

A Study on Order Generation in High Frequency Trading with Generative Adversarial Network

Thesis by
Yusuke Naritomi

Advisor
Professor Takanori Adachi

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
(Graduate School of Management)



Tokyo Metropolitan University
Tokyo, Japan

2022

(Defended December 22, 2022)

Acknowledgements

Foremost, I would like to express my sincere gratitude to Professor Takanori Adachi for the continuous support of my Ph. D. study and research, for his politeness, patience, and immense knowledge. His support helped me in all the time of research and writing of this thesis. Without his support this thesis would not exist.

This research was supported by Research Center for Quantitative Finance, Tokyo Metropolitan University.

Finally, the thesis would never be written without the understanding and the cooperation given by my family, especially by my wife, Mai as well as our kid, Miyo. I feel a deep gratitude to them.

**A Study on Order Generation in
High Frequency Trading with
Generative Adversarial Network**

by
Yusuke Naritomi

In Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

Abstract

This paper focuses on the generation of order data, which is high frequency financial time series data, and aims to construct highly realistic artificial market simulations.

In recent years, as seen in data assimilation, research on the fusion of computer simulations and real data has been active. With the development of machine learning, especially deep learning, it has become possible to simulate using deep learning models instead of physical models. This is surprising, because it means that simulations can be performed without knowing the underlying physical model if the deep learning model can be successfully trained. However, this requires a large amount of data to be trained.

This research focuses on high-frequency trading in the stock market and aims to construct a highly realistic artificial market by using a deep learning model to generate time series of micro orders placed by traders. The data generation method was modeled using the Wasserstein Generative Adversarial Network (WGAN).

As the first step, we studied the effectiveness of order data generation and data extension by WGAN, focusing only on the dynamics of microscopic order data. The Limit Order Book was calculated from artificially generated order time-

series data, and the price data was calculated by hypothetically execution of orders and was used as a data augmentation of the 1-tick ahead price forecasting model. For many individual stocks, the forecasting model with data augmentation was confirmed to be more accurate than the model without data augmentation, indicating its effectiveness. However, micro order dynamics alone isn't enough to represent macro quantities such as price time series and dynamics with long-term correlations, making it difficult to apply the model to long-term forecasts.

As the second step, we conducted research on the signature of price time series data, which is a macroscopic feature. The signature is a method for calculating features specific to the shape of a time series, and has recently begun to attract particular attention and is being applied in a variety of fields. However, since the paths of time series are assumed to be continuous, discrete data is converted to continuous paths by linear interpolation or other means before computing. Since the original data of financial time series data has discrete values, it is desirable to be able to handle them as discrete values directly. Therefore, we have extended the signature to be able to calculate the signature as discrete data and named it the discrete signature, and showed that it is effective for financial time series data.

As a final step, we propose Micro-Macro GAN as a coupled computation of the two GANs. It calls the micro order data generation method used in the first step, MicroGAN, and then calls the macro price data generation method using the signature described in the second step, MacroGAN. In other words, Micro-Macro GAN is a method for generating realistic data on both micro order data and macro price data. As a result, Micro-Macro GAN achieved the same level of results for generating order distribution as MicroGAN, and the better level of results than MicroGAN using the signature to evaluate macro quantities. However, the computational cost of learning was quite high, so further improvement is needed.

In order to realize a realistic artificial market simulation, reality is required for both micro order data and macro price data, which leads us to propose the Micro-Macro GAN. Future applications of the artificial market simulation pro-

posed in this paper may include data augmentation over various time horizons, evaluation in artificial markets as an alternative to back-testing, and validation as an environment for improving automated trading algorithms.

Contents

Acknowledgements	iii
Abstract	v
1 Global Introduction	1
1.1 Introduction	1
1.2 Research motivation	6
1.3 Purpose of research	6
1.4 Our contribution	7
1.5 Structure of thesis	7
2 Data Augmentation of High Frequency Financial Data Using Generative Adversarial Network	9
2.1 Introduction	9
2.2 Prerequisite	11
2.2.1 Setting	11
2.2.2 Wasserstein GAN	13
2.3 Method	14
2.3.1 Definition of events	14
2.3.2 Architecture of GAN	16
2.3.3 Generator network g_θ	16
2.3.4 Discriminator network f_w	17
2.3.5 Algorithm	18
2.4 Data	18

2.5	Result	20
2.5.1	Comparison of real and synthetic data generated by the GAN	20
2.5.2	Artificial market simulation with GAN	23
2.5.3	Verification of the effectiveness for data augmentation with GAN	23
2.5.4	Effect of noise dimension	27
2.6	Conclusion	30
3	Discrete Signature and its Application to Finance	33
3.1	Introduction	33
3.2	Discrete Signature	34
3.3	An implementation of discrete signature	42
3.3.1	The class Data	42
3.3.2	The class Words	43
3.3.3	The class Signature	43
3.4	An application of discrete signature to finance	46
3.4.1	Make a one-minute interval data stream	46
3.4.2	Time normalization	48
3.4.3	Make a discrete path for each session	49
3.4.4	Experiment and result	49
3.5	Concluding remarks	51
4	Stock Market Simulation by Micro-Macro GAN	53
4.1	Introduction	53
4.2	Related work	55
4.3	Prerequisite	56
4.3.1	Setting	56
4.3.2	Wasserstein GAN	58
4.3.3	Signature	59
4.4	Method	62
4.4.1	Definition	62

	xi
4.4.2 Overall picture of Micro-Macro GAN	63
4.4.3 Micro GAN	66
4.4.4 Macro GAN	67
4.4.5 Micro-Macro GAN Algorithm	68
4.5 Data	69
4.6 Result	70
4.6.1 Comparison of probability distributions and their distances	70
4.6.2 Comparison of price and volume dynamics	72
4.6.3 Timescales of price dynamics by the Micro GAN and the Micro-Macro GAN	73
4.6.4 Comparison of signatures	75
4.7 Conclusion	75
5 Global conclusion and discussion	77
Bibliography	79
Index	85

Chapter 1

Global Introduction

1.1 Introduction

The invention of money is said to be one of the great human inventions. A financial institution allows an individual or company to borrow money to purchase real estate, develop a new business, and so on. Before the invention of money, money was usually replaced by foodstuffs such as rice or precious metals such as gold and silver. Furthermore, in today's society, people use cash less and less, and we are shifting to a cashless society with credit cards and electronic money, where money has become a completely digital number. One of the advantages of money is that as long as the government guarantees the value of money, there is no clear expiration date. However, the relative value of assets fluctuates from time to time due to interest rates, exchange rates, and other factors, and it is also possible to increase assets by investing in financial instruments such as stocks, bonds, and derivatives. This is an important task because if we can successfully predict the future value of a financial instrument, we can avoid risk or increase our assets by investing in it. However, the financial instruments are finite, so if you want to invest, you need to act faster than other investors, and therefore it is important to obtain advantageous information before other investors do. The information data commonly used for future forecasting includes *technical data*, *fundamental data*, and more recently, *alternative data* such as news text. The

important thing to note here is that all of these data are accumulated and formed as a result of human activity or action.

Behind the price formation of stock prices are the various speculations of market participants. For example, if a company's performance is expected to grow, there will be many buy orders, and as a result, the stock price will rise. In other words, it is no exaggeration to say that the feelings of market participants are forming prices. They invest based on their preferences of investor judgment and data from the outside. Investors invest based on their own preferences and external data, and their behavior may also change depending on their physical condition on any given day. As described above, we understand that the price is determined by a very complex process.

If the principles of human behavior are understood, it is expected that the accuracy of future predictions will become considerably more accurate, and research linking *behavioral science* and finance has been active recently. However, as these studies proceed, the question ultimately arises as to what is the mechanism of *human intelligence*. If we can understand the mechanism of human intelligence, investment may leave human hands, and most tradings may be conducted by computer. However, modern science still has a long way to go in understanding human intelligence. In fact, there are many things that are not yet understood even by organisms that are much simpler than humans, such as insects.

Looking at the human body at the microscopic level, it is composed of about 60 trillion cells, and cells are composed of molecules such as DNA and proteins, and furthermore, molecules are composed of atoms. *Quantum mechanical forces* are at work between atoms, but at the microscopic scale (the level of cells and atoms), intelligence is seemingly non-existent. It is known that the definition of an organism must be "separated from the outside world by a membrane", "able to metabolize" and "able to replicate", but having all three does not necessarily mean that it is intelligent. What then is intelligence? One possibility is that when a large number of ignorant cells come together to form a complex network, the

result of the complex interactions may yield a life that has acquired the ability to learn, by being equipped with memory mechanisms and feedback mechanisms to respond to stress from the environment. Thus, this may have something to do with the human "mind".

In "*The Society of Mind*" [1], Marvin Minsky points out that the human "mind" is composed of a large number of ignorant agents, which acquire "mind" through their interaction. In other words, he points out that a collective network of ignorant agents produces intelligent agents, and when intelligent agents gather further, they become highly intelligent. This is very similar to a group of atoms, molecules, and cells that do not have a "mind" gathering together, and then the multicellular group gathers again and interacts with each other, eventually coming to have a "mind". In other words, a human being can be seen as a collection of various agents. Neural networks are simple mathematical models of neurons and their connections, and were proposed by Warren McCulloch and Walter Pitts [2] in 1943. The simplest model of neural networks is the *perceptron* that was developed by Frank Rosenblatt [3] in 1957. The perceptron inputs features to the input layer, multiplies the input by a weight corresponding to the strength of the connections between neurons. Finally, the output layer neurons pass the sum of these inputs through an activation function to produce the final result. However, the perceptron's inability to learn linearly inseparable problems was pointed out by Marvin Minsky and others, and it went into decline for a while. In 1986, David E. Rumelhart et al. [4] invented the *backpropagation* method, which made it possible to train a multi-layered perceptron and overcame the drawbacks of the previous method. However, it was difficult to prepare a large amount of training data at the time, and the backpropagation method alone was not successful because it over-trained the perceptron. With the advent of the *Dropout* method [5] and *ReLU* [6], these problems were solved with the help of a number of innovations and big data, and it became possible to increase the depth of the layers, giving birth to the technology known as *deep learning* [7]. In recent years, there have been significant developments in financial applications of deep learn-

ing models, such as stock price prediction models, automated trading models, and *natural language processing* models of financial texts.

In the field of financial time series, machine learning models, for example, stock prediction models, are often trained using only historical data and *back-tested* to evaluate the models. However, the behavior is unclear when data not in the historical data is given to the machine learning model. It is more serious when the machine learning model is overfitting the historical data. In this case, even if the difference between input data and the historical data is subtle, it would give the wrong output. There exist methods to avoid overfitting the historical data, for example, some of them are adding regularization terms such as *L1* and *L2* [8] or *data augmentation* [9]. Next, in cases that differ significantly from the historical data, such as the *Lehman* and the *Corona* shock, the predictions provided by the machine learning model will be completely unreliable. Not much has been done to address this problem in the financial field, but it might be solved if methods that do not use the historical data, such as *AlphaZero* [10], are applied.

An *artificial market* [11] is a virtual market that models mathematically multiple agents who maximize the output of their utility functions with different preferences, and on this basis these agents trade and it allows us to examine the micro dynamics of a financial market by virtually trading orders with multiple *agents* that have different preferences. This allows us to understand the mechanism of how micro-level order behavior leads to macro-level prices. However, there are various issues to be addressed, such as the problem of determining the number of agents, how to model agents, and how to determine agent parameters. This kind of method is called a *bottom-up approach*. It has been reported that artificial markets can be used to replicate macro-market prices by building up the micro-investor behavior of the agents, and can replicate the well-known statistical properties of real markets, known as "*stylized facts*". In addition, by simulating future market regulation in advance, the effects of regulation can be estimated. However, because it is very difficult to model agents as real investors, there is a large gap between reality and simulation though simulations can provide a range

of insights.

On the other hand, *imitation learning* or *inverse reinforcement learning* [12] methods are *top-down approaches* that allow agents to reflect data observed in the actual market. This is a method in which an agent modeled by machine learning learns behavioral rules of real data, which is called an expert policy. Furthermore, by extending this method to the *Generative Adversarial Network (GAN)* [13], it is possible to obtain expert policies without going through the reward function. In addition, there have been reports on extensions to *multi-agents* [14], but the application of this method to finance has not yet been reported. In an application of financial time series using GANs, Li et al. proposed a combination of artificial markets and the *Wasserstein GAN (WGAN)* [15], which is a GAN for solving optimal transportation problem [16]. They applied the WGAN to several U.S. stocks and showed that they could simulate them with high accuracy by comparing the statistical properties of real and synthetic data. In addition, *signature* [17] has been attracting attention as time series features in recent years. The signatures are feature that focus on the area of a time series, and It has been reported to be effective as a feature value of machine learning model. Furthermore, by combining the signatures and the GANs [18], time series have been successfully generated. However, no research has yet been done on data generation that integrates micro order-level phenomena to macro price level phenomena.

In recent years, research on the fusion of computer simulations with real data, as seen in *data assimilation* [19], has been active. In addition, with the development of deep learning, computer simulations are increasingly using deep learning models instead of physical models. In the field of finance, research has also been conducted to use deep learning models when simulating markets, for example, to emulate synthetic orders placed by virtual traders. However, it is fundamentally difficult to generate realistic price data.

1.2 Research motivation

We believe that the establishment of a method for realizing highly realistic artificial market simulation by successfully combining deep learning and *artificial market simulation* will be a major breakthrough in the field of financial data science. If such artificial market simulation becomes available, it will make a significant contribution to the fields of 1) *data augmentation*, 2) *algorithmic performance evaluation*, and 3) *algorithmic enhancement*, respectively. First, data augmentation can be achieved by generating a large amount of artificial data (e.g., time series of price data) by repeating the simulation of the market on a given day many times using an artificial market. Then we use the result as the training data of the forecast model. The addition of new artificial data in addition to past data is expected to avoid over-learning on past data and is expected to have a positive effect on regularization. Next is the performance evaluation of the algorithm, which has traditionally used historical order data to measure actual performance, but this has had the problem of not being able to accurately evaluate *market impact*. The last is the *reinforcement* of algorithm, and artificial markets can be used as an environment to reinforce new algorithm. This can be combined with reinforcement learning methods by using the artificial market as an environment. As mentioned above, the realization of highly realistic artificial market simulation is quite significant and valuable.

1.3 Purpose of research

The purpose of this study is to establish a method that can achieve highly realistic artificial market simulations. To do that achieve, this paper proposes a method for educating artificial markets using a top-down approach method with the *GAN*. As a first step, we focused on the micro order dynamics generated by traders and aimed to create an artificial market that can successfully reproduce them using the *WGAN*. We then transformed the order data generated by the *WGAN* into time series of price data, and verified whether it is effective as a data augmentation

to a 1-tick ahead forecasting model. In the second step, we tested the validity of signatures as features for macro price data. As a part of this, we attempted to extend the signatures to discrete paths. In the third step, we tested the effectiveness of the WGAN method, which generates micro dynamics, and the effectiveness of the *Sig-W GAN*, which uses the signatures described in the second step to generate macro price dynamics, as part of the third step. By coupling these two GANs, we aimed to realize a highly realistic artificial market.

1.4 Our contribution

We consider that we have made significant contributions in the following three points. 1) We devised an artificial market simulation that uses the WGAN to generate data on the micro order dynamics of investors, and this simulation is effective in extending data for forecasting 1-tick ahead stock price. 2) The signature used in the past was based on the assumption that the paths are continuous, but we have modified it so that it can handle discrete paths directly. This is a particularly important extension for financial time series. 3) We proposed a new method (*Micro-Macro GAN*) that couples two GANs for both micro and macro dynamics. This method can bring the artificial market simulation closer to a more realistic market by introducing the effect of long-term correlations.

1.5 Structure of thesis

Chapter 1 (this chapter) provides an overall description. In Chapter 2, we first focus on the micro dynamics of order data and discuss the effectiveness of order data generation and data augmentation using the WGAN. In Chapter 3, we focus on price changes, which are macro dynamics, and study the signatures, for obtaining an effective feature. We propose a discrete signature to apply to financial time series data and discuss its application to financial time series data. In Chapter 4, we propose a loss function using the WGAN for the micro dynamics

of order data and the signature for the macro dynamics of prices, and show the effectiveness of the Micro-Macro GAN as a data generation method for coupled micro and macro dynamics. In Chapter 5, we describe the overall discussion and conclusion.

Chapter 2

Data Augmentation of High Frequency Financial Data Using Generative Adversarial Network

2.1 Introduction

The stable functioning of financial markets is crucial to the management of the economy. However, any event that causes a small change in efficiency or stability, such as a major news shock or a regulatory change in the market, can have a very large impact. Successful development of mathematical models of financial markets can be simulated in advance using computers to help improve the design and regulation of financial markets, or to avoid crises when unexpected events occur. One approach to this is *artificial markets* [20, 21, 22].

An artificial market is a virtual market that mathematically models multiple investors who maximize their utility functions with different preferences, called *agents*, and on this basis these agents trade [20]. It has been reported that artificial markets can be used to replicate macro-market prices by building up the micro-investor behavior of the agents, and can replicate the well-known statistical properties of real markets, known as "*stylized facts*" [21]. In addition, by simulating future market regulation in advance, the effects of regulation can be estimated [22]. However, because it is very difficult to model agents as real investors, there is a large gap between reality and simulation though simulations can provide a range of insights. In practice, simulations that generate high fidelity

and realistic data are still lacking.

Generative Adversarial Networks (GANs) are a powerful and widely used method for creating realistic data [13]. GANs consist of two mechanisms called *generators* and *discriminators*. The generator generates fake data, and the discriminator is a classifier to distinguish the real data from the data generated by the generator. The generator learns by incorporating feedback from the discriminator and classifies the output of the discriminator as real. The discriminator, on the other hand, learns by feeding back its own discrimination results and classifies the output from the generator as both real and fake data. In this way, the generator and the discriminator can learn by competing with each other, so that the generator can eventually produce realistic data. In fact, there are many research reports that have succeeded in generating highly realistic data that is indistinguishable from the real thing even when viewed by humans [23]. Although GAN is primarily concerned with the creation of images, applications to financial time series have also been reported [24].

In recent years, some studies have incorporated data generated from GANs into training data for other models [25, 26, 27, 28]. M. F. Adar et al. [25] presented a data augmentation that uses GANs to generate synthetic medical images. Adding more data by data augmentation to the originally small amount of real data resulted in improved accuracy of the classification model. On the other hand, Tanaka et al. [26] showed that training a classifier using only the synthetic data of the original data set produced a more accurate classifier than training with the original data set. In an application of financial time series using GANs, Li et al. [29] proposed a combination of artificial markets and *Wasserstein GAN* (WGAN) [15, 30], which is a GAN for the optimal transportation problem [16]. They applied the WGAN to several U.S. stocks and showed that they could simulate them with high accuracy by comparing the statistical properties of real and synthetic data. Although research on data augmentation using GANs is ongoing, there is no research on data augmentation to high frequency financial data using GANs, yet.

The purpose of this study is to build a data augmentation technique to generate reality high-frequency financial data. As a first step, we constructed a framework for simulating artificial markets using GAN-generated high-frequency data. The target market is the cash stock market of the *Tokyo Stock Exchange* and we use *FLEX Full* [31] as high frequency financial data. We use WGAN, which has already been proven valid for financial time series data as a data generation method [29]. The GAN developed in this study consists of a convolutional neural network (CNN) and a *long-short term memory (LSTM)*. In order to demonstrate the effectiveness of the GAN, we used artificial market simulations using the GAN for data augmentation of execution prices. The data were used as training data for the LSTM-based model for predicting future stock price increases and decreases. Finally, we discuss the effectiveness of the forecasting model with and without data augmentation.

2.2 Prerequisite

Section 4.3 will explain a theoretical background of Wasserstein GAN.

2.2.1 Setting

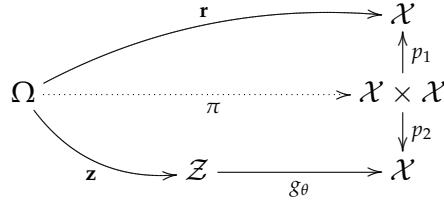
Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a fixed probability space, and $\bar{\mathcal{X}} := (\mathcal{X}, d_{\mathcal{X}})$ be a *Polish space*, that is, a complete separable metric space. Then, $\bar{\mathcal{X}}$ is considered as a measurable space $(\mathcal{X}, \mathcal{F}_{d_{\mathcal{X}}})$ equipped with a σ -field $\mathcal{F}_{d_{\mathcal{X}}}$ that is derived from a topology generated the distance $d_{\mathcal{X}}$.

Intuitively, the space $\bar{\mathcal{X}}$ shows a variety of possible aspects of the world we are thinking of.

More generally, $\bar{\mathcal{X}}$ may be considered as a (high-dimensional) *normed space*.

Definition 2.2.1. The *Reality* is a $\bar{\mathcal{X}}$ -valued random variable $\mathbf{r} : \Omega \rightarrow \mathcal{X}$.

The Reality \mathbf{r} shows the aspects that we do not have a direct way to know its complete description. It is like a concept of *population* in statistics theory. Our

Figure 2.2.1: \mathbf{r} and \mathbf{g}_θ

object is to seek an approximation of the probability distribution of \mathbf{r} by using a machine learning technique. However, an aspect space represented by a machine learning technique is not as rich as the real world $\tilde{\mathcal{X}}$. In the following, the aspect space represented by a machine learning method is considered as a Polish space $\tilde{\mathcal{Z}} := (\mathcal{Z}, d_{\mathcal{Z}})$ whose “dimension” is in general much lower than that of $\tilde{\mathcal{X}}$.

In Wasserstein GAN, we fix a $\tilde{\mathcal{Z}}$ -valued random variable $\mathbf{z} : \Omega \rightarrow \mathcal{Z}$, and then by providing a parameterized measurable function $g_\theta : \mathcal{Z} \rightarrow \mathcal{X}$, we update the parameters in order to approximate \mathbf{r} by

$$\mathbf{g}_\theta := g_\theta(\mathbf{z}) = g_\theta \circ \mathbf{z}. \quad (2.1)$$

The function g_θ is called a **generator**. The random variable \mathbf{z} is a “noise” with which the robustness of the generated artificial data is considered to be improved.

Please refer the following example for getting a concrete image for these setting.

Example 2.2.2. A probability distribution of a random variable $X : \Omega \rightarrow \mathcal{X}$ is written by $f_X : \mathcal{X} \rightarrow \mathbb{R}_+$.

1. $\mathcal{X} := \mathbb{R}^2$, $f_{\mathbf{r}}(x, y) := f_{N(\theta, 1)}(x) f_{U[0, 1]}(y)$,
2. $\mathcal{Z} := \mathbb{R}$, $\mathbf{z} \sim U[0, 1]$,
3. $g_\theta(y) := (\theta, y)$.

Next, we need a concept of “distance” for representing how close \mathbf{g}_θ to \mathbf{r} . By using this “distance”, we will implement a loss function for updating θ .

2.2.2 Wasserstein GAN

Definition 2.2.3. Let $\mathcal{X} \xleftarrow{p_1} \mathcal{X} \times \mathcal{X} \xrightarrow{p_2} \mathcal{X}$ be standard projections, and

$$X_1, X_2 : \Omega \rightarrow \mathcal{X}$$

be two \mathcal{X} -valued random variables.

1. The set of \mathcal{X}^2 -valued random variables $\Pi(X_1, X_2)$ is defined by

$$\begin{aligned} \Pi(X_1, X_2) := \{ \pi : \Omega \rightarrow (\mathcal{X}_1 \times \mathcal{X}_2, \mathcal{F}_{\mathcal{X}_1} \otimes \mathcal{F}_{\mathcal{X}_2}) \mid \\ p_1(\pi) \sim X_1 \cap p_2(\pi) \sim X_2 \}. \end{aligned} \quad (2.2)$$

2. A distance $W(X_1, X_2)$ between X_1 and X_2 is defined by

$$W(X_1, X_2) := \inf_{\pi \in \Pi(X_1, X_2)} \mathbb{E}[d_{\mathcal{X}}(p_1(\pi), p_2(\pi))]. \quad (2.3)$$

We call the distance a *Wasserstein distance*.

Note that the set $\Pi(X_1, X_2)$ is non-empty since it has at least an element (X_1, X_2) .

Originally, the Wasserstein distance was introduced in a context of the theory of optimal transportation problem [16]. Therefore, in the following we can develop our discussion in the context. However, we adopt a minimal concrete setting in this paper.

Definition 2.2.4. The set \mathcal{L}_1 is a set of all 1-Lipschitz measurable functions defined as following:

$$\begin{aligned} \mathcal{L}_1 := \{ f : \mathcal{X} \rightarrow \mathbb{R} \mid f \text{ is a measurable function} \cap \\ \forall x_1, x_2 \in \mathcal{X} . |f(x_1) - f(x_2)| \leq d_{\mathcal{X}}(x_1, x_2) \}. \end{aligned}$$

Theorem 2.2.5 (Kantorovich-Rubinstein's Duality: [15] Theorem 3). *The Wasserstein distance has a following representation.*

$$W(\mathbf{r}, \mathbf{g}_\theta) = \sup_{f \in \mathcal{L}_1} \left\{ \mathbb{E}[f(\mathbf{r})] - \mathbb{E}[f(\mathbf{g}_\theta)] \right\}. \quad (2.4)$$

Moreover, there exists a function $\tilde{f} \in \mathcal{L}_1$ that attains the contents of the absolute value in (2.4). Then, \tilde{f} satisfies

$$\nabla_\theta W(\mathbf{r}, \mathbf{g}_\theta) = -\mathbb{E}[\tilde{f}(\mathbf{g}_\theta)]. \quad (2.5)$$

Concretely speaking, we implement a function f by a neural network f_w , and then make it approach to the function \tilde{f} . The function f_w is called a **discriminator**.

Definition 2.2.6 (Wasserstein GAN). The **Wasserstein GAN** (or **WGAN**) is an algorithm for computing the following L .

$$L := \inf_{\theta} W(\mathbf{r}, \mathbf{g}_\theta) \doteq \inf_{\theta} \sup_{f_w \in \mathcal{L}_1} \left\{ \mathbb{E}[f_w(\mathbf{r})] - \mathbb{E}[f_w(\mathbf{g}_\theta)] \right\}. \quad (2.6)$$

In other words, Wasserstein GAN gives a method for obtaining optimal parameters w^* and θ^* by solving the min-max problem define in (2.6).

2.3 Method

2.3.1 Definition of events

Definition 2.3.1. [Order Events] An **order event** [32] is a quadruplet

$$\begin{aligned} (s, d, \Delta, v) &\in \{\text{MKT}, \text{LMT}, \text{CAN}\} \times \{\text{S}, \text{B}\} \\ &\times \{-L_{\min}, \dots, -1, 0, 1, \dots, L_{\max}\} \\ &\times \{100, 200, \dots, V_{\max}\}, \end{aligned}$$

where s denotes an order type (market order, limit order or cancel order), d denotes a trade direction (sell or buy), Δ denotes a price position (we set $L_{\max} = L_{\min} = 10$), and v denotes an order quantity (we set $V_{\max} = 2000$).

Definition 2.3.2. [Market Events] Assume that we have N_D order events

$$\{o_i := (s_i, b_i, \Delta_i, v_i)\}_{i=0,1,\dots,N_D-1}$$

on a day.

1. Let t_i be a time when the order event o_i occurs.
2. Let a_i , b_i and w_i be a best ask price, a best bid price, and a tick width at time t_i , respectively. a_i , b_i and w_i are defined as the values after the order at time t_i is reflected in the LOB.
3. An *market event* is a pentad

$$x_i := (t_i, o_i, a_i, b_i, w_i)$$

.

4. The limit price p_i of the market event x_i is defined by

$$p_i := \begin{cases} b_i + w_i \Delta_i & (d_i = S), \\ a_i - w_i \Delta_i & (d_i = B). \end{cases} \quad (2.7)$$

5. The bid-ask spread sp_i of the market event x_i is defined by

$$sp_i := (a_i - b_i) / w_i.$$

6. For $i > 0$,

$$\Delta t_i := t_i - t_{i-1} \in \mathbf{R}_+.$$

7. Let k be a fixed positive integer. For $i = k, k+1, \dots, N_D$ a *historical market event* at i is a sequence

$$H_{i-k:i} := \{x_j | i-k \leq j < i\}.$$

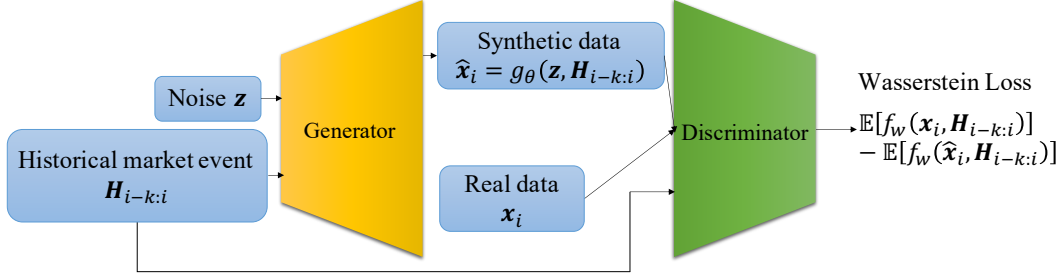


Figure 2.3.1: Architecture of GAN

We set $k = 20$ in this paper.

2.3.2 Architecture of GAN

Figure 2.3.1 shows an architecture of *GAN*, which contains two networks: a *generator* and a *discriminator*.

There are two inputs going to the generator network. The first is the noise vector $\mathbf{z} \in \mathbb{R}^{n_z}$ where n_z set to 100, and is generated using a uniform distribution $U[-1, 1]^{n_z}$ in this study. The second is the historical market event $\mathbf{H}_{i-k:i}$. The generator eventually outputs a synthetic market event

$$\hat{\mathbf{x}}_i := g_\theta(\mathbf{z}, \mathbf{H}_{i-k:i}).$$

On the other hand, the input of the discriminator network is either the synthetic market event $\hat{\mathbf{x}}_i$ or the real market event \mathbf{x}_i , and the discriminator outputs a scalar value

$$y_i := f_w(\mathbf{x}_i, \mathbf{H}_{i-k:i}).$$

Then the *Wasserstein distance* is calculated from their expected values.

2.3.3 Generator network g_θ

Figure 2.3.2 shows an architecture of the *generator* network. The generator learns a conditional probability $\mathbb{P}_r(\mathbf{x}_i | \mathbf{H}_{i-k:i})$. Firstly, the generator passes the historical market event $\mathbf{H}_{i-k:i}$ to the *Long Short Term Memory (LSTM)* layer as input,

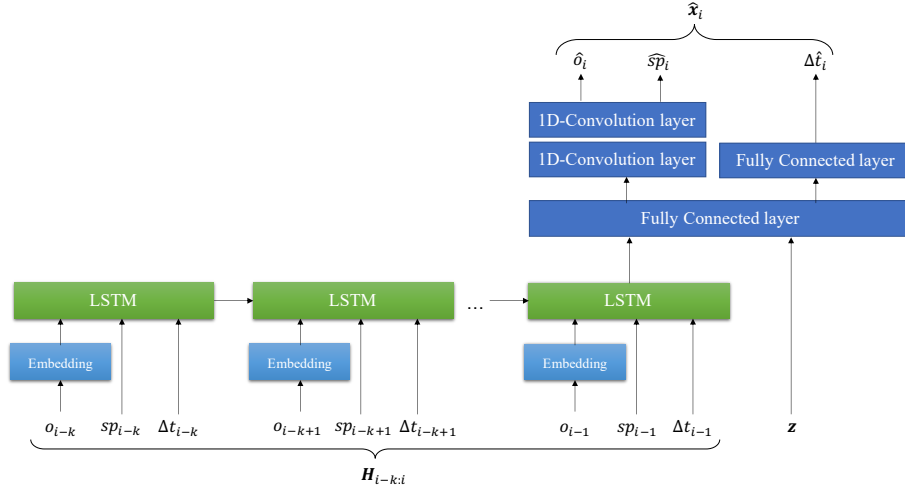


Figure 2.3.2: Generator network

where the order event IDs, o_i is represented as a one-hot vector and converted to a real vector e_i by an embedding layer, namely, $e_i = \text{Embedding}(o_i)$, where *Embedding* indicates the *embedding layer*. Secondly, the LSTM outputs a 1D feature vector, which is combined with the noise vector \mathbf{z} , and then the combined vector is passed to the *fully connected layers*. Finally, by using a two-layered 1D *convolutional neural network* (1D-CNN) and the fully connected layers, a next synthetic market event \hat{x}_i is generated.

2.3.4 Discriminator network f_w

Figure 2.3.3 shows an architecture of the *discriminator* network. In the same way as the generator in Figure 2.3.2, the discriminator also passes the historical market event $H_{t-k:i}$ to the *LSTM* layer as input. The synthetic market event \hat{x}_i or the real market event x_i is combined with the output vectors of the LSTM, passing to the *fully connected layer*. Furthermore, the output vector is used as the input to a two-layered 1D-CNN, and finally a scalar value y_i is obtained as an output by the fully connected layer.

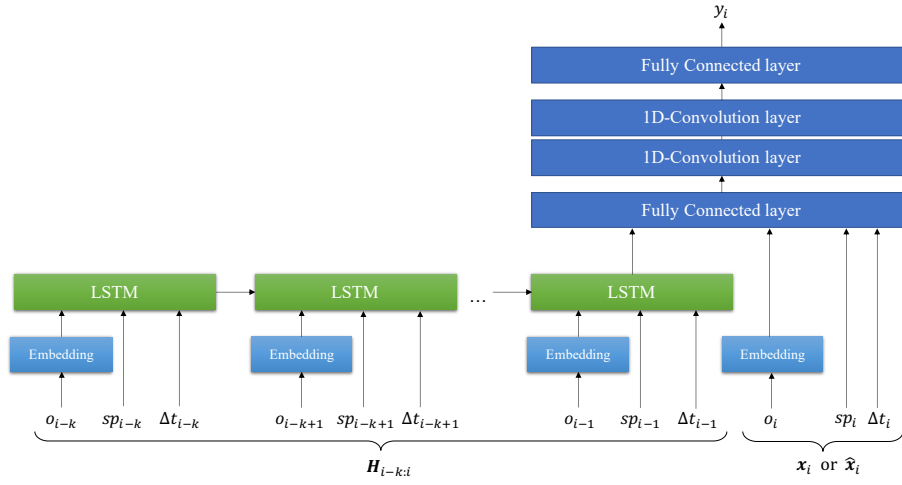


Figure 2.3.3: Discriminator network

2.3.5 Algorithm

The learning algorithm of the GAN is summarized in Algorithm 1. The number of discriminator iterations per generator iteration n_{dis} is set to 5, and we use WGAN with *gradient penalty* algorithm [30] to enforce 1-Lipschitz constraint on the discriminator and *Adam* algorithm [33] for updating parameters w and θ .

2.4 Data

The *TOPIX*, also known as the *Tokyo Stock Price Index*, is a capitalization weighted index of all companies listed on the First Section of the *Tokyo Stock Exchange*. The *Corona Shock* caused the TOPIX to fall sharply in February 2020 and March 2020 (Figure 2.4.1). As an in-sample period, we chose January 2020, before the Corona Shock occurred. To demonstrate the effectiveness of data augmentation by the GAN, as an out-of-sample period, we chose February 2020 - May 2020 including the period of the Corona Shock. The target stocks are the TOPIX CORE 30, which consists of 30 stocks with particularly high market capitalization and liquidity among all the stocks on the First Section of the Tokyo Stock Exchange, as of October 31, 2019. Table 2.4.1 shows the average of the number of executions,

Algorithm 1 We use default values of $\lambda = 0.1$, $N = 100000$, $n_{dis} = 5$, $m = 32$, $\alpha = 0.001$, $\beta_1 = 0.5$, $\beta_2 = 0.9$, $k = 20$, and $n_z = 100$ (default).

Require: The gradient penalty coefficient λ , the number of discriminator iterations per generator iteration n_{dis} , the batch size m , Adam hyperparameters α , β_1 , β_2 , initial discriminator parameter w_0 , initial generator parameter θ_0 , real data size S , history length k , and noise dimension n_z .

```

for  $n = 1, \dots, N$  do
  for  $t = 1, \dots, n_{dis}$  do
    for  $b = 1, \dots, m$  do
       $i \leftarrow$  random integer in  $[0, S)$ 
      Sample real data  $\{\mathbf{x}_i, H_{i-k:i}\} \sim \mathbb{P}_r$ , random variable  $\mathbf{z} \sim U[-1, 1]^{n_z}$ ,  $\epsilon \sim U[0, 1]$ .
       $\tilde{\mathbf{x}}_i \leftarrow g_\theta(\mathbf{z}, H_{i-k:i})$ 
       $\hat{\mathbf{x}}_i \leftarrow \epsilon \mathbf{x}_i + (1 - \epsilon) \tilde{\mathbf{x}}_i$ 
       $L^{(b)} \leftarrow f_w(\tilde{\mathbf{x}}, H_{i-k:i}) - f_w(\mathbf{x}, H_{i-k:i}) + \lambda(\|\nabla_{\hat{\mathbf{x}}} f_w(\hat{\mathbf{x}}, H_{i-k:i})\|_2 - 1)^2$ 
    end for
     $L \leftarrow \frac{1}{m} \sum_{b=1}^m L^{(i)}$ 
     $w \leftarrow \text{Adam}(\nabla_w L, w, ff, fi_1, fi_2)$ 
  end for
  Sample a batch real data  $\{\mathbf{x}_i^{(b)}, H_{i-k:i}^{(b)}\} \sim \mathbb{P}_r$ , a batch of random variables  $\{\mathbf{z}^{(b)}\} \sim U[-1, 1]^{n_z}$ 
   $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{b=1}^m -f_w(g_\theta(\mathbf{z}^{(b)}, H_{i-k:i}^{(b)}), w, ff, fi_1, fi_2))$ 
end for

```

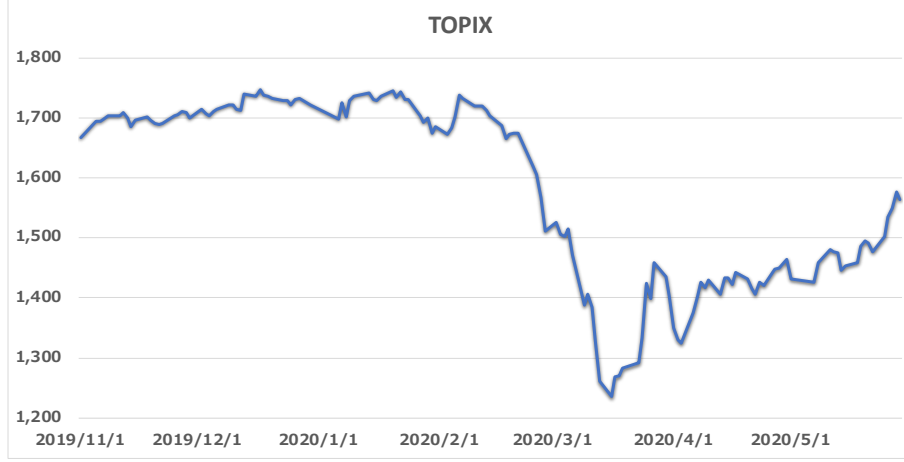


Figure 2.4.1: Topix from November 2019 to May 2020

the number of orders, the number of volumes, and the amount of trading for the TOPIX CORE 30 in January 2020. In order to build the GAN model for trading times, as time zone, we chose 9:10 to 11:20 or 12:40 to 14:50. The FLEX Full data was converted to the order events o_i for $i = 0, 1, \dots, N_D - 1$ as shown in Table 2.4.2, and the limit order books were constructed from the order events. Finally, we obtained the market events x_i for $i = 0, 1, \dots, N_D - 1$ from these data. For training the GAN, we used *Amazon Web Service's Elastic Compute Cloud (EC2)* to build a high-performance virtual environment in the cloud [34]. We prepared six virtualization environments and learned the GAN by computing for about two days.

2.5 Result

2.5.1 Comparison of real and synthetic data generated by the GAN

Using the market events for the TOPIX CORE 30 (Training period: 6 January 2020 to 31 January 2020), we conducted the learning of the GAN for each stock by applying the algorithm 1. First, the probability distribution of order events o_i generated by the GAN was compared with the real data. Figure 2.5.1 shows two stocks, Takeda Pharmaceutical Company (4502) and KDDI (9433), which (a) and

Table 2.4.1: Average number of daily trades, number of orders, volume, and value of trades for the TOPIX CORE 30 in January 2020

Code	Corporation	Executions	Orders	Volume	Amount
2914	JT	4977	100424	4814973	11,419,512,328
3382	7&I HD	3521	59272	1901452	7,900,263,468
4063	Shin-Etsu Chem	2538	50622	1057300	13,026,381,236
4452	Kao	3631	50293	1146010	10,354,898,073
4502	Takeda	4719	90479	3445115	14,918,853,710
4503	Astellas	4817	82134	4744215	8,951,999,315
4568	Daiichi Sankyo	4581	70093	1480463	10,777,673,315
6098	Recruit HD	5071	118355	3453047	14,709,340,121
6501	Hitachi	4904	96909	3117205	14,056,219,842
6758	Sony	12805	258010	7128531	55,782,578,747
6861	Keyence	2261	32365	488263	19,085,349,578
6954	Fanuc	2899	54124	831742	16,936,117,394
6981	Murata	7833	133742	3198900	21,281,004,178
7203	Toyota	6524	177461	4509815	34,791,232,942
7267	Honda	6085	104400	3818731	11,362,196,507
7751	Canon	3804	74100	3466610	10,434,161,300
7974	Nintendo	4235	65574	1019126	43,454,054,842
8031	Mitsui	3775	76353	3726252	7,307,010,197
8058	Mitsubishi	5258	90001	4096331	11,756,207,302
8306	MUFG	11409	176964	38506526	22,159,210,992
8316	SMFG	4346	94812	3925273	15,473,888,236
8411	Mizuho FG	4924	77808	71090500	11,738,384,326
8766	Tokyo Marine	2747	47302	1279900	7,759,158,326
8802	Mitsubishi Estate	5561	92416	4072468	8,713,773,465
9020	JRE	2627	49989	840626	8,299,716,463
9022	JRC	1092	22990	272126	5,941,747,763
9432	NTT	4034	73093	3595031	10,057,936,515
9433	KDDI	4081	71734	4107547	13,427,893,231
9437	NTT Docomo	3356	58879	3379300	10,429,018,673
9984	SBG	15102	274085	12684663	60,466,736,100

(b) indicate the real data, and (c) and (d) indicate the synthetic data for the top 30 with the highest frequency. Several events generated by the GAN did not appear in the top 30 of the real data, however, we find that the order type, the order quantity, the order direction and the order position are similar. In the same way, the probability distribution of the difference between current and previous order arrival time generated by the GAN was compared with the real data. The real data (a) and (b) in Figure 2.5.2 was close to Poisson's distribution in both cases. In addition, we extracted the distribution of the order type (LMT, MKT, CAN) and the order direction (Buy/Sell) from the distribution of order events (Figure 2.5.3). The synthetic distribution was almost same as the real distribution from Figure 2.5.3. From the above, the events generated from the GAN were close to the real ones and It showed that the GAN was able to successfully learn the real data.

Table 2.4.2: Example of a sequence for order events

Order event number i	Time t_i	Order event $o_i = (s_i, d_i, \Delta_i, v_i)$
23435	10:10:00.095024	CAN:S:1:200
23436	10:10:00.210141	MKT:S:0:700
23437	10:10:00.211171	CAN:B:3:100
23438	10:10:00.211664	CAN:B:2:100
23439	10:10:00.212166	LMT:S:2:100
23440	10:10:00.212589	LMT:S:2:100
23441	10:10:00.212982	CAN:B:3:100
23442	10:10:00.213446	CAN:B:2:300
23443	10:10:00.213911	LMT:S:2:100
23444	10:10:00.214334	CAN:B:2:200
23445	10:10:00.214765	LMT:S:2:200
23446	10:10:00.215137	CAN:B:3:300
23447	10:10:00.215594	LMT:S:2:500

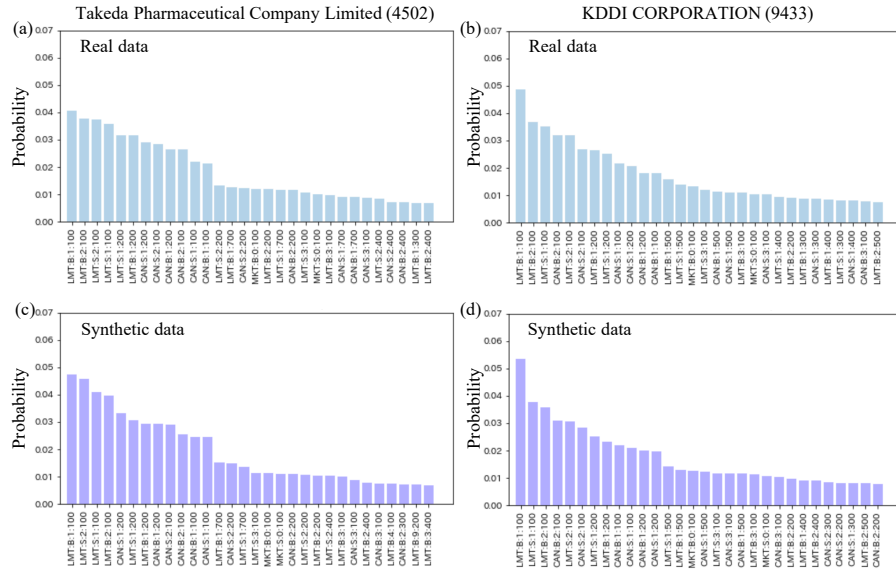


Figure 2.5.1: Comparison of probability distributions of order events generated by the GAN

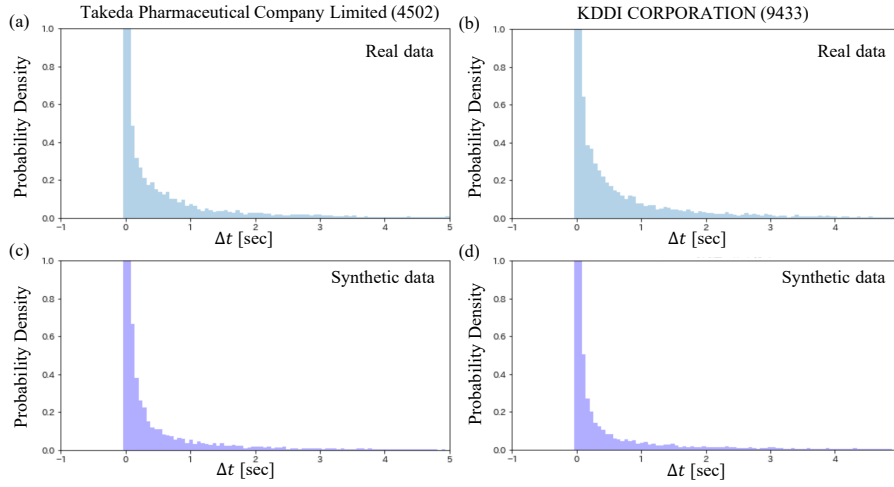


Figure 2.5.2: Comparison of probability distributions of the difference between current and previous order arrival time Δt generated by the GAN

2.5.2 Artificial market simulation with GAN

We have developed a program that constructs the limit order book (LOB) from the sequence of market events generated by the GAN and makes virtual executions. From this program, it is possible to simulate the market artificially by reflecting the output to input of the GAN (Figure 2.5.4). Figure 2.5.5 shows the time evolution of real and artificial execution price from 9:10 am on January 10, 2020 as initial condition. In this way, the GAN can semi-infinitely increase the execution prices, and in addition to the real data, the data generated by the GAN can be used as training data of another machine learning models. This is called Data Augmentation. An example of improving the accuracy of the model by the data augmentation with GANs is reported [25, 26].

2.5.3 Verification of the effectiveness for data augmentation with GAN

We performed the data augmentation of the execution prices with the GAN. The verification was done using the LSTM model, which is generally used as a time series model, to determine whether the execution price will up or down in the future. In the training data of the LSTM, the execution prices data that are the same

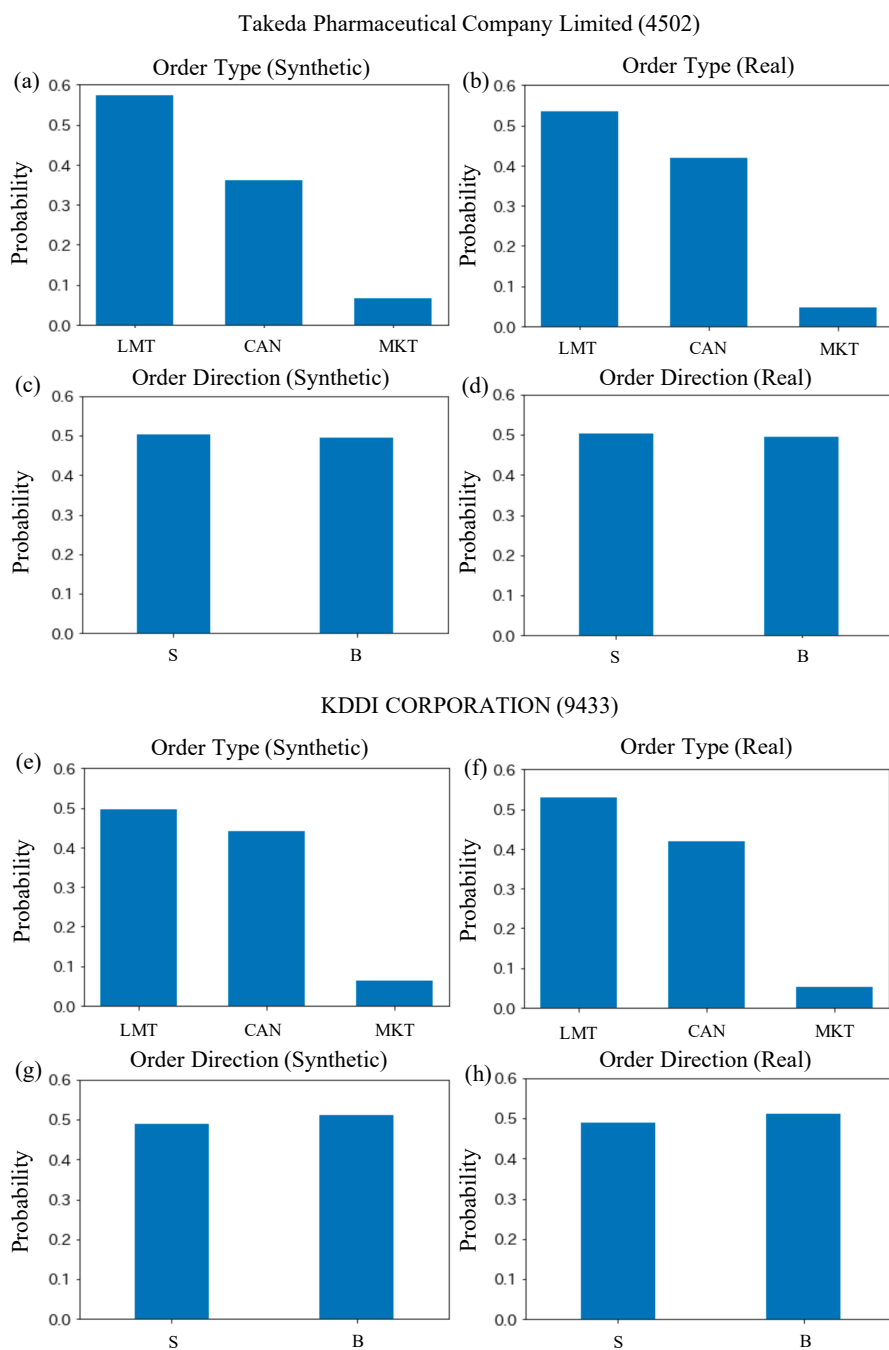


Figure 2.5.3: Comparison of probability distributions of the order type and the order direction generated by the GAN

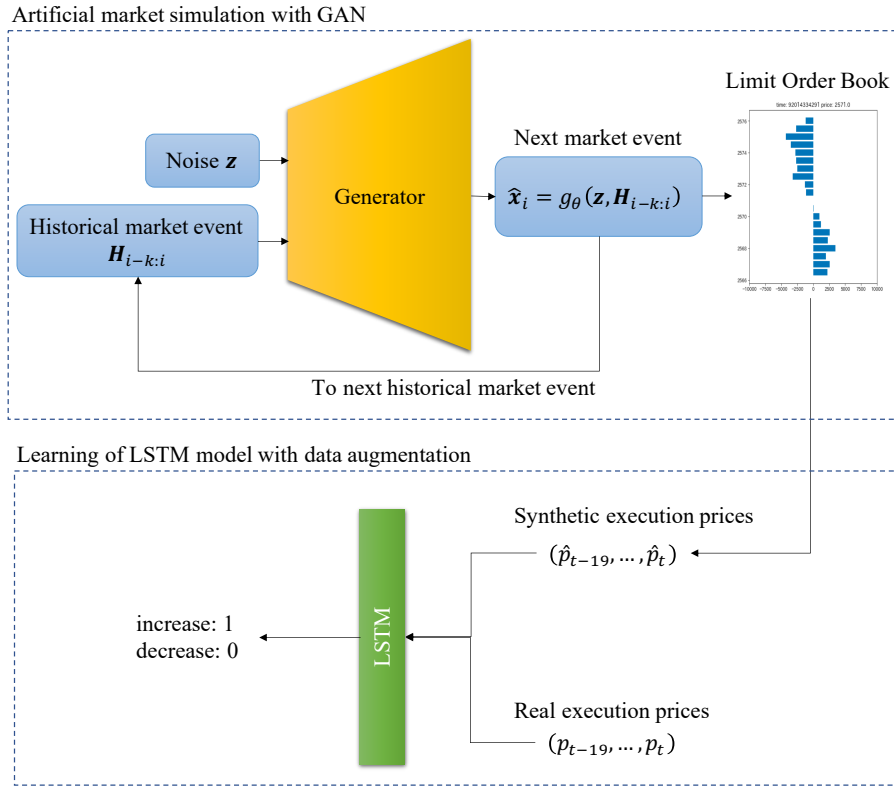


Figure 2.5.4: Daigram of artificial market simulation with GAN

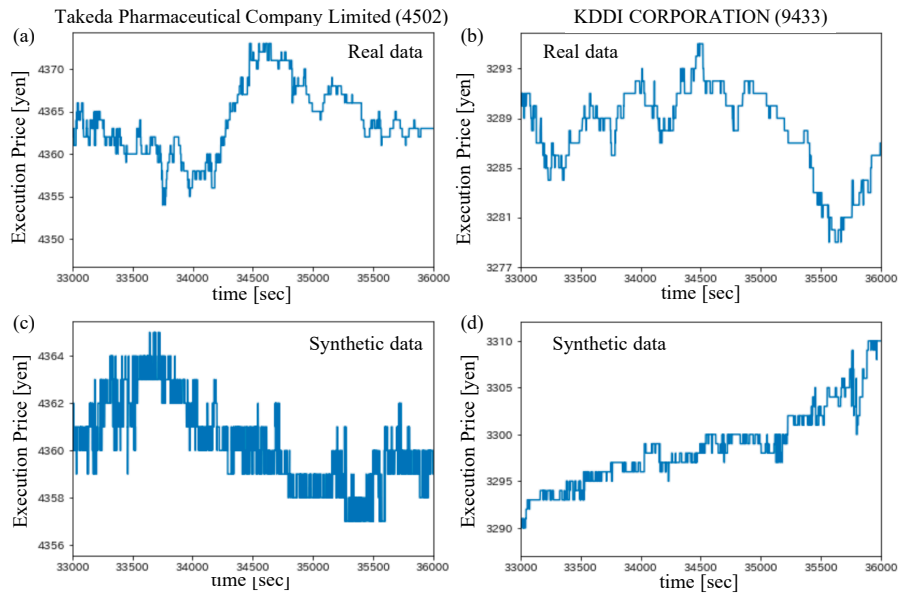


Figure 2.5.5: Time evolution of real and artificial execution price

as the previous and the current execution price from these data was removed, therefore only that was necessarily changed. The input for the LSTM is the last 20 execution price data and the output is 1 if the price is up and 0 if it is down. In-sample data was used during January 2020, as in the case of GAN, and the training data and the test data were split 7 to 3, and the early stopping to get the best model was used to terminate the training. We selected the best model by the early stopping as the final model. In addition, this final model is applied to the out-of-sample period (January - May 2020).

We investigated how much it affects accuracy of the LSTM model for each stocks with or without the data augmentation. First, Table 2.5.1 shows the results of training without the data augmentation. January's result represents the accuracy in the test data while the results for February through May represent the accuracy in the out-of-sample. In most stocks, the accuracy of the LSTM is largely reduced due to the Corona Shock (February - March 2020). Next, Table 2.5.2 shows the result with the data augmentation. The number of the real data and the data augmented by the GAN were added in a 1:10 ratio. The data augmentation was performed by randomly selecting the initial LOB of the morning or the afternoon session within the in-sample, and then at the end of the morning or the afternoon session. Even with the data augmentation, the accuracy is similarly worse than without the data augmentation in the case of the Corona Shock. The result suggests that the data augmentation may not include a Corona-Shock-like event, and this problem will be solved in the future. Finally, the result of the difference in the accuracy with and without the data augmentation is shown in Table 2.5.3 where black and red numbers mean positive and negative, respectively. From this result, the accuracy with the data augmentation was improved more than without for many stocks. However, the accuracy of some of stocks such as Kao, Daiichi Sankyo, Keyence and others was decreased due to the data augmentation. This reason is thought to be because the data generated by the GAN is quite far from the reality. It may be improved with the learning of GAN. In addition, the accuracy for each stocks was increased significantly even under the

Table 2.5.1: Accuracy of increase/decrease for stock prices using LSTM model without data augmentation

	JT	7 & IHD	Shin-Etsu	Kao	Takeda	Astellas
January	0.706	0.501	0.572	0.528	0.591	0.516
February	0.645	0.495	0.536	0.521	0.570	0.508
March	0.534	0.496	0.471	0.535	0.542	0.503
April	0.559	0.491	0.507	0.525	0.584	0.503
May	0.595	0.504	0.554	0.519	0.587	0.514

	Daiichi Sankyo	Recruit HD	Hitachi	Sony	Keyence	Fanuc
January	0.550	0.549	0.508	0.567	0.504	0.508
February	0.548	0.548	0.509	0.541	0.497	0.500
March	0.555	0.484	0.489	0.491	0.500	0.506
April	0.544	0.484	0.492	0.511	0.500	0.504
May	0.552	0.519	0.504	0.542	0.503	0.507

	Murata	Toyota	Honda	Canon	Nintendo	Mitsui
January	0.516	0.571	0.514	0.648	0.550	0.608
February	0.487	0.529	0.501	0.565	0.544	0.605
March	0.481	0.479	0.487	0.510	0.494	0.519
April	0.491	0.513	0.490	0.515	0.525	0.525
May	0.498	0.555	0.494	0.563	0.532	0.536

	Mitsubishi	MUFG	SMFG	Mizuho FG	Tokyo Marine	Mitsubishi Estate
January	0.581	0.753	0.699	0.952	0.506	0.505
February	0.554	0.682	0.669	0.920	0.491	0.502
March	0.509	0.563	0.507	0.780	0.494	0.498
April	0.539	0.592	0.507	0.746	0.496	0.495
May	0.523	0.635	0.523	0.850	0.506	0.495

	JRE	JRC	NTT	KDDI	NTT Docomo	SBG
January	0.498	0.504	0.507	0.651	0.676	0.647
February	0.485	0.510	0.496	0.599	0.617	0.591
March	0.477	0.489	0.489	0.505	0.502	0.541
April	0.485	0.490	0.499	0.576	0.551	0.587
May	0.477	0.492	0.507	0.618	0.623	0.594

Corona Shock, such as Takeda, Mitsubishi UFJ, and Mizuho FG. As a reason for the improved accuracy, these results suggest that the data augmentation works as a regularization. While machine learning models typically prevent overfitting by adding regularization terms to the loss function, our results show that adding synthetic data can have the same effect.

2.5.4 Effect of noise dimension

Table 2.5.4 shows the effect of the dimension of the vector $\mathbf{z} \in \mathbb{R}^{n_z}$. The five selected stocks have lower prediction accuracy at $n_z = 100$. In addition to $n_z = 100$, we have tried $n_z = 5, 10, 20$, and 50 , and we see that $n_z = 10$ is particularly good. Considering the significance of the dimension effect of \mathbf{z} , we would expect it to be a parameter that controls the reality of the market events produced by the

Table 2.5.2: Accuracy of increase/decrease for stock prices using LSTM model with data augmentation

	JT	7 & I HD	Shin-Etsu	Kao	Takeda	Astellas
January	0.713	0.578	0.598	0.475	0.636	0.622
February	0.647	0.526	0.553	0.489	0.609	0.584
March	0.539	0.484	0.455	0.465	0.559	0.507
April	0.560	0.510	0.512	0.478	0.623	0.564
May	0.597	0.534	0.572	0.491	0.636	0.564

	Daiichi Sankyo	Recruit HD	Hitachi	Sony	Keyence	Fanuc
January	0.478	0.574	0.607	0.606	0.515	0.557
February	0.485	0.561	0.568	0.572	0.491	0.506
March	0.471	0.501	0.478	0.490	0.456	0.488
April	0.477	0.509	0.493	0.528	0.484	0.497
May	0.479	0.543	0.564	0.570	0.486	0.509

	Murata	Toyota	Honda	Canon	Nintendo	Mitsui
January	0.545	0.542	0.639	0.698	0.571	0.650
February	0.505	0.519	0.544	0.601	0.561	0.643
March	0.490	0.480	0.463	0.515	0.509	0.531
April	0.497	0.510	0.511	0.532	0.544	0.546
May	0.514	0.528	0.515	0.600	0.553	0.565

	Mitsubishi	MUFG	SMFG	Mizuho FG	Tokyo Marine	Mitsubishi Estate
January	0.621	0.789	0.729	0.956	0.540	0.592
February	0.588	0.726	0.699	0.927	0.527	0.555
March	0.509	0.601	0.520	0.812	0.488	0.505
April	0.563	0.644	0.517	0.790	0.492	0.504
May	0.546	0.687	0.534	0.867	0.567	0.547

	JRE	JRC	NTT	KDDI	NTT Docomo	SBG
January	0.550	0.539	0.622	0.691	0.716	0.698
February	0.534	0.473	0.554	0.633	0.651	0.638
March	0.517	0.464	0.532	0.515	0.507	0.570
April	0.525	0.460	0.538	0.615	0.574	0.627
May	0.516	0.496	0.621	0.658	0.654	0.636

generative model. That is, the lower dimensions are expected to generate different types of reality-appropriate events, while the higher dimensions are expected to output events with diluted and averaged features. To test this, we generated events with different generators of random number seeds using the LOB at 9:10 on January 20, 2020 as the initial condition. The results are shown in Figure 2.5.6. Each simulation was performed 100 times. The results show that the lower the dimension, the greater the variety of events that occur. From the above, it can be seen that the dimension of \mathbf{z} acts as a parameter to control the diversity of events.

Table 2.5.3: Difference accuracy with and without. data augmentation

	JT	7 & I HD	Shin-Etsu	Kao	Takeda	Astellas
January	0.007	0.077	0.026	(0.053)	0.045	0.105
February	0.002	0.031	0.016	(0.031)	0.039	0.075
March	0.005	(0.012)	(0.016)	(0.070)	0.017	0.003
April	0.002	0.020	0.004	(0.047)	0.039	0.061
May	0.002	0.029	0.018	(0.028)	0.048	0.051

	Daichi Sankyo	Recruit HD	Hitachi	Sony	Keyence	Fanuc
January	(0.071)	0.025	0.098	0.039	0.010	0.048
February	(0.063)	0.013	0.058	0.031	(0.006)	0.005
March	(0.084)	0.017	(0.012)	(0.001)	(0.043)	(0.018)
April	(0.067)	0.025	0.001	0.016	(0.016)	(0.007)
May	(0.072)	0.024	0.060	0.029	(0.018)	0.002

	Murata	Toyota	Honda	Canon	Nintendo	Mitsui
January	0.029	(0.029)	0.125	0.050	0.021	0.042
February	0.018	(0.011)	0.042	0.036	0.017	0.038
March	0.009	0.001	(0.024)	0.005	0.015	0.012
April	0.006	(0.003)	0.021	0.018	0.019	0.022
May	0.016	(0.026)	0.021	0.037	0.021	0.029

	Mitsubishi	MUFG	SMFG	Mizuho FG	Tokyo Marine	Mitsubishi Estate
January	0.040	0.036	0.030	0.004	0.034	0.087
February	0.033	0.044	0.030	0.007	0.036	0.053
March	0.001	0.038	0.013	0.032	(0.006)	0.007
April	0.025	0.052	0.010	0.043	(0.005)	0.009
May	0.024	0.052	0.011	0.018	0.061	0.052

	JRE	JRC	NTT	KDDI	NTT Docomo	SBG
January	0.051	0.035	0.115	0.039	0.040	0.051
February	0.049	(0.038)	0.058	0.035	0.035	0.046
March	0.040	(0.025)	0.043	0.010	0.005	0.029
April	0.039	(0.030)	0.040	0.039	0.023	0.040
May	0.040	0.004	0.114	0.040	0.031	0.042

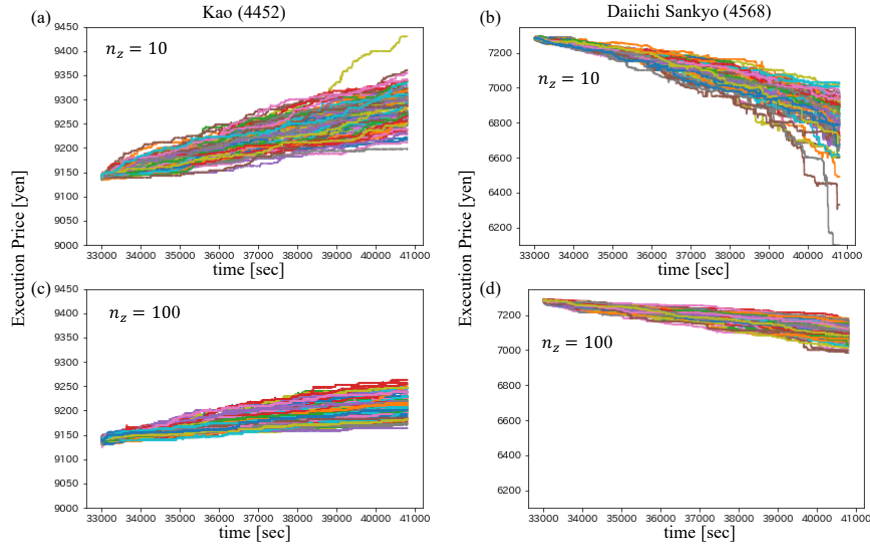
Figure 2.5.6: Simulation results for $n_z = 10$ and $n_z = 100$ using the LOB at 9:10 on January 20, 2020 as the initial condition: the case of Kao and Daiichi Sankyo.

Table 2.5.4: Comparison of accuracy when changing noise dimension

Corp	Month	Number of z dimensions				
		5	10	20	50	100
Kao	January	0.529	0.533	0.522	0.536	0.475
	February	0.524	0.531	0.517	0.528	0.489
	March	0.514	0.530	0.509	0.526	0.465
	April	0.517	0.523	0.510	0.522	0.478
	May	0.520	0.534	0.514	0.527	0.491
	Average	0.521	0.530	0.514	0.528	0.480
Daiichi Sankyo	January	0.547	0.556	0.557	0.546	0.478
	February	0.535	0.551	0.544	0.530	0.485
	March	0.539	0.557	0.545	0.542	0.471
	April	0.529	0.544	0.528	0.526	0.477
	May	0.541	0.556	0.546	0.532	0.479
	Average	0.538	0.553	0.544	0.535	0.478
Toyota	January	0.607	0.608	0.601	0.609	0.542
	February	0.557	0.556	0.553	0.558	0.519
	March	0.477	0.483	0.482	0.480	0.480
	April	0.535	0.535	0.533	0.535	0.510
	May	0.583	0.583	0.575	0.584	0.528
	Average	0.552	0.553	0.549	0.553	0.516
Keyence	January	0.523	0.523	0.526	0.513	0.515
	February	0.505	0.512	0.501	0.499	0.491
	March	0.484	0.485	0.469	0.483	0.456
	April	0.496	0.505	0.489	0.499	0.484
	May	0.502	0.506	0.493	0.502	0.486
	Average	0.502	0.506	0.495	0.499	0.486
JRC	January	0.544	0.529	0.541	0.541	0.539
	February	0.476	0.470	0.473	0.481	0.473
	March	0.467	0.471	0.470	0.472	0.464
	April	0.464	0.476	0.468	0.476	0.460
	May	0.494	0.498	0.503	0.499	0.496
	Average	0.489	0.489	0.491	0.494	0.486

2.6 Conclusion

In this study, we applied the artificial market simulation with the GAN using FLEX Full of the TOPIX CORE 30 as the high frequency financial data and obtained the execution prices from the result of simulations for each stocks. The prediction model was trained to predict the increase/decrease of stock prices using the LSTM model and investigated how much it affects the accuracy of the LSTM model for each stocks with or without the data augmentation. As a result, using the data augmentation, there are many models whose accuracy are

improved better than models trained with only real data. We further investigated the effect of noise dimension and found that the output results of the GAN differ significantly depending on the difference of noise dimension. In order to clarify this, a theoretical approach to the dependence of noise dimension will be needed in the future.

Overall, we conclude that the artificial market simulation of high frequency financial data with the GAN could be more than worth a try.

Chapter 3

Discrete Signature and its Application to Finance

3.1 Introduction

The signature, one of the key concepts of rough path theory, is recently considered as a means to find an appropriate feature set in machine learning systems [36]. It may become a powerful tool when combining with traditional machine learning techniques such as deep learning. In this paper, we introduce a new concept called discrete signatures, and apply it to some financial problems.

In Section 3.2, we introduce a concept of flat discrete signatures that is a simple discretization of the traditional signatures defined in [17], but with the **head-tail** transformation that is an enlargement method of the underlying alphabet set. We show that the **head-tail** transformation, just like the lead-lag transformation of streams, provides the quadratic variation of any component of the original process. This is important since the quadratic variation has a high relevance in financial applications. When applying flat discrete signatures to time-series analysis, we often encounter the necessity of treating data closer to the present time as more important than older data. In order to address this problem, we generalize flat discrete signatures to reflect the fact. The resulting version is called discrete signatures.

In Section 3.3, we will make a brief explanation about how we implement the signatures. Actually, an implementation of signatures was made by Patrick

Kidger and Terry Lyons as a Python-usable library called Signatory workable with PyTorch, which is written in C++ [38]. We will present yet another, but a very simple implementation using Python by adopting discrete signatures.

In Section 3.4, as an example of applications of discrete signature to finance, we consider the problem of judging whether a given price-shares process is of the morning or of the afternoon session in Tokyo Stock Exchange. We make a logistic regression with components of discrete signatures as features or explanatory variables. Then we will see that our result is as good as the regression with the whole raw data set with much fewer data points.

3.2 Discrete Signature

Throughout this paper, we fix the discrete time domain

$$\mathcal{T} := \{t_0, t_1, t_2, \dots\} \quad (3.1)$$

with

$$0 = t_0 < t_1 < \dots < t_n < t_{n+1} < \dots$$

and a *discrete* path X in \mathbb{R}^d for some fixed positive integer d , which can be written like

$$\begin{array}{ccc} \mathcal{T} & \xrightarrow{X} & \mathbb{R}^d \\ \Psi & & \Psi \\ t & \longmapsto & X_t = (X_t^1, \dots, X_t^d). \end{array} \quad (3.2)$$

Definition 3.2.1. [Word]

$$I := \{1, 2, \dots, d\}, \quad (3.3)$$

$$I^* := \bigcup_{k=0}^{\infty} I^k. \quad (3.4)$$

We call an element of I an *alphabet* and an element of I^* a *word* or a *multi-index*.

The unique element of I^0 is denoted by λ , which is the word with length 0, or

the *empty word*.

The *concatenation* of two words $u \in I^j$ and $v \in I^k$, denoted by $u \otimes v$, is the word $w \in I^{j+k}$ defined by for $i \in \{1, 2, \dots, j+k\}$,

$$w_i := \begin{cases} u_i & \text{if } 1 \leq i \leq j, \\ v_{i-j} & \text{if } j+1 \leq i \leq j+k, \end{cases} \quad (3.5)$$

where w_i stands for $w(i)$.

We usually focus a finite subset of I^* such as

$$I^{\leq k} := \bigcup_{\ell=0}^k I^\ell. \quad (3.6)$$

First, we will see the traditional definition of (continuous) signatures.

Definition 3.2.2. [Signature [17]] Let \mathbb{R}_+ be the continuous time domain starting from 0, I be an alphabet set and $\tilde{X} : \mathbb{R}_+ \rightarrow \mathbb{R}^I$ be a path. Let $a, b \in \mathbb{R}_+$ with $a < b$.

1. For $w \in I^*$, $S(\tilde{X})_{a,b}^w \in \mathbb{R}$ is defined inductively by

$$S(\tilde{X})_{a,b}^\lambda := 1, \quad (3.7)$$

$$S(\tilde{X})_{a,b}^{w \otimes i} := \int_a^b S(\tilde{X})_{a,t}^w d\tilde{X}_t^i \quad (\text{for } i \in I), \quad (3.8)$$

where

$$d\tilde{X}_t^i := \dot{\tilde{X}}_t^i dt. \quad (3.9)$$

2. The (traditional) *signature* of \tilde{X} over $[a, b]$ is a function $S(\tilde{X})_{a,b} : I^* \rightarrow \mathbb{R}$ defined by

$$S(\tilde{X})_{a,b}(w) := S(\tilde{X})_{a,b}^w \quad (3.10)$$

for $w \in I^*$.

Because we have a discrete path $X : \mathcal{T} \rightarrow \mathbb{R}^d$, we have to convert X to an appropriate continuous time path \tilde{X} before computing its signature.

One of the natural ways to accomplish this is an interpolation. If we adopt the linear interpolation to fill the values between t_n and t_{n+1} , we have for $t_n \leq t < t_{n+1}$

$$\tilde{X}_t^i := \frac{X_{t_n}^i(t_{n+1} - t) + X_{t_{n+1}}^i(t - t_n)}{t_{n+1} - t_n}. \quad (3.11)$$

Then for $t_n \leq t < t_{n+1}$,

$$\dot{\tilde{X}}_t^i = \frac{X_{t_{n+1}}^i - X_{t_n}^i}{t_{n+1} - t_n}. \quad (3.12)$$

Therefore, for $m, n \in \mathbb{N}$ with $m < n$, and $i \in I$,

$$\begin{aligned} S(\tilde{X})_{t_m, t_n}^{w \otimes i} &= \int_{t_m}^{t_n} S(\tilde{X})_{t_m, t}^w \dot{\tilde{X}}_t^i dt = \sum_{\ell=m}^{n-1} \int_{t_\ell}^{t_{\ell+1}} S(\tilde{X})_{t_m, t}^w \dot{\tilde{X}}_t^i dt \\ &= \sum_{\ell=m}^{n-1} \frac{X_{t_{\ell+1}}^i - X_{t_\ell}^i}{t_{\ell+1} - t_\ell} \int_{t_\ell}^{t_{\ell+1}} S(\tilde{X})_{t_m, t}^w dt = \sum_{\ell=m}^{n-1} \frac{X_{t_{\ell+1}}^i - X_{t_\ell}^i}{t_{\ell+1} - t_\ell} \tilde{S}_\ell(t_{\ell+1} - t_\ell) \\ &= \sum_{\ell=m}^{n-1} \tilde{S}_\ell(\tilde{X}_{t_{\ell+1}}^i - \tilde{X}_{t_\ell}^i), \end{aligned}$$

for some value \tilde{S}_ℓ that satisfies

$$\tilde{S}_\ell \in \{S(\tilde{X})_{t_m, s}^w \mid t_\ell \leq s \leq t_{\ell+1}\} \quad (3.13)$$

by the mean-value theorem.

Note that some of candidates of \tilde{S}_ℓ are

$$S(\tilde{X})_{t_m, t_\ell}^w, \quad \frac{1}{2}(S(\tilde{X})_{t_m, t_\ell}^w + S(\tilde{X})_{t_m, t_{\ell+1}}^w), \quad S(\tilde{X})_{t_m, t_{\ell+1}}^w. \quad (3.14)$$

Definition 3.2.3. For the alphabet set I , we define the *extended* alphabet set \bar{I} by

$$\bar{I} := I \times \{-, +\}. \quad (3.15)$$

For an alphabet $i \in I$, we call the extended alphabets $i^- := (i, -) \in \bar{I}$ and $i^+ := (i, +) \in \bar{I}$ the **head** and the **tail** of i , respectively.

In the following definition, we will assign the first and the last candidates in

(3.14) to **heads** and **tails**.

Definition 3.2.4. [Flat Discrete Signature] Let I be an alphabet set, $X : \mathcal{T} \rightarrow \mathbb{R}^I$ be a discrete path, and $m, n \in \mathbb{N}$ with $m < n$.

1. For $w \in \bar{I}^*$ and $i \in I$, $S(X)_{t_m, t_n}^w \in \mathbb{R}$ is defined inductively by

$$S(X)_{t_m, t_n}^\lambda := 1, \quad (3.16)$$

$$S(X)_{t_m, t_n}^{w \otimes i^-} := \sum_{\ell=m}^{n-1} S(X)_{t_m, t_\ell}^w (X_{t_{\ell+1}}^i - X_{t_\ell}^i), \quad (3.17)$$

$$S(X)_{t_m, t_n}^{w \otimes i^+} := \sum_{\ell=m}^{n-1} S(X)_{t_m, t_{\ell+1}}^w (X_{t_{\ell+1}}^i - X_{t_\ell}^i). \quad (3.18)$$

2. The **flat discrete signature** of X over $[t_m, t_n]$ is a function $S(X)_{t_m, t_n} : \bar{I}^* \rightarrow \mathbb{R}$ defined by for $w \in \bar{I}^*$,

$$S(X)_{t_m, t_n}(w) := S(X)_{t_m, t_n}^w. \quad (3.19)$$

Proposition 3.2.5. For $i, i_1, i_2, i_3 \in I$, $* \in \{-, +\}$ and $m, n \in \mathbb{N}$ with $m < n$,

$$S(X)_{t_m, t_n}^{i^*} = \sum_{m \leq \ell < n} (X_{t_{\ell+1}}^i - X_{t_\ell}^i) = X_{t_n}^i - X_{t_m}^i, \quad (3.20)$$

$$S(X)_{t_m, t_n}^{i_1^* \otimes i_2^-} = \sum_{m \leq \ell_1 < \ell_2 < n} (X_{t_{\ell_1+1}}^{i_1} - X_{t_{\ell_1}}^{i_1})(X_{t_{\ell_2+1}}^{i_2} - X_{t_{\ell_2}}^{i_2}), \quad (3.21)$$

$$S(X)_{t_m, t_n}^{i_1^* \otimes i_2^+} = \sum_{m \leq \ell_1 \leq \ell_2 < n} (X_{t_{\ell_1+1}}^{i_1} - X_{t_{\ell_1}}^{i_1})(X_{t_{\ell_2+1}}^{i_2} - X_{t_{\ell_2}}^{i_2}), \quad (3.22)$$

$$S(X)_{t_m, t_n}^{i_1^* \otimes i_2^- \otimes i_3^-} = \sum_{m \leq \ell_1 < \ell_2 < \ell_3 < n} (X_{t_{\ell_1+1}}^{i_1} - X_{t_{\ell_1}}^{i_1})(X_{t_{\ell_2+1}}^{i_2} - X_{t_{\ell_2}}^{i_2})(X_{t_{\ell_3+1}}^{i_3} - X_{t_{\ell_3}}^{i_3}), \quad (3.23)$$

$$S(X)_{t_m, t_n}^{i_1^* \otimes i_2^- \otimes i_3^+} = \sum_{m \leq \ell_1 < \ell_2 \leq \ell_3 < n} (X_{t_{\ell_1+1}}^{i_1} - X_{t_{\ell_1}}^{i_1})(X_{t_{\ell_2+1}}^{i_2} - X_{t_{\ell_2}}^{i_2})(X_{t_{\ell_3+1}}^{i_3} - X_{t_{\ell_3}}^{i_3}), \quad (3.24)$$

$$S(X)_{t_m, t_n}^{i_1^* \otimes i_2^+ \otimes i_3^-} = \sum_{m \leq \ell_1 \leq \ell_2 < \ell_3 < n} (X_{t_{\ell_1+1}}^{i_1} - X_{t_{\ell_1}}^{i_1})(X_{t_{\ell_2+1}}^{i_2} - X_{t_{\ell_2}}^{i_2})(X_{t_{\ell_3+1}}^{i_3} - X_{t_{\ell_3}}^{i_3}), \quad (3.25)$$

$$S(X)_{t_m, t_n}^{i_1^* \otimes i_2^+ \otimes i_3^+} = \sum_{m \leq \ell_1 \leq \ell_2 \leq \ell_3 < n} (X_{t_{\ell_1+1}}^{i_1} - X_{t_{\ell_1}}^{i_1})(X_{t_{\ell_2+1}}^{i_2} - X_{t_{\ell_2}}^{i_2})(X_{t_{\ell_3+1}}^{i_3} - X_{t_{\ell_3}}^{i_3}). \quad (3.26)$$

Proof. Straightforward. □

You may notice the correspondence between $\{-, +\}$ and $\{<, \leq\}$ in the ranges of summations in Proposition 3.2.5.

Example 3.2.6. Suppose that we observed 2 dimensional data in Table 3.2.1 with $I = \{1, 2\}$.

Table 3.2.1: Input data stream

t	0	1	1.5	2.5	3
X^1	1	3		5	8
X^2	1	4	2		6

We will fill the missing data in in Table 3.2.1 with their latest values like the data in Table 3.2.2.

Table 3.2.2: Filled data stream

t	0	1	1.5	2.5	3
X^1	1	3	3	5	8
X^2	1	4	2	2	6

Then, the initial segment of the signature $S(X)_{0,3}$ whose words length is less than or equal to 2, has the following values, where $*$ $\in \{-, +\}$.

$$S(X)_{0,3}^\lambda = 1,$$

$$S(X)_{0,3}^{1*} = 7, \quad S(X)_{0,3}^{2*} = 5,$$

$$S(X)_{0,3}^{1*1-} = 16, \quad S(X)_{0,3}^{1*1+} = 33, \quad S(X)_{0,3}^{1*2-} = 12, \quad S(X)_{0,3}^{1*2+} = 30,$$

$$S(X)_{0,3}^{2*1-} = 5, \quad S(X)_{0,3}^{2*1+} = 23, \quad S(X)_{0,3}^{2*2-} = -2, \quad S(X)_{0,3}^{2*2+} = 27.$$

In [37], the quadratic variation of any component of the original process X is provided by introducing the lead-lag transformation of streams. Since the

quadratic variation has a high relevance in financial applications, this result was crucial.

The following theorem shows that our **head-tail** transformation also provides a similar functionality.

Theorem 3.2.7. For $i \in I$, $* \in \{-, +\}$ and $m, n \in \mathbb{N}$ with $m < n$,

$$S(X)_{t_m, t_n}^{i* \otimes i^+} - S(X)_{t_m, t_n}^{i* \otimes i^-} = \sum_{m \leq \ell < n} (X_{t_{\ell+1}}^i - X_{t_\ell}^i)^2. \quad (3.27)$$

Proof. By (3.21) and (3.22), we have

$$\begin{aligned} & S(X)_{t_m, t_n}^{i* \otimes i^+} - S(X)_{t_m, t_n}^{i* \otimes i^-} \\ &= \sum_{m \leq \ell_1 \leq \ell_2 < n} (X_{t_{\ell_1+1}}^i - X_{t_{\ell_1}}^i)(X_{t_{\ell_2+1}}^i - X_{t_{\ell_2}}^i) - \sum_{m \leq \ell_1 < \ell_2 < n} (X_{t_{\ell_1+1}}^i - X_{t_{\ell_1}}^i)(X_{t_{\ell_2+1}}^i - X_{t_{\ell_2}}^i) \\ &= \sum_{m \leq \ell_1 = \ell_2 < n} (X_{t_{\ell_1+1}}^i - X_{t_{\ell_1}}^i)(X_{t_{\ell_2+1}}^i - X_{t_{\ell_2}}^i) = \sum_{m \leq \ell < n} (X_{t_{\ell+1}}^i - X_{t_\ell}^i)^2. \end{aligned}$$

□

When applying signatures to time-series analysis, we often encounter the necessity of treating data closer to the present time as more important than older data. Let us think to generalize flat discrete signatures to reflect the fact.

Now for $m < n$, we can rewrite (3.18) as follows.

$$S(X)_{t_m, t_n}^{w \otimes i^+} = \sum_{\ell=m}^{n-1} (X_{t_{\ell+1}}^i - X_{t_\ell}^i) S(X)_{t_m, t_{\ell+1}}^w = S(X)_{t_m, t_{n-1}}^{w \otimes i^+} + (X_{t_n}^i - X_{t_{n-1}}^i) S(X)_{t_m, t_n}^w. \quad (3.28)$$

We can read (3.28) as “First $S(X)_{t_m, t_{n-1}}^{w \otimes i^+}$ is computed at time t_{n-1} , and then $(t_n - t_{n-1})$ later, $S(X)_{t_m, t_n}^w$ and $S(X)_{t_m, t_n}^{w \otimes i^+}$ are calculated using the (slightly outdated) $S(X)_{t_m, t_{n-1}}^{w \otimes i^+}$ ”.

Similarly, we can rewrite (3.17) as follows.

$$S(X)_{t_m, t_n}^{w \otimes i^-} = \sum_{\ell=m}^{n-1} (X_{t_{\ell+1}}^i - X_{t_\ell}^i) S(X)_{t_m, t_\ell}^w = S(X)_{t_m, t_{n-1}}^{w \otimes i^-} + (X_{t_n}^i - X_{t_{n-1}}^i) S(X)_{t_m, t_{n-1}}^w. \quad (3.29)$$

This time, we can read (3.29) as “First $S(X)_{t_m, t_{n-1}}^{w \otimes i^-}$ and $S(X)_{t_m, t_{n-1}}^w$ are computed at time t_{n-1} , and then $(t_n - t_{n-1})$ later, $S(X)_{t_m, t_n}^{w \otimes i^+}$ is calculated using the (slightly outdated) $S(X)_{t_m, t_{n-1}}^{w \otimes i^-}$ and $S(X)_{t_m, t_{n-1}}^w$ ”.

In the following definition, a generalized version of flat discrete signatures is defined by calculating the outdated terms with a weight of 1 or less, taking into account the elapsed time.

Definition 3.2.8. [discrete Signature] Let I be an alphabet set, $X : \mathcal{T} \rightarrow \mathbb{R}^I$ be a discrete path, $m, n \in \mathbb{N}$ with $m < n$, and $\mu \geq 0$.

1. For $w \in \bar{I}^*$ and $i \in I$, $S^\mu(X)_{t_m, t_n}^w \in \mathbb{R}$ is defined inductively by

$$S^\mu(X)_{t_m, t_n}^\lambda := 1, \quad (3.30)$$

$$S^\mu(X)_{t_m, t_m}^w := \begin{cases} 1 & \text{if } w = \lambda, \\ 0 & \text{otherwise,} \end{cases} \quad (3.31)$$

$$S^\mu(X)_{t_m, t_n}^{w \otimes i^-} := e^{-\mu(t_n - t_{n-1})} (S^\mu(X)_{t_m, t_{n-1}}^{w \otimes i^-} + (X_{t_n}^i - X_{t_{n-1}}^i) S^\mu(X)_{t_m, t_{n-1}}^w), \quad (3.32)$$

$$S^\mu(X)_{t_m, t_n}^{w \otimes i^+} := e^{-\mu(t_n - t_{n-1})} S^\mu(X)_{t_m, t_{n-1}}^{w \otimes i^+} + (X_{t_n}^i - X_{t_{n-1}}^i) S^\mu(X)_{t_m, t_{n-1}}^w. \quad (3.33)$$

2. The *discrete signature* of X with the decay rate μ over $[t_m, t_n]$ is a function

$S^\mu(X)_{t_m, t_n} : \bar{I}^* \rightarrow \mathbb{R}$ defined by for $w \in \bar{I}^*$,

$$S^\mu(X)_{t_m, t_n}(w) := S^\mu(X)_{t_m, t_n}^w. \quad (3.34)$$

Note that $S^0(X)_{t_m, t_n} = S(X)_{t_m, t_n}$.

Proposition 3.2.9. For $i, i_1, i_2 \in I$, $m, n \in \mathbb{N}$ with $m < n$, and $\mu > 0$,

$$S^\mu(X)_{t_m, t_n}^{i-} = \sum_{m \leq \ell < n} e^{-\mu(t_n - t_\ell)} (X_{t_{\ell+1}}^i - X_{t_\ell}^i), \quad (3.35)$$

$$S^\mu(X)_{t_m, t_n}^{i+} = \sum_{m \leq \ell < n} e^{-\mu(t_n - t_{\ell+1})} (X_{t_{\ell+1}}^i - X_{t_\ell}^i), \quad (3.36)$$

$$S^\mu(X)_{t_m, t_n}^{i_1^- \otimes i_2^-} = \sum_{m \leq \ell_1 < \ell_2 < n} e^{-\mu(t_n - t_{\ell_1})} (X_{t_{\ell_1+1}}^{i_1} - X_{t_{\ell_1}}^{i_1}) (X_{t_{\ell_2+1}}^{i_2} - X_{t_{\ell_2}}^{i_2}), \quad (3.37)$$

$$S^\mu(X)_{t_m, t_n}^{i_1^+ \otimes i_2^+} = \sum_{m \leq \ell_1 \leq \ell_2 < n} e^{-\mu(t_n - t_{\ell_1})} (X_{t_{\ell_1+1}}^{i_1} - X_{t_{\ell_1}}^{i_1}) (X_{t_{\ell_2+1}}^{i_2} - X_{t_{\ell_2}}^{i_2}), \quad (3.38)$$

$$S^\mu(X)_{t_m, t_n}^{i_1^+ \otimes i_2^-} = \sum_{m \leq \ell_1 < \ell_2 < n} e^{-\mu(t_n - t_{\ell_1+1})} (X_{t_{\ell_1+1}}^{i_1} - X_{t_{\ell_1}}^{i_1}) (X_{t_{\ell_2+1}}^{i_2} - X_{t_{\ell_2}}^{i_2}), \quad (3.39)$$

$$S^\mu(X)_{t_m, t_n}^{i_1^+ \otimes i_2^+} = \sum_{m \leq \ell_1 \leq \ell_2 < n} e^{-\mu(t_n - t_{\ell_1+1})} (X_{t_{\ell_1+1}}^{i_1} - X_{t_{\ell_1}}^{i_1}) (X_{t_{\ell_2+1}}^{i_2} - X_{t_{\ell_2}}^{i_2}). \quad (3.40)$$

Proof. By induction on n . □

We have a similar result as Theorem 3.2.7 for discrete signatures, which tells that discrete signatures can represent “weighted” quadratic variations. Actually, the result is a generalization of Theorem 3.2.7.

Theorem 3.2.10. For $i \in I$, and $m, n \in \mathbb{N}$ with $m < n$,

$$S^\mu(X)_{t_m, t_n}^{i- \otimes i^+} - S^\mu(X)_{t_m, t_n}^{i- \otimes i^-} = \sum_{m \leq \ell < n} e^{-\mu(t_n - t_\ell)} (X_{t_{\ell+1}}^i - X_{t_\ell}^i)^2, \quad (3.41)$$

$$S^\mu(X)_{t_m, t_n}^{i^+ \otimes i^+} - S^\mu(X)_{t_m, t_n}^{i^+ \otimes i^-} = \sum_{m \leq \ell < n} e^{-\mu(t_n - t_{\ell+1})} (X_{t_{\ell+1}}^i - X_{t_\ell}^i)^2. \quad (3.42)$$

Proof. The proof is exactly same as that of Theorem 3.2.7, by using Proposition 3.2.9. □

Example 3.2.11. Using the same data in Example 3.2.6, the initial segment of the discrete signature $S^\mu(X)_{0,3}$ with the decay rate $\mu = \log 2 \doteq 0.693$ (half-life = 1)

whose words length is less than or equal to 2, has the following values.

$$\begin{aligned}
S^\mu(X)_{0,3}^\lambda &= 1, \\
S^\mu(X)_{0,3}^{1-} &= 3.08, & S^\mu(X)_{0,3}^{1+} &= 4.91, & S^\mu(X)_{0,3}^{2-} &= 2.70, & S^\mu(X)_{0,3}^{2+} &= 4.04, \\
S^\mu(X)_{0,3}^{1-1-} &= 3.37, & S^\mu(X)_{0,3}^{1-1+} &= 11.65, & S^\mu(X)_{0,3}^{1-2-} &= 3.33, & S^\mu(X)_{0,3}^{1-2+} &= 12.56, \\
S^\mu(X)_{0,3}^{1+1-} &= 6.74, & S^\mu(X)_{0,3}^{1+1+} &= 19.57, & S^\mu(X)_{0,3}^{1+2-} &= 6.66, & S^\mu(X)_{0,3}^{1+2+} &= 20.16, \\
S^\mu(X)_{0,3}^{2-1-} &= -0.63, & S^\mu(X)_{0,3}^{2-1+} &= 8.61, & S^\mu(X)_{0,3}^{2-2-} &= -1.25, & S^\mu(X)_{0,3}^{2-2+} &= 12.19, \\
S^\mu(X)_{0,3}^{2+1-} &= 0.21, & S^\mu(X)_{0,3}^{2+1+} &= 13.71, & S^\mu(X)_{0,3}^{2+2-} &= -1.33, & S^\mu(X)_{0,3}^{2+2+} &= 18.34.
\end{aligned}$$

3.3 An implementation of discrete signature

In this section, we will make a brief description about an implementation of discrete signature with Python [35]. You can see the whole code **sig.py** and the data **sample1.dat** in Table 3.2.1 at <https://github.com/takanori-adachi/discrete-signature>.

Let us explain the functionality of classes in **sig.py** in the following subsections.

3.3.1 The class Data

Suppose we have a data stream like the following tab-separated records, which is corresponding to the data in Table 3.2.1.

; time	event_type	value
0.0	1	1.0
0.0	2	1.0
1.0	1	3.0
1.0	2	4.0
1.5	2	2.0
2.5	1	5.0
3.0	1	8.0
3.0	2	6.0

The class **Data** will perform the conversion from the above data stream to the filled data specified in Table 3.2.2. It reads the input stream (raw data) from a file and stores it into a list `self.raw_data`. Then, it collects the elements of I (the set of event types, `self.I`), \bar{I} (the set of extended event types, `self.barI`) and $\mathcal{T}(\text{timedomain}, \text{self.T})$, converting them into the internal integer values, and preparing dictionaries for the conversions. It finally creates I -dimensional discrete path X , or `self.X`.

The method `w2mi` converts a word to a list of integers representing alphabets, or elements of I containing in the word. Conversely, `mi2w` converts a list of integers to the corresponding word. The data member `t2i` is the dictionary converting from an actual time to its corresponding index.

3.3.2 The class Words

The class **Words** generates the set $I^{\leq k}$ as a list of its elements (words). The resulting list of elements of the set $I^{\leq k}$ is stored in the data member `self.Istar`.

In the flat case, i.e. when $\mu = 0$, we have

$$S(X)_{t_m, t_n}^{i^- \otimes w} = S(X)_{t_m, t_n}^{i^+ \otimes w} \quad (3.43)$$

for $i \in I$ and $w \in \bar{I}$ by Proposition 3.2.5. Therefore, we can identify i^- and i^+ for the first alphabet $i \in I$. So, we prepare a separate universe of words `self.IstarHalf` for the case $\mu = 0$.

3.3.3 The class Signature

A signature is initialized with a **Data** object `data` and the maximum length of words `k`. The class **Signature** encapsulate the heart of the computation of discrete signatures.

```

1 class Signature(object): # discrete signature
2     def __init__(self, data, k):
3         self.data = data
```

```

4     self.k = k # maximum length of words
5
6     def sig(self, t1, t2, w):
7         return(self.sig0(data.t2i[t1], data.t2i[t2], data.w2mi(w)))
8
9     def sig0(self, m, n, iss):
10        v = 1.0
11        if len(iss) > 0:
12            w = iss[: -1]
13            i = iss[len(iss) - 1]
14            j, s = self.i2js(i)
15            if s == 0: # HEAD
16                v = self.mu_delta_t[n-1] * (self.sig0(m, n-1, iss)
17                    + data.delta_X[n-1, j] * self.sig0(m, n-1, w))
18            else: # TAIL
19                v = self.mu_delta_t[n-1] * self.sig0(m, n-1, iss)
20                    + data.delta_X[n-1, j] * self.sig0(m, n, w)
21        return(v)

```

where $\text{mu_delta_t}[n]$ is $e^{-\mu(t_{n+1}-t_n)}$, and $\text{delta_X}[n-1, j]$ is a data member defined in the class **Data** as $X_{t_{n+1}}^i - X_{t_n}^j$. The function `i2js` converts a given index specifying an element of $\bar{I} = I \times \{+, -\}$ to a pair (j, s) where $j \in I$ and $s = 0$ if $i = j^-$, and $s = 1$ if $i = j^+$.

The function `sig` simply calls another function `sig0` after converting its arguments to corresponding internal representations. The function `sig0` is a straightforward implementation of equations (3.30), (3.31), (3.32) and (3.33). Note that it uses the recursive call technique.

This simple implementation, however, is not so efficient. In fact, in the recursive call of the function `sig0`, it repeats computations many times for the same arguments, which is simply a waste of time.

In order to avoid this extra computation, we will introduce a container object `Signature.v` for holding results of computation so far.

First, we introduce the container object `Words.v`. It consists of binary and

multinary tree structures. For each word $w \in \bar{I}^{\leq k}$, we have a pair

$$c_w := (b_w, r_w), \quad (3.44)$$

where r_w is the value of the signature at w , and b_w is a boolean value that indicates whether r_w has been calculated or not. The intermediate container v_w is defined by the following recursive definition.

$$\begin{aligned} v_w &:= (c_w, (v_{w \otimes i_1}, \dots, v_{w \otimes i_{\bar{d}}})) , & (\text{for } w \in \bar{I}^{\leq(k-1)}) \\ v_w &:= (c_w, ()) , & (\text{for } w \in \bar{I}^k) \end{aligned}$$

where \bar{d} is the cardinality of \bar{I} and $\{i_1, \dots, i_{\bar{d}}\} = \bar{I}$. Then, the container `Words.v` is defined by v_λ .

Next, we construct a container `Signature.v` which is a double list of `Words.v`. For each pair of time $(t_m, t_n) \in \mathcal{T}$ with $m < n$. The function `Signature.get_c` retrieves c_w from $v_{m,n}$ for the word w whose index is `iss`. Using `Signature.v`, the function `Signature.sig0` can be rewritten as:

```

1  def sig0(self, m, n, iss): # faster algorithm using container self.v
2      c = self.get_c(m, n, iss)
3      if c[0]: # if already computed
4          return c[1] # return its value
5      # otherwise, compute from scratch
6      v = 1.0
7      if len(iss) > 0:
8          w = iss[: -1]
9          i = iss[len(iss) - 1]
10         j, s = self.i2js(i)
11         if s == 0: # HEAD
12             v = self.mu_delta_t[n-1] * (self.sig0(m, n-1, iss)
13                 + data.delta_X[n-1, j] * self.sig0(m, n-1, w))
14         else: # TAIL
15             v = self.mu_delta_t[n-1] * self.sig0(m, n-1, iss)

```

```

16         + data.delta_X[n-1,j] * self.sig0(m, n, w)
17     c[0] = True # it is computed
18     c[1] = v # and its value is 'v'
19     return(v)

```

We will use this faster version in Section 3.4.

3.4 An application of discrete signature to finance

As an example of applications of discrete signature to finance, in this section, we consider the problem of judging whether a given price-shares process is of the morning or of the afternoon session in Tokyo Stock Exchange (TSE). TSE has morning (9:00-11:30) and afternoon (12:30-15:00) sessions each trading day. Therefore, each session has 2 hours and 30 minutes.

We use FLEX Full historical data bought from the Japan Exchange Group (JPX) as the raw data. FLEX Full data consists of high frequency tick data from which we can extract several micro dynamic data such as *ita* data or limit order book data. The time resolution of FLEX Full data is currently 1 microsecond, or 10^{-6} second. In the following, time is displayed in minutes. For example, "09:12:34.567890" is represented by the value $9 \times 60 + 12 + 34.567890/60 = 552.5761315$.

3.4.1 Make a one-minute interval data stream

We extract data stream

$$\mathcal{D} = \{D_t\} \quad (3.45)$$

from FLEX Full data, where t is an observed time in minutes, and, each D_t consists of the following five components:

$$\begin{aligned}
 D_t.P^a & - \text{ best ask price,} \\
 D_t.P^b & - \text{ best bid price,} \\
 D_t.S^a & - \text{ the total of ask side shares,} \\
 D_t.S^b & - \text{ the total of bid side shares,} \\
 D_t.V & - \text{ accumulated execution volume.}
 \end{aligned}$$

We will generate a substream of $\{D_t\}$ at one-minute interval for each trading session.

First, let us define index sets of one minute interval blocks from the original data by for $n \in \mathbb{N} := \{0, 1, 2, \dots\}$,

$$J_n := \{t \mid n \leq t < n+1 \text{ and } D_t \in \mathcal{D}\}, \quad (3.46)$$

$$\bar{J}_n := \{t \mid n \leq t \leq n+1 \text{ and } D_t \in \mathcal{D}\}. \quad (3.47)$$

Next, define pairs of times denoting open and close times of the session.

$$(N_0, N_1) \in \{(9 \times 60, 11.5 \times 60), (12.5 \times 60, 15 \times 60)\}, \quad (3.48)$$

$$N := N_1 - N_0 = 150. \quad (3.49)$$

If $D_{t_{\max J_{N_0}}}.V = 0$, i.e. the security had not been open in the first minute of the session, we do not use the session as data and throw it away. By assuming $D_{t_{\max J_{N_0}}}.V > 0$, we pick D_t for each $n = N_0, N_0 + 1, \dots, N_1$, which is called \bar{D}_n , by

the following procedure:

```

 $\bar{D}_{N_0} := D_{\min J_{N_0}}$ 
for  $n$  in range  $(N_0 + 1, N_1)$  :
    if  $J_{n-1} = \emptyset$  :  $\bar{D}_n := \bar{D}_{n-1}$ 
    else :  $\bar{D}_n := D_{\max J_{n-1}}$ 
if  $\bar{J}_{N_1-1} = \emptyset$  :  $\bar{D}_{N_1} := \bar{D}_{N_1-1}$ 
else :  $\bar{D}_{N_1} := D_{\max J_{N_1-1}}$ 

```

Then, we got a one-minute interval data stream

$$\{\bar{D}_n\}_{n=N_0, \dots, N_1} \quad (3.50)$$

for each session.

3.4.2 Time normalization

Since our problem is to detect time-related information of the given data stream, we will eliminate clues by normalizing the time. The followings are normalized time and its corresponding components. For $n = 0, 1, \dots, N$,

$$t_n := \frac{n}{N}, \quad (3.51)$$

$$P_{t_n}^a := \bar{D}_{N_0+n} \cdot P^a, \quad (3.52)$$

$$P_{t_n}^b := \bar{D}_{N_0+n} \cdot P^b, \quad (3.53)$$

$$S_{t_n}^a := \bar{D}_{N_0+n} \cdot S^a, \quad (3.54)$$

$$S_{t_n}^b := \bar{D}_{N_0+n} \cdot S^b, \quad (3.55)$$

$$V_{t_n} := \bar{D}_{N_0+n} \cdot V. \quad (3.56)$$

Then, our time domain is

$$\mathcal{T} := \{t_0, t_1, \dots, t_N\}. \quad (3.57)$$

3.4.3 Make a discrete path for each session

We introduce some other statistics. For $t \in \mathcal{T}$,

$$p_t := \ln \frac{P_t^a + P_t^b}{2}, \quad (\text{logarithm of mid-price}) \quad (3.58)$$

$$s_t := P_t^a - P_t^b. \quad (\text{spread}) \quad (3.59)$$

Next, we construct a discrete path

$$X := (X^1, X^2, X^3, X^4) : \mathcal{T} \rightarrow \mathbb{R}^I \quad (3.60)$$

with

$$I := \{1, 2, 3, 4\} \quad (3.61)$$

from which we will compute its discrete signature. For $t \in \mathcal{T}$,

$$X_t^1 := \frac{p_t - \langle p \rangle}{\sqrt{\langle p^2 \rangle - \langle p \rangle^2}}, \quad (\text{normalized logarithm of mid-price}) \quad (3.62)$$

$$X_t^2 := \frac{s_t - \langle s \rangle}{\sqrt{\langle s^2 \rangle - \langle s \rangle^2}}, \quad (\text{normalized spread}) \quad (3.63)$$

$$X_t^3 := \frac{S_t^a - S_t^b}{S_t^a + S_t^b}, \quad (\text{normalized imbalance}) \quad (3.64)$$

$$X_t^4 := \frac{V_t}{V_1}, \quad (\text{normalized accumulated volume}) \quad (3.65)$$

where $\langle x \rangle := \frac{1}{N} \sum_{t \in \mathcal{T}} x_t$ for any sequence $\{x_t\}_{t \in \mathcal{T}}$.

3.4.4 Experiment and result

In the experiment, we used data from January 2020 to July 2021 for 30 names in TOPIX CORE 30. After shuffling date, we use 80% of the whole data for training, and use 20% for test.

The calculated signature is used to determine the morning and afternoon sessions using logistic regression by which a binary decision was made, with 0 for the morning and 1 for the afternoon.

The set of event types or statistics is I defined in (3.61). We pick the seven sorts of feature sets as subsets of $\bar{I}^{\leq k}$, $\overline{\{1\}}^{\leq k}$, $\overline{\{2\}}^{\leq k}$, $\overline{\{3\}}^{\leq k}$, $\overline{\{4\}}^{\leq k}$, $\bar{I}^{\leq k}$ itself, $\overline{\{2,4\}}^{\leq k}$ and $\{w \in \bar{I}^{\leq k} \mid w \sim / [4^- 4^+] /\}$, where “ $w \sim / [4^- 4^+] /\$ ” means “ w matches the pattern $[4^- 4^+]$ ”. In other words, it means “ w contains the (extended) alphabets 4^- or 4^+ ”. We check these patterns for $k = 1, 2, 3$.

Table 3.4.1 shows the accuracy of logistic regression adopting members of the feature set as its explanatory variables.

Table 3.4.1: Computation with Signature

Feature set	Accuracy			Number of features		
	$k=1$	$k=2$	$k=3$	$k=1$	$k=2$	$k=3$
$\overline{\{1\}}^{\leq k}$	50.72%	55.46%	55.91%	1	3	7
$\overline{\{2\}}^{\leq k}$	72.51%	75.54%	83.08%	1	3	7
$\overline{\{3\}}^{\leq k}$	55.04%	58.96%	59.14%	1	3	7
$\overline{\{4\}}^{\leq k}$	90.18%	93.01%	97.46%	1	3	7
$\bar{I}^{\leq k}$	89.63%	98.86%	99.51%	4	36	292
$\overline{\{2,4\}}^{\leq k}$	89.58%	98.84%	99.82%	2	10	42
$\{w \in \bar{I}^{\leq k} \mid w \sim / [4^- 4^+] /\}$	90.18%	97.84%	99.55%	1	15	163

The statistics “4” (normalized cumulative volume) apparently made the best performance, and the statistics “2” (normalized spread) is next. That is why we tried the $\{2,4\}$ case and the last case that treats only words containing “4”. You may see that the values of the sixth and the last cases are better than that of the whole set $\bar{I}^{\leq k}$ case at $k = 3$, while the number of features of the $\{2,4\}$ case and the last case are much less than the whole set case.

Let us mention the computation speed of obtaining the signature in Table 3.4.1. The workstation we used for the computation has 2 CPUs. Each CPU has 48 cores, and each core can handle 2 threads. So, the total number of threads is 192, which is the number of affordable distributed parallel processing. We used 150 threads out of 192 for our computation in order to avoid overwhelming the

tasks of other users. The computation of all components of $\bar{I}^{\leq 4}$ of the signature took 68 minutes and 33.266 seconds.

In order to evaluate the result in Table 3.4.1 fairly, we also performed logistic regression using the raw data as it is without using signature as a comparison. The result is shown in Table 3.4.2.

Table 3.4.2: Computation without Signature

Statistics	Accuracy	Number of features
Normalized logarithm of mid-price	60.14%	151
Normalized spread	88.18%	151
Normalized imbalance	66.90%	151
Normalized cumulative volume	99.73%	151
All	99.64%	604

One of the most important points in the comparison is the number of features required to achieve good accuracy. For example, in $k = 3$ cases, the logistic regression using all raw 604 data points performs 99.64% accuracy while the logistic regression using 42 components of the discrete signature specified by $\overline{\{2,4\}}^{\leq 3}$ performs 99.82% accuracy which is slightly better than the former case. In other words, the regression with the feature set specified by the signature can achieve almost the same level of good results as the regression with the whole raw data set with much fewer data points.

3.5 Concluding remarks

We would like to leave a few remarks before finishing this paper.

The lead-lag transformation needs to double the cardinality n of the time domain \mathcal{T} while our **head-tail** transformation needs to double the cardinality d of the alphabet set I . Then, the ratio of computation times of these two methods will be $\frac{d^{2n}}{(2d)^n} = \left(\frac{d}{2}\right)^n$. Therefore, the lead-lag transformation will take more time than

ours when $d > 2$.

We used the pattern “[4⁻4⁺]” in Table 3.4.1 for specifying the subset of $\bar{I}^{\leq k}$. In general, a subset of I^* is called a language in Mathematical Language Theory [39]. There are some popular languages in this sense including regular languages and context-free languages. By modifying the class **Words** with the Python built-in library **re**, we can easily extend it to handle regular languages, i.e. languages generated by regular expressions. This gives us a more possibility to specify smaller and more appropriate feature sets instead of using whole $\bar{I}^{\leq k}$ whose cardinality is 292 when $k = 3$ in Section 3.4.

Chapter 4

Stock Market Simulation by Micro-Macro GAN

4.1 Introduction

In the field of financial time series, *machine learning* models, for example, stock prediction models, are often trained using only historical data and *back-tested* to evaluate the models. However, the behavior of the machine learning model when given data that is not in the historical data is unclear. It is more serious when the machine learning model is overfitting the historical data. In this case, even if the difference between input data and the historical data is subtle, it would give the wrong output. There exist methods to avoid overfitting the historical data, for example, some of them are adding regularization terms such as *L1* and *L2* [8] or *data augmentation* [9]. In cases that differ significantly from the historical data, such as the Lehman and the *Corona shock*, the predictions provided by the machine learning model will be completely unreliable. Not much has been done to address completely this problem in the financial field, but it may be solved if methods that do not use the historical data, such as *AlphaZero* [10], are applied.

Artificial markets are means to model financial markets on a computer [20, 21, 22] and allow us to examine the micro order dynamics by virtual multiple *agents* with different preferences. The artificial markets allow us to understand the mechanism of how micro-level order behavior leads to macro-level prices. However, there are various problems, such as how to determine the number

of agents, how to model agents, and how to determine agent parameters. This method of artificially modeling agents is called a *bottom-up approach*.

On the other hand, imitation learning or inverse reinforcement learning methods are *top-down approaches* that is methods to model agents to reflect data observed in the actual market and learn behavioral rules which is called an expert policy from the real data by machine learning. However, these methods require the calculation of the reward function at each learning step, which takes an enormous amount of computation time before learning converges. As a new method that incorporates the *Generative Adversarial Network (GAN)* [13] methodology, it is possible to obtain expert policies without going through the reward function [12]. In addition, there have been reports on extensions to multi-agents [14], but its application to finance has not yet been reported.

In an application of financial time series using GANs, Li et al. [29] proposed a combination of artificial markets and the *Wasserstein GAN (WGAN)* [15, 30], which is a GAN for solving optimal transportation problem [16]. They applied the WGAN to several U.S. stocks and showed that they could simulate them with high accuracy by comparing the statistical properties of real and synthetic data. In addition, signatures have been attracting attention as time series features in recent years [40]. The signatures are features that focus on the area of a time series, and it has been reported to be effective as a machine learning feature [41]. Furthermore, by combining the signatures and GANs, which is called *Sig-W GAN*, time series have been successfully generated [18]. However, no research has yet been done on data generation that integrates micro order dynamics to macro price dynamics.

In this study, we propose a new market simulation by machine learning model to generate synthetic price dynamics data from synthetic order dynamics data. The market simulation built by machine learning model, which is called *Micro-Macro GAN*, is trained by coupling two mechanisms. The first mechanism which is trained by using the Wasserstein Generative Adversarial Network (WGAN) generates micro order dynamics and is called the *Micro GAN*. The second mechanism which is trained by using the Sig-W GAN generates macro price dynamics

from the micro order dynamics generated by the Micro GAN and is called the *Macro GAN*. The price dynamics data is converted to the signature, and finally, the Sig-W [29] metric is computed as a loss function. As a demonstration, training was performed on Toyota Motor Corporation taken from the Flex Full data provided by *the Japan Exchange Group (JPX)* [31]. The time series data generated from the Micro GAN and the Micro-Macro GAN were compared to the real time series data. The results showed that the Micro-Macro GAN results were similar to the real time series data than the Micro GAN results.

4.2 Related work

As attempt to reproduce a real market using an artificial market, for example, K. Izumi et al. [42] constructed the *artificial market for foreign exchange* as a *multi-agent simulation* and showed that the rapid depreciation or appreciation of the yen in the 1995 could be reproduced in the multi-agent simulation by learning each agent who use a genetic algorithm which input news and economic indicators as features. There are *agent simulation* models with assumption that agents do not have intelligence. which is called a *zero-intelligence* model [43, 44, 45] and the orders are placed randomly by the agent. For example, S. Maslov [45] proposed a model of random order placement in a continuous double auction trading rule. As a result, he succeeded in reproducing statistical properties widely seen in financial markets, such as fat tails and volatility clustering, using only a zero-intelligence model and assumed that these statistical properties are characteristics produced by the trading rules of the market.

S. Vyetenko et al [46] focused on the artificial market simulation for training and testing machine learning models and proposed a framework for evaluating the simulation using statistical properties that should be satisfied as a realistic market. The framework is expected to be used as a robustness check and benchmark for *artificial market simulations* in the future.

In recent years, research has also been conducted on models that use the

GANs to generate the orders placed by traders [47, 29]. Li et al. [29] proposed *Stock-GAN* that uses the WGAN to generate synthetic orders. Time series data of price and volume generated by the synthetic order dynamics were compared some statistical properties with realistic, showing that they were close. A. Colletta et al. [47] constructed a synthetic market by representing a single generative model which is called *world-agent* that mimics multi-traders. It is possible to generate meaningful orders in response to experimental agents on the constructed synthetic market.

The Stock-GAN and the world-agent are similar to the Micro GAN, which supports the usage of the Micro GAN as a baseline against the Micro-Macro GAN. Their models do not include a mechanism for learning *long-term memory*, and it is fundamentally difficult to represent the macro dynamics such as the prices from the orders. In order to solve the problem, this paper introduces a mechanism to learn long-term memory by reflecting the price dynamics information to the generator using the Macro GAN.

4.3 Prerequisite

Section 4.3 will explain a theoretical background of Wasserstein GAN.

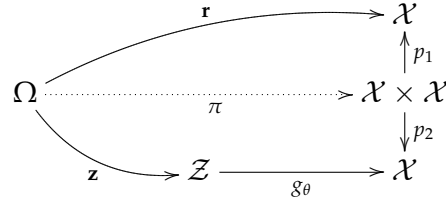
4.3.1 Setting

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a fixed probability space, and $\bar{\mathcal{X}} := (\mathcal{X}, d_{\mathcal{X}})$ be a *Polish space*, that is, a complete separable metric space. Then, $\bar{\mathcal{X}}$ is considered as a measurable space $(\mathcal{X}, \mathcal{F}_{d_{\mathcal{X}}})$ equipped with a σ -field $\mathcal{F}_{d_{\mathcal{X}}}$ that is derived from a topology generated by the distance $d_{\mathcal{X}}$.

Intuitively, the space $\bar{\mathcal{X}}$ shows a variety of possible aspects of the world we are thinking of.

More generally, $\bar{\mathcal{X}}$ may be considered as a (high-dimensional) *normed space*.

Definition 4.3.1. The *Reality* is a $\bar{\mathcal{X}}$ -valued random variable $\mathbf{r} : \Omega \rightarrow \mathcal{X}$.

Figure 4.3.1: \mathbf{r} and \mathbf{g}_θ

The Reality \mathbf{r} shows the aspects that we do not have a direct way to know its complete description. It is like a concept of *population* in statistics theory. Our object is to seek an approximation of the probability distribution of \mathbf{r} by using a machine learning technique. However, an aspect space represented by a machine learning technique is not as rich as the real world $\bar{\mathcal{X}}$. In the following, the aspect space represented by a machine learning method is considered as a Polish space $\bar{\mathcal{Z}} := (\mathcal{Z}, d_{\mathcal{Z}})$ whose “dimension” is in general much lower than that of $\bar{\mathcal{X}}$.

In Wasserstein GAN, we fix a $\bar{\mathcal{Z}}$ -valued random variable $\mathbf{z} : \Omega \rightarrow \mathcal{Z}$, and then by providing a parameterized measurable function $g_\theta : \mathcal{Z} \rightarrow \mathcal{X}$, we update the parameters in order to approximate \mathbf{r} by

$$\mathbf{g}_\theta := g_\theta(\mathbf{z}) = g_\theta \circ \mathbf{z}. \quad (4.1)$$

The function g_θ is called a *generator*. The random variable \mathbf{z} is a “noise” with which the robustness of the generated artificial data is considered to be improved.

Please refer the following example for getting a concrete image for these setting.

Example 4.3.2. A probability distribution of a random variable $X : \Omega \rightarrow \mathcal{X}$ is written by $f_X : \mathcal{X} \rightarrow \mathbb{R}_+$.

1. $\mathcal{X} := \mathbb{R}^2$, $f_{\mathbf{r}}(x, y) := f_{N(\theta, 1)}(x)f_{U[0, 1]}(y)$,
2. $\mathcal{Z} := \mathbb{R}$, $\mathbf{z} \sim U[0, 1]$,

$$3. g_\theta(y) := (\theta, y).$$

Next, we need a concept of “distance” for representing how close \mathbf{g}_θ to \mathbf{r} . By using this “distance”, we will implement a loss function for updating θ .

4.3.2 Wasserstein GAN

Definition 4.3.3. Let $\mathcal{X} \xleftarrow{p_1} \mathcal{X} \times \mathcal{X} \xrightarrow{p_2} \mathcal{X}$ be standard projections, and

$$X_1, X_2 : \Omega \rightarrow \mathcal{X}$$

be two \mathcal{X} -valued random variables.

1. The set of \mathcal{X}^2 -valued random variables $\Pi(X_1, X_2)$ is defined by

$$\begin{aligned} \Pi(X_1, X_2) := \{ \pi : \Omega \rightarrow (\mathcal{X} \times \mathcal{X}, \mathcal{F}_{\mathcal{X}} \otimes \mathcal{F}_{\mathcal{X}}) \mid \\ p_1(\pi) \sim X_1 \wedge p_2(\pi) \sim X_2 \}. \end{aligned} \quad (4.2)$$

2. A distance $W(X_1, X_2)$ between X_1 and X_2 is defined by

$$W(X_1, X_2) := \inf_{\pi \in \Pi(X_1, X_2)} \mathbb{E}[d_{\mathcal{X}}(p_1(\pi), p_2(\pi))]. \quad (4.3)$$

We call the distance a *Wasserstein distance*.

Note that the set $\Pi(X_1, X_2)$ is non-empty since it has at least an element (X_1, X_2) .

Originally, the Wasserstein distance was introduced in a context of the theory of optimal transportation problem [16]. Therefore, in the following we can develop our discussion in the context. However, we adopt a minimal concrete setting in this paper.

Definition 4.3.4. The set \mathcal{L}_1 is a set of all 1-Lipschitz measurable functions defined

as following:

$$\mathcal{L}_1 := \{f : \mathcal{X} \rightarrow \mathbb{R} \mid f \text{ is a measurable function such that} \\ \forall x_1, x_2 \in \mathcal{X} . |f(x_1) - f(x_2)| \leq d_{\mathcal{X}}(x_1, x_2)\}.$$

Theorem 4.3.5 (Kantorovich-Rubinstein's Duality: [15] Theorem 3). *The Wasserstein distance has a following representation.*

$$W(\mathbf{r}, \mathbf{g}_\theta) = \sup_{f \in \mathcal{L}_1} \left\{ \mathbb{E}[f(\mathbf{r})] - \mathbb{E}[f(\mathbf{g}_\theta)] \right\}. \quad (4.4)$$

Moreover, there exists a function $\tilde{f} \in \mathcal{L}_1$ that attains the contents of the absolute value in (4.4). Then, \tilde{f} satisfies

$$\nabla_\theta W(\mathbf{r}, \mathbf{g}_\theta) = -\mathbb{E}[\tilde{f}(\mathbf{g}_\theta)]. \quad (4.5)$$

Concretely speaking, we implement a function f by a neural network f_w , and then bring it closer to the function \tilde{f} . The function f_w is called a *discriminator*.

Definition 4.3.6 (Wasserstein GAN). The *Wasserstein GAN* (or *WGAN*) is an algorithm for computing the following L .

$$L := \inf_{\theta} W(\mathbf{r}, \mathbf{g}_\theta) \doteq \inf_{\theta} \sup_{f_w \in \mathcal{L}_1} \left\{ \mathbb{E}[f_w(\mathbf{r})] - \mathbb{E}[f_w(\mathbf{g}_\theta)] \right\}. \quad (4.6)$$

In other words, Wasserstein GAN gives a method for obtaining optimal parameters w^* and θ^* by solving the min-max problem define in (4.6).

4.3.3 Signature

Let \mathcal{T} be a time domain such as

$$\mathcal{T} := \mathbb{R}_+ := [0, \infty). \quad (4.7)$$

and $x = \{x_t\}_{t \in \mathcal{T}}$ be a path in \mathbb{R}^d for some fixed positive integer d , which can be written like

$$\begin{array}{ccc} \mathcal{T} & \xrightarrow{x} & \mathbb{R}^d \\ \Psi & & \Psi \\ t & \longmapsto & x_t = (x_t^1, \dots, x_t^d). \end{array} \quad (4.8)$$

Definition 4.3.7. [Word]

$$I := \{1, 2, \dots, d\}, \quad (4.9)$$

$$I^* := \bigcup_{k=0}^{\infty} I^k. \quad (4.10)$$

We call an element of I an *alphabet* and an element of I^* a *word* or a *multi-index*. The unique element of I^0 is denoted by λ , which is the word with length 0, or the *empty word*. The *concatenation* of two words $u \in I^j$ and $v \in I^k$, denoted by $u \otimes v$, is the word $w \in I^{j+k}$ defined by for $i \in \{1, 2, \dots, j+k\}$,

$$w_i := \begin{cases} u_i & \text{if } 1 \leq i \leq j, \\ v_{i-j} & \text{if } j+1 \leq i \leq j+k, \end{cases} \quad (4.11)$$

where w_i stands for $w(i)$.

We usually focus a finite subset of I^* such as

$$I^{\leq M} := \bigcup_{\ell=0}^M I^\ell. \quad (4.12)$$

First, we will see the traditional definition of signatures.

Definition 4.3.8. [Signature [17]] Let \mathbb{R}_+ be the continuous time domain starting from 0, I be an alphabet set and $x : \mathcal{T} \rightarrow \mathbb{R}^I$ be a path. Let $a, b \in \mathbb{R}_+$ with $a < b$.

1. For $w \in I^*$, $S(x)_{a,b}^w \in \mathbb{R}$ is defined inductively by

$$S(x)_{a,b}^\lambda := 1, \quad (4.13)$$

$$S(x)_{a,b}^{w \otimes i} := \int_a^b S(x)_{a,t}^w dx_t^i \quad (\text{for } i \in I), \quad (4.14)$$

where

$$dx_t^i := \dot{X}_t^i dt. \quad (4.15)$$

2. The **signature** of x over $[a, b]$ is a function $S(x)_{a,b} : I^* \rightarrow \mathbb{R}$ defined by

$$S(x)_{a,b}(w) := S(x)_{a,b}^w \quad (4.16)$$

for $w \in I^*$.

Note that the signature of a \mathbb{R}^I -valued stochastic process $\{X_t\}_{t \in \mathcal{T}}$ can be thought as a \mathbb{R} -valued (generalized) stochastic process

$$\{S(X)_{a,b}^w\}_{w \in I^*}. \quad (4.17)$$

Next, we introduce the Signature Wasserstein-1 (Sig-W) metric proposed by H. Li et al [18].

Definition 4.3.9. Let X_1, X_2 be \mathbb{R}^I -valued stochastic processes and M be an integer specifying the depth of the signature.

$$\text{Sig-W}^{(M)}(X_1, X_2) := \left(\sum_{w \in I^{\leq M}} (\mathbb{E}[S(X_1)_{a,b}^w - S(X_2)_{a,b}^w])^2 \right)^{1/2}. \quad (4.18)$$

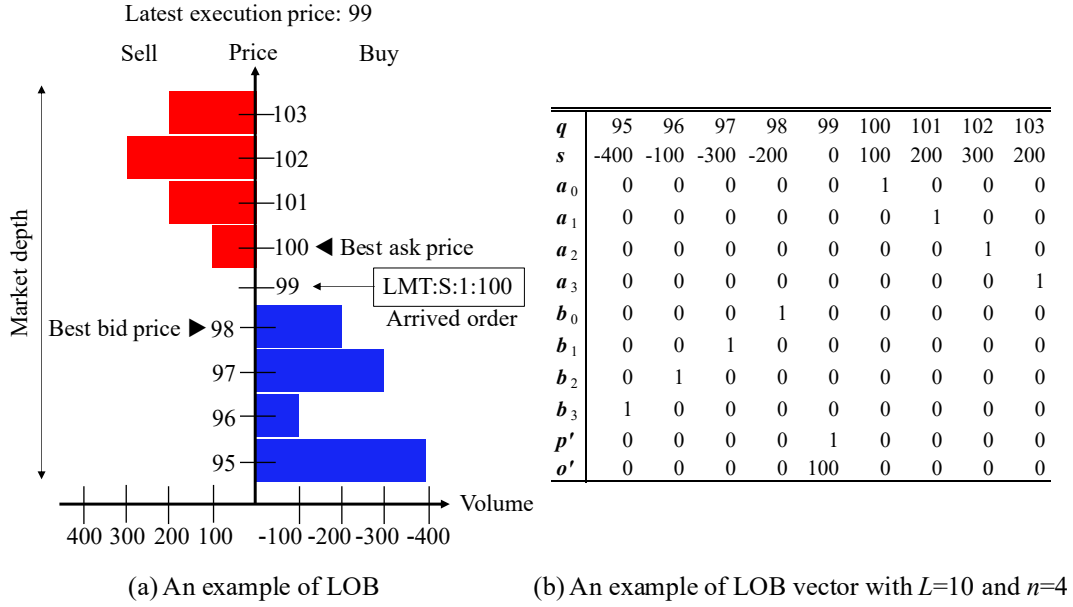


Figure 4.4.1: Limit order book and its vector

4.4 Method

4.4.1 Definition

Definition 4.4.1. [Order Event] An *order event* [32] is a quadruple o defined as

$$\begin{aligned}
 o := (r, d, \Delta, v) \in & \{\text{LMT}\} \times \{\text{S}, \text{B}\} \times \{-L_{\min}, \dots, -1, 0, 1, \dots, L_{\max}\} \\
 & \times \{100, 200, \dots, V_{\max}\} \\
 & \cup \{\text{MKT}\} \times \{\text{S}, \text{B}\} \times \{0\} \times \{100, 200, \dots, V_{\max}\} \\
 & \cup \{\text{CAN}\} \times \{\text{S}, \text{B}\} \times \{1, \dots, L_{\max}\} \\
 & \times \{100, 200, \dots, V_{\max}\},
 \end{aligned}$$

where r is an order type (market order, limit order or cancel order), d is a trade direction (sell or buy), Δ is a price position (we set $L_{\max} = L_{\min} = 10$), and v is an order quantity (minimum unit 100 and we set $V_{\max} = 2000$). Thus, the order event is represented as one-hot vectors o and the number of dimensions is $2 \times 21 \times 20 + 2 \times 20 + 2 \times 10 \times 20 = 1280$, which is called an order vector.

Definition 4.4.2. [LOB vector] An *LOB vector*, l is defined as

$$l := (q, s, p', a, b, o') \in \mathbb{R}^{L \times (2n+4)}.$$

$q \in \mathbb{R}^L$ is a vector with components of quote prices and $s \in \mathbb{R}^L$ is a vector with components of quote shares, which is called a quote share vector. $p' \in \mathbb{R}^L$ is a one-hot vector that is 1 if it is for the position of the latest execution price, otherwise 0. $a := (a_0, \dots, a_{n-1}) \in \mathbb{R}^{L \times n}$ and $b := (b_0, \dots, b_{n-1}) \in \mathbb{R}^{L \times n}$ are position vectors that are aligned from the 0th quote (best quote) to the $n - 1$ th quote, thus, for $i = 0, \dots, n - 1$, a_i is 1 if there is an ask quote, otherwise 0, and b_i is 1 if there is a bid quote, otherwise 0. $o' \in \mathbb{R}^L$ is an order position vector whose element is 1 if it is for the position of order which is calculated from both Δ and the best quote position, otherwise 0. L is a market depth and n is the number of quotes to be considered. We set $L = 40$ and $n = 20$. The LOB vector represents the feature vector of LOB and is calculated from the market simulator.

An example of the LOB vector is shown in Figure 4.4.1 (b) when the LOB with the arrived order (LMT:S:1:100) is as shown in Figure 4.4.1 (a).

Definition 4.4.3. [Market vector] *Market vector*, m_i at order arrival time $t_i \in \mathbb{R}$ is defined as,

$$\{m_i := (t_i, p_i, v_i, o_i, s_i)\}_{i=0,1,\dots,N_D-1},$$

where N_D is the number of order events contained in the time interval of a specified day (This time interval is called “specified interval”), $p_i \in \mathbb{R}$ is the current price value, $v_i \in \mathbb{R}$ is an execution volume, o_i is the order vector and s_i is the quote share vector.

4.4.2 Overall picture of Micro-Macro GAN

Figure 4.4.2 shows an overall picture of *Micro-Macro GAN*. Firstly, the *micro generator* (Figure 4.4.2 (a)) receives two inputs, z_i and m_i , respectively. $z_i \in \mathbb{R}^{n_z}$ is a noise vector, where n_z is 100, and each z_i is a random variable that follows an inde-

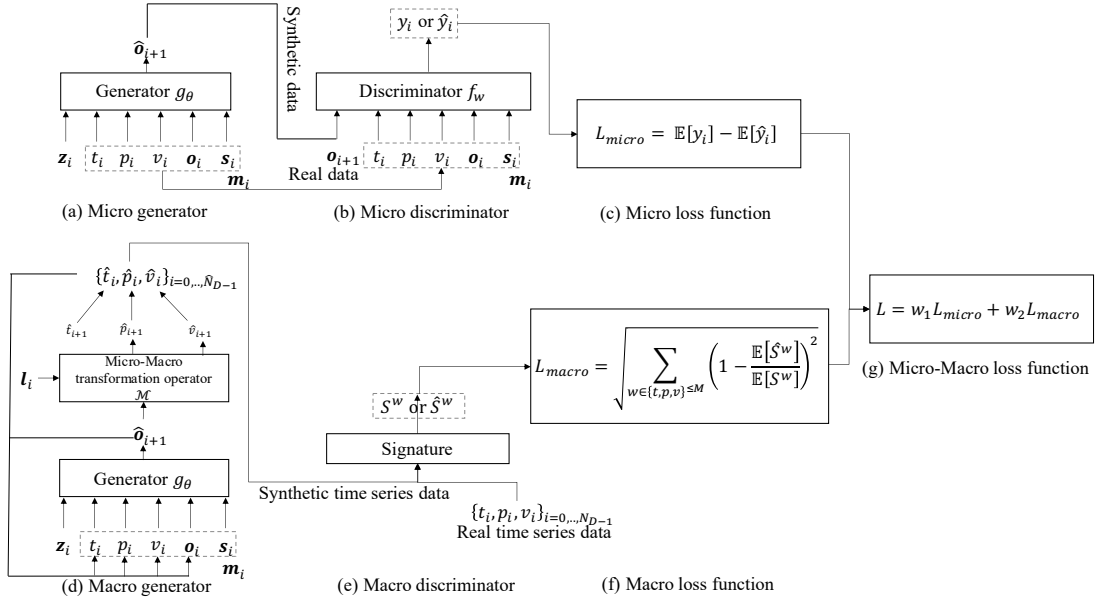


Figure 4.4.2: Overall picture of Micro-Macro GAN

pendent standard normal distribution, and \mathbf{m}_i is the market vector as indicated in Definition 4.3. The micro generator g_θ with parameter θ outputs a synthetic next order vector,

$$\hat{\mathbf{o}}_{i+1} := g_\theta(\mathbf{z}_i, \mathbf{m}_i).$$

The *micro discriminator* f_w (Figure 4.4.2 (b)) with parameter w is trained by receiving either the synthetic next order vector, $\hat{\mathbf{o}}_{i+1}$ or the real next order vector, \mathbf{o}_{i+1} , and its outputs, \hat{y}_i and y_i , are a scalar value, respectively,

$$\hat{y}_i := f_w(\hat{\mathbf{o}}_{i+1}),$$

$$y_i := f_w(\mathbf{o}_{i+1}).$$

On the other hand, in the *macro generator* (Figure 4.4.2 (d)), the order vector $\hat{\mathbf{o}}_{i+1}$ as micro variable is transformed into the execution price \hat{p}_{i+1} and the execution volume \hat{v}_{i+1} as macro variables using the LOB vector \mathbf{l}_i ,

$$(\hat{p}_{i+1}, \hat{v}_{i+1}) := \mathcal{M}(\hat{\mathbf{o}}_{i+1}, \mathbf{l}_i),$$

where \mathcal{M} is the *micro-macro transformation operator*. There are two methods to implement this operator: a deterministic rule-based method such the *continuous double auction* and a neural network-like function approximation method [29]. In this study, we selected the deterministic rule-based method. Let $X := \{t_i, p_i, v_i\}_{i=0, \dots, N_D-1}$ and $\hat{X} := \{\hat{t}_i, \hat{p}_i, \hat{v}_i\}_{i=0, \dots, \hat{N}_D-1}$ be vectors of tuples of real and synthetic macro variables of the specified interval, respectively, and the signatures (Figure 4.4.2 (e)) calculated from them are

$$S^w := \{S(X)_{0, N_D-1}^w\}_{w \in \{t, p, v\}^{\leq M}}, \quad (4.19)$$

$$\hat{S}^w := \{S(\hat{X})_{0, \hat{N}_D-1}^w\}_{w \in \{t, p, v\}^{\leq M}}, \quad (4.20)$$

where \hat{t}_i is the order arrival time and \hat{N}_D is the number of the synthetic order events in the specified interval.

The *Wasserstein loss function* for the Micro GAN (Figure 4.4.2 (c)) is calculated as follow,

$$L_{micro} := \mathbb{E}[y_i] - \mathbb{E}[\hat{y}_i]. \quad (4.21)$$

On the other hand, the loss function for the Macro GAN uses $\text{Sig-W}^{(M)}$, and is rewritten as a relative error (Figure 4.4.2 (f)) calculated as follow,

$$L_{macro} := \left(\sum_{w \in \{t, p, v\}^{\leq M}} \left(1 - \frac{\mathbb{E}[\hat{S}^w]}{\mathbb{E}[S^w]} \right)^2 \right)^{1/2}. \quad (4.22)$$

We set $M = 2$. Although the synthetic time \hat{t} should be generated by probability distribution models such as hazard functions or the GANs, the real time was used, $\hat{t} = t$. Finally, the loss function for the Micro-Macro GAN (Figure 4.4.2 (g)) is calculated as follow,

$$L := w_1 L_{micro} + w_2 L_{macro}, \quad (4.23)$$

where w_1 and w_2 are positive values such that $w_1 + w_2 = 1$, specifying the weights of the Micro and the Macro GANs.

4.4.3 Micro GAN

The *Micro GAN* is a GAN for generating the order as the micro variable, and is formulated as the *conditional WGAN (CWGAN)*, in which a single generator generates the orders placed by a large number of traders. The Stock-GAN [29] and the world-agent [47] are similar to the Micro GAN, which supports the usage of the Micro GAN as a baseline against the Micro-MacroGAN.

Micro generator g_θ

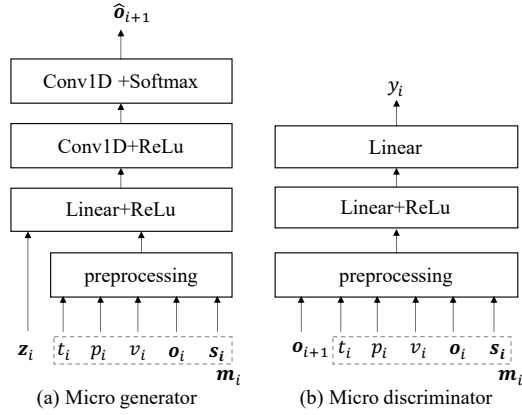


Figure 4.4.3: Micro GAN network

Figure 4.4.3 (a) shows the *micro generator* network which generates the synthetic next order vector \hat{o}_{i+1} and trains a conditional probability, $\mathbb{P}(o_{i+1}|m_i)$. First, m_i is converted to $\tilde{m} := (\tilde{t}_i, \tilde{p}_i, \tilde{v}_i, \tilde{o}_i, \tilde{s}_i)$ by the preprocessing layer,

$$\tilde{t}_i := \frac{t_i}{T},$$

$$\tilde{p}_i := p_i,$$

$$\tilde{v}_i := \log_{10}(1 + v_i),$$

$$\tilde{o}_i := \text{Embedding}(o_i),$$

$$\tilde{s}_i := \frac{s_i}{\sum_{j=0}^{L-1} s_{ij}},$$

where T indicates the terminal time, in units of seconds and we set $T = 900$ (15 minutes), o_i converted to a 100 dimensional real vector \tilde{o}_i by an *embedding layer*, $\tilde{o}_i = \text{Embedding}(o_i)$. Furthermore, \tilde{m}_i is combined with \mathbf{z} , which is passed to a *linear layer* and finally \hat{o}_{i+1} is generated by a two-layered 1-dimensional *convolutional neural network* (1D-CNN).

Micro discriminator f_w

Figure 4.4.3 (b) shows the *micro discriminator* network. In the same way, m_i is converted to \tilde{m} by the preprocessing layer, and is combined with the synthetic order vector \hat{o}_{i+1} or the real order vector, o_{i+1} . The combined vector is passed to a two-layered linear layer and finally a scalar value \hat{y}_i or y_i is output.

4.4.4 Macro GAN

The *Macro GAN* consists of a *macro generator* and a *macro discriminator* and is formulated by the *Sig-W GAN* to generate price dynamics including long periods that are difficult to represent in the *Micro GAN*.

Macro generator

the *macro generator* consists of the same generator as the generator used in the micro generator and the *micro-macro translation operator* \mathcal{M} . The weight parameters θ to be trained include only the micro generator g_θ , and \mathcal{M} contains no weight parameters. \mathcal{M} is implemented using algebraic operations to represent the execution rules in a continuous double auction.

Macro discriminator

The *macro discriminator* calculates the signatures, S^w and \hat{S}^w , using Eq(4.19-4.20). Thus, the macro discriminator does not include weight parameters.

4.4.5 Micro-Macro GAN Algorithm

The learning algorithm of the Micro-Macro GAN is summarized in Algorithm 2. The number of micro discriminator iterations per a micro generator iteration n_{dis} is set to 5, and we use the WGAN algorithm [15] for the Micro GAN. On the other hand, we use Sig-W algorithm for the Macro GAN. w_1 and w_2 are weight parameters of the Micro GAN and the Macro GAN. We use *Adam* algorithm [33] for updating parameters w and θ .

Algorithm 2 We use default values of $N = 10$, $n_{dis} = 5$, $b = 64$, $E = 8$, $\alpha = 0.05$, $\beta_1 = 0.5$, $\beta_2 = 0.9$, $w_1 = 0.5$, $w_2 = 0.5$, $M = 2$ and $n_z = 100$.

Require: N is the number of epochs, n_{dis} is the number of discriminator iterations per generator iteration, b is the batch size, α, β_1 and β_2 are Adam hyperparameters, w_0 is initial discriminator parameter, θ_0 is initial generator parameter, w_1 is weight of Micro GAN, w_2 is weight of Macro GAN, M is the depth of the signature, and n_z is noise dimension.

for $n = 1, \dots, N$ **do**

for $d = 1, \dots, n_{dis}$ **do**

 Sample a batch real data $\{\mathbf{o}_{i+1}^{(b)}, \mathbf{m}_i^{(b)}\} \sim \mathbb{P}$, a batch of random variables $\{\mathbf{z}_i^{(b)}\} \sim N(0, 1)^{n_z}$.

$L_d^{micro} \leftarrow \frac{w_1}{m} \sum_{b=1}^m f_w(g(\mathbf{z}_i^{(b)}, \mathbf{m}_i^{(b)}))$

$w \leftarrow \text{Adam}(\nabla_w L_d^{micro}, \text{ff}, \text{f}_1, \text{f}_2)$

end for

 Sample a batch real data $\{\mathbf{o}_{i+1}^{(b)}, \mathbf{m}_i^{(b)}\} \sim \cdot$, a batch of random variables $\{\mathbf{z}_i^{(b)}\} \sim N(0, 1)^{n_z}$.

$L_g^{micro} \leftarrow -\frac{1}{m} \sum_{b=1}^m f_w(g(\mathbf{z}_i^{(b)}, \mathbf{x}_i^{(b)}))$

for $e = 1, \dots, E$ **do**

 Get a date at random, initialize a market simulator with its date, set \mathbf{m}_0 , and get the real data X_1 .

$\hat{t}_0 \leftarrow 0$, $\hat{p}_0 \leftarrow 0$, $\hat{v}_0 \leftarrow 0$

for $i = 0, \dots, \hat{N}_D^e - 1$ **do**

 Sample a random variables $\{\mathbf{z}_i\} \sim N(0, 1)^{n_z}$.

$\hat{\mathbf{o}}_{i+1} \leftarrow g(\mathbf{z}_i, \mathbf{m}_i)$

 Set I_i from the market simulator and \hat{t}_{i+1} from other generator model (in this paper, $\hat{t}_{i+1} = t_{i+1}$).

$(\hat{p}_{i+1}, \hat{v}_{i+1}) \leftarrow \mathcal{M}(\hat{\mathbf{o}}_{i+1}, I_i)$

 Update the market simulator using $\hat{\mathbf{o}}_{i+1}$ and set \mathbf{m}_i .

end for

$S_e^w \leftarrow \{S(X_1)_{0, N_D-1}^w\}_{w \in \{t, p, v\} \leq M}$

$\hat{S}_e^w \leftarrow \{S(X_2)_{0, \hat{N}_D-1}^w\}_{w \in \{t, p, v\} \leq M}$

end for

$\bar{S}^w \leftarrow \frac{1}{E} \sum_{e=1}^E S_e^w$, $\bar{\hat{S}}^w \leftarrow \frac{1}{E} \sum_{e=1}^E \hat{S}_e^w$

$L_g^{macro} \leftarrow \sqrt{\sum_{w \in \{t, p, v\} \leq M} \left(1 - \frac{\bar{\hat{S}}^w}{\bar{S}^w}\right)^2}$

$L_g \leftarrow w_1 L_g^{micro} + w_2 L_g^{macro}$

$\theta \leftarrow \text{Adam}(\nabla_\theta L_g, \text{ff}, \text{f}_1, \text{f}_2)$

end for

4.5 Data

We select the *Tokyo Stock Exchange (TSE)* as a target market and use *FLEX Full* data [31] which is historical data of high frequency trading provided by the *Japan Exchange Group (JPX)*. To demonstrate the effectiveness of the Micro-Macro GAN, we select Toyota Motor Corporation (7203:Tokyo), January 2020 - July 2021 as a

highly liquid stock. Since the algorithm is currently very computationally expensive, we focus on the first 15 minutes of the opening time which is 9:00 for training the Micro-Macro GAN. The FLEX Full data is converted to the order vectors \mathbf{o}_i for $i = 0, 1, \dots, N_D - 1$, and the LOBs are constructed from the order events. Table 4.5.1 is a summary of the trading date, the number of orders, the execution price and the execution volume from 9:00 to 9:15, which is excluded days when special quotes occur.

Table 4.5.1: Number of Orders, Price at 9:15 and Volume at 9:15 for Toyota Motor Corp in July 2021

Date	Number of orders	Price at 9:15	Volume at 9:15
2021-07-01	20207	-25	496400
2021-07-05	26890	32	399900
2021-07-06	15257	-33	377100
2021-07-07	26157	-23	723100
2021-07-08	18782	0	341600
2021-07-09	34697	36	601000
2021-07-13	17418	-5	385000
2021-07-14	30739	97	765800
2021-07-15	18669	-4	337500
2021-07-16	21570	73	381100
2021-07-19	28308	-14	654300
2021-07-20	33433	-58	864400
2021-07-21	20882	43	525500
2021-07-27	22683	-4	485000
2021-07-28	23960	48	427300
2021-07-30	24198	73	396100

4.6 Result

4.6.1 Comparison of probability distributions and their distances

Fig 4.6.1 (a) shows the top 30 most probable real order events. It also shows that the most probable orders are limit or cancel orders, with low volume and close to the best ask or the best bid position. On the other hand, market orders are located at top13 and 14. Also, within the top30, all order volumes are small and near the best price, we can assume that most orders are placed by high-frequency traders.

Figure 4.6.1 (b) shows the probability distribution after 100,000 epochs using

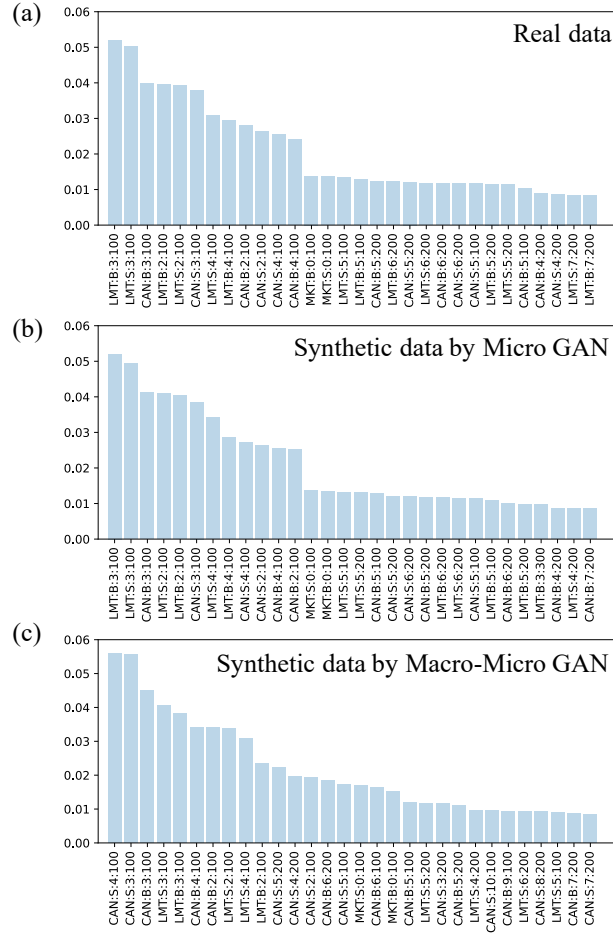


Figure 4.6.1: Comparison of order frequency

the Micro GAN algorithm ($w_1 = 1, w_2 = 0$). Comparing with the real probability distribution in Figure 4.6.1 (b), two distributions are almost identical in the top order events. This indicates that Micro GAN is well trained.

Figure 4.6.1 (c) shows the probability distribution after 10 epochs using the Micro-Macro GAN algorithm with the pre-trained parameters, w_0 and θ_0 , after which performed 100,000 epochs using only the Micro GAN algorithm. It can be seen that the probability distribution is similar to the real one.

The results of measuring the distance to the real probability distribution using the Kullback-Leibler and L2 distances are shown in Table 4.6.1. Both the Micro GAN and the Micro-Macro GAN were confirmed to be quite similar to the real distribution.

Table 4.6.1: Probability distribution distances

	Micro GAN	Micro-Macro GAN
Kullback–Leibler distance	1.85.E-02	2.04.E-01
L2 distance	1.26.E-04	3.87.E-03

4.6.2 Comparison of price and volume dynamics

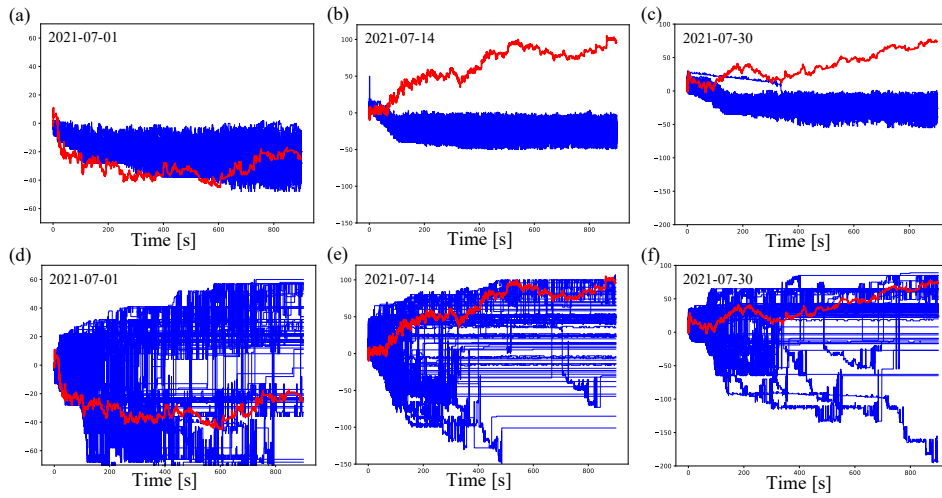


Figure 4.6.2: Comparison of price by the Micro GAN and the Micro-Macro GAN

Figure 4.6.2 and 4.6.3 show the results of the price and volume dynamics obtained by the market simulation on July 1, 2021, July 15, 2021, and July 28, 2021. The market simulations were run 100 times on each date and their results are indicated by the blue lines while the real data is indicated by the red line. Figure 4.6.2 (a), (b) and (c) show the price curves by the Micro GAN, which shows down curves for all dates, but their curves are similar. Figure 4.6.2 (d), (e) and (f) show the price curves by the Micro-Macro GAN, which are diverse for all dates. On the other hand, Figure 4.6.3 (a), (b) and (c) show the volume curves by the Micro GAN, which shows generally higher than or roughly equivalent to the real curve. Figure 4.6.3 (d), (e) and (f) show the volume curves by the Micro-Macro GAN, which are diverse for all dates just like those of the price curves. As a result, although the probability distribution of the Micro GAN closely resembles the real

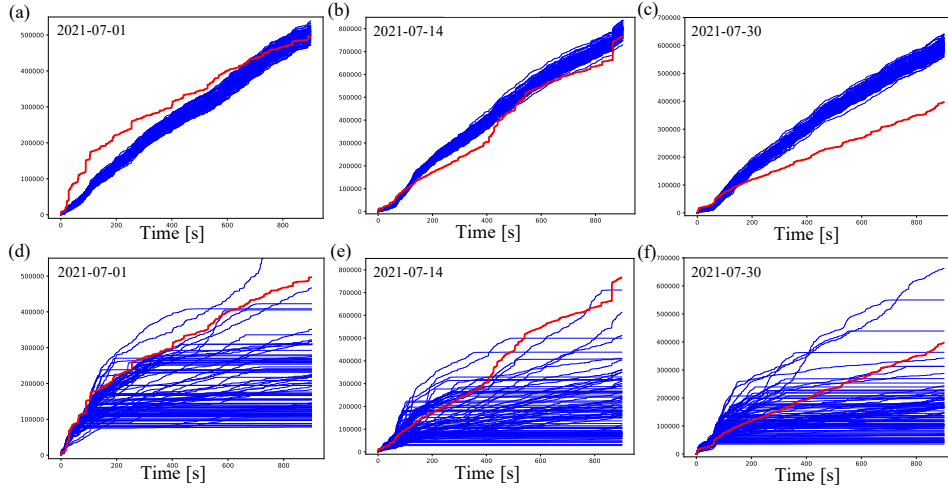


Figure 4.6.3: Comparison of volume by the Micro GAN and the Micro-Macro GAN

data, its dynamics are different from the real data, while the Micro-Macro GAN learns the signatures for each date and successfully generates various price and volume curves.

4.6.3 Timescales of price dynamics by the Micro GAN and the Micro-Macro GAN

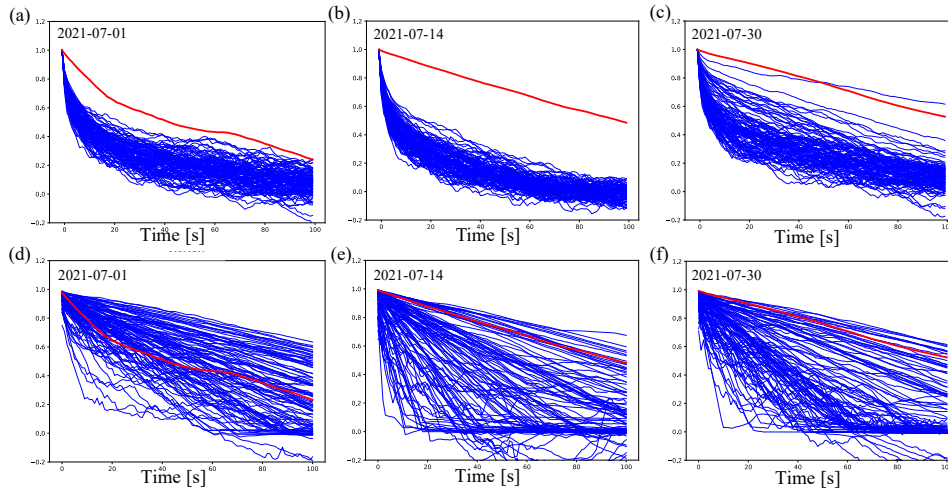


Figure 4.6.4: Autocorrelation function

Sampling was performed every second from the generated price time series data, $\{\hat{p}(t)\}_{t=0,1,\dots,900}$, where the unit of t is seconds, 0 means 9:00 and 900 means 9:15. To investigate the timescale of price dynamics, autocorrelation functions,

$$C(k) := \langle \hat{p}(t) \hat{p}(t+k) \rangle / \langle \hat{p}(t) \hat{p}(t) \rangle,$$

are computed, where $\langle \cdot \cdot \cdot \rangle$ is the time average, k is lag-time that set from 0 to 100. Figures 4.6.4 shows autocorrelation functions of the price dynamics generated by the Micro GAN and the Micro-Macro GAN in blue lines and red lines are the real.

The autocorrelation functions computed from the price time series data by the Micro GAN (Figure 4.6.2 (a), (b) and (c)) are shown in Figures 4.6.4 (a), (b) and (c). Because the result of the autocorrelation function by the Micro GAN is mostly smaller than the real, it can be seen that the dynamics generated by the Micro GAN are shorter-term than the real. On the other hands, the autocorrelation functions by the Micro-Macro GAN (Figure 4.6.2 (d), (e) and (f)) are shown in Figures 4.6.4 (d), (e) and (f). It can be seen that the Micro-Macro GAN results contain dynamics with more time scales than the Micro GAN, some of which appear similar to be real.

Table 4.6.2: Average signatures of $w \in \{t, p, v\}^{\leq 2}$

	t	p	v	$t-t$	$t-p$	$t-v$	$p-t$	$p-p$	$p-v$	$v-t$	$v-p$	$v-v$
real	900	14.8	510068.8	405000.0	5727.9	190961811.9	7547.1	1012.5	2981806.3	268100064.4	3428312.5	142789756687.5
Micro GAN	900	-24.7	605267.0	405000.0	-1856.6	239850579.0	-20352.6	344.2	-13153827.3	304889718.0	-1785981.7	192023330816.0
Micro -Macro GAN	900	14.7	161973.4	405000.0	7078.8	21960658.4	6109.6	639.8	-2463620.4	123815434.3	3922888.9	18748018048.0

Table 4.6.3: Mean squared error with real signatures

	t	p	v	$t-t$	$t-p$	$t-v$	$p-t$	$p-p$	$p-v$	$v-t$	$v-p$	$v-v$
① Micro GAN	0.0	39.4	95198.2	0.0	7584.5	48888767.1	27899.7	668.3	16135633.6	36789653.6	5214294.2	49233574128.5
② Micro -Macro GAN	0.0	0.1	348095.3	0.0	1350.9	169001153.5	1437.5	372.7	5445426.6	144284630.1	494576.4	124041738639.5
Difference (①-②)	0.0	39.3	-252897.1	0.0	6233.6	-120112386.4	26462.2	295.6	10690206.9	-107494976.5	4719717.8	-74808164511.0

4.6.4 Comparison of signatures

Table 4.6.2 shows the average signatures of the real data and the synthetic data by generated by the Micro GAN and the Micro-Macro GAN up to $M = 2$ of time t , price p and volume v . Table 4.6.3 shows the root square error of the Micro GAN and the Micro-Macro GAN with the real signature. The results show that the Micro-Macro GAN has smaller errors than the Micro GAN for the signature components including price, while the Micro GAN has smaller errors than the Micro-Macro GAN for the signature components including only volume. This is because, unlike the price curves, the volume curves are always monotonically increasing, and if the probability distribution of market orders are similar to the real data, the volume curves are also expected to be similar to the real data.

4.7 Conclusion

In this study, we proposed a new market simulation method using the Micro-Macro GAN. As a demonstration experiment, Toyota Motor Corporation was taken from the Flex Full data provided by the JPX and trained. The probability distribution of orders was similar to the real one, and market simulation by the Micro-Macro GAN was performed to generate various time series data of price and volume. Comparison of the signatures showed that the price signatures were similar to the real ones, but the volume signatures were different. Due to the limitation of the computational resources, only the Micro-Macro GAN results with $M = 2$ were obtained. However, more detailed results may be obtained if M is increased to 3 or more. Actually, we plan to increase the computational efficiency of the program code and computing resources to get more realistic curves. Our ultimate goal is to run the simulation until 15:00, the closing time of the TSE trading session.

Overall, we conclude that the market simulation with the Micro-Macro GAN could be more than worth a try.

Chapter 5

Global conclusion and discussion

Computer simulations are widely used in a variety of fields, including finance, life sciences, physics, oceanography, and meteorology. One of the greatest advantages of simulation is to be able to predict the future. However, in order to run a simulation, the underlying physical model must be modeled by humans themselves (deductive method or bottom-up approach). In recent years, with the development of deep learning technology, it has become possible to represent physical models by deep learning models themselves learning from data, even without knowing the physical model, as long as a large amount of data is available (inductive method, top-down approach). One of the simulation methods in finance is artificial market. While conventional artificial market simulations require modeling of order events placed by traders, we have attempted to model them with GANs. We have also tried to realize a highly realistic artificial market simulation with a new top-down approach.

In Chapter 2, we showed that it is possible to generate data that reproduces the distribution of order events by first focusing only on the order events generated by traders. As a verification of the effectiveness of this method, we confirmed that the accuracy of the 1-tick ahead price prediction model was improved by the execution prices generated by the artificial market simulation. As a result, it was shown that the data generated artificially by the artificial market simulation was effective in the actual short-term forecast model.

In Chapter 3, we focused on price changes, which are macro dynamics, and studied its signature, which is an effective feature. We proposed a discrete signature to apply to financial time series data, and showed that it is effective for financial time series data.

In Chapter 4, we proposed a method for generating micro order data, which we named Micro GAN, and a method for generating macro price time series data using the signature, which we named Macro GAN, and proposed Micro-Macro GAN was proposed as a coupled computation of the two GANs. We found that the resulting order distribution is comparable to Micro GAN for the evaluation of micro quantities, and is better than Micro GAN using the signature for the evaluation of macro quantities.

However, the artificial market simulation by GAN is only obtained by learning from historical data, and is still insufficient for events that are not in the historical data. Therefore, the results of this study are based on the interpolation of events that have not existed in the past represented by the historical data, so the results are not expected to be effective for events that cannot be represented by the interpolation. For example, the results of the data augmentation in Chapter 2 show an overall increase in the accuracy, but not as big as are expected. If a large shock event such as the Corona Shock can be generated in the artificial market, the accuracy would have increased more. This slight increase is likely due to the interpolative expression of the data augmentation.

As a simulation of future markets, a framework that can address the case of a significant deviation from the current state by stressing the artificial market is considered necessary. However, a fundamental solution to this problem could be a method that does not use historical data, such as an artificial market learning only from itself and in a short period of time, like Alpha Go. If we had such a solution, stock trading would be fully automated and completely out of human hands.

Bibliography

Bibliography

- [1] M. Minsky, *Society Of Mind*, Simon & Schuster, 1988.
- [2] W. S. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity, *The bulletin of mathematical biophysics*, 5(4), 115-133, 1943.
- [3] F. Rosenblatt, *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*, *Psychological Review*, 65(6), 386-408, 1958.
- [4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, *Nature*, 323(6088), 533-536, 1986.
- [5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, *Journal of Machine Learning Research* ,15, 1929-1958, 2014.
- [6] R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit, *Nature*, 405, 947-951, 2000.
- [7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, The MIT Press, 2016.
- [8] A. Y. Ng, Feature selection, L1 vs. L2 regularization, and rotational invariance, In *Proceedings of the twenty-first international conference on Machine learning*, ACM, 78, 2004.
- [9] J. Wang, L. Perez, The Effectiveness of Data Augmentation in Image Classification using Deep Learning, *arXiv:1712.04621*, 2017.

- [10] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan and D. Hassabis, Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm , arXiv:1712.01815, 2017.
- [11] K. Izumi, Artificial Market (in Japanese), Morikita Publishing, 2003.
- [12] J. Ho and S. Ermon, Generative Adversarial Imitation Learning, arXiv:1606.03476, 2016.
- [13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, Generative adversarial nets. In Advances in Neural Information Processing Systems 27, 2672-2680. Curran Associates, Inc., 2014.
- [14] J. Song, H. Ren, D. Sadigh and S. Ermon, Multi-Agent Generative Adversarial Imitation Learning, arXiv:1807.09936, 2018.
- [15] M. Arjovsky, S. Chintala, and L. Bottou, Wasserstein GAN, arXiv:1701.07875, 2017.
- [16] C. Villani, Topics in Optimal Transportation, American Mathematical Society, 58, 2003.
- [17] T. Lyons, M. Caruana, and T. Lévy, Differential Equations Driven by Rough Paths, Number 1908 in Lecture Notes in Mathematics. Springer-Verlag, 2007.
- [18] H. Li, L. Szpruch, M. Sabate-Vidales, B. Xiao, M. Wiese and S. Liao, Sig-Wasserstein GANs for Time Series Generation, arXiv:2111.01207, 2021.
- [19] K. Law, A. Stuart, and K. Zygalakis, Data Assimilation: A Mathematical Introduction, Springer, 2015.
- [20] C. Chiarella, G. Iori, and J. Perello, The impact of heterogeneous trading rules on the limit order book and order flows, Journal of Economic Dynamics and Control 33(3), 525-537, 2009.

- [21] B. LeBaron, Agent-based computational finance, Handbook of Computational Economics. Elsevier, 2006.
- [22] T. Mizuta, S. Hayakawa, K. Izumi, and S. Yoshimura, Simulation Study on Effects of Tick Size Difference in Stock Markets Competition, Proceedings of AESCS2013, 2013.
- [23] A. Radford, L. Metz, and S. Chintala, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, arXiv:1511.06434, 2016.
- [24] A. Koshiyama, N. Firoozye, and P. Treleaven, Generative Adversarial Networks for Financial Trading Strategies Fine-Tuning and Combination, arXiv:1901.01751, 2019.
- [25] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, Synthetic Data Augmentation using GAN for Improved Liver Lesion Classification, arXiv:1801.02385, 2018.
- [26] F. Tanaka and C. Aranha, Data Augmentation Using GANs, arXiv:1904.09135, 2019.
- [27] Y. X. Wang, R. Girshick, M. Hebert, and B. Hariharan, Low-Short Learning from Imaginary Data, arXiv:1801.05401, 2018.
- [28] A. Antoniou, A. Storkey, and H. Edwards, Data Augmentation Generative Adversarial Networks, arXiv:1711.04340, 2017.
- [29] J. Li, X. Wang, Y. Lin, A. Sinha, and M. P. Wellman, Generating Realistic Stock Market Order Streams, arXiv:2006.04212, 2020.
- [30] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, Improved Training of Wasserstein GANs, arXiv:1704.00028, 2017.
- [31] <https://www.jpx.co.jp/markets/paid-info-equities/realtime/index.html>
- [32] T. Adachi, Algorithmic Trading (in Japanese), Asakura Publishing, 2018.

- [33] D. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, arXiv:1412.6980, 2014.
- [34] <https://aws.amazon.com/jp/ec2/>
- [35] D. M. Beazley, Python Distilled, Addison-Wesley, 2022.
- [36] I. Chevyrev and A. Kormilitzin, A primer on the signature method in machine learning, arXiv:1603.03788, 2016.
- [37] L. G. Gyurkó, T. Lyons, M. Kontkowski, and Field, J, Extracting information from the signature of a financial data stream, arXiv:1307.7244, 2014.
- [38] P. Kidger and T. Lyons, Signatory: differentiable computations of the signature and logsignature transforms, on both CPU and GPU, In *International Conference on Learning Representations*, 2021.
- [39] M. Sipser, Introduction to the Theory of Computation, Cengage Learning, 3rd edition, 2013.
- [40] B. I. Chevyrev and T. Lyons, Characteristic Functions of Measures on Geometric Rough Paths, The Annals of Probability, no. 6, 4049-4082, 2016.
- [41] J. Morrill, A. Fermanian, P. Kidger and T. Lyons, A Generalised Signature Method for Multivariate Time Series Feature, arXiv:2006.00873, 2021.
- [42] K. Izumi and K. Ueda, Phase Transition in a Foreign Exchange Market - Analysis Based on an Artificial Market Approach, IEEE Transactions on Evolutionary Computation, vol. 5, no. 5, 2001.
- [43] D. K. Gode and S. Sunder, Allocative Efficiency of Markets with Zero-Intelligence Traders: Market as a Partial Substitute for Individual Rationality, Journal of Political Economy, vol. 101, no. 1, 119-137, 1993.
- [44] J. D. Farmer, P. Patell and I. I. Zovko, The predictive power of zero intelligence in financial markets, Proceedings of the National Academy of Sciences, vol. 102, no. 6, 2254-2259, 2005.

- [45] S. Maslov, Simple Model of a Limit Order-Driven Market, *Physica A*, vol. 278, 571-578, 2000.
- [46] S. Vyetenko, D. Byrd, N. Petosa, M. Mahfouz, D Dervovic, M. Veloso and T. H. Balch, Get Real: Realism Metrics for Robust Limit Order Book Market Simulations, arXiv:1912.04941, 2019.
- [47] A. Coletta, M. Prata, M. Conti, E. Mercanti and N. Bartolini, Towards Realistic Market Simulations: a Generative Adversarial Networks Approach, arXiv:2110.13287, 2021.

Index

A		C	
Adam	18, 68	concatenation	35, 60
agent simulation	55	conditional WGAN	66
agents	4, 9, 53	continuous double auction	65
algorithmic enhancement	6	convolutional neural network	17, 67
algorithmic performance evaluation	6	Corona	4
alphabet	34, 60	Corona Shock	18
AlphaZero	4, 53	Corona shock	53
alternative data	1	CWGAN	66
Amazon Web Service's Elastic Com-		D	
pute Cloud	20	data assimilation	5
artificial market	4, 55	data augmentation	4, 6, 53
artificial market simulation	6	deep learning	3
artificial market simulations	55	discrete signature	40
artificial markets	9	discriminator	14, 16, 17, 59
		discriminators	10
		Dropout	3
B		E	
back-tested	4, 53	EC2	20
backpropagation	3	embedding layer	17, 67
behavioral science	2	empty word	35, 60
bottom-up approach	4, 54	extended	36

INDEX

86

F

flat discrete signature	37
FLEX Full	11, 69
foreign exchange	55
fully connected layer	17
fully connected layers	17
fundamental data	1

G

GAN	5, 6, 16, 54
GANs	10
Generative Adversarial Network	5, 54
Generative Adversarial Networks	10
generator	12, 16, 57
generators	10
gradient penalty	18

H

historical market event	15
human intelligence	2

I

imitation learning	5
inverse reinforcement learning	5

L

L1	4, 53
L2	4, 53
Lehman	4
linear layer	67
LOB vector	63

Long Short Term Memory	16
long-short term memory	11
long-term memory	56
LSTM	11, 16, 17

M

machine learning	53
macro discriminator	67
Macro GAN	55, 67
macro generator	64, 67
market event	15
market impact	6
Market vector	63
micro discriminator	64, 67
Micro GAN	54, 66, 67
micro generator	63, 66
Micro-Macro GAN	7, 63
Micro-Macro GAN,	54
micro-macro transformation operator	65
micro-macro translation operator	67
multi-agent simulation	55
multi-agents	5
multi-index	34, 60

N

natural language processing	4
normed space	11, 56

O

order event	14, 62
-------------	--------

P

perceptron	3
Polish space	11, 56
population	11, 57

Q

Quantum mechanical forces	2
---------------------------	---

R

Reality	11, 56
reinforcement	6
ReLU	3

S

Sig-W GAN	7, 54, 67
signature	5, 35, 61
Stock-GAN	56
stylized facts	4, 9

T

technical data	1
The Society of Mind	3
Tokyo Stock Exchange	11, 18, 55, 69
Tokyo Stock Price Index	18
top-down approaches	5, 54
TOPIX	18
TSE	55, 69

W

Wasserstein distance	13, 16, 58
Wasserstein GAN	5, 10, 14, 54, 59
Wasserstein loss function	65
WGAN	5, 6, 10, 14, 54, 59
word	34, 60
world-agent	56

Z

zero-intelligence	55
-------------------	----