selection problems.

In our work, we primarily focus on the simulated market with adverse selection risks. The main reason to study this type of questions is that the market environment is totally under control. In the real market, adverse selection risk is only one of the frictions. Directly using the real-market data in the study of the adverse selection problems requires deep understanding on nearly all kinds of frictions that are present in the data so that these noise signals can be cleaned out. Obviously, this is impossible. There are a lot of unknown factors underneath the market data, such as inventory risks, financial bubbling, financial crisis and so on. Therefore, virtual market allows us to understand the market changes and the machine learning model's behaviors. Here, we deliberately design the market rules so that only adverse selection risk issue exists in the market, and apply deep reinforcement learning techniques to study the market dynamics. The main idea is to explore the formation of the market equilibrium while multiple market maker agents compete with each other. In the meanwhile, we further study the behaviors of those trained individual agents from the microscopic perspective, trying to understand what policy it learns from the competitions with and without the Glosten-Milgrom agent in the market.

Chapter 2

Market Making Under Adverse Selection Risks

Let us first review some of the notions used in market making.

2.1 Efficient Market

Efficient market hypothesis [1][3] is the current mainstream hypothesis used in the financial market. It states that the prices of the asset reflect all publicly available information. That's said if all trading parties share symmetric information and frictions are minor effects in the market, then the prices of the asset reflect its expected value conditioned on the publicly available information up to time t and therefore,

$$p_t = E[v_t|I_t]$$

where p_t is the price of the asset at time t, v_t is "true" value or the expected present value priced under the full information condtion at time t. Note that the return at time t could be written in this form,

$$r_t = p_t - p_{t-1} = E[v_t|I_t] - E[v_{t-1}|I_{t-1}]$$

By applying the tower rule, we obtain that $r_t = v_t - v_{t-1}$. This suggests that returns are serially uncorrelated and markets are efficient.

2.2 Market with Microstructure Effects

Unlike the efficient market hypothesis, the market with microstructure effects usually assumes the existence of various kinds of frictions in the market. The frictions mainly come from two sides, the cost to operate the market and the asymmetric information distribution among traders. Let's extend the formulation used in the previous section by adding an error term to the price.

$$p_t = E[v_t|I_t] + \epsilon_t$$

Then the return at time t is described as,

$$r_t = p_t - p_{t-1} = E[v_t|I_t] - E[v_{t-1}|I_{t-1}] + \epsilon_t - \epsilon_{t-1}$$

The serially correlated term $\rho(\epsilon_t, \epsilon_{t-1})$ suggests the inefficiency of the market with the presence of the market frictions. This inefficiency also breaks the assumption made by the efficient market hypothesis and suggests that the prices of the asset do not reflect all publicly available information. One of the factors is the asymmetric information problem. Different traders have their own set of information. This creates asymmetry of information in the market. Therefore, in the thesis, we will focus on the study of asymmetric information.

2.3 Asymmetric Information Paradigm

Let's use the paradigm introduced by Glosten and Milgrom. In the Glosten-Milgrom's scheme, the payoff of the asset is drawn from two-point distribution. Without loss of generality, we simplify the payoff to be 0 with probability $1 - p$ and 1 with p. In this model, the market has three different types of traders, noise traders, informed traders and market makers. The noise traders are the groups of uninformed traders so they buy or sell with equal probabilities. The second type is the informed traders. This models the effect that the information possessed by different traders is asymmetric. Therefore, the informed traders know the payoff at the next time step perfectly. Lastly, the market makers are risk-neutral and competitive traders. In this setting, traders come to the market in the sequential order, so they trade once and exit the market after the trade. After a trader arrives in the market, it is either an informed trader with probability π or an uninformed/noise trader with probability $1-\pi$. There are only two types of orders it can execute, buy or sell. If the incoming trader is an informed trader, as it possesses the information about the payoff, it can submit a buy order if the payoff is 1 and a sell order if the payoff is 0. However, if the incoming trader is a noise trader, the only behavior it has is to randomly buy or sell each with probability 0.5. Since the order size is fixed in this scenario, we can normalize it to be one.

There are several advantages with this setting. It does not assume normality, and this feature makes it more flexible in applications. Also, it assumes that traders arrive in sequential orders. Therefore, market makers can observe every single trade, instead of the aggregated order flows. This is a more accurate description of trading process in the real-world and more suitable for empirical studies and real-world data. This setting also puts some restrictions. Since each trader is only allowed to trade once. The optimal strategy for each trader is to execute greedy orders and there is no incentive for them to place orders strategically. Also, in our setting, the order size is fixed, it does not allow for strategies that apply various order sizes in the market.

2.4 Glosten-Milgrom Pricing Model

In the Glosten-Milgrom model [5], it models the bid/ask prices that market makers should quote in the market equilibrium. The market makers quote a bid price at which they want to trade against a sell order and an ask price at which they want to trade against a buy order. These two prices are not necessarily the same as they contain different information about the payoff. The derivation of these prices are in the following. As market makers are risk-neutral and competitive, they set the prices to be the conditional expectation of the payoff given the previous history of all orders. That is, for the bid price,

$$p_{b,t} = E[d|\mathcal{H}_{t-1}, sell]$$

where d is the payoff, $\mathcal{H}_{t-1}$ is the history of all previous orders up to time $t-1$. The same applies to the ask price.

$$p_{a,t} = E[d|\mathcal{H}_{t-1}, buy]$$

The bid price and the ask price can be derived by the above formulation.

Lemma 2.4.1. *Let π be the probability that the incoming trader is an informed trader. Therefore, at time t, the bid price and the ask price can be modeled as,*

$$p_{b,t} = \frac{\frac{1}{2}(1-\pi)p_{t-1}}{\frac{1}{2}(1-\pi)p_{t-1} + (\pi + \frac{1}{2}(1-\pi))(1-p_{t-1}))}$$

and,

$$p_{a,t} = \frac{(\pi + \frac{1}{2}(1-\pi))p_{t-1}}{(\pi + \frac{1}{2}(1-\pi))p_{t-1} + \frac{1}{2}(1-\pi)(1-p_{t-1})}$$

where p_{t-1} is the transaction price, which is the bid price or the ask price depending on the incoming order, at time $t-1$.

Proof. Let's start with the bid price. As d is either 1 or 0,

$$p_{b,t} = E[d|\mathcal{H}_{t-1}, sell] \tag{2.1}$$

$$= P[d=1|\mathcal{H}_{t-1}, sell] \tag{2.2}$$

By the conditional probability and the total probability, we have,

$$P[d=1|\mathcal{H}_{t-1}, sell] = \frac{P[d=1, sell|\mathcal{H}_{t-1}]}{P[sell|\mathcal{H}_{t-1}]} \tag{2.3}$$
$$= \frac{P[d=1, sell|\mathcal{H}_{t-1}]}{P[d=1, sell|\mathcal{H}_{t-1}] + P[d=0, sell|\mathcal{H}_{t-1}]} \tag{2.4}$$

where,

$$P[d=1, sell|\mathcal{H}_{t-1}] = P[sell|d=1, \mathcal{H}_{t-1}]P[d=1|\mathcal{H}_{t-1}] \tag{2.5}$$
$$= \frac{1}{2}(1-\pi)P[d=1|\mathcal{H}_{t-1}] \tag{2.6}$$
$$= \frac{1}{2}(1-\pi)E[d|\mathcal{H}_{t-1}] \tag{2.7}$$
$$= \frac{1}{2}(1-\pi)p_{t-1} \tag{2.8}$$

$$P[d=0, sell|\mathcal{H}_{t-1}] = P[sell|d=0, \mathcal{H}_{t-1}]P[d=0|\mathcal{H}_{t-1}] \tag{2.9}$$
$$= (\pi + \frac{1}{2}(1-\pi))P[d=0|\mathcal{H}_{t-1}] \tag{2.10}$$
$$= (\pi + \frac{1}{2}(1-\pi))(1 - P[d=1|\mathcal{H}_{t-1}]) \tag{2.11}$$
$$= (\pi + \frac{1}{2}(1-\pi))(1 - p_{t-1}) \tag{2.12}$$

Therefore, putting these equations together, we have the bid price,

$$p_{b,t} = \frac{\frac{1}{2}(1-\pi)p_{t-1}}{\frac{1}{2}(1-\pi)p_{t-1} + (\pi + \frac{1}{2}(1-\pi))(1-p_{t-1})} \tag{2.13}$$

The same idea applies to the ask price. □

From the above lemma, we obtain the bid price and the ask price at time t as a function of the transaction price at time $t-1$.

Chapter 3

Deep Reinforcement Learning and Market Makers

After the appearance of AlphaGo, deep reinforcement learning receives many attentions from various fields. In this thesis, we apply the deep reinforcement learning techniques to the field of market making. In the market, reinforcement learning(RL) agents are asked to quote prices. The agents are designed to map to the continuous action space. For this reason, deep deterministic policy gradient method is used to solve this task. Before takling about the deep deterministic policy gradient [11], let's first introduce several key notions.

3.1 Policy Gradient Method

The policy gradient method [18] is one of the fundamental alogrithms in the field of deep reinforcement learning. It provides an end-to-end optimization to the parametrizable policy network. The policy network is modeled by the function $\pi_\theta(a|s)$ and parametrized by θ, where θ is any kind of functions, $a \in \mathcal{A}$ is the action and $s \in \mathcal{S}$ is the state. The main idea of policy gradient is to compute the gradient of the objective function given by the policy and optimize θ for achieving the best reward.

Specifically, the objective function is defined as the following,

$$J(\theta) = V^{\pi_\theta}(s_0) = \Sigma_a q^\pi(s_0, a)\pi_\theta(a|s_0)$$

where $V^\pi(s)$ is the value function for π_θ, π_θ is the probability distribution of all possible actions and $q^\pi(s, a)$ is the long-term value function. Now we can derive the gradient of the objective function.

Theorem 3.1.1. *(Policy Gradient Theorem [17]) For any differentiable policy $\pi_\theta(a|s)$, the policy gradient is*

$$\nabla_\theta J(\theta) \propto \Sigma_s \mu(s) \Sigma_a q^\pi(s, a) \nabla_\theta \pi_\theta(a|s)$$

Proof. The proof is quite simple. Let's start with the state-value function, $V^\pi(s)$, for any $s \in \mathcal{S}$,

$$\begin{aligned}
&\nabla_\theta V^{\pi_\theta}(s) \qquad (3.1)\\
&= \nabla_\theta \Sigma_a q^\pi(s, a)\pi_\theta(a|s)\\
&= \Sigma_a(\nabla_\theta q^\pi(s, a)\pi_\theta(a|s) + q^\pi(s, a)\nabla_\theta \pi_\theta(a|s))\\
&= \Sigma_a(\nabla_\theta(\Sigma_{r,s'} P(s', r|s, a)(r + V^{\pi_\theta}(s')))\pi_\theta(a|s) + q^\pi(s, a)\nabla_\theta \pi_\theta(a|s))\\
&= \Sigma_a((\Sigma_{s'} P(s'|s, a)\nabla_\theta V^{\pi_\theta}(s')\pi_\theta(a|s) + q^\pi(s, a)\nabla_\theta \pi_\theta(a|s))
\end{aligned}$$

By unrolling the recursion and taking $\beta(s) = \Sigma_a(q^\pi(s, a)\nabla_\theta \pi_\theta(a|s))$, we obtain,

$$\begin{aligned}
&\nabla_\theta V^{\pi_\theta}(s) \qquad (3.2)\\
&= \Sigma_a \Sigma_{s'} P(s'|s, a)\pi_\theta(a|s)\nabla_\theta V^{\pi_\theta}(s') + \beta(s)\\
&= \Sigma_{s'} P_\pi(s \to s', 1)\nabla_\theta V^{\pi_\theta}(s') + \beta(s)\\
&= \beta(s) + \Sigma_{s'} P_\pi(s \to s', 1)\beta(s') + ...\\
&= \Sigma_{s'} \Sigma_{k=0}^{\infty} P^\pi(s \to s', k)\beta(s')
\end{aligned}$$

where $P^{\pi}(s \to s', k)$ is the probability of transitioning from state s to state s' in k steps under policy π.
Therefore,

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta V^{\pi}(s_0) \\
&= \Sigma_{s'}\nu(s')\Sigma_a(q^{\pi}(s', a)\nabla_\theta \pi_\theta(a|s')) \\
&= (\Sigma_{s'}\nu(s'))\Sigma_{s'}\frac{\nu(s')}{\Sigma_{s'}\nu(s')}\Sigma_a(q^{\pi}(s', a)\nabla_\theta \pi_\theta(a|s')) \\
&\propto \Sigma_{s'}\mu(s')\Sigma_a(q^{\pi}(s', a)\nabla_\theta \pi_\theta(a|s'))
\end{aligned} \tag{3.3}$$

where $\nu(s') = \Sigma_{k=0}^{\infty} P^{\pi}(s \to s', k)$ and $\mu(s') = \frac{\nu(s')}{\Sigma_{s'}\nu(s')}$ is the stationary distribution under π. □

3.2 Deterministic Policy Gradient

As described in the previous section, the policy network there is the probability distribution over all possible actions. In this section, we introduce the idea of deterministic policy gradient [16], in which the policy is modeled as a deterministic decision. Thus, we have the following objective function,

$$J(\theta) = \int_{\mathcal{S}} p^{\mu_\theta}(s) q(s, \mu_\theta(s)) ds$$

where $\mu(s)$ is a deterministic policy, $P^{\mu_\theta}(s)$ is the discounted state distribution, $p^{\mu_\theta}('s') = \int_{\mathcal{S}} \Sigma_{k=0}^{\infty} \gamma^{k-1} p_0(s) P^{\pi}(s \to s', k) ds$ and γ is the discounted factor. It's worth noting that the policy is deterministic. The variations only come from different states. So, the gradient of this objective function is derived as the following,

Theorem 3.2.1. *(Deterministic Policy Gradient) Assume the MDP satisfies conditions, then*

$$\nabla_\theta J(\theta) = \int_{\mathcal{S}} p^{\pi}(s) \nabla_\theta \mu_\theta(s) \nabla_a q^{\mu}(s, a)|_{a=\mu_\theta(s)} ds$$

The proof of the theorem is in the original paper [16]. it's also shown that the deterministic policy is simply a special case of the stochastic policy. This opens up the opportunies for the deterministic policy gradient methods to be used in any policy gradient algorithm. The key advantage of the DPG over traditional stochastic policy gradient method is that the deterministic policy gradient removes the integration over all actions. It is more sample efficient and drastically reduce the time required for training a RL algorithm.

3.3 Actor-critic Algorithm

In the policy gradient method introduced above, there are two main components in the objective function, the policy model π and the value function q^π. The idea of actor-critic method [9] is to further model the value function in addition to the policy model. The parametrization of the value function can also help learn the policy and stablize the policy gradients.

In the actor-critic method, there are two models, the actor network and the critic network. The critic network is in charge of the value updates. It sends error signal to the actor network based on the reward signal from the environment. The actor network is also called the policy network, where it outputs actions based on the current states. The updates of the actor network are guided by the tuple (error signal, actions, states), where error signals are provided by the critic network. The detailed algorithm is described in their original work [9].

3.4 Deep Deterministic Policy Gradient

Deep deterministic policy gradient method(DDPG) [11] is one of the most well-known deep reinforcement learning methods in the field. DDPG is a model-free, off policy actor-critic algorithm that uses deep neural networks to learn policies in continuous action spaces. Based on the idea of DPG and actor-critic algorithm, DDPG further incorporates the two main ideas from deep Q-networks [13] [14], experience replay and target networks.

Experience Replay The experience replay is a technique to build a replay buffer that stores experiences, the tuple of the reward, the state and the action, of the RL agent during training. Since the consecutive experiences are temporally correlated, this temporal correlation will lead to large variance in the estimation of the value function. In order to break the temporal correlation within different training episodes, the replay buffer is introduced. With the replay buffer, the training is conducted by randomly sampling experiences from the buffer instead of using temporally-correlated experiences within episodes.

Target Networks In addition to the actor and critic networks, DDPG creates a target actor network and a target critic network for them respectively. It uses the target actor network and the target critic network as the reference value to update the critic networks. The issue with not using the target networks is that during the training, the parameters of the both actor and critic networks are changing. The changing error signals will let both networks diverge eventually. The introduction of the target networks is to put regularizations on the learning processes and greatly increases the stability of the training.

Other than these ideas, it further introduces the idea of action explorations in the continuous domain. The exploration policy it constructed directly uses the additive noise to the actor policy, where the additive noise is modeled by the Ornstein-Uhlenbeck process to produce temporally-correlated explorations. The details of the algorithm is described in appendix Algorithm 1.

Chapter 4

Experiments

4.1 Multi-agent Simulation Framework

As previously mentioned, our study involves multiple market maker agents competing with each other. Lets's consider the following market condition. Think about a dealer market with n market makers, m investors/traders and a single traded security. The payoff of the asset at time t is sampled to be 0 with probability $1 - p_{t-1}$ and 1 with probability p_{t-1}. As described previously, traders arrive sequentially and only one trades at a time. Therefore, a trade occur only between one of the market makers and the arrived trader.

4.1.1 Market Makers

For any market maker i at time t, it observes the transaction price the trader trades at the previous time step $t - 1$ and all the previous order histories up to time $t - 1$. Based on these information, the market makers decide to quote the ask price and the bid price using its own pricing model. After submitting these prices, the market makers receive reward from the environment in the form of PnL's.

- Observations

    1. The transaction price p_{t-1} the trader trades at time $t - 1$.

2. The order histories up to time $t-1$, specifically the number of buys and sells up to $t-1$.

- Actions

1. Use pricing model to obtain $p_{a,t}$ and $p_{b,t}$. For the neural agent, it outputs the pricing factors $\epsilon_{a,t}$ and $\epsilon_{b,t}$ such that $p_{a,t} = p_{t-1}(1+\epsilon_{a,t})$ and $p_{b,t} = p_{t-1}(1+\epsilon_{b,t})$.

- Rewards

1. Type 1 reward is the spread PnL, the environment rewards the winning agent the spread $p_{a,t}-p_{t-1}$ if the incoming order is a buy order or $p_{t-1}-p_{b,t}$ if it is a sell order.
2. Type 2 reward considers the asset payoff. The winning agent gets the reward $p_{a,t}-d$ or $d-p_{b,t}$.

Note that type 1 reward is the traditional description of the rewards gained by market makers. In the rest of the paper, we will use type 2 reward. The sense of "win" in the market is defined as the one who quotes the lowest ask price if the incoming order is a buy or the one who quotes the highest bid price if the incoming order is a sell. The environment described above is mainly for evaluations. In the training phase, we also apply regret terms to each losing agent as described in chapter 3 both to stablize and speed up the training.

4.1.2 Traders

There are two types of traders in the market, noise traders and informed traders. The probability distribution for the noise traders is $1-\pi$ and that for the informed traders is π. As there is only one trader at each time step, we sample the type of traders with above probability distribution. The noise traders have no information about the asset payoff, so they randomly pick the order type, sell or buy, with probability exactly 0.5. Unlike the noise traders, the informed traders know the payoff accurately

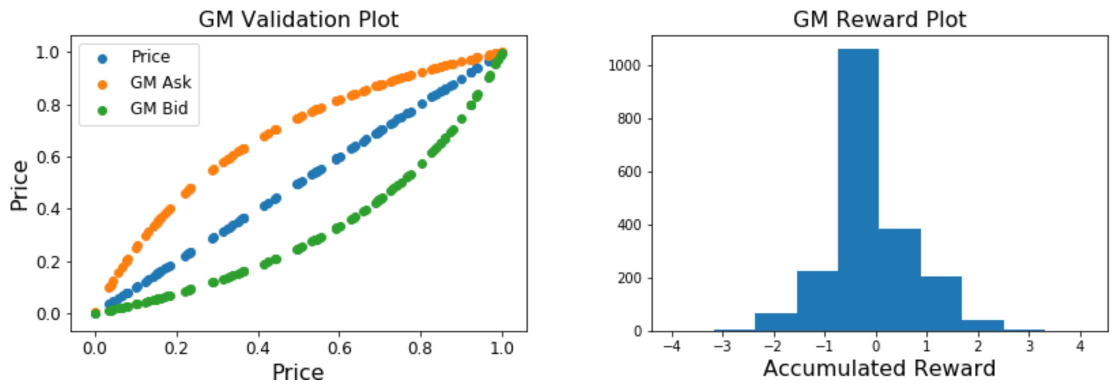


Figure 4-1: Left: The scatter plot shows the Glosten-Milgrom model's prediction of the ask and bid price. Right: This is the histogram of accumulated reward. The reward is calculated using the type 2 reward functions. It is accumulated over time horizon 10 iterations in the dynamic simulations. The averaged reward is -0.01348, which is very close to the expected reward, 0.

and they will submit a buy order if the asset payoff is 1 and a sell order if the payoff is 0. For each trader, it will greedly pick the market maker that offers the best price. No strategic move is allowed for traders.

4.1.3 Single-period Modeling and Dynamic Modeling

Both the single-period simulation and the dynmaic simulation use the same pipeline. The detailed procedures are described in the Algorithm 2 in Appendix. It only differs in the pricing part. Since in the single period model, there is no order history. The only thing it considers is the previous transaction price. However, the dynamic modeling simulates the whole episode of the events. In addition to the transaction prices, it also has the accumulated order history. For both simulations, rewards are only delivered at the end of each simulation episode.

4.1.4 Validation using the Glosten-Milgrom Pricing Model

The validation of the framework uses the theoretical solution provided by Glosten-Milgrom model. Here we only do dynamic simulations in that for this theoretical model the dynamic simulation is the generalization of the single-period simulation. Since GM model is risk-neutral and competitive, in equlibrium the GM model could provide an expected reward of 0 in the current market. In Figure 4-1, the figure on the left provides the ask and bid curve obtained from the GM model and the figure on the right shows the distribution of the delivered rewards. The model fits in the framework well in the sense that the averaged reward is very close to the expected reward. These simulation results validate the framework as the basic market environment, and allow us to further explore the market using neural agents.

4.2 Deep RL-based Market Makers

The main objective of this work is to model market makers using neural network agents and study the effects of adverse selection and informed traders on the behaviors of market makers. In our setting, we apply the standard deep deterministic policy gradient(DDPG) method to train the RL-based market makers. DDPG is one of the state-of-the-art methods used to the continous action main, and the market, obviously, only takes continuous prices. So in our case we build neural approximators to model the pricing strategy and use DDPG to optimize the strategy under our market conditions. In our model, the input to the neural network is the transaction time at the last time step and the all previous order history up to the last time step. The ouput of the neural network are the pricing factors $\epsilon_{a,t}$ and $\epsilon_{b,t}$ where $\epsilon_{a,t}, \epsilon_{b,t} \in [-1, 1]$. The ask price $p_{a,t}$ and the bid price $p_{b,t}$ at time t are modeled as,

$$p_{a,t} = p_{t-1}(1 + \epsilon_{a,t})$$

and

$$p_{b,t} = p_{t-1}(1 + \epsilon_{b,t})$$

The design of the function is inspired by the previous work [4]. We have also considered another output function, in which we directly use the output of the neural agents as the ask and bid prices. This output model does not work for almost all the time. The the ask prices will be pushed to 1 and the bid prices will be pushed to 0 after training. So in this thesis, we will only use the output model described above.

4.2.1 Reward Formulations

The design of the reward functions is based on two main ideas. The first one is inspired by the settings used in the Glosten-Milgrom model. In the equilibrium the difference between the actual asset payoff and the expected payoff converges to zero. The second idea comes from the fact that there are multiple agents competing with each other. Therefore, there should be a regret term for the agents who can observe all previous orders but cannot win the order.

So the basic reward formulation uses difference between the current asset payoff and the trading price. That is, if the incoming order is a buy order then the neural agents should quote a ask price. If there is only one agent in the market, the reward signal is simply $r_t = p_{a,t} - d$. If there are multiple agents competing with each other, there are two different types of reward functions, one for the winning agents and one for the losing agents. For the case that the agent places an ask order, the winning agent is the one that places the lowest ask price in the market. Note that the winning agent is unique. If there are multiple agents place the same ask prices, the winning agent is uniformly sampled from one of them. So, the winning agent gets reward $r_t = p_{a,b}^{win} - d$. For each losing agent i, they get reward $r_t = \epsilon_a d(p_{a,t}^{win} - p_{a,t}^{i})$, where the reward for losing agents could be seen as the regret term of not winning the order and ϵ_a is the parameter that controls the magnitude of the whole regret term. The same idea applies to the case that the agent places a bid order, with the difference that the winning agent gets reward $r_t = d - p_{b,t}^{win}$ and each losing agent i gets reward $r_t = \epsilon_a(1-d)(p_{b,t}^{i} - p_{b,t}^{win})$, where ϵ_b is the controlling factor for the regret term in the bid case.

The reasons to design of the regret terms are twofold. It provides explicit competitions for losing agents. In the original design, 0 rewards are applied to those who lose the orders. However, this will lead to the case in which all neural agents choose to not compete and end up getting some random behaviors. The second goal here is to stablize the convergence of the DDPG algorithm. DDPG does not behave well under the sparse reward setting. If we choose to use 0 rewards to losing agents, then the number of experiences required to train the neural agents will increase exponentially.

4.2.2 Other Training Techniques

In addition to the reward functions described above, several other training techniques are applied to stablize the training process.

Balancing Training Samples DDPG uses memory to store all training samples, and these training samples are collected from previous experiments. DDPG algorithm draws a minibatch randomly from the memory storage. So this random draw will potential produce the bias that the minibatch contains more reward signals about either asking or bidding. This unbalanced problem lets the trained network only perform well on either asking or bidding, but not both. In order to reduce the bias induced by the unbalanced training samples, we balance the training samples for each iteration.

Additive Competing Factor In the multiagent environment, the regret terms in the reward functions for losing agents have some edge case, in which the losing agent places the same price as the winning agent but loses in the lottery step. We add a small competing epsilon $\epsilon_c = 0.001$ to the target price $p_{a,t}^{win} - \epsilon_c$ or $p_{b,t}^{win} + \epsilon_c$. This modification encourages neural agents to compete with each other independently and avoids the case that they make agreement on some unreasonable prices.

Cyclic Training Procedure When multiple neural agents are present in the market, only one agent is allowed to get trained at one iteration. Unlike the target network training method, the cyclic training procedure lets neural agents to compete in turn. In our case, we let each neural agent to get trained up to 500 iterations and repeat this process until all of them converge. This technique is to make sure that each agent is competing with stable agents. Otherwise, if we train all the agents at each iteration, the agents are constantly competing with other changing agents. This will make the whole training process very unstable.

Chapter 5

Experiments

5.1 Experiment Settings

The experiments conducted in this study cover two kinds of competitions: 1. the competition between the GM model and the neural agent; 2. the competitions among three neural agents. Both experiments are run in the single-period setting and in the dynamic setting. In the DDPG algorithm, both actor and critic networks use 2-layer neural networks with 50 nodes in the first layer and 30 nodes in the second layer. The batch size is 64. The activation function for the actor network is tanh function. The optimizer is Adam with learning rate 1e-3. The evaluation is conducted after the neural agents converge and the negative regret rewards are set to be zero in the evaluation phase. The rewards calculated in the evaluation phase are averaged over 500 simulations.

5.2 Results and Analysis

The evaluation metrics for the task uses averaged reward over 500 simulations. In this market, there are two driving forces determining the averaged reward for market makers, the intention to make profits and the competition pressures. The first kind is the natural driver for market makers to quote prices. As for-profit agents, market makers' ultimate goal is to earn money from market making. This leads them to quote higher ask prices and lower bid prices. The second driving force comes from the competitions from other competitors. If the agents lose the order at certain time, they will receive 0 award as they do not have the chance to trade. Therefore, as long as the agents can have positive expected rewards, they will quote lower ask prices and higher bid prices. In equilibrium, these two counter forces will be offset and agents are expected to have 0 expected rewards.

Glosten-Milgrom model competes with one neural agent We first conduct experiments on the competitions between the Glosten-Milgrom model and the neural agent. Note that the Glosten-Milgrom model is used as a trained agent to compete with the neural agent. For both single-period simulations and dynamic simulations, the evaluation only calculates the reward from the competitions among neural agents, and the Glosten-Milgrom prices are not involved. In this case, there is only one neural agent. The reward is simply the difference between the asset payoff and the ask/bid prices outputted from the neural agent.

From Figure 5-1, in the single-period simulation, the figure on the left shows that the shape of ask/bid prices given by the neural agent is somewhat similar to the shape of the GM prices. The neural agent can capture pricing behaviors from the Glosten-Milgrom model under this simple setting. Also, the averaged reward for the neural agent is very promising. The result is 0.05226 and thus very close to the expected reward. In the dynamic simulation, the figure on the right gives the distribution of the ask prices and bid prices given by the neural agent. One session in the the dynamic simulation runs in 5 iterations, and will early stop if the transaction price reaches 0 or 1. The shape of ask prices is very similar to the ask prices obtained from the single-period simulation. The bid prices are very distinct. In Figure 5-2, the bid prices are splitted by the time step. As shown in the plot, for the same transaction price, the neural agent will give different bid price at different time step. This shows that the neural agent is developing some kind of strategies conditioned on not only the transaction price but also the order history. Surprisingly, this strategy gives averaged reward 0.0005346.

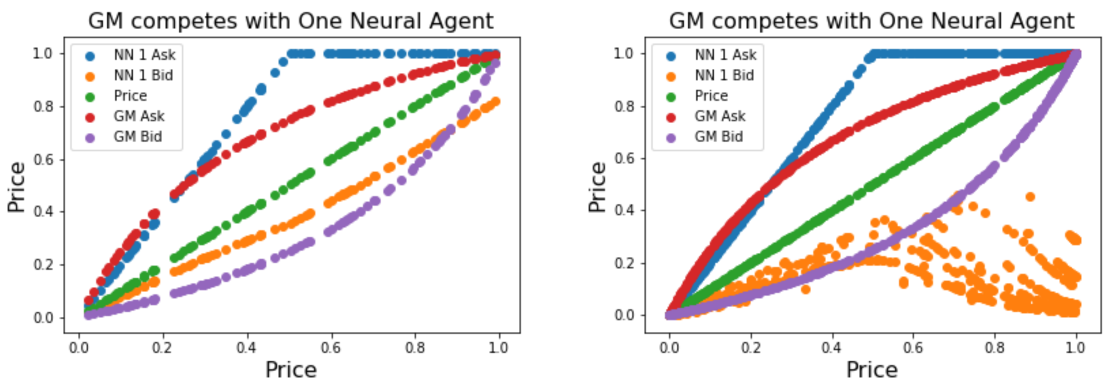


Figure 5-1: Left: The price formed from the competitions between the Glosten-Milgrom model and the neural agent under the single-period simulation. Right: The price formed from the competitions between the Glosten-Milgrom model and the neural agent under the dynamic simulation.

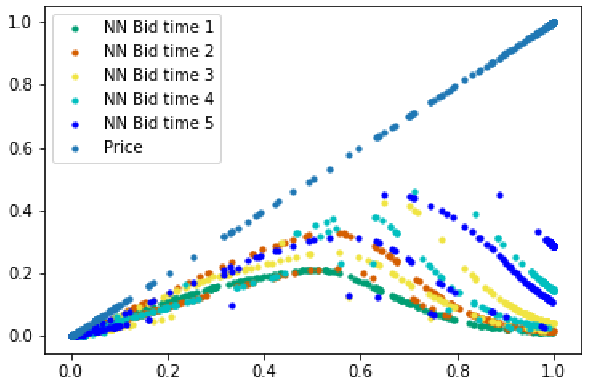


Figure 5-2: In this plot, we split the bid prices at different time step.

Competitions among three neural agents The interesting part comes from the competitions among three neural agents. In this scenario, there is no trained agent, such as the Glosten-Milgrom model, participanting in the training phase. Only three neural agents compete with each other. In Figure 5-3, the experiments shown in the figures are under the single-period setting. The plot on the left demonstrates three sets of ask prices and bid prices from each neural agent. For all three neural agents,

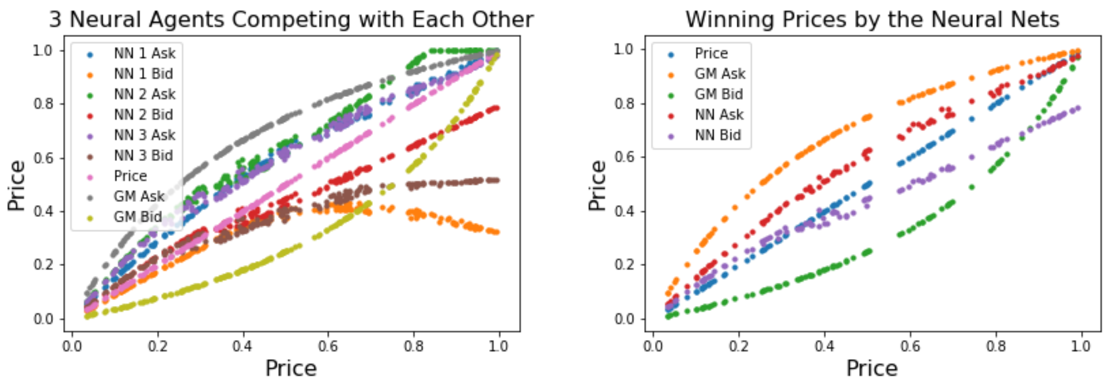


Figure 5-3: Left: The price formed from the competitions among the three neural agents under the single-period simulation. Right: The price formed from the competitions among the three neural agents under the dynamic simulation. For both plots, the GM ask/bid is used as a reference and not involved in the training phase.

the curves of ask prices are intertwined together. It shows that the competitions among these three neural agents force them to reach the price equilibrium. However, on the bid side, one curve dominates the competition. The plot on the right is the winning strategy taken from these three neural agents. Surprisingly, this strategy gives averaged reward 0.0697. In addition, the averaged rewards for each neural agent are plotted in Figure 5-4 respectively. The averaged rewards for each of them are, (-0.0220, -0.0586, -0.0358).

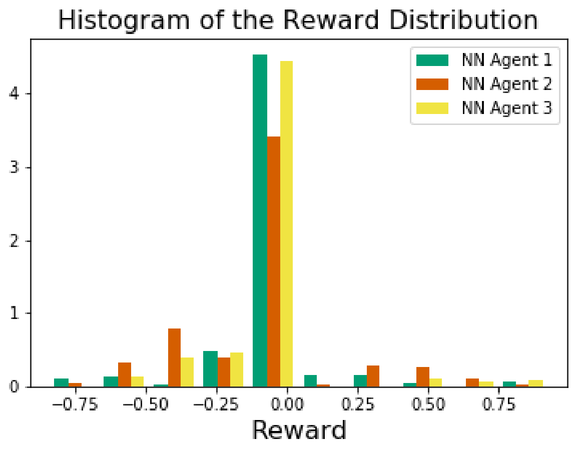


Figure 5-4: The histogram shows the reward distribution for all three neural agents.

Howevever, in 5-5, the dynamic simulation shows divergent behaviors. The ask curves look reasonable, but the bid curves are somewhat random. Multiple tricks and various hyperparameters are tested under this setting, but none of them worked well.

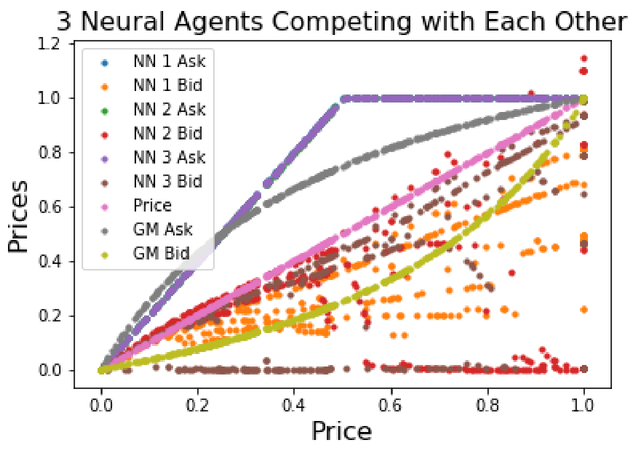
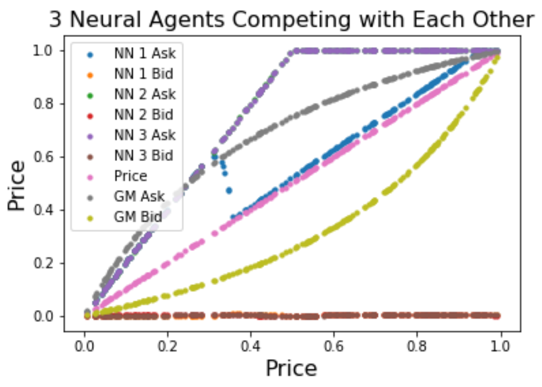


Figure 5-5: The price disitribution under the dynamic simulation settings

Some failure case Given the techniques to stablize the training introduced previously, the neural agents are still not very stable during the training phase. From Figure 5-6, it is the result that the neural agents converge to in the single-period simulation. These ask prices and bid prices show that the competitions among the three neural agents are not intense enough. This is suggesting that the model is very sensitive to the magnitude factor ϵ_a and ϵ_b in the regret terms. Even though the curves look very strange, the winning strategy shown on the right still gives an averaged reward 0.0412. The result is very surprising. Any of the three neural agents has the chance to make more profits, but all of them choose to stay greedy. For instance, one of the three neural agents could simply make a little bit high bid prices and get all the profits instead of making agreement with the rest.

Discussion The proposed framework allows us to test various competition settings and see their equilibrium. It is worth noting that even though Glosten-Milgrom model provides a generally optimal solution to the market with asymmetric information, it is not the unique solution. The GM model solves the problem independent of the competitors' behaviors, but this is not the only solution. There are other solutions



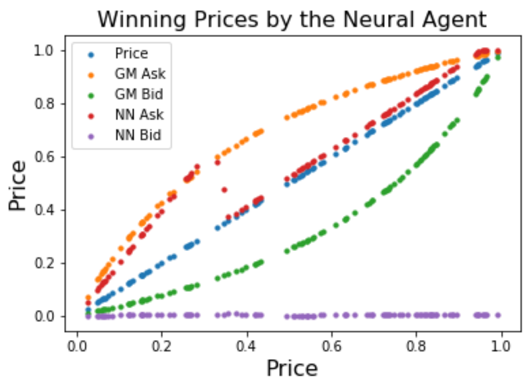


Figure 5-6: The price distribution formed by the comeptitions among three neural agents under the single-period simulation setting. Left: the price distribution for all three neural agents. Right: the price distribution derived from the winning strategy.

conditioned on other competitors' behaviors. One good example is the plot shown on the Figure 5-3. Our work provides a different perspective on the issue of adverse selection risks. It suggests that the market equilibrium exists not only in the form described in the Glosten-Milgrom model. There are other kinds of market equilibrium conditioned on the rest of players in the market.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

In the thesis, we present the mutliagent environment under the asymmetric information assumptions. Several competition scenarios, with different competition schemes and single-period/multiple-period modeling, have been tested and analyzed. It could be concluded that unlike the theoretical model, the Glosten-Milgrom model, neural agents are able to capture pricing strategy that could reach market equilibrium as well. The policies learnt by neural agents are conditioned on the pricing strategies of the rest of the players in the market, whereas the policy derived from the Glosten-Milgrom model is a generally optimal strategy and is independent of the pricing behaviors of other players participating in the market. This allows us to look at the problem of asymmetric information from a multiagent perspective.

6.2 Future Extension

There are many potential interesting directions to extend this thesis. A natural idea is to further stablize the training environment. Several tricks or modification could be done here, changing the training method and designing a different kind of reward functions. Current training method uses DDPG as the main approach to train neural agents. However, there are several limitations about DDPG. DDPG usually learns unstable policies, so it could be observed that the trained policy has many variations and some of them quote very strange prices. A better choice is normalized advantage functions(NAF) [6], which is also a variant of Q-learning. NAF will learn stable policies. The design of reward functions also plays a key role here. In addition to the canonical reward functions described in the environment, it is possible to apply cross-entropy penalities on the regret term instead of the linear comparison introduced in the thesis. Other than stablizing the environment, more microstructure features could be included in the environment, such as private information and inventory controls. Private information describes the scenario in which each market maker possesses different kind of information about the market and it could be quantifies in the reward functions to guide market maker's quoting behaviors. The concept of inventory controls is usually introduced when market makers quote multiple ask/bid prices and they need to maintain the number of the prices they quote depending on the drift of the market price.

Appendix A

Tables

Experiment Type	ϵ_a	ϵ_b
GM agent with one neural agent, single-period	1.2	1.4
GM agent with one neural agent, dynamic	1.4	1.7
Three neural agents, single period	1.3	1.7
Three neural agents, dynamic	1.7	2.2

Table A.1: Hyperparameters used in the regret term in the reward function.

Appendix B

Figures

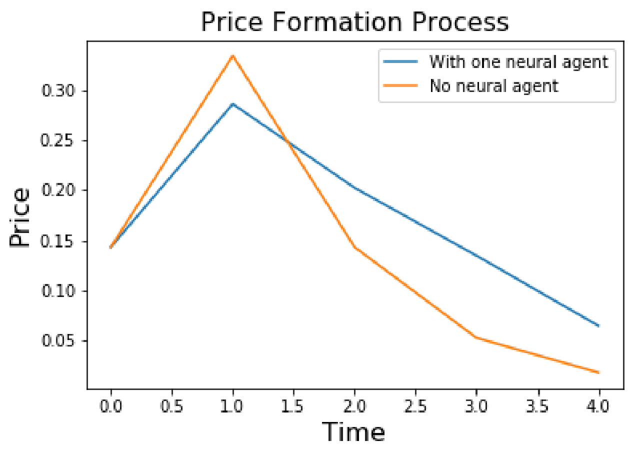


Figure B-1: Here is the plot showing price formation process under the dynamic simulation. The blue line is formed in the market with GM agent and one neural agent and the orange line describes the market with only one GM agent.

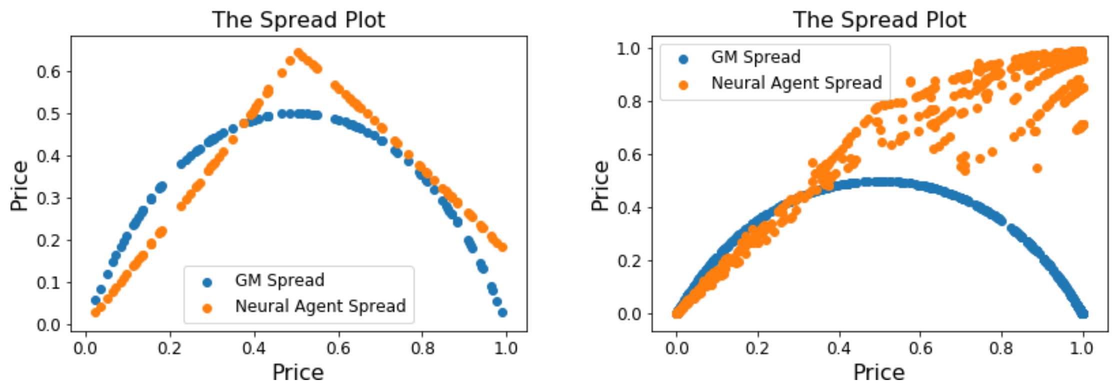


Figure B-2: Left: it is the spread curve for the market with GM agent and one neural agent under the single-period simulation. Right: it is the spread curve with the same condition but under the dynamic simulation.

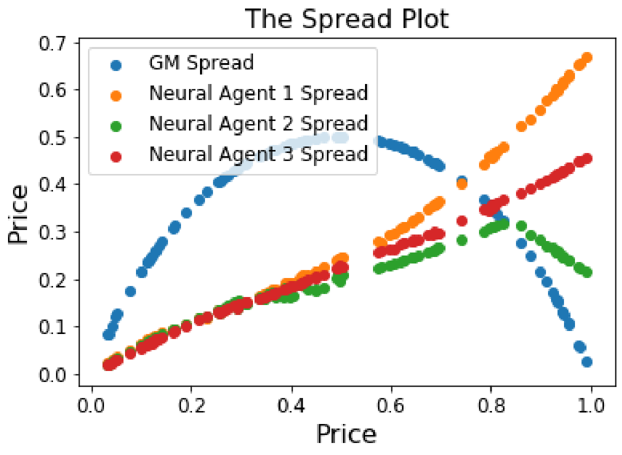


Figure B-3: It is the spread curve for the market with three neural agents under the single-period simulation. The GM spread is taken as a reference here.

Appendix C

Algorithms

Algorithm 1 Deep Deterministic Policy Gradient

```
    Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
 2: Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$.
    Initialize replay buffer R.
 4: **for** e **do**pisode = 1, M
        Initialize a random process $\mathcal{N}$ for action exploration.
 6:     Receive initial observation state $s_1$.
        **for** t **do** = 1, T
 8:         Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise.
            Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$.
10:         Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$.
            Sample a random minibatch of N transition $(s_i, a_i, r_i, s_{i+1})$ from R.
12:         Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$.
            Update critic by minimizing the loss: $L = \frac{1}{N}\Sigma_i(y_i - Q(s_i, a_i|\theta^Q))^2$
14:         Update the actor policy using the sampled policy gradient:
                $$\nabla_{\theta^\mu} J(\theta) \approx \frac{1}{N}\Sigma_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$
            Update the target networks:
                $$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
                $$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$
16:     **end for**
    **end for**
```

Algorithm 2 Market with asymmetric information

```
1: Time horizon T
2: Uniformly sample initial price $p_0 \sim [0, 1]$
3: The density of informed traders $\pi$
4: for $t = 0, \ldots, T - 1$ do
5:     Sample the asset payoff $d_t \sim Bernoulli(p_t)$.
6:     Sample the type of traders, tdr, with density $\pi$ and $1 - \pi$.
7:     if tdr is the informed trader then
8:         if the asset payoff $d_t$ is 1 then
9:             Submit a buy order
10:        else
11:            Submit a sell order
12:        end if
13:    else
14:        Uniformly sample a random number $k$ from $[0, 1]$.
15:        if k $< 0.5$ then
16:            Submit a buy order
17:        else
18:            Submit a sell order
19:        end if
20:    end if
21:    if previously submitted order is a buy order then
22:        For each agent $i$, let the agent make an ask quote $p^i_{a,t}$.
23:    else
24:        For each agent $i$, let the agent make a bid quote $p^i_{b,t}$.
25:    end if
26:    Set the agent who provides the best price to be the winning agent
27:    Calculate the reward for each agent.
28:    Set the transaction price $p_{t+1} = min_i\{p^i_{a,t}\}$ or $p_{t+1} = max_i\{p^i_{a,t}\}$ depending
   on the submitted order.
29: end for
```

Bibliography

[1] Louis Bachelier. *The Theory of Speculation.* PhD thesis, University of Paris, 1900.

[2] Bruno Biais, Larry Glosten, and Chester Spatt. Market microstructure: A survey of microfoundations, empirical results, and policy implications. *Journal of Financial Markets*, 8(2):217–264, 2005.

[3] Eugene Fama. Efficient capital markets: A review of theory and empirical work. *Journal of Financial Markets*, 25(2):383, 1970.

[4] Sumitra Ganesh, Nelson Vadori, Mengda Xu, Hua Zheng, Prashant Reddy, and Manuela Veloso. Reinforcement learning for market making in a multi-agent dealer market. In *33rd Conference on Neural Information Processing Systems*, 2019.

[5] Lawrence R. Glosten and Paul R. Milgrom. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of financial economics*, 14(1):71–100, 1985.

[6] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *arXiv:1603.00748*, 2016.

[7] Olivier Guéant, Charles-Albert Lehalle, and Joaquin Fernandez Tapia. Dealing with the inventory risk: a solution to the market making problem. *Mathematics and Financial Economics*, 7(4):477–507, 2013.

[8] Thomas Ho and Hans Stoll. Optimal dealer pricing under transactions and return uncertainty. *Journal of financial economics*, 9(1):47–73, 1981.

[9] Vijay R. Konda. *Actor-Critic Algorithms.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2002.

[10] Albert S. Kyle. Continuous auctions and insider trading. *Econometrica*, 53(6):1315–1335, 1985.

[11] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations*, 2016.

[12] Ananth Madhavan. Market microstructure: A survey. *Journal of Financial Markets*, 3(3):205–258, 2000.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *Proceedings of the 31 st International Conference on Machine Learning, Beijing, China*, 2014.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[15] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, and Karen Simonyan Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[16] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, and Daan Wierstra et al.. Deterministic policy gradient algorithms. In *Proceedings of the 31 st International Conference on Machine Learning, Beijing, China*, 2014.

[17] Richard S Sutton and Andrew G. Barto. *Introduction to reinforcement learning*, chapter 1.2. MIT Press, 1998.

[18] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000.