MASTER THESIS

A Study of Data-driven Market Simulator

Songyan Hou

Supervisor: Prof. Josef Teichmann

Department of Mathematics
ETH Zürich

Submitted: March 8, 2021

Abstract

Data-driven market generator based on deep learning model provides a more flexible framework to stimulate financial time series. We briefly discuss the generative modelling approaches as well as evaluation metrics for financial time series and point out their main challenges and difficulties, such as small data set problem and irregular sampling problem. In order to address these problems in financial time series stimulation, we introduce a market generator based on variational autoencoders (VAEs) with signature features, which is utilized from rough paths theory. Furthermore, signature features are universal, which helps to construct a maximum mean discrepancy (MMD) test based on signature kernel. This MMD test overcome difficulties in evaluation between distributions on time series and can be computed efficiently by recursive algorithm. We further improve the performance of market generator by considering student-t prior VAEs and introducing a novel neural network structure with signature map embedded. Moreover, we accelerate the market generator by inventing a signature inversion algorithm based on autoencoder structure. Also, we present a generative model based on ridge regression on signature features, which provide very accurate stimulation given the distribution of driven processes. Finally, we present the numerical implementations of market generator on rough models as well as all other algorithms to support our results numerically.

Contents

1 Introduction **4**
- 1.1 Financial times-series stimulation . . . 5
  - 1.1.1 Classical and modern model . . . 5
  - 1.1.2 Challenges of financial time series stimulation . . . 6

2 Signature feature **8**
- 2.1 Signature . . . 8
  - 2.1.1 Signature map . . . 10
  - 2.1.2 Signature stream . . . 11
- 2.2 Log-signature . . . 13
  - 2.2.1 Lie Series . . . 14
- 2.3 Universality and characteristics . . . 15
- 2.4 Discrete signature feature . . . 18
  - 2.4.1 Kernel trick . . . 19
- 2.5 Low-rank-signature . . . 20

3 Variational Autoencoders **22**
- 3.1 Variational autoencoders . . . 22
  - 3.1.1 Variational inference . . . 23
  - 3.1.2 Reparameterization Trick . . . 24
- 3.2 Variations of VAEs . . . 26
  - 3.2.1 Conditional VAEs . . . 26
  - 3.2.2 $\beta-$VAEs . . . 27
  - 3.2.3 Student-VAEs . . . 27
  - 3.2.4 Gaussian process VAEs . . . 28

4 Market Generator **29**
4.1 Time series splitting 30
4.2 Path to signature features 31
4.3 Variational autoencoder on signature features 32
4.4 Signature features to path 32
4.5 Evaluation of model 33

5 Numerical Implementation **34**
5.1 Inversion algorithm 34
5.1.1 Single log-signature inversion 34
5.1.2 Multiple log-signatures inversion 36
5.2 Evaluation 38
5.2.1 Separating distributions 39
5.3 Market generator 41
5.3.1 Lead-lag based VAEs 42
5.3.2 Time-augmentation based VAEs 44
5.3.3 Student-t prior 47
5.4 Path stimulation with signature 49
5.5 Learning from path to path 51

Acknowledgement

I would like to thank my supervisor Professor Josef Teichmann for endless help and instructions. Furthermore, I thank him very much for this interesting topic and directing me to the right direction, but also his encouragement motivating me to explore my own research interests. It is a great honor and pleasure to work with him on this fantastic topic. Also, I want to express my sincere thanks to my parents for their great love and endless support. Finally, I want to thank all my friends. Their accompany and help is one of my greatest memory during my master study.

1. Introduction

There is growing interests in applications of deep leaning to quantitative finance in [6, 12, 18, 17, 29, 31, 50, 59], where the power of deep neural network in pricing, hedging, modeling, and portfolio optimization have been proven. In all fantastic applications above, deep solvers are trained on market data to learn an optimal control when faced with market distributions. However, in practice, market data are usually scare for training. One approach is to use classical model to generate data but classical model are recently challenged by real market data because classical model are usually too simple and static to reflect the market. Another approach is the data-driven market generator, which learns from limited market data to generate sufficient samples which has similar distribution to the real market data. In this thesis, we develop a flexible generative model based on variational autoencoder (VAEs) [38] and signatures of paths [24], which is first introduced in [13]. This model is capable to generate realistic synthetic data with scare data, and also this model can generate market data conditioned on history data. we improve the performance of this model in both speed as well as performance, and we fix some numerical weakness in practical implementations. Furthermore, we introduce a novel structure of neural networks in this thesis.

The thesis is organized as follows: In this chapter, we state the times-series stimulation problem with an outline of motivations and potential applications such as data anonymisation [40]. Furthermore, we discuss the challenges of market generating in financial scenario such as roughness [25]. Then we contrast the main differences between classical methods and data-driven generators, also pointing out new challenges for generative models, such as the choice of evaluation metrics. In chapter 2, we give a brief introduction to the signature feature of a path, focusing on the universality of the normalized signature map [14] as well as properties of multiple variations of signature features. In chapter 3, we give a brief introduction to the variational autoencoders (VAEs) [37] and some variations of VAEs.

In chapter 4, we present our methodology to construct the market generator with details. In chapter 5, we provide numerical implementations of evaluation based on signatures, inverting algorithm from log-signatures to paths, performance of our market generator on rough models, and paths stimulation with signatures. The algorithm developed in this paper and all details of numerical implementations are available in the Github repository provided with this paper Github repository: `https://github.com/justinhou95/market_simulator`.

1.1 Financial times-series stimulation

The recent developments of deep learning in financial applications is one of the main motivation to construct a good market generator because the quality of input data determines the performance of deep solvers. There are many other situations where market simulator are need.

- **Data anonymisation:** financial and medical data are usually confidential. If only the distribution of data matters, data generator can generate the distribution indifferent from the true data without leaking the true data.
- **Small original training data sets:** financial data are usually expensive and scare. Data generator augment the data sets for more complex application avoid over fitting to the observed data.
- **Back testing:** back testing is important for trading strategy and the market generator avoid overfitting to historical data.
- **Risk management:** market simulator can be used to generate synthetic paths to estimate various risk metrics, such as Value at Risk (VaR).

Motived by these applications, many market generator have been developed, so we shortly recall the history.

1.1.1 Classical and modern model

Numerical simulation of financial time series has rich literatures.

- **Classical models:** Classical models includes stochastic market models and autoregressive models. The advantages of classical models is that they are explainable and reliable but they are less flexible for complicated data. A generalization of classical models is to add more parameters and consider their weighted average [19].

- **Modern models:** Generative models in machine learning learn a map from randomness to synthetic data which has the same distribution of true data. Therefore, modern model implicitly approximates the true distribution without explicitly computing the distribution.

Among all generative models, there are two classes of deep generative models which are very successful and widely used: Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs). GANs were first introduced in 2014 by Goodfellow et al. in [47] and soon gained it's reputation in generating high resolution images and images classification tasks. VAEs were first introduced in 2013 by Kingma et al. in [37], as a variational bayesian inference with autoencoder structure which was proposed in [5]. GANs are undoubtedly the most popular model in generative models and have been proven successful in many areas . On the other hand, recent theoretical connections between autoencoders and variational bayesian inference draw more attentions to VAEs as well, see [38]. Usually, GANs are capable to generate high precision samples but they suffer from data-hunger and instability in training phase, see [4]. On the other hand, VAEs are easy to train and don't suffer from mode collapse, but mostly produce smoother (blurry in images scenario) samples, see [33]. In our market generating task, recent works using GANs to generate markets data includes the following: [8, 18, 28, 12, 57, 58]. For VAEs, to date the only literature using VAEs for data generation is [13]. In this thesis, we focus on VAEs models due to challenges that arise in the setting of small data set of financial time series.

1.1.2 Challenges of financial time series stimulation

In the scenario of financial time series, a number of features of financial data, commonly referred to as stylized facts [16], need to be considered carefully when designing market generator.

- **Small data set:** financial data like S&P500 and DAX are relatively small data set.
- **Non-stationary:** financial data are in general non-stationary, therefore it need special preprocessing to satisfies i.i.d. assumptions for training data.
- **Heavy tails and aggregational Gaussianity:** financial data are believed non-gaussian and heavy-tail general, see [3].
- **No arbitrage** sometimes financial models are required to satisfies no arbitrage condition.

- **Volatility clustering:** absence of autocorrelation of asset returns, but slow decay of autocorrelation in absolute returns, see [16].
- **Discrete representation of continuous path:** finite horizon observation might fail in characterizing specific properties in financial data. For example, one can construct examples that are indistinguishable from one another on a finite set of marginals, but leads to arbitrarily different hedging strategies and option prices in continuous time, see [11].
- **Leverage effect:** Asset returns has negative correlation with the volatility of asset returns, see [10].
- **Roughness:** Financial data are observed rough in values as well as volatility, see [25].
- **Irregular sampling:** missing data and highly oscillatory data are common in financial scenario, see [36].

On one hand, the market generator should be robust enough to overcome problems in financial data (small data set, non-stationary, irregular sampling, non-robust representation). On the other hand, the market generator should be expressive enough to generate rough, heavy tails synthetic data. These challenges motivate us to introduce the signature feature map. Signature features have the following properties: (i) efficiently representing data which are important for hedging problems, see (2.20) (ii) robustness in representing data, see [11] (iii) robustness in irregular sampling, see [43] (iv) Invariant under translation and reparameterization, see Proposition 2.5 (v) uniqueness up to tree-like equivalence, see Theorem 2.6 (vi) universality, see 2.9 (vii) efficiently computation by recursive algorithm, see [39] and powerful python packages available. Moreover, signatures provides a generic way evaluating the closeness between different distributions of time series by considering the Maximum Mean Discrepancy (MMD) test [26] based on signature kernel introduced by Chevyrev and Oberhauser in [14]. Traditional distributional metrics and divergences usually have the following weakness: (i) lack of universality because metric is designed for specific application (ii) need of explicit distribution (KL-divergence and Wasserstein metric) (iii) being hard to compute or even ard to define on paths space (iii) fail to characterize the hedging performance of paths (iv) need of stationary assumption. However, Maximum Mean Discrepancy (MMD) test based on signature kernel overcomes all the difficulties stated above because signature is defined on path space, and more importantly, the universality of the normalized signature map [14]. Next, we provide an introduction with details on properties of signatures in the next chapter.

2. Signature feature

In this chapter, we give an introduction to the signature feature of a path and some variation of signature feature such as log-signature, low-rank-signature. Moreover, we discuss the expectation of signatures of a stochastic process and emphasize their characteristic ability of the law of stochastic process. For simplicity and readability, we restrict our introduction on the space of continuous bounded variation paths from a compact time interval J to a Euclidean space $E = \mathbb{R}^d$ denoted by $\mathcal{C}_0^1(J, E)$. We refer the discussion of semi-martingales and rough paths taking values in Banach space E to [24, 46] and kindly invite interested readers to the comprehensive and rigorous introduction of signature feature in [24, 46].

2.1 Signature

We start the introduction from the tensor algebra in which our signature feature takes value. We consider the non-commuting formal power series of tensors with formal indeterminates as basis of E.

Definition 2.1 (Formal power series)**.** We denote $T((E))$ the space of formal power series of tensors in E that

$$T((E)) = \left\{ \mathbf{a} = (\mathbf{a}_k)_{k\geqslant 0} \colon \mathbf{a}_k \in E^{\otimes k} \right\} = \bigoplus_{m\geqslant 0} E^{\otimes m} \tag{2.1}$$

endowed with addition and multiplication defined as follows: Let $\mathbf{a} = (\mathbf{a}_k)_{k\geqslant 0}, \mathbf{b} = (\mathbf{b}_k)_{k\geqslant 0} \in T((E))$ and $\lambda \in \mathbb{R}$. Then

$$\begin{aligned} \mathbf{a} + \mathbf{b} &= (\mathbf{a}_k + \mathbf{b}_k)_{k\geqslant 0} \\ \mathbf{a} \otimes \mathbf{b} &= \Big(\sum_{i+j=k} \mathbf{a}_i \otimes \mathbf{b}_j \Big)_{k\geqslant 0} \\ \lambda \mathbf{a} &= (\lambda \mathbf{a}_k)_{k\geqslant 0} \end{aligned} \tag{2.2}$$

Let $T^{(n)}(E)$ denote the truncated tensor algebra space up to order n

$$T^{(n)}(E) = \bigoplus_{k \leqslant n} E^{\otimes k} \tag{2.3}$$

Let $e_1, \cdots, e_d$ be a finite basis of $E = \mathbb{R}^d$. Then $\mathbf{a} \in T((E))$ has the following linear form

$$\mathbf{a} = \sum_{k \geqslant 0} \Big(\sum_{i_1, \cdots, i_k = 1}^{d} a_{i_1, \cdots, i_k} e_{i_1} \otimes \cdots \otimes e_{i_k} \Big), \quad a_{i_1, \cdots, i_k} \in \mathbb{R}. \tag{2.4}$$

and $T^{(n)}(E)$ can be considered as a subspace of $T((E))$.

Definition 2.2. We denote $\mathbf{T}(E)$ the Banach space

$$\mathbf{T}(E) \coloneqq \Big\{ \mathbf{t} \in T((E)) \colon \|\mathbf{t}\|_{\mathbf{T}(E)} \coloneqq \sqrt{\sum_{k \geqslant 0} \|\mathbf{t}_k\|^2_{E^{\otimes k}}} < \infty \Big\}. \tag{2.5}$$

Similarly, we denote $\mathbf{T}^{(n)}(E)$ the truncation of $\mathbf{T}(E)$ up to order n.

Definition 2.3 (Signature)**.** We denote $\mathbf{Sig}_J \colon \mathcal{C}_0^1(J, E) \to \mathbf{T}(E)$ the signature map such that for all $X \in \mathcal{C}_0^1(J, E)$

$$\mathbf{Sig}_J(X) = (1, \mathbf{s}_1, \cdots) \in \mathbf{T}(E) \tag{2.6}$$

where

$$\begin{aligned} \mathbf{s}_k &= \int_{t_1 < \cdots < t_k \in J} dX_{t_1} \otimes \cdots \otimes dX_{t_k} \\ &= \sum_{i_1, \cdots, i_k = 1}^{d} \int_{t_1 < \cdots < t_k \in J} dX^{i_1}_{t_1} \cdots dX^{i_k}_{t_k} \cdot e_{i_1} \otimes \cdots \otimes e_{i_k} \end{aligned} \tag{2.7}$$

Further more, we let $\mathbf{Sig}_J^{(n)}$ denote the truncated signature map up to order n

$$\mathbf{Sig}_J^{(n)}(X) = (1, \mathbf{s}_1, \cdots, \mathbf{s}_M) \in \mathbf{T}^{(n)}(E). \tag{2.8}$$

Example 2.4. *Let $X_t = t\mathbf{x} \in \mathbb{R}^d$, then*

$$\mathbf{Sig}_{[0,1]}(X) = (1, \mathbf{x}, \frac{\mathbf{x}^{\otimes 2}}{2!}, \cdots) \in \mathbf{T}(E) \tag{2.9}$$

2.1.1 Signature map

We expect a good feature map to capture important information while ignoring irrelevant ones. First observation is the the signature feature is invariant of starting point because $d(X_t - X_0) = dX_t$. Moreover, it is invariant of reparametrization.

Proposition 2.5 (Invariant under reparametrization)**.** *Let $X \in \mathcal{C}_0^1([S_1, T_1], E)$ and $\tau\colon [S_1, T_1] \to [S_2, T_2]$ a non-decreasing surjective reparametrization. Then*

$$\mathbf{Sig}_{[S_2,T_2]}(X_{\tau(\cdot)}) = \mathbf{Sig}_{[S_1,T_1]}(X) \tag{2.10}$$

Proof.

$$\begin{aligned}\mathbf{Sig}_{[S_2,T_2]}(X_{\tau(\cdot)})_k &= \int_{\tau(t_1)<\cdots<\tau(t_k)\in[S_2,T_2]} dX_{\tau(t_1)} \otimes \cdots \otimes dX_{\tau(t_k)} \\ &= \int_{t_1<\cdots<t_k\in[S_1,T_1]} dX_{t_1} \otimes \cdots \otimes dX_{t_k)} = \mathbf{Sig}_{[S_1,T_1]}(X)\end{aligned} \tag{2.11}$$

□

From the invariant property of reparametrization, we notice that signature map is not injective. However, signature feature is injective up to tree-like equivalence $\sim_t$ which we detailed in [7], and obviously time-reparametrization is included in tree-like equivalence. We define $\mathcal{P}_0^1 = \mathcal{C}_0^1([0, T], E)/\sim_t$ the quotient space up to tree-like equivalence endowed with quotient metric.

Theorem 2.6 (Weak uniqueness [7])**.** $\mathbf{Sig}_{[0,T]}$ *is injective on $\mathcal{P}_0^1$.*

Proof. See [7]. □

If we add time as an additional strictly increasing coordinate into path ie. $\overline{X_t} = (t, X_t) \in \overline{E} \coloneqq \mathbb{R} \oplus E$, then $\mathbf{Sig}_{[0,T]}$ is injective on the time-augmented space.

Corollary 2.7 (Uniqueness [7])**.** $\mathbf{Sig}_{[0,T]}$ *is injective on $\mathcal{C}_0^1([0, T], \overline{E})$.*

Proof. Since the first coordinate is strictly increasing, the only tree-like equivalent path is itself i.e. $\mathcal{P}_0^1 = \mathcal{C}_0^1([0, T], \overline{E})$, which concludes the proof. □

Proposition 2.8. $\mathbf{Sig}_{[0,1]}$ *is neither surjective nor the range of which a linear subspace of $\mathbf{T}(E)$.*

Proof. Let $X \in \mathcal{C}_0^1([0,T],E)$ and w.l.o.g assume $X_0 = 0$. By integration by part

$$\begin{aligned}\mathbf{Sig}_{[0,1]}(X)_{1,2} + \mathbf{Sig}_{[0,1]}(X)_{2,1} &= \int_{t_1<t_2\in[0,1]} dX^1_{t_1}dX^2_{t_2} + \int_{t_1<t_2\in[0,1]} dX^1_{t_1}dX^2_{t_2} \\ &= \int_{t\in[0,1]} d(X^1_t X^2_t) = \mathbf{Sig}_{[0,1]}(X)_1\ \mathbf{Sig}_{[0,1]}(X)_2.\end{aligned} \tag{2.12}$$

Thus, $\mathbf{Sig}_{[0,1]}$ is neither surjective nor the range of it a linear subspace of $\mathbf{T}(E)$. □

Theorem 2.9 (Weak universality)**.** *Let A be a compact set of $\mathbf{Sig}(\mathcal{C}_0^1(J,E))$, then for all $f\colon A \to \mathbb{R}$ continuous and for all $\epsilon > 0$, there exists a linear functional $L \in \mathbf{T}(E)^*$ such that*

$$\sup_{\mathbf{a}\in A} \|f(\mathbf{a}) - L(\mathbf{a})\| \leqslant \epsilon \tag{2.13}$$

Proof. Proof relies on the Stone-Weierstrass theorem and the shuffle product property of the signature detailed in [43]. □

2.1.2 Signature stream

Instead of viewing signature as a static object, we can consider signature stream of a path $X \in \mathcal{C}_0^1([0,T],E)$ as a process $\big(\mathbf{Sig}_{[0,t]}(X)\big)_{[0,T]}$ taking values in $\mathbf{T}(E)$.

Proposition 2.10. *Let $X \in \mathcal{C}_0^1([0,T],E)$ and define $\pi_n\colon \mathbf{T}(E) \to \mathbf{T}^{(n)}(E)$ the projection such that for all $\mathbf{x} \in \mathbf{T}(E)$*

$$\pi_n\big((\mathbf{x}_k)_{k\geqslant 0}\big) = (\mathbf{x}_k)_{k\leqslant n} \tag{2.14}$$

then $S_t = \mathbf{Sig}^{(n)}_{[0,t]}(X)$ satisfies for all $t \in [0,T]$ that

$$dS_t = \pi_n(S_t \otimes dX_t), \quad S_0 = (1,0,\cdots), \tag{2.15}$$

and moreover $(S_t)_{t\in[0,T]}$ is the unique solution of (2.15).

Proof. See Lemma 2.10 in [46]. □

Definition 2.11. Let $X \in \mathcal{C}_0^1([0,s],E)$ and $X \in \mathcal{C}_0^1([s,t],E)$. The concatenated path $X \star Y \in \mathcal{C}_0^1([0,t],E)$ is defined by

$$(X \star Y)_u = \begin{cases} X_u & u \in [0,s] \\ Y_u + (X_s - Y_s) & u \in [s,t] \end{cases} \tag{2.16}$$

Theorem 2.12 (Chen's identity)**.** *Let $X \in \mathcal{C}_0^1([0,s],E)$ and $Y \in \mathcal{C}_0^1([s,t],E)$. Then*

$$\mathbf{Sig}_{[0,t]}(X \star Y) = \mathbf{Sig}_{[0,s]}(X) \otimes \mathbf{Sig}_{[s,t]}(Y) \tag{2.17}$$

Proof. See Theorem 2.9 in [46]. □

Example 2.13. *Let X be linear on $[n, n+1]$ and let $X_{n+1} - X_n = \mathbf{x}_n$ for $n \in \mathbb{N}$, then*

$$\mathbf{Sig}_{[0,N]}(X) = \bigotimes_{n \leqslant N} (1, \mathbf{x}_n, \frac{\mathbf{x}_n^{\otimes 2}}{2!}, \cdots) \tag{2.18}$$

Example 2.14 (Linear controlled differential equation)**.** *Let $E = \mathbb{R}^d, W = \mathbb{R}^n$. let $X \in \mathcal{C}_0^1([0,T],E)$ and let $B \colon E \to \mathbf{L}(W)$ be a bounded linear map. Consider*

$$dY_t = B(dX_t)(Y_t), \quad Y_0 \in W \tag{2.19}$$

If we denote $B^k \coloneqq B(e_k)$, $k = 1, \cdots, d$ then

$$dY_t = \sum_{k=1}^{d} B^k(Y_t) dX_t^k, \quad Y_0 \in W. \tag{2.20}$$

It follows from Picard's iteration that

$$\begin{aligned} Y_t^n &= \left(I + \sum_{k=1}^{n} B^{\otimes k} \int_{t_1 < \cdots < t_k \in [0,t]} dX_{t_1} \otimes \cdots \otimes dX_{t_k} \right) Y_0 \\ &= \left(I + \sum_{k=1}^{n} \sum_{i_1, \cdots, i_k = 1}^{d} B^{i_k} \cdots B^{i_1} \int_{t_1 < \cdots < t_k \in [0,t]} dX_{t_1}^{i_1} \cdots dX_{t_k}^{i_k} \right) Y_0. \end{aligned} \tag{2.21}$$

Let the variation of $X \in \mathcal{C}_0^1([0,T],E)$ denoted by $\|X\|_{[0,T]}$, then

$$\left\| \int_{t_1 < \cdots < t_k \in [0,t]} dX_{t_1} \otimes \cdots \otimes dX_{t_k} \right\|_{E^{\otimes k}} \leqslant \frac{\|X\|_{[0,T]}^k}{k!}. \tag{2.22}$$

Therefore, Y_t^n converges to Y_t as $n \to \infty$ i.e.

$$\|Y_t - Y_t^n\|_W \leqslant \sum_{k>n} \frac{\|B\|_{\mathcal{L}(E,\mathcal{L}(W))}^k \|X\|_{[0,T]}^k}{k!} \leqslant \frac{\|B\|_{\mathcal{L}(E,\mathcal{L}(W))}^{n+1} \|X\|_{[0,T]}^{n+1}}{n!} \to 0, \quad \text{as } n \to \infty \tag{2.23}$$

and

$$Y_t = \left(I + \sum_{k=1}^{\infty} B^{\otimes k} \int_{t_1 < \cdots < t_k \in [0,t]} dX_{t_1} \otimes \cdots \otimes dX_{t_k} \right) Y_0. \tag{2.24}$$

In the language of signature

$$Y_t = \Big(\sum_{k=0}^{\infty} B^{\otimes k} \Big) \Big(\mathbf{Sig}_{[0,t]}(X) \Big) Y_0. \tag{2.25}$$

which implies that the solution of controlled SDE could be written as a linear function on signature stream of control path. This implies that signature stream is a promising feature for controlled ODE.

Remark. Similar result holds for smooth vector field with some additional boundness assumption and the proof is in alignment with the argument above, see [43].

Despite nice properties as feature map, signature feature suffers from dimension explosion w.r.t truncation order. Consider a path $X \in \mathcal{C}_0^1([0,T], \mathbb{R}^d)$ and the truncated signature of X up to order n lies in $\mathbf{T}^{(n)}(\mathbb{R}^d) \subseteq \bigoplus_{k \leqslant n} (\mathbb{R}^d)^{\otimes k}$ which has dimension $1 + d + \cdots + d^n = \frac{d^{n+1}-d}{d-1}$ growing exponentially w.r.t the truncation order n. Therefore, we would like to explore some low-rank approximations of signature feature.

2.2 Log-signature

From Proposition 2.8, we know that $\mathbf{Sig}_{[0,1]}$ is neither surjective nor the range of which a linear subspace of $\mathbf{T}(E)$. In Example 2.4, for $X_t = t\mathbf{x} \in \mathbb{R}^d$, then

$$\mathbf{Sig}_{[0,1]}(X) = \sum_{k \geqslant 0} \frac{\mathbf{x}^{\otimes k}}{k!} \tag{2.26}$$

which is the power series expansion of exponential function. Motivated by this, we curve the signature space by taking 'logarithm' which we defined below.

Definition 2.15. Define $\mathbf{T}_1(E) = \{\mathbf{a} \in \mathbf{T}(E) \colon \mathbf{a}_0 = 1\}$. Let $\mathbf{a} \in \mathbf{T}_1(E)$, we define the exponential be

$$\mathbf{exp}(\mathbf{a}) = \sum_{n \geqslant 0} \frac{\mathbf{a}^{\otimes n}}{n!}. \tag{2.27}$$

and we define the logarithm be

$$\mathbf{log}(\mathbf{a}) = \mathbf{log}(1 + \mathbf{t}) = \sum_{n \geqslant 1} \frac{(-1)^{n-1}}{n} \mathbf{t}^{\otimes \mathbf{n}}. \tag{2.28}$$

Lemma 2.16. $\mathbf{exp}(\cdot)$ *and* $\mathbf{log}(\cdot)$ *are inverse of each other on* $\mathbf{T}_1(E)$.

Proof. See Lemma 2.21 in [46]. □

Therefore, taking logarithm of signature loses no information.

Definition 2.17 (Log-signature)**.** We denote $\mathbf{LogSig}_J \colon \mathcal{C}_0^1(J,E) \to \mathbf{T}(E)$ the log-signature map such that for all $X \in \mathcal{C}_0^1(J,E)$

$$\mathbf{LogSig}_J(X) = \mathbf{log}\big(\mathbf{Sig}_J(X)\big) \tag{2.29}$$

Let $\mathbf{LogSig}_J^{(n)}$ denote the truncated signature map up to order n

$$\mathbf{LogSig}_J^{(n)}(X) = \pi_n\big(\mathbf{LogSig}_J(X)\big) \tag{2.30}$$

where π_n define in (2.14).

2.2.1 Lie Series

Besides storing the same information, thanks to the shuffle product property of signature, log-signature space is linear and admits a more concrete representation. Precisely speaking, log-signature space is a linear subspace of Lie formal series over E. Recall the Lie bracket $[\cdot,\cdot]$ on $T((E))$ such that

$$[\mathbf{a},\mathbf{b}] = \mathbf{a}\otimes\mathbf{b} - \mathbf{b}\otimes\mathbf{a} \tag{2.31}$$

If F_1 and F_2 are two linear subspaces of $T((E))$, let us denote by $[F1,F2]$ the linear span of all the elements of the form $[\mathbf{a},\mathbf{b}]$, where $\mathbf{a}\in F_1$ and $\mathbf{b}\in F_2$.

Definition 2.18 (Lie formal series)**.** We denote $\mathcal{L}((E))$ the space of Lie formal series over E such that

$$\mathcal{L}((E)) = \{\mathbf{l}\in T((E)) \colon \mathbf{l}_n \in L_n\} \tag{2.32}$$

where

$$L_0 = 0, L_1 = [E,E], L_2 = [E,L_1], \cdots, L_n = [E,L_{n-1}], \cdots \tag{2.33}$$

and denote $\mathcal{L}^{(n)}(E) = \pi_n(\mathcal{L}((E)))$ the truncated Lie series up to order n.

Theorem 2.19. *The range of log-signature is a linear subspace of Lie series over* E.

$$\mathbf{LogSig}(\mathcal{C}_0^1) \subseteq \mathcal{L}((E)). \tag{2.34}$$

Proof. See Theorem 2.23 in [46]. □

Theorem 2.20. *The range of truncated log-signature is the truncated Lie series over E up to the same order*

$$\mathbf{LogSig}^{(n)}(\mathcal{C}_0^1) = \mathcal{L}^{(n)}(E) \tag{2.35}$$

Proof. See Proposition 2.27 in [46]. □

With Lie series form, we compute the dimension of truncated log-signature space.

Proposition 2.21. *The dimension of the space of truncated log-signature up to order n is*

$$w(d,n) = \sum_{k=1}^{n} \frac{1}{k} \sum_{i|k} \mu\Big(\frac{k}{i}\Big) d^i \tag{2.36}$$

which is the Witt's formula and μ is the Möbius function.

Proof. See Corollary 4.14 in [49]. □

If we let $d = 5$ and $n = 5$, then $\dim(\mathbf{Sig}) = 3905$ while $\dim(\mathbf{LogSig}) = 829$. In summary, log-signature curves the signature space and reduces the redundancy without losing any information. However, there is no free lunch because log-signature loses the universality of signature, therefore requiring nonlinear models.

Example 2.22. *For $X_t = t\mathbf{x} \in \mathbb{R}^d$, then*

$$\mathbf{Sig}_{[0,1]}(X) = \mathbf{exp}(\mathbf{x}), \quad \mathbf{LogSig}_{[0,1]}(X) = \mathbf{x}. \tag{2.37}$$

Log-signature pick non-redundant information in signature.

2.3 Universality and characteristics

Recall the weak universality of signature, we restrict ourselves on a compact set of $\mathbf{Sig}(\mathcal{C}_0^1(J,E))$. To generalize the theorem beyond compactness, we would like to normalize the set $\mathbf{Sig}(\mathcal{C}_0^1(J,E))$ into a bounded ball. The first idea comes to mind is to normalize signatures by scaling them on $\mathbf{T}(E)$. However, because signature map is neither surjective nor the range of it a linear subspace, scaled tensor of a signature might not be signature. Thus, we drive ourselves out of the region of $\mathbf{Sig}(\mathcal{C}_0^1(J,E))$ where we have the shuffle product property. An alternative is to normalize the

signature by scaling the path. For $X \in \mathcal{C}_0^1(J, E)$ and $\lambda > 0$, the signature of the scaled path

$$\begin{aligned}\mathbf{Sig}_J(\lambda X)_k &= \int_{t_1 \leqslant \cdots \leqslant t_k \in J} d\lambda X_{t_1} \otimes \cdots \otimes d\lambda X_{t_k} \\ &= \lambda^k \int_{t_1 \leqslant \cdots \leqslant t_k \in J} dX_{t_1} \otimes \cdots \otimes dX_{t_m} = \lambda^k \mathbf{Sig}_J(X)_k\end{aligned} \tag{2.38}$$

Thus, the norm of the scaled path

$$\left\|\mathbf{Sig}_J(\lambda X)\right\|_{\mathbf{T}(E)}^2 = \sum_{k \geqslant 0} \lambda^{2k} \|\mathbf{Sig}_J(X)_k\|_{E^{\otimes k}}^2 \tag{2.39}$$

Therefore, we define the scaling map δ_λ on $\mathbf{T}(V)$ such that for all $\mathbf{a} \in \mathbf{T}(V)$

$$\delta_\lambda(\mathbf{a}) = (\lambda^k \mathbf{a}^k)_{k \geqslant 0} \tag{2.40}$$

Definition 2.23. A tensor normalization is a continuous injective map

$$\Lambda \colon \mathbf{T}(V) \to \{\mathbf{t} \in \mathbf{T}(V) \colon \|\mathbf{t}\|_{\mathbf{T}(E)} \leqslant K\} \tag{2.41}$$

$$\mathbf{t} \mapsto \delta_{\lambda(\mathbf{t})}(\mathbf{t}) \tag{2.42}$$

where $K > 0$ and $\lambda \colon \mathbf{T}(V) \to (0, \infty)$ a function.

The existence of tensor normalization is not trivial because of the nonlinear relationship between the scaling factor $\lambda(\mathbf{t})$ and the norm of scaled tensor $\|\delta_{\lambda(\mathbf{t})}(\mathbf{t})\|$ shown above. The proof of existence and the construction methodology can be found in [14].

Theorem 2.24 ([14])**.** *Let $\Lambda \colon \mathbf{T}(E) \to \mathbf{T}(E)$ be a tensor normalization. The normalized signature*

$$\Phi = \Lambda \circ \mathbf{Sig}_J \tag{2.43}$$

(i) is a continuous injection from $\mathcal{P}_0^1$ in to a bounded subset of $\mathbf{T}(E)$,

(ii) is universal to $C_b(\mathcal{P}_0^1, \mathbb{R})$, equipped with the strict topology,

(iii) is characteristic to the space of finite regular Borel measures on $\mathcal{P}_0^1$.

Proof. See Proposition 4.1 in [14]. □

Corollary 2.25. *Let X a stochastic process on $[0,1]$ and measurable $(\Omega, \mathcal{F})$. Let $\mathbb{P}$ and $\mathbb{Q}$ be two regular probability measures such that $X \in \mathcal{P}_0^1$ almost surely. Then*

$$\mathbb{E}_{\mathbb{P}}[\Phi(X)] = \mathbb{E}_{\mathbb{Q}}[\Phi(X)] \quad \textit{iff} \quad \mathbb{P} = \mathbb{Q} \tag{2.44}$$

Proof. Proof directly from (iii) of Theorem 2.24. □

Remark. If we consider the regular probability measure on time augmented path space $\mathcal{C}_0^1([0,T], \overline{E})$, similar result holds since $\mathcal{C}_0^1([0,T], \overline{E}) = \mathcal{P}_0^1$.

Definition 2.26 (Maximum mean distance)**.** We define the maximum mean distance (MMD) as

$$d_{\mathcal{G}}(\mathbb{P}, \mathbb{Q}) = \sup_{f \in \mathcal{G}} |\mathbb{E}_{\mathbb{P}}[f] - \mathbb{E}_{\mathbb{Q}}[f]| \tag{2.45}$$

where $\mathcal{G} \subseteq \mathbb{R}^{\mathcal{P}_0^1}$.

Let $\mathcal{G} = \{\langle \mathbf{l}, \Phi(\cdot) \rangle_{\mathbf{T}_1} : \mathbf{l} \in \mathbf{T}_1, \|\mathbf{l}\|_{\mathbf{T}_1} \leqslant 1\}$, then by Corollary 2.25

$$d_{\mathcal{G}}(\mathbb{P}, \mathbb{Q}) = 0 \quad \text{iff} \quad \mathbb{P} = \mathbb{Q} \tag{2.46}$$

which implies that a metric, see [23]. Moreover

$$\begin{aligned} d_{\mathcal{G}}(\mathbb{P}, \mathbb{Q}) &= \sup_{f \in \mathcal{G}} |\mathbb{E}_{\mathbb{P}}[f] - \mathbb{E}_{\mathbb{Q}}[f]| \\ &= \sup_{f \in \mathcal{G}} \Big| \mathbb{E}_{(X,Y) \sim \mathbb{P} \otimes \mathbb{Q}}[f(X) - f(Y)] \Big| \\ &= \sup_{\mathbf{l} \in \mathbf{T}_1, \|\mathbf{l}\|_{\mathbf{T}_1} \leqslant 1} \Big| \mathbb{E}_{(X,Y) \sim \mathbb{P} \otimes \mathbb{Q}} \big[\langle \mathbf{l}, \Phi(X) - \Phi(Y) \rangle_{\mathbf{T}_1} \big] \Big| \end{aligned} \tag{2.47}$$

Since $\mathbb{E}_{(X,Y) \sim \mathbb{P} \otimes \mathbb{Q}} \big[\Phi(X) - \Phi(Y) \big] \in \mathbf{T}_1$, then

$$d_{\mathcal{G}}(\mathbb{P}, \mathbb{Q}) = \mathbb{E}\big[\langle \Phi(X) - \Phi(Y), \Phi(X') - \Phi(Y') \rangle_{\mathbf{T}_1} \big] \tag{2.48}$$

where X, Y, X', Y' are independent with $X, X' \sim \mathbb{P}$ and $Y, Y' \sim \mathbb{Q}$

Definition 2.27. We denote $\mathbf{k_{Sig}} \colon \mathbf{T}_1 \times \mathbf{T}_1 \to \mathbb{R}$ the signature kernel such that

$$\mathbf{k_{Sig}}(\cdot, \cdot) = \langle \Phi(\cdot), \Phi(\cdot) \rangle_{\mathbf{T}_1} \tag{2.49}$$

Then we can rewrite the MMD as

$$d_{\mathcal{G}}(\mathbb{P}, \mathbb{Q}) = \mathbb{E}\big[\mathbf{k_{Sig}}(X, X') \big] - 2\mathbb{E}\big[\mathbf{k_{Sig}}(X, Y) \big] + \mathbb{E}\big[\mathbf{k_{Sig}}(Y, Y') \big] \tag{2.50}$$

where X, Y, X', Y' are independent with $X, X' \sim \mathbb{P}$ and $Y, Y' \sim \mathbb{Q}$. Similar to the kernel trick in machine learning, the kernel representation of MMD provides a very efficient way evaluating the metric, which we will elaborate in the next section.

2.4 Discrete signature feature

Let $X \in \mathcal{C}_0^1([0,1],E)$ and consider a discrete sequence $\mathbf{X} = (X_0, X_{1/N}, \cdots, X_1) \in E^{N+1}$. We extend the concept of signature on discrete sequence E^{N+1} by computing signature of the linear interpolation of sequence. From Example 2.13 we know the signature of piecewise-linear path is

$$\mathbf{exp}(\mathbf{X}_1 - \mathbf{X}_0) \otimes \cdots \otimes \mathbf{exp}(\mathbf{X}_N - \mathbf{X}_{N-1}) \tag{2.51}$$

Therefore we define the discrete signature as follows

Definition 2.28. Let $\mathbf{X} = (\mathbf{X}_i)_{i=0}^N \in E^{N+1}$ and let $\triangle\mathbf{X}_i = \mathbf{X}_i - \mathbf{X}_{i-1}$. We denote $\mathbf{sig}_{[0,N]}$ the discrete signature of order m and depth n such that

$$\mathbf{sig}_{[0,N]}(\mathbf{X}) = \pi_n\bigg(\bigotimes_{i=1}^N \pi_m\Big(\mathbf{exp}\big(\triangle\mathbf{X}_i\big)\Big)\bigg) \tag{2.52}$$

In particular, if $m = 1$ we call it growing discrete signature

$$\begin{aligned}\mathbf{sig}_{[0,N]}(\mathbf{X}) &= \pi_n\bigg(\prod_{i=1}^N \big(1 + \triangle\mathbf{X}_i\big)\bigg) \\ &= \sum_{k=0}^n \sum_{i_1<\cdots<i_k=1}^N \triangle\mathbf{X}_{i_1} \otimes \cdots \otimes \triangle\mathbf{X}_{i_k}.\end{aligned} \tag{2.53}$$

If $m > 1$, we call it high order discrete signature, and if $m \geqslant n$, we call it truncated discrete signature

$$\begin{aligned}\mathbf{sig}_{[0,N]}(\mathbf{X}) &= \pi_n\bigg(\prod_{i=1}^N \mathbf{exp}\big(\triangle\mathbf{X}_i\big)\bigg) \\ &= \sum_{k=0}^n \sum_{i_1\leqslant\cdots\leqslant i_k=1}^N \triangle\mathbf{X}_{i_1} \otimes \cdots \otimes \triangle\mathbf{X}_{i_k}\end{aligned} \tag{2.54}$$

Lemma 2.29. *Let $X \in \mathcal{C}_0^1([0,1],E)$ and $\mathbf{X} = (X_{i/N})_{i=0}^N \in E^{N+1}$, then*

$$\|\mathbf{Sig}_{[0,1]}(X) - \mathbf{sig}_{[0,N]}(\mathbf{X})\|_{\mathbf{T}_1} \to 0 \quad as \quad N \to 0 \tag{2.55}$$

Proof. See Theorem 5 in [39]. □

Remark. If we let X to be the Brownian motion (not in $\mathcal{C}_0^1$), the growing discrete signature converges to the signature defined with Ito integral, while the high order discrete signature converges to the signature defined with Stratonovich integral. More generally, if we want to approximate a geometric p-rough path, we need high order discrete signature at least order $\lfloor p \rfloor$, see [39].

2.4.1 Kernel trick

In the same fashion of horner algorithm

$$\begin{aligned} &a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_n x^n \\ &\quad = a_0 + x\Big(a_1 + x\Big(a_2 + x\big(a_3 + \cdots + x(a_{n-1} + x\, a_n)\cdots\big)\Big)\Big). \end{aligned} \tag{2.56}$$

we can reduce the computational complexity of discrete signature by dynamic programming principle.

Proposition 2.30. *Let* $\mathbf{X} = (\mathbf{X}_i)_{i=0}^N \in E^{N+1}$ *and let* $\mathbf{sig}_{[0,N]}$ *be the growing signature*

$$\begin{aligned} \mathbf{sig}_{[0,N]}(\mathbf{X}) &= \mathbf{sig}_{[0,N-1]}(\mathbf{X}) \otimes (1 + \triangle \mathbf{X}_N) \\ &= 1 + \sum_{i_1=1}^{N} \triangle \mathbf{X}_{i_1} \Big(1 + \sum_{i_2=i_1+1}^{N} \triangle \mathbf{X}_{i_2}(\cdots)\Big) \end{aligned} \tag{2.57}$$

This is too surprise because we just rewrite the Chen's identity. However, this greatly reduce the complexity when computing the kernel.

Definition 2.31. Let $\mathbf{sig}_{[0,N]}$ be the discrete signature, then we denote $\mathbf{k}_{\mathbf{sig}}$ the discrete signature kernel such that

$$\mathbf{k}_{\mathbf{sig}}(\cdot,\cdot) = \langle \mathbf{sig}_{[0,N]}(\cdot), \mathbf{sig}_{[0,N]}(\cdot) \rangle_{\mathbf{T}_1} \tag{2.58}$$

Proposition 2.32. *Let* $\mathbf{X} = (\mathbf{X}_i)_{i=0}^N \in E^{N+1}$ *and let* $\mathbf{Y} = (\mathbf{Y}_i)_{i=0}^N \in E^{N+1}$. *Let* $\mathbf{k}_{\mathbf{sig}}$ *be the kernel of discrete growing signature*

$$\mathbf{k}_{\mathbf{sig}}(\mathbf{X},\mathbf{Y}) = 1 + \sum_{\substack{i_1=1 \\ j_1=1}}^{N} \langle \triangle \mathbf{X}_{i_1}, \triangle \mathbf{Y}_{j_1} \rangle \Big(1 + \sum_{\substack{i_2=i_1+1 \\ j_2=j_1+1}}^{N} \langle \triangle \mathbf{X}_{i_2}, \triangle \mathbf{Y}_{j_2} \rangle (\cdots)\Big) \tag{2.59}$$

Remark. Similar formulas hold for high order discrete signature and kernel, see [39].

This result is important for numerical implementation in two aspects. First is that we avoid computing the tensor but the inner product directly by kernel trick. If somehow (for better characteristic capacity) you would like lift paths to a RKHS space (H, κ) before applying the signature kernel, this formula avoids explicitly compute $\triangle \kappa_X$ and $\triangle \kappa_Y$. but approximating with

$$\langle \triangle \kappa_{X_i}, \triangle \kappa_{Y_j} \rangle \approx \kappa(X_{i+1}, Y_{j+1}) + \kappa(X_i, Y_j) - \kappa(X_i, Y_{j+1}) - \kappa(X_{i+1}, Y_j), \tag{2.60}$$

which reduce a potentially infinite computational complexity if (H, κ) is an infinite dimensional space, see [14]. Secondly, the recursive structure in time implies that computing the inner product of signature stream is as cheap as computing the inner product of signature. This recursive feature can also be used for more general inner product between tensor and discrete signature.

2.5 Low-rank-signature

We extend the recursive method from discrete signature to tensor with similar recursive structure. Let $\mathbf{l} = (\mathbf{l}_k)_{k=0}^{\infty} \in \mathbf{T}_1(E)$, let $\mathbf{X} = (\mathbf{X}_i)_{i=0}^{N} \in E^{N+1}$,and let $\mathbf{sig}_{[0,N]}$ be the growing signature. If $\mathbf{l}_k = \mathbf{l}_{k-1} \otimes l_k$ then

$$\begin{aligned}
\langle \mathbf{l}, \mathbf{sig}_{[0,N]}(\mathbf{X}) \rangle_{\mathbf{T}_1(E)} &= \sum_{k \geqslant 0} \langle \mathbf{l}_k, \mathbf{sig}^{(k)}_{[0,N]}(\mathbf{X}) \rangle_{\mathbf{T}_1(E)} \\
&= \sum_{k \geqslant 0} \sum_{i=1}^{N} \langle \mathbf{l}_{k-1}, \mathbf{sig}^{(k-1)}_{[0,i-1]}(\mathbf{X}) \rangle_{\mathbf{T}_1(E)} \cdot \langle l_k, \triangle \mathbf{X}_i \rangle_{\mathbf{T}_1(E)} \\
&= 1 + \sum_{\substack{i_1=1 \\ j_1=1}}^{N} \langle l_{i_1}, \triangle \mathbf{X}_{j_1} \rangle \Big(1 + \sum_{\substack{i_2=i_1+1 \\ j_2=j_1+1}}^{N} \langle l_{i_2}, \triangle \mathbf{X}_{j_2} \rangle (\cdots) \Big)
\end{aligned} \tag{2.61}$$

This is exactly the formula (2.59) in Proposition 2.32. In the intermediate step of computing the inner product, we have obtained the value of $\langle \mathbf{l}_k, \mathbf{sig}^{(k)}_{[0,N]}(\mathbf{X}) \rangle_{\mathbf{T}_1(E)}$ for all k. Therefore, if we consider a sequence of L many tensors such that $\mathbf{l}_k = \mathbf{l}_{k-1} \otimes l_k$, then we can leverage the recursive structure in order and compute their inner product with signature in linear computational complexity w.r.t L.

Remark. This result can also be generalized to higher order case by considering the recursive algorithm computing high order discrete signature and kernel, see [39]. However, in the paper [54] introducing low-rank tensor projection of ordered sequential data, only first order recursive structure is considered. We believe that higher order recursive structure also exists and we wish to prove this claim in further research.

If we only assume $\mathbf{l}_k = l_{k,1} \otimes \cdots \otimes l_{k,k}$, then we lose the recursive in order, but still we have the recursive in time of discrete signature and the kernel trick.

$$\langle \mathbf{l}_k, \mathbf{sig}^{(k)}_{[0,N]}(\mathbf{X}) \rangle_{\mathbf{T}_1(E)} = \sum_{i=1}^{N} \langle l_{k,1} \otimes \cdots \otimes l_{k-1,k}, \mathbf{sig}^{(k-1)}_{[0,i-1]}(\mathbf{X}) \rangle_{\mathbf{T}_1(E)} \cdot \langle l_{k,k}, \triangle \mathbf{X}_i \rangle_{\mathbf{T}_1(E)} \tag{2.62}$$

Definition 2.33. We call $\mathbf{l} \in \mathbf{T}_1$ rank-1 tensor if

$$\mathbf{l} = l_1 \otimes \cdots \otimes l_k, \quad \text{for some } k \geqslant 1 \tag{2.63}$$

Similar to log-signature, the inner product between truncated signature and rank-1 tensor map the signature space to a low dimensional space $\mathbb{R}$. This projection is linear, computational cheap, and more importantly trainable. The choice of rank-1 tensors gives us the freedom to choose the output space dimension as well as the direction of projection.

Definition 2.34. Let $(\mathbf{l}_l)_{l=0}^L$ be a sequence of rank-1 tensors in $\mathbf{T_1}$ and let $\mathbf{sig}_{[0,N]}$ the discrete signature, then we denote $\mathbf{LRsig}_{[0,N]}$ the low-rank signature such that

$$\mathbf{LRsig}_{[0,N]} = \left(\langle \mathbf{l}_l, \mathbf{sig}_{[0,N]} \rangle_{\mathbf{T}_1(E)}\right)_{l=0}^L \in \mathbb{R}^L \tag{2.64}$$

and we define the low-rank signature stream $(\mathbf{LRsig}_{[0,i]})_{i=0}^N$.

From the recursive algorithm above, we know that computing the low-rank signature stream is as cheap as computing the low-rank signature. Unlike signature stream, low-rank signature stream still in a low dimensional space, so we may again apply the low-rank signature stream to it. Thus, we may stack many levels of signature stream on the original path and this has been proved successful dealing with sequential data, see [54].

3. Variational Autoencoders

In this chapter, we give a brief introduction to the variational autoencoders (VAEs) [37] and some variations of VAE such as conditional VAEs (CVAEs) [52], $\beta-$VAEs [30], student-VAEs [2], and gaussian process VAEs (GP-VAEs) [21]. Variational autoencoders is a generative model which simulates how the data is generated under certain presumed distribution. In short, VAEs are variational bayesian inference on latent variable models with likelihood and posterior described by neural networks. In the scenario of generating time series, there are in general two frameworks to apply variational autoencoders. First is to assume a bayesian model with both observable variables and latent variables taking values in space of time series [21] [54], for example taking latent space as a gaussian process in GP-VAEs. Another approach is to first consider an appropriate feature map from time series to a finite dimensional features space and apply VAE on this feature space. After the training of VAE on features space, we generate new samples on the feature space and transform those samples on feature space back to time series with the inverse map of the feature map [13]. In this thesis, we focus on the second one. More details and background information on VAEs can be found in [38].

3.1 Variational autoencoders

Variational autoencoders are variational inference on deep latent variable model. Deep latent variable models consist of observable random variable x, latent random variable z, and parameter θ. The joint distribution $p_\theta(x, z)$ of the deep latent variable model is parameterized by neural networks. For example, we may choose z be standard normal distribution and x a normal distribution with mean $\mu = \mathbf{NN}(z)$ and variance 1, where $\mathbf{NN}$ is a neural network. Moreover, we call $p_\theta(z) = p(z)$ the *prior* (independent with θ), $p_\theta(z|x)$ the *posterior*, $p_\theta(x|z)$ the *conditional distribution* and $p_\theta(x)$ the *marginal likelihood*. The big advantage of deep latent variable model

is that even when the prior $p(z)$ and conditional distribution $p(x|z)$ are explicit and simple, the marginal likelihood $p_\theta(x)$ can be very expressive due to the universal approximating capacity of neural network. However, the price to pay for expressive marginal likelihood is the intractability of marginal likelihood, namely having no analytic solution or efficient estimator for it. By Bayes' rule

$$p_\theta(x) = \frac{p_\theta(x,z)}{p_\theta(z|x)}.$$

The joint distribution is not hard to write down since

$$p_\theta(x,z) = p\big(x|\mathbf{NN}_\theta(z)\big)p(z)$$

but the posterior $p_\theta(z|x)$ in general has no analytic solution or efficient estimator. Therefore, we need to leverage the idea of variational inference, introducing a distribution $q_\phi(z|x)$ approximating the true posterior $p_\theta(z|x)$.

3.1.1 Variational inference

Let $x_1, \cdots, x_n$ be i.i.d. observable samples drawn from a random variable x. Maximizing the log-marginal likelihood by Bayes's rule, we obtain

$$\begin{aligned}
\log p_\theta(x_i) &= \log\Big(\frac{p_\theta(x_i,z)}{p_\theta(z|x_i)}\Big) \\
&= \log\Big(\frac{p_\theta(x_i,z)}{q_\phi(z|x_i)}\frac{q_\phi(z|x_i)}{p_\theta(z|x_i)}\Big) \\
&= \mathbb{E}_{q_\phi(z|x_i)}\bigg[\log\Big(\frac{p_\theta(x_i,z)}{q_\phi(z|x_i)}\Big) + \log\Big(\frac{q_\phi(z|x_i)}{p_\theta(z|x_i)}\Big)\bigg] \\
&= \mathbb{E}_{q_\phi(z|x_i)}\bigg[\log\Big(\frac{p_\theta(x_i,z)}{q_\phi(z|x_i)}\Big) + \log\Big(\frac{q_\phi(z|x_i)}{p_\theta(z|x_i)}\Big)\bigg] \\
&= \mathbb{E}_{q_\phi(z|x_i)}\bigg[\log\Big(\frac{p_\theta(x_i,z)}{q_\phi(z|x_i)}\Big)\bigg] + KL\big(q_\phi(z|x_i)||p_\theta(z|x_i)\big) \\
&\geqslant \mathbb{E}_{q_\phi(z|x_i)}\bigg[\log\Big(\frac{p_\theta(x_i,z)}{q_\phi(z|x_i)}\Big)\bigg]
\end{aligned} \tag{3.1}$$

Let us denote that $\mathcal{L}_{\theta,\phi}(x)$ the *evidence lower bound* (ELBO) such that

$$\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x)}\bigg[\log\Big(\frac{p_\theta(x,z)}{q_\phi(z|x)}\Big)\bigg] \tag{3.2}$$

Notice that

$$\log p_\theta(x) = \max_\phi \mathcal{L}_{\theta,\phi}(x). \tag{3.3}$$

Thus, we can solve the maximization problem of ELBO instead of the marginal likelihood. Also if

$$q^*_\phi(z|x) \in \arg\max \mathcal{L}_{\theta,\phi}(x) \tag{3.4}$$

then it satisfies that $q^*_\phi(z|x) = p_\theta(z|x)$ because

$$KL\big(q_\phi(z|x_i)\big|\big|p_\theta(z|x_i)\big) = 0 \quad \text{iff} \quad q_\phi(z|x) = p_\theta(z|x). \tag{3.5}$$

Therefore maximizing the ELBO also gives us the true posterior. Moreover, observe that

$$\begin{aligned}
\mathcal{L}_{\theta,\phi}(x) &= \mathbb{E}_{q_\phi(z|x)}\left[\log\Big(\frac{p_\theta(x,z)}{q_\phi(z|x)}\Big)\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log\Big(\frac{p_\theta(x|z)p(z)}{q_\phi(z|x)}\Big)\right] \\
&= -KL\big(q_\phi(z|x)\big|\big|p_\theta(z)\big) + \mathbb{E}_{q_\phi(z|x)}\big[p_\theta(x|z)\big].
\end{aligned} \tag{3.6}$$

Thus, we are minimizing the KL-distance between approximating posterior and true prior, meanwhile maximizing expected conditional distribution under approximating posterior distribution.

3.1.2 Reparameterization Trick

In order the solve the variational problem, we compute the gradient of the objective function w.r.t θ and ϕ. However, the gradient w.r.t ϕ is in general hard to compute because the expectation also depends on ϕ.

$$\begin{aligned}
\nabla_\phi \mathcal{L}_{\theta,\phi}(x) &= \nabla_\phi \mathbb{E}_{q_\phi(z|x)}\big[\log p_\theta(x,z) - \log q_\phi(z|x)\big] \\
&\neq \mathbb{E}_{q_\phi(z|x)}\big[\nabla_\phi(\log p_\theta(x,z) - \log q_\phi(z|x))\big]
\end{aligned} \tag{3.7}$$

Therefore, we need to factorize the randomness taking expectation out of the parameter ϕ. So we consider

$$\tilde{z} = g(\epsilon, \phi, x) \tag{3.8}$$

where g is a differentiable function and ϵ is independent with ϕ and x. In this case, for all differentiable function f

$$\begin{aligned}\nabla_\phi \mathcal{L}_{\theta,\phi}(x) &= \nabla_\phi \mathbb{E}_{q_\phi(z|x)}\Big[f(x,z)\Big] \\ &= \nabla_\phi \mathbb{E}_{g(\epsilon,\phi,x))}\Big[f(x,z)\Big] \\ &= \nabla_\phi \mathbb{E}_\epsilon\Big[f\big(x,g(\epsilon,\phi,x)\big)\Big] \\ &= \mathbb{E}_\epsilon\Big[\nabla_\phi f\big(x,g(\epsilon,\phi,x)\big)\Big]\end{aligned} \tag{3.9}$$

Let $\mathbf{NN}_\phi$ and $\mathbf{NN}_\theta$ be two neural networks, then by reparameterization trick, we define

$$\tilde{z} = g(\epsilon, \mathbf{NN}_\phi(x)) \tag{3.10}$$

Thus, we compute the gradient by backward propagation and Monte Carlo

$$\begin{aligned}\nabla_\phi \mathcal{L}_{\theta,\phi}(x) &= \mathbb{E}_\epsilon\Big[\nabla_\phi f\big(x,g(\epsilon,\mathbf{NN}_\phi(x))\big)\Big] \\ &\approx \frac{1}{L}\sum_{i=1}^{L}\nabla_\phi f\big(x,g(\epsilon_i,\mathbf{NN}_\phi(x))\big)\end{aligned} \tag{3.11}$$

In some case, the KL-divergence term even has a analytic formula.

Example 3.1. *Consider a deep latent variable model with gaussian prior and gaussian conditional distribution:*

$$\begin{gathered}q_\phi(z|x) = \mathcal{N}\Big(z;\mu_x,\sigma_x^2\Big) \\ \mu_x = [\mu_1,\cdots,\mu_d]^T, \quad \sigma_x^2 = Diag([\sigma_1^2,\cdots,\sigma_d^2]) \\ p_\theta(z) = \mathcal{N}\Big(z;0,I_d\Big)\end{gathered} \tag{3.12}$$

Then

$$\begin{aligned}-KL\big(q_\phi(z|x)\big|\big|p_\theta(z)\big) &= \mathbb{E}_{q_\phi(z|x)}\big[\log p_\theta(z)\big] - \mathbb{E}_{q_\phi(z|x)}\big[\log q_\phi(z|x)\big] \\ &= \int \log p_\theta(z) q_\phi(z|x) dz - \int \log p_\theta(z|x) q_\phi(z|x) dz \\ &= \frac{1}{2}\sum_{i=1}^{d}\Big(1+\log(\sigma_i^2) - \mu_i^2 - \sigma_i^2\Big)\end{aligned} \tag{3.13}$$

3.2 Variations of VAEs

The classical VAEs assume gaussian prior and ELBO loss. There are multiple variations of classical VAEs suitable for different scenario. In this section we briefly introduce some variations of VAEs which we use as a block of our market generator introduced in the next chapter.

3.2.1 Conditional VAEs

Condition VAEs [52] are typically used when your data points are labeled and you want to generate data points in a specific class. In financial time-series generating task, we aim at generating the distribution of future stock process given the information known, namely the conditional distribution of time-series. Therefore, we apply the conditional VAEs with conditions being the information known and data being the time series. Let c be conditions, x be time series, and z be latent variable. Let $\mathbf{NN}_\phi$ and $\mathbf{NN}_\theta$ be two neural networks. Then conditional VAEs has the following structure

$$\begin{aligned} q_\phi(z|x,c) &= q\Big(z; \mathbf{NN}_\phi(x,c)\Big) \\ p_\theta(z|c) &= p(z) \\ p_\theta(x|z,c) &= p\Big(x; \mathbf{NN}_\theta(z,c)\Big) \end{aligned} \tag{3.14}$$

$$\max_{\theta,\phi} \sum_{i=1}^{m} \mathcal{L}_{\theta,\phi}(x_i) = \max_{\theta,\phi} \sum_{i=1}^{m} \mathbb{E}_{q_\phi(z|x_i,c_i)} \left[\log \left(\frac{p_\theta(x_i, z|c_i)}{q_\phi(z|x_i,c_i)} \right) \right] \tag{3.15}$$

Or we can let $y = (x, c)$ then

$$\max_{\theta,\phi} \sum_{i=1}^{m} \mathcal{L}_{\theta,\phi}(y_i) = \max_{\theta,\phi} \sum_{i=1}^{m} \mathbb{E}_{q_\phi(z|y_i)} \left[\log \left(\frac{p_\theta(y_i, z|c_i)}{q_\phi(z|y_i)} \right) \right] \tag{3.16}$$

This is mathematically the same by adding the reconstruction loss between c and reconstructed c, which is zero, because identity gives reconstruction. However, it becomes different if the reconstruction of c and x share some layers of neural network. In this case, there is a tradeoff between expressive capacity and zero reconstruction loss of c.

3.2.2 $\beta-$VAEs

$\beta-$VAEs [30] add an adjustable parameter β to balance the KL-divergence term and the likelihood term in the ELBO loss. We denote $\mathcal{L}^{\beta}_{\theta,\phi}$ the β-ELBO loss such that

$$\mathcal{L}_{\theta,\phi} = -\beta KL\big(q_\phi(z|x)\big|\big|p_\theta(z)\big) + \mathbb{E}_{q_\phi(z|x)}\big[p_\theta(x|z)\big]. \tag{3.17}$$

An interpretation of β-VAEs is that $\mathcal{L}_{\theta,\phi}$ is a log likelihood with penalty on KL-distance between posterior and prior. This similar to Ridge regression which maximizes the log likelihood with penalty on L^2-distance between parameters to zero. β-VAEs is also closely related to the information bottleneck principle, see [30].

3.2.3 Student-VAEs

In financial scenario, heavy tail distribution are commonly observed in the returns of stock prices. Also, normal priors could be outlier-resistant and never reject outliers in data modeling. However Student's t-distributions can generate heavy tails is used in conditional distribution and can reject outliers if used as prior. This motivates us to implement a VAE with Student's t distribution in prior [2] as well as conditional distribution. Recall student's t distribution

$$p(z;\mu,\sigma,\nu) = \frac{\Gamma(\frac{\mu+p}{2})}{\Gamma(\frac{\mu}{2})\sqrt{(\pi\nu)^p}\sigma}\left(1+\frac{(z-\mu)^T\sigma^{-2}(z-\mu)}{\nu}\right)^{-\frac{\nu+p}{2}} \tag{3.18}$$

We consider the following variational autoencoder

$$\begin{aligned} q_\phi(z|x) &= \mathrm{St}\Big(z;\mathbf{NN}_{\phi,\mu}(x),\mathbf{NN}_{\phi,\sigma}(x),\nu_\theta\Big) \\ p_\theta(z) &= \mathrm{St}\Big(z;0,I_d,\nu_\theta\Big) \\ p_\theta(x|z) &= \mathrm{St}\Big(x;\mathbf{NN}_\theta(z),I_d,1\Big) \end{aligned} \tag{3.19}$$

In this case we have no simple expression of ELBO as normal case, but still we have an analytic expression of KL-divergence term, see [1]. More general, we can consider when prior is stacked independent student-t distribution with different parameter ν, i.e.

$$\begin{aligned} q_\phi(z|x) &= \prod_i \mathrm{St}\Big(z;\mathbf{NN}_{\phi,\mu_i}(x),\mathbf{NN}_{\phi,\sigma_i}(x),\mathbf{NN}_{\phi,\nu_i}(x)\Big) \\ p_\theta(z) &= \prod_i \mathrm{St}\Big(z;0,1,\nu_{\theta,i}\Big) \\ p_\theta(x|z) &= \mathrm{St}\Big(x;\mathbf{NN}_\theta(z),I_d,1\Big) \end{aligned} \tag{3.20}$$

In this case, we no longer have a analytic formula of the KL-divergence term, so we can use monte carlo to approximate the expectation.

3.2.4 Gaussian process VAEs

All variations of VAEs above are designed for samples and latent variables on Euclidean space. Gaussian process VAEs [21]generalize the idea of variational autoencoder by choosing a latent time series space with prior to be a distribution of gaussian process. Let $(\mathbf{x}_t)_{t\in J} = \mathbf{x} \in \text{Seq}(\mathbb{R}^d)$ be a time series random variable and $(\mathbf{z}_t)_{t\in J'} = \mathbf{z} \in \text{Seq}(\mathbb{R}^{d'})$ be a latent time series random variable. We consider the following model

$$\begin{aligned} \mu_i &= \mathbf{NN}_{\phi,\mu,i}(\mathbf{x}) \\ \Lambda_i &= \mathbf{NN}_{\phi,\Lambda,i}(\mathbf{x}) \\ q_\phi(\mathbf{z}|\mathbf{x}) &= \prod_i \mathcal{GP}(\mathbf{z}; \mu_i, \Lambda_i^{-1}) \\ p_\theta(\mathbf{z}) &= \mathcal{GP}(\mathbf{z}; m(\cdot), k(\cdot,\cdot)) \\ p_\theta(\mathbf{x}_t|\mathbf{z}) &= \mathcal{N}\Big(\mathbf{x}_t; \mathbf{NN}_\theta(\mathbf{z}), \sigma^2 I_d\Big) \end{aligned} \tag{3.21}$$

The encoding part from $\mathbf{x}$ to $\mathbf{z}$ is almost the same as VAEs but instead of generating independent normal random variables, we generate independent gaussian process and stack them to multi-dimensional gaussian process. The decoding part is applying a neural network transformation to the latent time series. The prior can be chosen as any gaussian process. An observation in [21] is that classical RBF kernel does not reflect the dynamics of data very well. However a mixture kernel

$$\begin{aligned} &k(\lambda|\alpha,\beta,\lambda) = \int p(\lambda|\alpha,\beta) k_{RBF}(r,\lambda) d\lambda^{(\alpha-1)} \\ &k_{RBF}(r,\lambda) = \exp(-\lambda r^2/2) \\ &p(\lambda|\alpha,\beta) \propto \lambda \exp(\alpha\lambda/\beta) \end{aligned} \tag{3.22}$$

is successful if used in the context of robust dynamic topic modeling where similar multi-scale time dynamics occurs.

4. Market Generator

In this chapter, we combine signature features and variational autoencoders introduced above to construct a market generator. Recall our problem setting: by observing one realization of stochastic process, we wish to infer the distribution of the process and stimulate paths under the learnt distribution. For example, given the S&P index of last 12 months, we wish to construct market generator which generates possible paths of S&P index of next months for us, and hopefully generating paths of S&P index to the next two months. Our algorithm can be subdivided into the following five parts, of which we present a brief overview below:

(Step 1): **Time series splitting:** Split single realization of path $(x_{0:N})$ to multiple subpaths $(x_{i:i+n})_{i=1}^{m}$. This subdivision helps to generate multiple samples for training.

(Step 2): **Path to signature features:** Applying the signature feature map to m subpaths $(x_{i:i+n})_{i=1}$ gives m samples on $\mathbb{R}^d$, where d depends on the choice of signature feature map.

(Step 3): **Variational autoencoder on signature features:** Train a variational autoencoder on the signature feature space and generate samples on the signature feature space.

(Step 4): **Signature features to path:** Inverse samples on signature feature space to samples on paths space.

(Step 5): **Evaluation of model:** Compare the distribution of generated paths to true paths.

4.1 Time series splitting

In many generative machine learning tasks, such as images generating, abundant of data points are available for training. However, for financial time series, scarcity of data can be a systemic problem due to the following reasons. Firstly, financial time series data has time inconsistency problem. Time inconsistency problem is well studied in behavioral economics where preferences change from time to time. Inconsistency in dynamic implies that if we use very old data, we are actually using data from a different dynamic, in another word, samples from a different distribution [16]. This is not a problem in image classification because a image of cat drawn 100 years before also looks like a cat today, but the dynamic of stocks market 100 years before can be greatly different from the one today. The second problem is that financial data usually suffers from roughness in paths. Therefore, a frequent observation results in a rougher path which leads to noise as well as difficulties in stable numerical methods. The last problem is that usually we only have one realization of path in financial scenario, unlike speech generating multiple experiments can be conducted. Therefore, if we need i.i.d samples, we need to split the path and assume further stationary condition to guarantee the independence among samples. If we consider stock prices under Black-scholes model

$$S_t = S_0 \exp\left\{ (\mu - \frac{\sigma^2}{2})t + \sigma W_t \right\}$$

and if we consider the log-return X_t i.e.

$$X_t = (\mu - \frac{\sigma^2}{2})t + \sigma W_t$$

then we consider the difference of log return

$$X_{t+\triangle t} - X_t = (\mu - \frac{\sigma^2}{2})\triangle t + \sigma(W_{t+\triangle t} - W_t).$$

Since Brownian motion has increment increment, splitting the difference of log return gives i.i.d samples even we choose length of subpath to be 1. Inspired by this idea, we preprocess the stock price by taking the difference of log return before splitting. Given the preprocessed path, the longer we choose the length of subpath, fewer samples we have. The more samples we have, the more we violate the independence. Therefore, there is actually a tradeoff in the choice of length of subpaths. In our experiments, we subdivide the time series into intervals: (i) 1 day (ii) 1 week (iii) 1 month. Mathematically speaking, given a path $x_{0:N}$ of with $N+1$ times observation.

$$x_i = x_{t_{i-1}:t_i}, \quad i = 1, \cdots, m$$

where $0 = t_0 \leqslant t_1 \leqslant \cdots \leqslant t_{m+1} = N$. Note that this is not a equal length division since days in months different and there might be some days data missing. However, this inconsistency in division as well as missing data is not a problem to our model, because we will later consider the log-signature feature of each subpath. The dimension of log-signature only depends on our truncation order and the calculation of log-signature is robust to missing data.

4.2 Path to signature features

After splitting path to samples of subpaths, we apply a feature map on each subpath and here our feature map is the log-signature. We choose signature based feature and in particular the log-signature motivated by the following reasons

(i) Signature type features stores path information in an efficient way by Example 2.14.

(ii) Expected normalized signature characterize the law of stochastic process by Corollary 2.25.

(iii) Signature can be directly used to compute signature kernel evaluating the MMD distance between distribution on paths by (2.48).

(iv) Log-signature has the same information as signature but store them in a more compact way, by Proposition 2.21.

(v) The image of signature space is not a linear subspace of tensor space but the range of truncated log-signature is the truncated Lie series over E up to the same order by Theorem 2.20.

(vi) Tensor-algebra exponential of the generated log-signatures recover the group-like (shuffle product) property of signature, see [46].

(vii) The numerical log-signature is more robust than signature.

(viii) Signature features eliminate pricing and hedging ambiguities problem addressed in and signature can be directly used for pricing, see [11].

We additional apply a lead-lag transformation [20] to path before computing the log-signature features of the subpath. This is because the signature of a path after lead-lag transformation helps to capture the volatility of a path, which is of vital importance in finance.

4.3 Variational autoencoder on signature features

Given m many samples on log-signature space denoted by $(X_i)_{i=1}^m$, we build a generative model and train it with samples. We feed log-signatures of samples in to the variational autoencoder for training. We consider student-t prior in order to recover a heavy-tail distribution observed commonly in financial data. Also, we optimize the β-ELBO loss in training to balance the reconstruction loss and KL-divergence. If we further want to learn the conditional distribution w.r.t previous information, we can apply conditional VAEs to generate conditional distribution. In the numerical implementation, we also include a signature computing layer in the decoded part of VAE in order to make sure the output of VAE lie in the right subset. Details of configuration of VAE are introduced in the next chapter as well as the source code in Github repository: `https://github.com/justinhou95/market_simulator`.

4.4 Signature features to path

After the training of model, we generate samples on the log-signature space. Now we need to inverse a log-signature to a path. The signature of a path uniquely determines the path itself up to tree-like equivalence but reverting signature to path is a computationally highly non-trivial task and currently a topic of active research. In [13], they develop an evolutionary algorithm by mimicking to some extent biological evolution. They start with an initial population of random paths, and iteratively select the paths whose signatures are closest to the target signature, and breed these paths and introduce mutations to generate a new generation of paths until we get a population of paths whose signature are close to the target signature. However, the revolution algorithm is very slow in optimization and reverting task of each signature need a restart of the evolution procedure. Therefore, we develop a new neural network based solution to inverse the signature. Our method is much faster than evolution algorithm and can be trained to deal with a collection of signatures once trained. Our approach is to construct an autoencoder with encoder be a neural network and decoder be the log-signature transformation. We train our autoencoder by minimizing the L^2 error between input log-signature with the output of autoencoder. Once trained, we consider the encoder as a good inverse function of log-signature. This method can be trained w.r.t one samples in order to reverse this single sample in log-signature space. Moreover, the parameter in encoder does not change too much when we are training it to reverse different samples. Therefore, we can leverage the parameter trained before to start our new training for the next sample, which accel-

erates the training procedure. Also, we can pre-trained the model on a collection of log-signature and use the pre-trained model to inverse a sample without training.

4.5 Evaluation of model

We evaluate the goodness of market generator by computing the MMD difference between the true distribution $\mathbb{P}$ and the generated distribution $\mathbb{Q}$ on path space.

$$d_{\mathcal{G}}(\mathbb{P},\mathbb{Q}) = \mathbb{E}\big[\mathbf{k_{Sig}}(X,X')\big] - 2\mathbb{E}\big[\mathbf{k_{Sig}}(X,Y)\big] + \mathbb{E}\big[\mathbf{k_{Sig}}(Y,Y')\big] \tag{4.1}$$

where X, Y, X', Y' are independent with $X, X' \sim \mathbb{P}$ and $Y, Y' \sim \mathbb{Q}$. For discrete approximation, we compute the MMD test statistic:

$$T_U^2(X_1,\cdots,X_m;Y_1,\cdots,Y_m) \coloneqq \frac{1}{m(m-1)}\sum_{i,j,i\neq j}\mathbf{k_{Sig}}(X_i,X_j) - \frac{2}{mn}\sum_{i,j}\mathbf{k_{Sig}}(X_i,Y_j) + \frac{1}{n(n-1)}\sum_{i,j,i\neq j}\mathbf{k_{Sig}}(Y_i,Y_j)$$

[26, 27] prove that T_U^2 is an unbiased estimators for $d_{\mathcal{G}}$ and provide a confidence interval of rejecting $\mathbb{P} = \mathbb{Q}$.

Theorem 4.1. *Assume that $m = n$, and $0 \leqslant \mathbf{k_{Sig}}(\cdot,\cdot) \leqslant K$, then*

$$\mathbb{P}\left\{T_U^2(X_1,\cdots,X_m;Y_1,\cdots,Y_m) - d_{\mathcal{G}}(\mathbb{P},\mathbb{Q}) > t\right\} \leqslant \exp\left(\frac{-t\lfloor m/2\rfloor}{8K^2}\right)$$

They also pointed out that the choice of threshold is conservative and can be improved by using data-dependent bounds. An alternative is to apply a permutation test. We refer to the MMD testing literature for many more details and improvements [26, 27, 15, 51, 32, 53]. Details of MMD computing are postpone to the next chapter with numerical results, where we introduce some technical improvement to overcome the instability of normalized signature kernel in scaling problem.

5. Numerical Implementation

In this chapter, we present numerical implementations of our algorithm. We present numerical comparisons between different evaluation metrics and inversion algorithms. Then we show numerical results of market generator introduced last chapter. At the end of this chapter, we supplement numerical examples highlighting the efficiency of signature features and provide a new approach to stimulate time series.

5.1 Inversion algorithm

In this section, we display numerical implementations of our neural methods inverting log-signature to path and compare our method with the evolution method first introduced in [13]. We show with numerical facts that our neural method outperforms the evolution method in multiple aspects. Also, we point out that the time augmentation outperforms lead-lag transformation in the log-signature inversion task of geometric brownian motion. At last, we show that our neural network can be pertained and then be used to inverse a collection of log-signature without training.

5.1.1 Single log-signature inversion

We first consider a 2-dimensional Black-scholes model of independent assets $S_t = (S_t^1, S_t^2)$, where

$$S_t^k = e^{W_t^k - \frac{t}{2}}, \quad k = 1, 2$$

with W^1 and W^2 being independent brownian motions. We sample the 2-d path on $t = \frac{i}{N}$, $i = 0, \cdots N$ where $N = 20$. Then we compute the discrete log-signature of the discrete stream. Now we apply our neural method to inverse the log-signature in order to retrieve the original path.

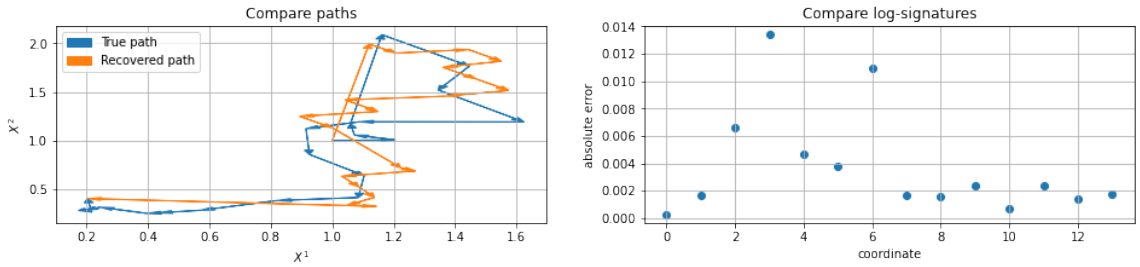

Figure 5.1: Inverse 2-d path with neural method

The L^2 error of log-signatures between real path and recovered path are always small after training. But sometimes (frequency like 1 time in 5 experiments) we found that the recovered path is not close to the original one even though their log-signatures are very close under L^2 norm. Possible reasons for this phenomenon follows.

(i) The path embeds almost tree-like structure. This can be solved by time augmentation.

(ii) L^2-error on log-signature space is not a good loss. We can weight different coordinates. However, there is actually no good reason to explain why should we do so and also there is no convincing way to prefix weights.

(iii) Neural-network is not well designed and generates paths that are too wild and prior on paths space should be added. A more delicate study of neural network structure should be studied in future work.

A direct fix is to apply time augmentation and we present the numerical improvement below.

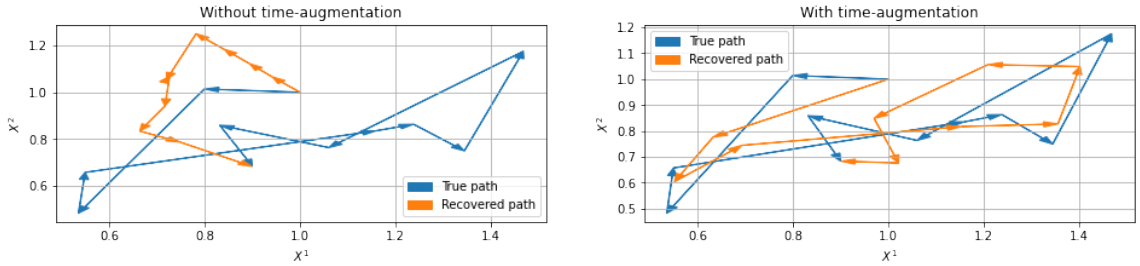

Figure 5.2: Inverse 2-d path with neural method

Next, we consider 1-d path $X_t = S_t^1$ with observation $(X_i)_{i=0}^N$ on $t = \frac{i}{N}$, $i = 0, \cdots N$ where $N = 20$. We apply lead-lag transformation on $(X_i)_{i=0}^N$ which gives a 2-dimensional path $(\tilde{X}_i)_{i=0}^N$. Then we can use the same procedure as before. We compare our method with the evolution algorithm by not only comparing the retrieved path and comparing the replication strategy of call option w.r.t the retrieved path.

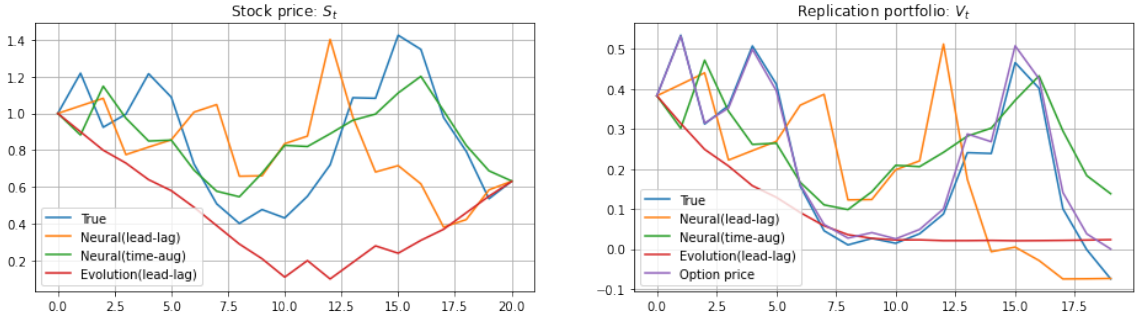

Figure 5.3: Neural method VS evolution algorithm

The neural method with lead-lag slightly performs better than the evolution algorithm with lead-lag in most cases, but the difference is negligible. However, the neural method is much faster! It takes around 5 seconds to retrieve a path but evolution algorithm takes around 300 seconds. Moreover, we observe that the neural method with time augmentation outperforms the other two methods and it is also more stable across experiments.

5.1.2 Multiple log-signatures inversion

By the uniqueness of log-signature after time-augmentation, log-signature map has a unique inverse mapping. We train an autoencoder in order to learn this inverse map on a data set consisting of brownian paths $(W_i)_{i=0}^N$ on $t = \frac{i}{N}$, $i = 0, \cdots N$ where $N = 20$, together with the time augmentation trick.

Figure 5.4: Neural method on training set

Then, we test the trained autoencoder on a collection of brownian paths independent with training paths.

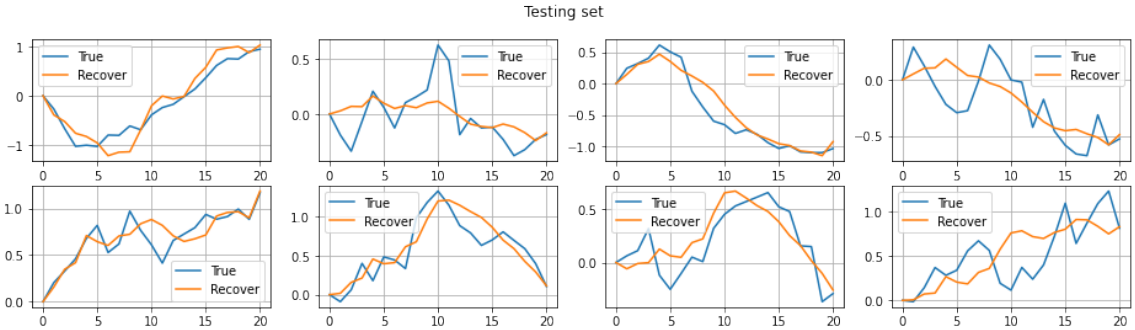

Figure 5.5: Neural method on testing set

Next, we test the trained autoencoder on a collection of geometric brownian paths independent with training paths.

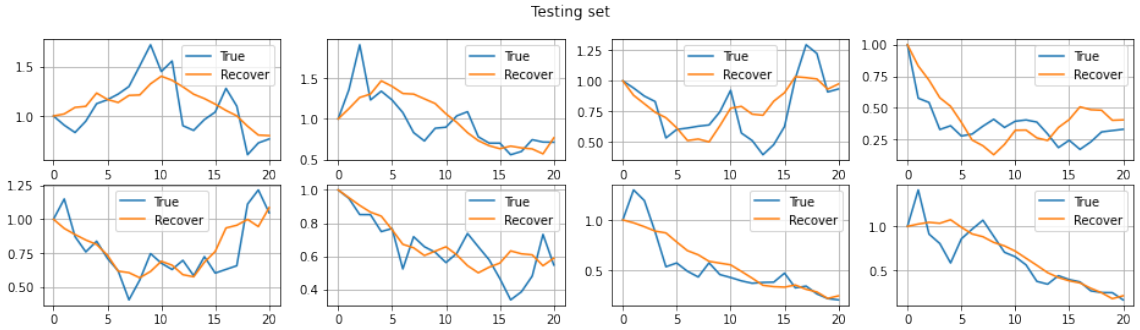

Figure 5.6: Neural method on testing set

We observe that the neural method not only performs very well on test processes, but also keeps stable and robust across different distributions. Given that the design of our neural network is just a very simple ReLU neural network with 3 hidden layers, this result is very surprising. This inspires us to pre-train the autoencoder on a large set do not change the parameter when inverting log-signatures. This pre-training approach also enables us to design a penalty on latent space because we actually know original paths of the training data. This idea should be made rigorous and implemented in future work.

5.2 Evaluation

As we have discussed in previous chapter, efficient evaluation of the similarity between two distributions of time series is not a easy task. Through this paper, we applied the MMD test based on signature kernels introduced in [14] to characterize the similarity. In this section, we discuss numerical implementation of signature kernels and variations of signature kernels, meanwhile comparing their performance in differentiating distributions of time series.

- **Signature kernel**: inner product of signature
- **Normalized kernel (one in [14])**: before applying inner product, normalize all the signature **Sig** with Λ such that $\|\Lambda(\mathbf{Sig})\|_{\mathbf{T}_1} = \psi(\|\mathbf{Sig}\|_{\mathbf{T}_1})$, where

$$\psi(\sqrt{x}) = \begin{cases} x & \text{if } x \leqslant M \\ M + M^{1+a}(M^{-a} - x^{-a})/a & \text{if } x > M \end{cases}$$

- **Batch normalized kernel**: before applying normalized kernel, pre-normalize all the signature in batch with δ_λ such that $\|\delta_\lambda(\mathbf{Sig})\|_{\mathbf{T}_1} \leqslant C$.
- **Lifted signature kernel**: lift the paths to RKHS space (H, κ) and compute the signature kernel by the following approximation

$$\langle \triangle\kappa_{X_i}, \triangle\kappa_{Y_j} \rangle \approx \kappa(X_{i+1}, Y_{j+1}) + \kappa(X_i, Y_j) - \kappa(X_i, Y_{j+1}) - \kappa(X_{i+1}, Y_j), \quad (5.1)$$

Notice that even though normalized signature kernel scales all samples in a 'smooth' (in fact Lipschitz here) way but all signatures are scaled by different factor. A direct consequence is that X will be consider as the same as λX for all $\lambda > 1$ if the norm of signatures are large. To overcome this scaling problem, we first normalize all signatures by the mean of scaling factor among all samples. This method works very well in practice when the norm of signatures are large. An alternative to to improve performance is to consider Lifted signature kernel. We first lift paths to a RKHS space and then compute the signature. It is surprising that this kernel can also be computed efficiently with the approximation (5.1) and the recursive algorithm in [14]. We show that this lifting greatly increase the performance but the drawback is that this kernel can not be computed from signature.

5.2.1 Separating distributions

Firstly, we compare the signature kernel with the signature kernel after lifting. We choose the same numerical case as in [14], the task of differentiating the following two distributions.

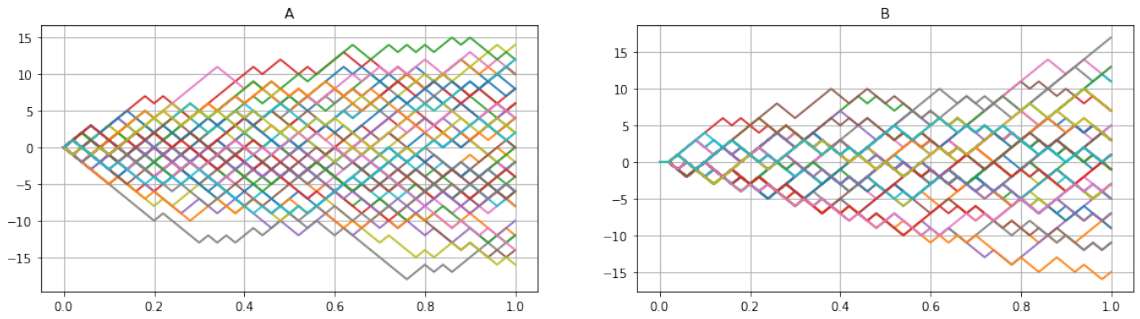

Figure 5.7: Random walks

We consider the exponential lifted signature kernel and compare it with signature kernel.

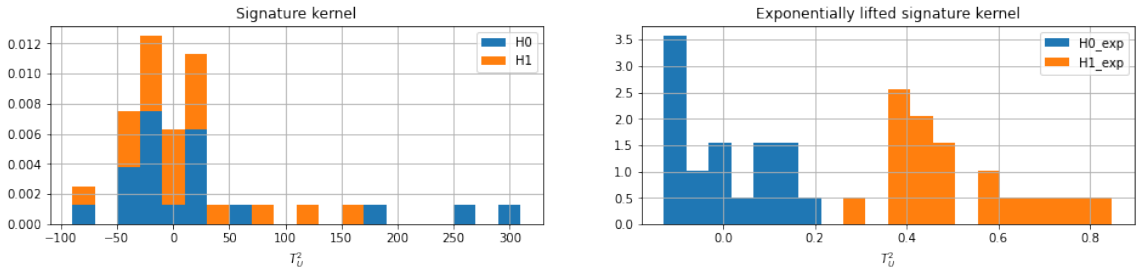

Figure 5.8: Random walks

Exponential lifted signature kernel outperforms the signature kernel. Moreover, we also try some different RKHS space such as square exponential. But we found exponential lifted signature kernel usually outperform other choices, which is the same conclusion drawn in [14]. Next, we apply all of the four kernels introduced above to compare time series generated by rough Bergomi model with different parameters.

$$\begin{aligned} dX_t &= -\frac{1}{2}V_t dt + \sqrt{V_t}dW_t, \quad X_0 = \log(S_0) \\ dV_t &= \xi_0 \mathcal{E}(2\nu C_H \mathcal{V}_t), \quad V_0, v, \xi_0 > 0 \\ \mathcal{V}_t &= \int_0^t (t-u)^{H-1/2} dZ_u, \quad H \in (0, 1/2) \\ \langle Z, W \rangle_t &= \rho t, \quad \rho \in (-1, 1) \end{aligned} \tag{5.2}$$

where $X = \log(S)$, and $\mathcal{E}(\cdot)$ denotes the stochastic exponential and $C_H = \frac{2H\Gamma(\frac{3}{2}-H)}{\Gamma(H+\frac{1}{2})\Gamma(2-2H)}$.

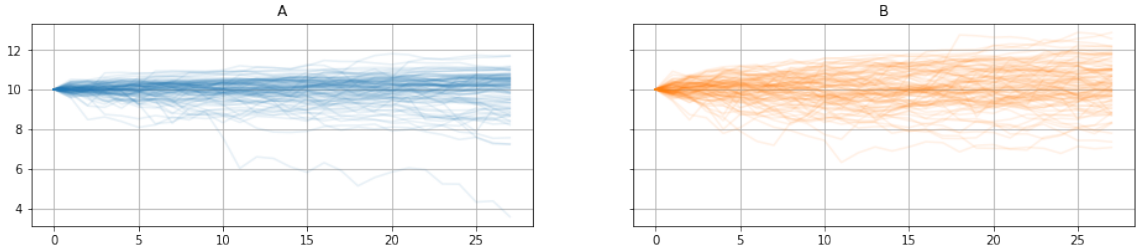

Figure 5.9: Rough Bergomi model

We consider the following two cases: (i) X and Y from the same distribution A (ii) X from the same distribution A and Y from the distribution B. We compute the MMD statistic using the above four kernel and compare their performance.

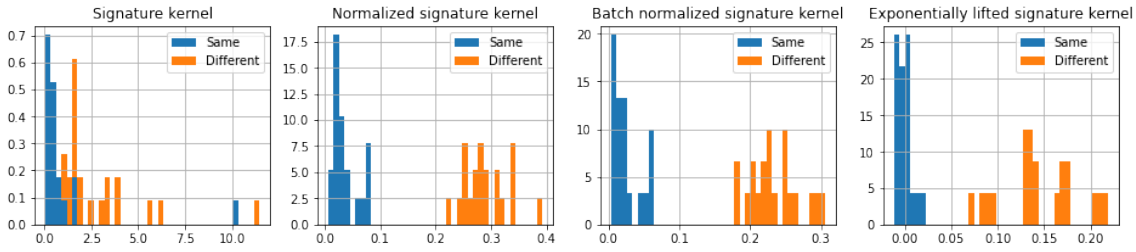

Figure 5.10: Support of T_U^2 of two cases

We notice that signature kernel fails and exponentially lifted signature kernel performs better than normalized signature kernel. Moreover, we observe that the normalized signature kernel also fails if we amplify values of paths. This motivates us to design the batch normalized signature kernel which uniformly scale signature to a considerable range. Now, we consider the same numerical problem of rough Bergomi model with value of paths amplified by scale 10.

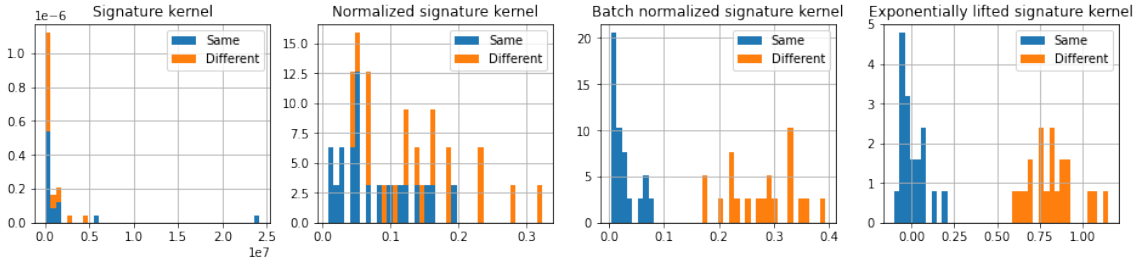

Figure 5.11: Stability in scaling

Normalized signature kernel fails but batch normalized signature kernel is stable across scaling. Although, exponential lifted signature kernel outperforms other kernels in all experiments, it can not be computed directly form signature. Therefore, in most cases, we prefer to apply the batch normalized signature kernel when only signatures are available.

5.3 Market generator

In this section we present the numerical implementation of market generator algorithms introduced in the above chapter. First we present numerical results by using VAEs with only rectified neural network structure. Then we present the numerical

results by using VAEs with signature structure embedded. At last, we discuss the difference between normal prior and heavy tail prior. We first consider the rough Bergomi model and sample 100 paths of rough Bergomi model with the same distribution and we consider two different preprocessing options: (i) lead-lag transformation (ii) time augmentation.

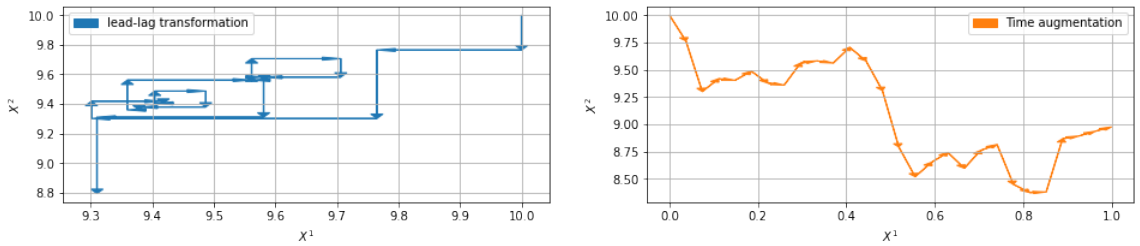

Figure 5.12: Lead-lag transformation VS time augmentation

5.3.1 Lead-lag based VAEs

First we consider the lead-lag transformation. We compute the log-signature of the lead-lag transformed paths and feed them to a β-variational autoencoder with normal prior. After training, we obtain the following in-sample results.

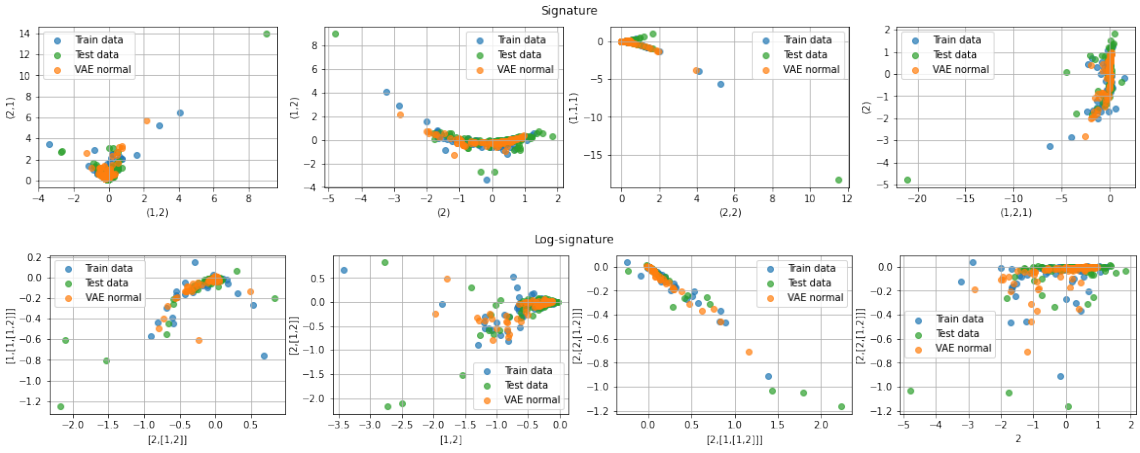

Figure 5.13: Signature and log-signature

Also we obtain the following out-sample result. Here the different data is the data generated by a rough Bergomi model but with different parameter.

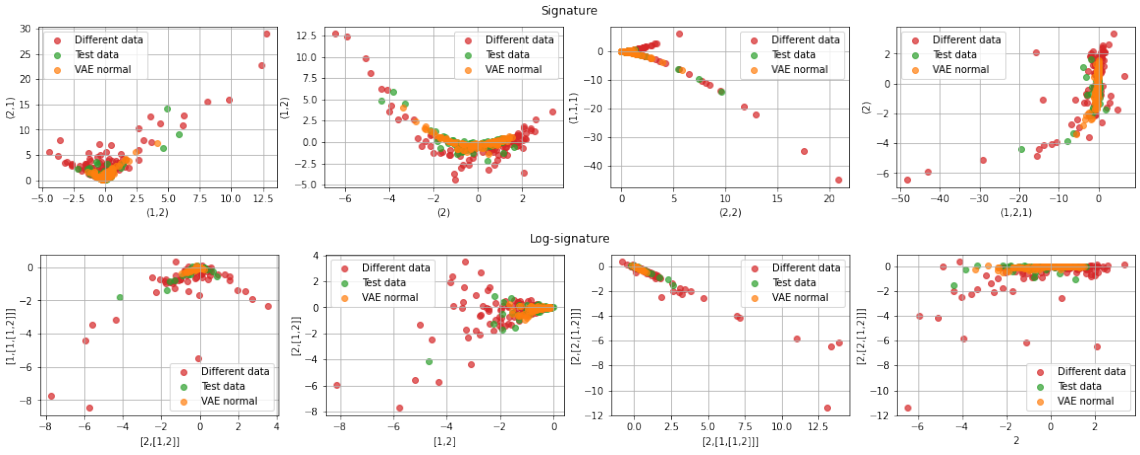

Figure 5.14: Signature and log-signature

The comparison through visualization looks promising and next we quantify the comparison by computing the MMD statistics based on batch normalized signature kernel introduced before. We compute the MMD between (i) two independent random variables of true distribution (ii) true distribution and generated distribution (iii) true distribution with a different distribution

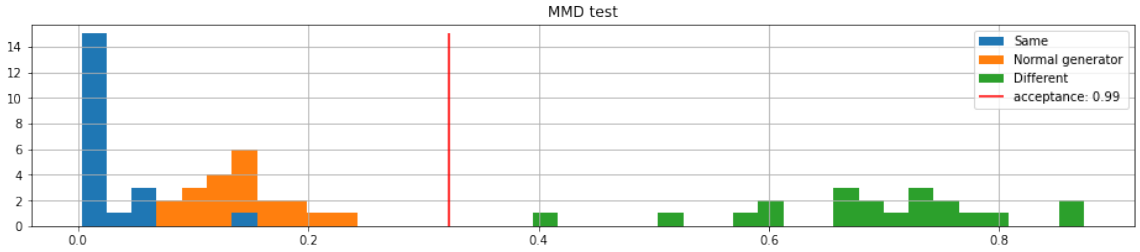

Figure 5.15: MMD tests on signatures

This implies that we have successfully generated a similar distribution on log-signature space. Next, we inverse samples on log-signature space to path space with the neural method we introduced before. We compute the following three distributions of paths (i) testing paths (ii) paths retrieved from testing log-signature (iii) paths retrieved from generated log-signature.

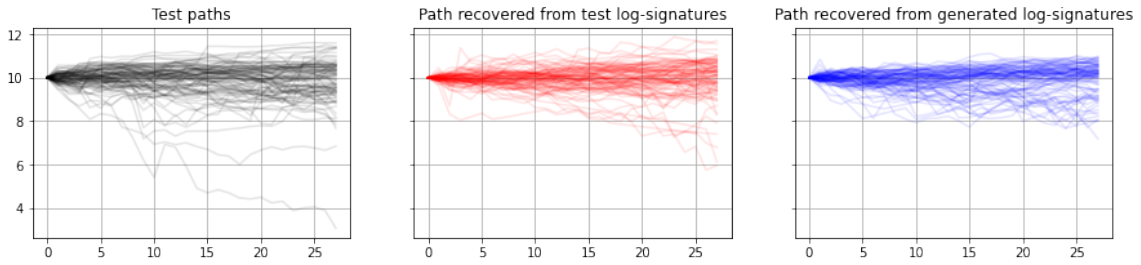

Figure 5.16: Paths

Now we can apply the exponentially lifted signature kernel MMD test to compare the differences between them. We benchmark the result with (i) 20 MMD tests between training paths and testing paths (ii) 20 MMD tests between training paths and paths generated by a different distribution

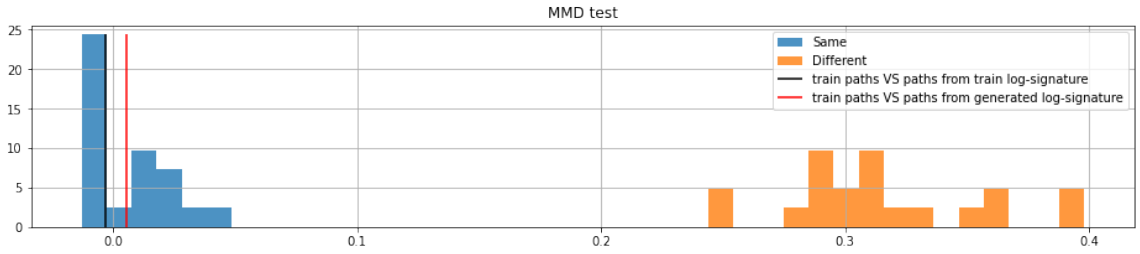

Figure 5.17: MMD test on paths

The above results imply that we successfully generate an indifferent distribution of paths, which solves our task.

5.3.2 Time-augmentation based VAEs

Motivated by the fact that time-augmentation outperforms the the lead-lag transformation in inverting log-signature. We compute the log-signature of the time augmented paths and feed them to a β-variational autoencoder with normal prior. If we use the same VAE as before, we obtain the following in-sample result.

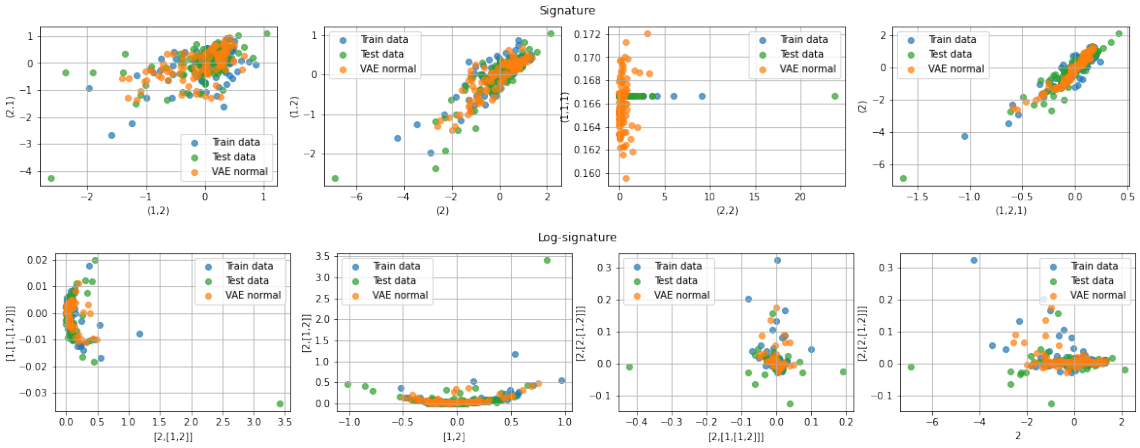

Figure 5.18: Signature and log-signature

This is because that log-signatures of the transformed paths lie in a subset of log-signature space but normal generator hardly capture this substructure. This is essentially the same problem when using signatures as feature space. Therefore, we add the decoder structure introduced in the neural method inverting log-signature to the decoder part of our VAE to makes sure outputs lie in the right subset, and we obtain the following in-sample result.

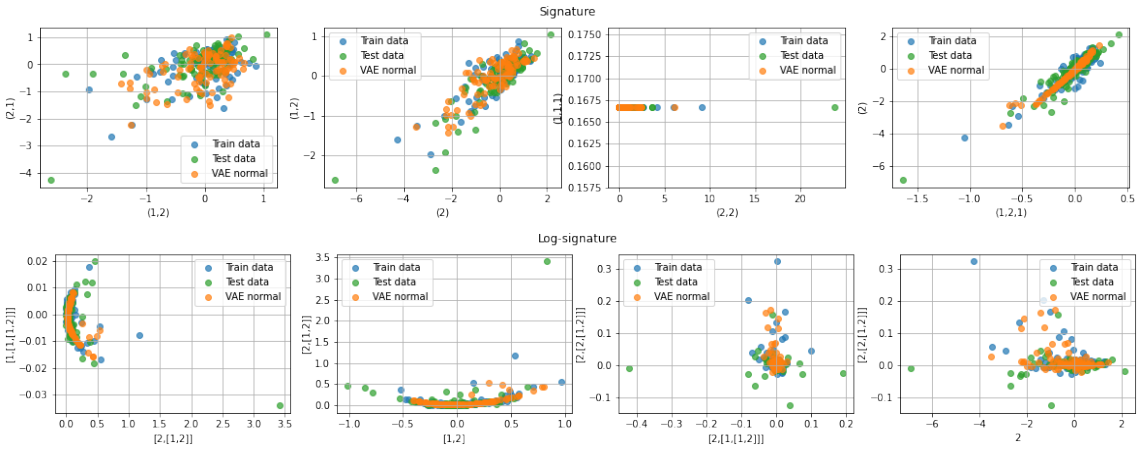

Figure 5.19: Signature and log-signature

Also, we obtain the following out-sample result.

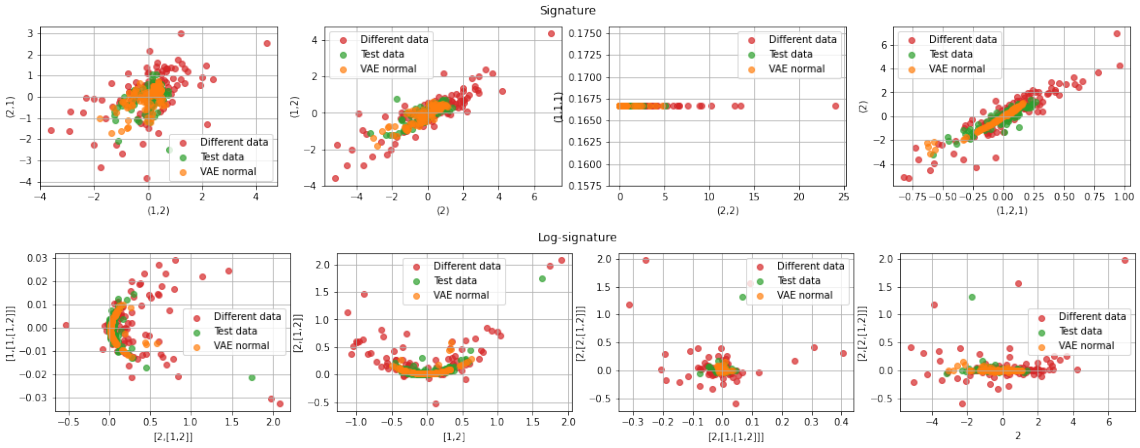

Figure 5.20: Signature and log-signature

Next we quantify the comparison by computing the MMD based on batch normalized signature kernel introduced before.

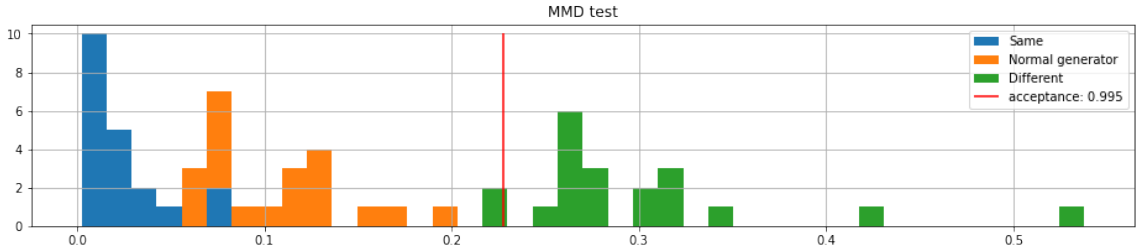

Figure 5.21: MMD tests on signatures

This implies that we have successfully generated similar distribution on log-signature space. Next, we inverse samples on log-signature space to paths with the neural method we introduced before. We compute the following three distributions of paths (i) testing paths (ii) paths retrieved from testing log-signature (iii) paths retrieved from generated log-signature

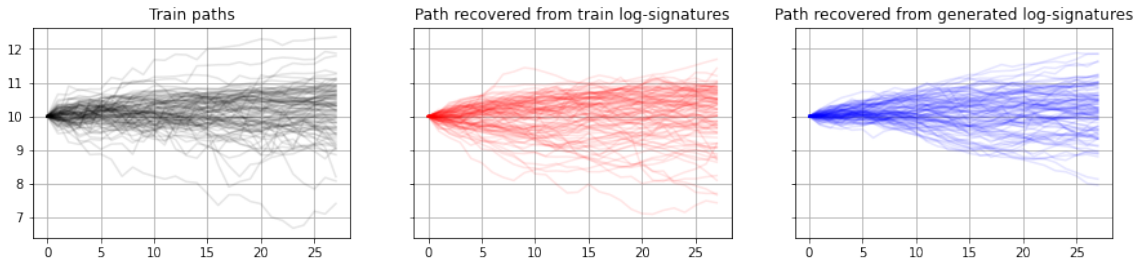

Figure 5.22: Paths

Then we apply the exponentially lifted signature kernel MMD test to compare the differences between paths.

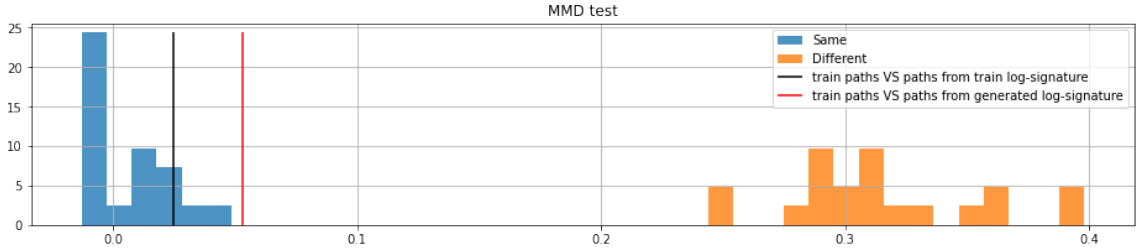

Figure 5.23: MMD test on paths

In summary, Lead-lag based VAE and time augmented VAE both successfully generate similar distribution. Even though from the numerical results from neural method, the time augmentation outperforms the lead-lag transformation for geometric brownian motion, we observe that paths retrieved from time-augmentation is less rougher than paths retrieved from lead-lag transformation. Moreover, the VAE structure embedding signature computing structure can also be used for lead-lag transformation and this should improve the performance of lead-lag based VAE, which is also a promising numerical study to do in future work.

5.3.3 Student-t prior

In this subsection, we present the numerical comparison between VAE with normal prior and VAE with student-t prior. Firstly, we visualize the generated samples on signature space as well as log-signature space.

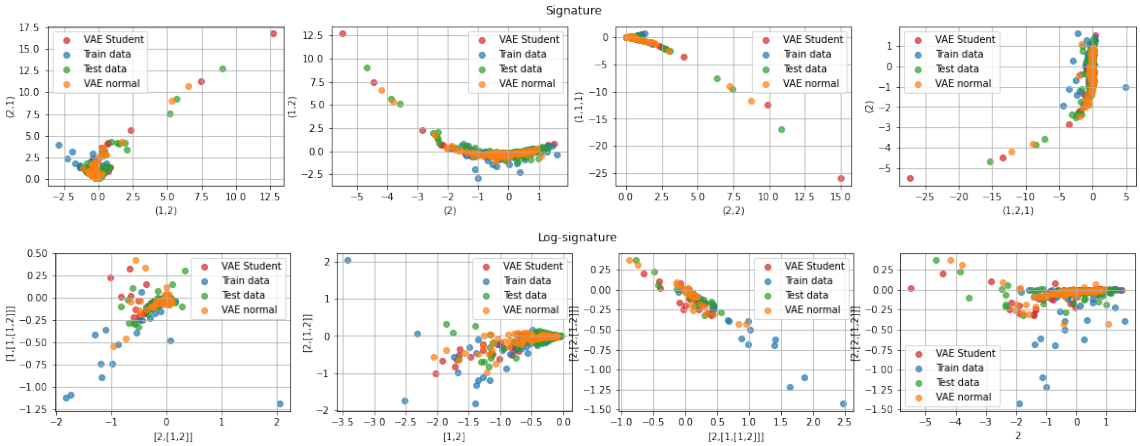

Figure 5.24: Signature and log-signature

Then we quantify the difference between VAE with normal prior and VAE with student-t prior with MMD test based on batch normalized signature kernel.

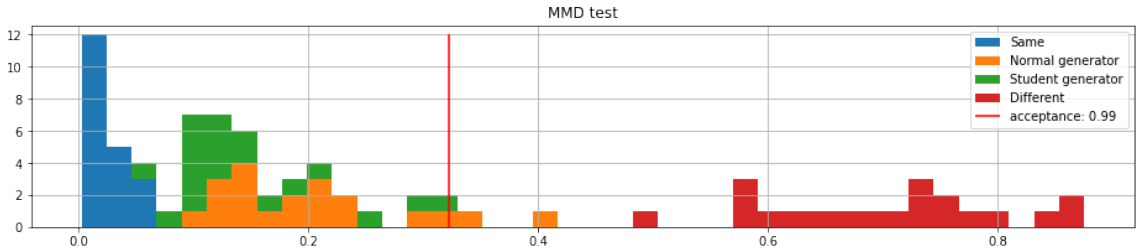

Figure 5.25: MMD test

We observe that student-t prior in general outperforms the normal prior but the difference is negligible. However, this proves that prior of VAE matters and a following question is that what prior is the best? In another word, we are interested in the property of the signature distribution, for example, whether the distribution on the signature space is heavy-tail or not, which should be studied in future work.

5.4 Path stimulation with signature

In this section, we consider numerical stimulations of the following stochastic differential equation

$$dX_t = \sum_{k=1}^{m} B^k X_t dW_t^k, \quad X_t \in \mathbb{R}^n, W_t \in \mathbb{R}^m, B^k \in \mathbb{R}^{n\times n}$$

where W_t is a Brownian motion on $\mathbb{R}^m$. We choose $m = n = 2$ and

$$B^1 = \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix}, \quad B^2 = \begin{bmatrix} 0.4 & 0.3 \\ 0.2 & 0.1 \end{bmatrix}.$$

Then we stimulate the differential equation with signatures.

$$\begin{aligned} X_t &= \sum_{d=0}^{\infty} \sum_{i_1,\cdots,i_d=1}^{n} \left(\int_{0\leqslant t_1 \leqslant \cdots \leqslant t_d \leqslant t} dW_{t_1}^{i_1} \cdots dW_{t_d}^{i_d} \right) B^{i_d} \cdots B^{i_1} X_0 \\ &= \sum_{d=0}^{\infty} \sum_{i_1,\cdots,i_d=1}^{n} \mathbf{Sig}_{[0,t]}^{i_1,\cdots,i_d}(W) B^{i_d} \cdots B^{i_1} \cdot X_0 \end{aligned}$$

We approximate the signature by discrete growing signature. This is important because if we use higher order, we implicitly assume Stratonovich integral which is not the case here. Or we are using a Milstein scheme and implicitly assume the commutativity of B^1 and B^2 if we use second order discrete truncated signature.

$$\begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix} \begin{bmatrix} 0.4 & 0.3 \\ 0.2 & 0.1 \end{bmatrix} - \begin{bmatrix} 0.4 & 0.3 \\ 0.2 & 0.1 \end{bmatrix} \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix} = \begin{bmatrix} -0.05 & -0.15 \\ 0.15 & 0.05 \end{bmatrix} \neq 0$$

Unfortunately, all python packages we know available so far (including esig `https://esig.readthedocs.io/`, iisignature [48], and Signatory [35]) only support truncated discrete signature but not support growing signature. We approximate the true path with Euler approximation on a very fine grid.

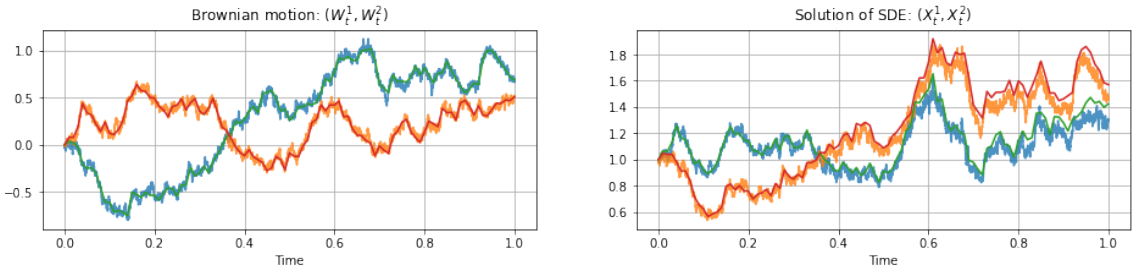

Figure 5.26: Solution of SDE stimulated with Euler method on fine grid

Now we compare stimulations between growing discrete signature and truncated discrete signature.

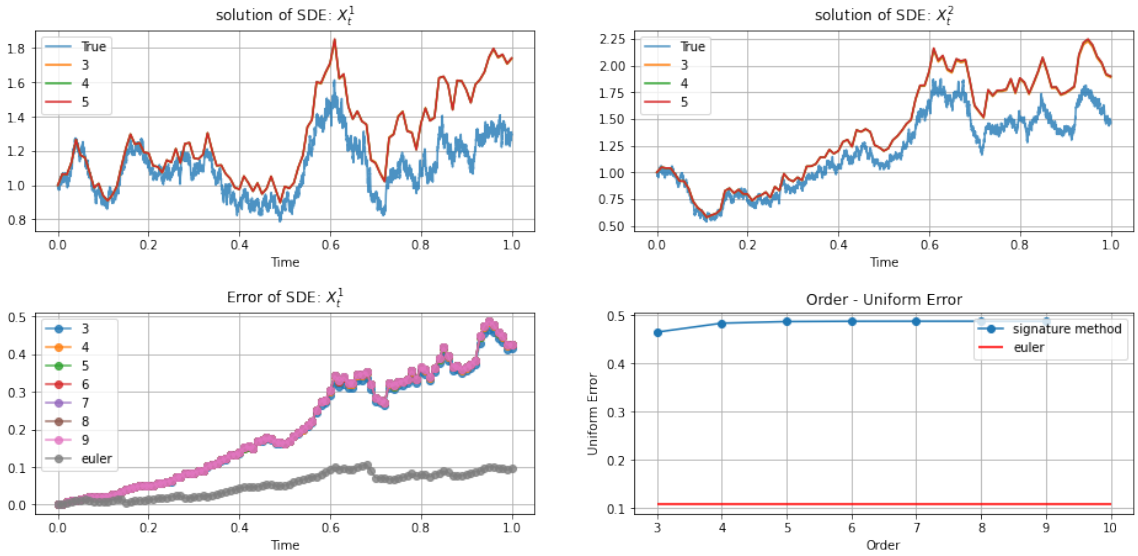

Figure 5.27: Truncated discrete signature

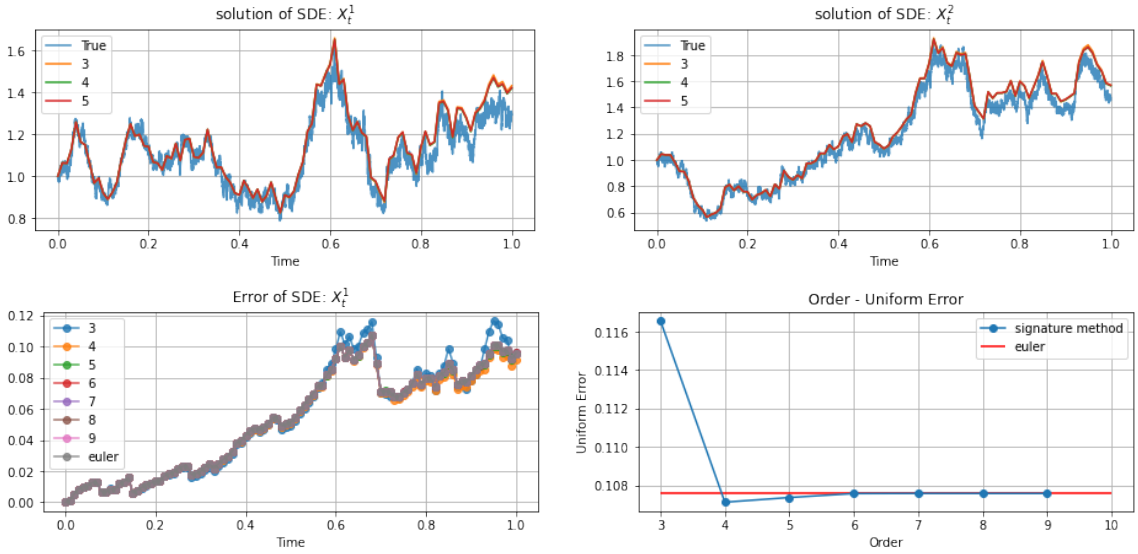

Figure 5.28: Growing discrete signature

Growing discrete signature outperforms the truncated signature. As the order of truncated discrete signature increases, it tends to it's limit of approximation, which is the error caused by mis-computing the quadratic variation term using Stratonovich integral. As the order of growing discrete signature increases, the performance tends to the Euler method. This is useful because if we know the signature of our control at T, then multiplying the signature with vector field gives a stimulation as good as Euler stimulation running from 0 to T. The key here is the linear relation between signature and the value of SDE. Inversely, this linear relation can be easily learnt by a simple linear regression.

5.5 Learning from path to path

Consider controlled differential equations

$$dX_t = \sum_{k=1}^{m} V^k(X_t) dW_t^k, \quad X_t \in \mathbb{R}^d \tag{5.3}$$

By generalized Taylor's expansion we have

$$\begin{aligned} X_t^{(i)} &= \sum_{m=0}^{\infty} \sum_{i_1,\cdots,i_m=1}^{d} \mathbf{Sig}_{[0,t]}^{i_1,\cdots,i_m}(W) \sum_{j=1}^{d} M_{i,j}^{i_1,\cdots,i_m} X_s^{(j)} \\ &= \sum_{m=0}^{\infty} \sum_{i_1,\cdots,i_m=1}^{d} \sum_{j=1}^{d} M_{i,j}^{i_1,\cdots,i_m} \mathbf{Sig}_{[0s,t]}^{i_1,\cdots,i_m}(W) X_0^{(j)} \\ &= \langle M_i, \mathbf{Sig}_{[s,t]} \otimes X_s \rangle \end{aligned} \tag{5.4}$$

Where $M_{i,j}^{i_1,\cdots,i_m}$ are matrixes depending on s. If we let $\mathcal{X} = \mathbf{Sig}_{[s,t]} \otimes X_s$ and $\mathcal{Y} = X_t$, we can learn the relation in (5.4) by a linear regression on $(\mathcal{X}, \mathcal{Y})$. This idea was first shown to me in the machine learning in finance course by Prof. Josef Teichmann, where he also introduced reservoir computing which is closely related to this approach here if you consider the signature stream as a reservoir. We refer interested readers on signature and reservoir computing to discussion in [???]. Now, we consider $m = d = 2$ and the following dynamic

$$V^1(X) = \begin{bmatrix} 2|X^{(2)}|^{0.7} \\ X^{(2)} \end{bmatrix}, \quad V^2(X) = \begin{bmatrix} 2|X^{(2)}|^{0.7} \\ 0 \end{bmatrix}, \quad X_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

For reservoir space, we consider

$$dR_t = \sum_{k=1}^{m} (A^k R_t + b^k) dW_t^k, \quad R_t \in \mathbb{R}^d \tag{5.5}$$

where A^k and b^k are random matrix and vector. We stimulate the real solution X with Euler scheme on $[0, 1]$ with 2000 grid points.

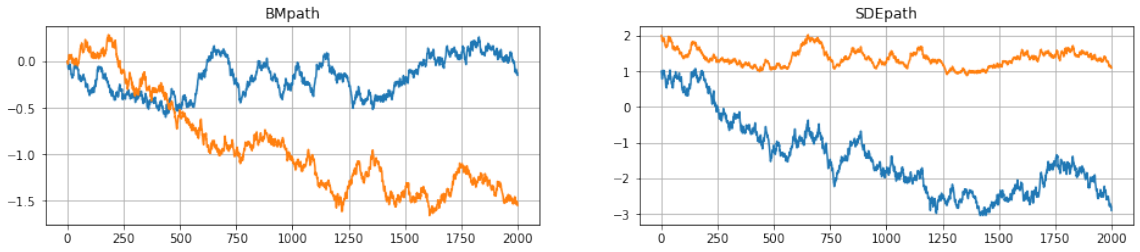

Figure 5.29: Solution of SDE stimulated with Euler method

We choose first $t = 1, \cdots 1000$ to be training samples, and $t = 1001, \cdots 2000$ to be test samples. We use ridge regression to learn matrixes $M_{i,j}^{i_1,\cdots,i_m}$ on training samples, and generate path on testing samples with learnt matrixes. We compared the generated path with true path under the same random seed.

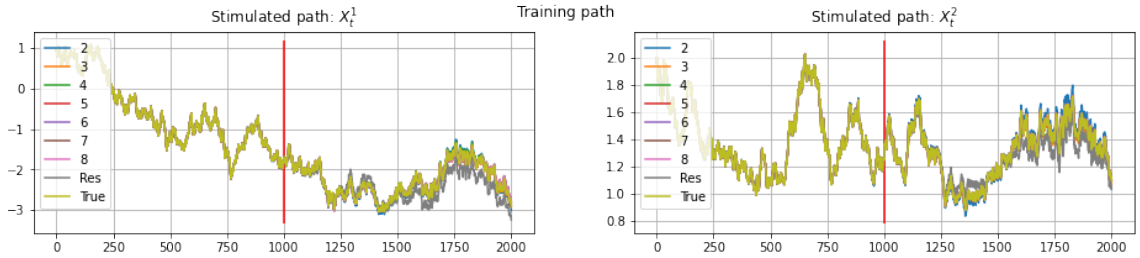

Figure 5.30: Training path

Furthermore, we generate path starting from 0 with another random seed rs and compare the generated path with true path under the same random seed rs.

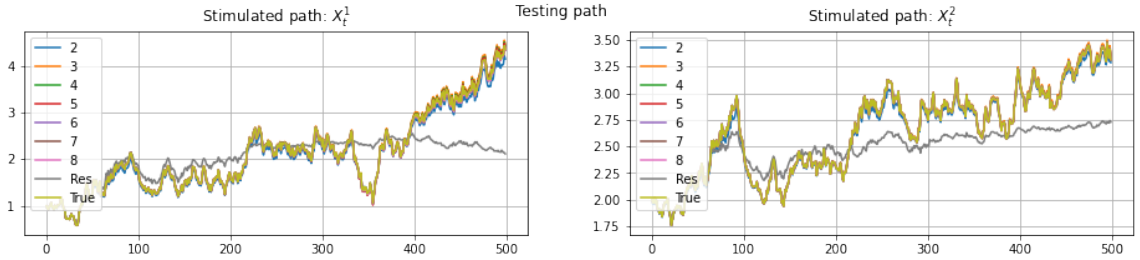

Figure 5.31: Testing path

Now we present the relation between errors and the order of growing discrete signature

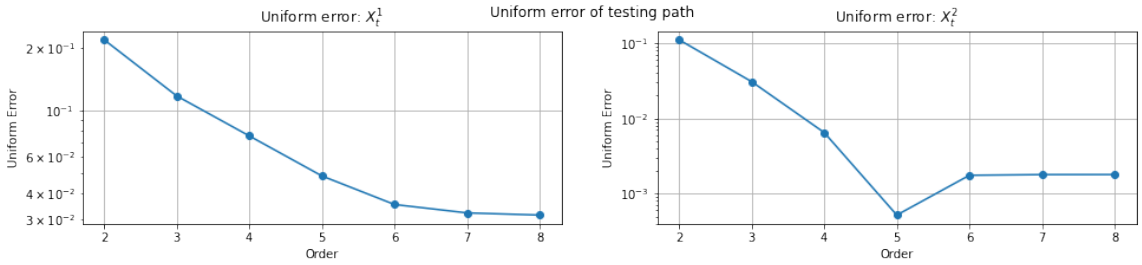

Figure 5.32: Order - Error

As the order of discrete signature increases, the error decreases and usually touches the limit when order comes to 6. Next, we use generators we just learnt to generate new samples and compared generators based on different truncation order with MMD distance between samples generated by true dynamic.

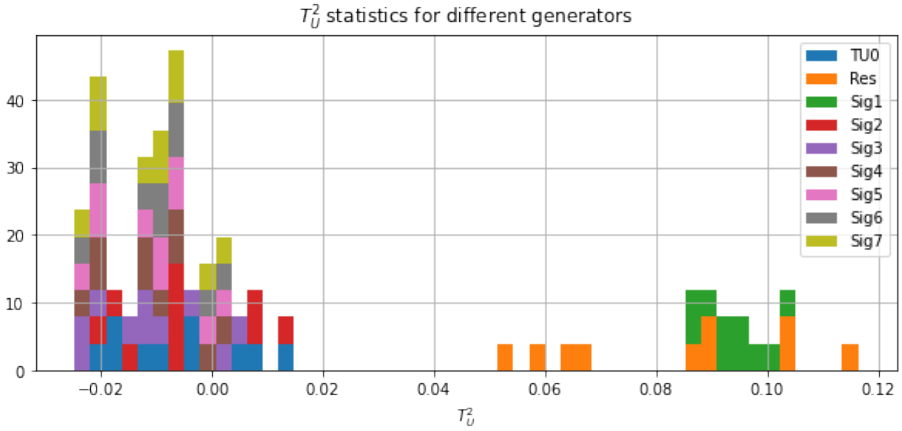

Figure 5.33: T_U^2 statistics of different generators

We found that the market generator learnt by signature of higher order is as good as the real dynamic. In fact, the linear relation not only helps to build a nice model, but also contributes to the training procedure because many robust solvers for linear regression are available. If we simply use Adam algorithm to train the same linear model, the performance is not as good as what we achieved by closed-form solver.

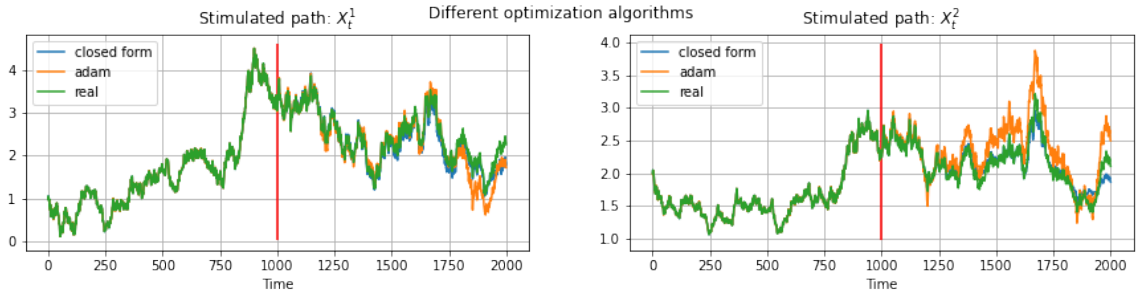

Figure 5.34: Different optimization algorithms

However, it should be pointed out that this linear relation only holds when we consider path starting from the same initial point because matrixes $M_{i,j}^{i_1,\cdots,i_m}$ depends on the gradient of vector field at the starting point X_s. Therefore, unless the the vector field is linear (gradient independent of X_s), our model can not be directly applied if samples are paths starting from different initial points. We consider the same controlled differential equation and we split path in to subpaths before construct training data. Since the first dimension is driven by nonlinear dynamic, the performance of training is greatly interfered by splitting data, while even though the second dimension is driven by linear dynamic, the performance still decreases as we split path into subpaths.

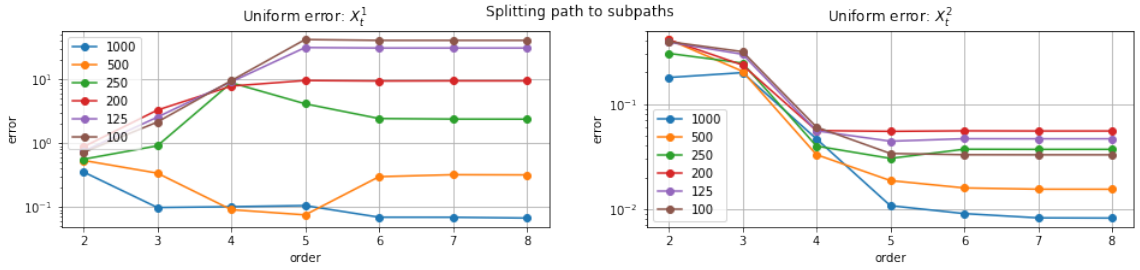

Figure 5.35: Splitting paths

We propose to fix this non-linearity problem by approximate the relation between X_s and $M_{i,j}^{i_1,\cdots,i_m}$ by some nonlinear function and more research need to be done in future work. Also, we need to keep in mind that this is not a generative model itself because we are learning the dependency between driving process and controlled process by knowing their value under same random seed. Therefore, in order to generate new samples, we still need to know the distribution of driving process in

advance. This might be useful if the target time series you wish to generate is too complicated, but you know that it is driven by some simple process, of which you know the distribution. In this case, as long as you can observe both the target time series and the simple process, you can apply this methodology to generate new samples.

Bibliography

[1] Abiri, N., and Ohlsson, M. The advantage of using student's t-priors in variational autoencoders.

[2] Abiri, N., and Ohlsson, M. Variational auto-encoders with student's t-prior. *arXiv preprint arXiv:2004.02581* (2020).

[3] Amaral, L. A., Plerou, V., Gopikrishnan, P., Meyer, M., and Stanley, H. E. The distribution of returns of stock prices. *International Journal of Theoretical and Applied Finance 3*, 03 (2000), 365–369.

[4] Arjovsky, M., and Bottou, L. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862* (2017).

[5] Ballard, D. H. Modular learning in neural networks. In *AAAI* (1987), pp. 279–284.

[6] Bayer, C., Horvath, B., Muguruza, A., Stemper, B., and Tomas, M. On deep pricing and calibration of (rough) stochastic volatility models. *Preprint*.

[7] Boedihardjo, H., Geng, X., Lyons, T., and Yang, D. The signature of a rough path: uniqueness. *Advances in Mathematics 293* (2016), 720–737.

[8] Bonnier, P., Kidger, P., Arribas, I. P., Salvi, C., and Lyons, T. Deep signature transforms. *arXiv preprint arXiv:1905.08494* (2019).

[9] Bonnier, P., Liu, C., and Oberhauser, H. Adapted topologies and higher rank signatures. *arXiv preprint arXiv:2005.08897* (2020).

[10] Bouchaud, J.-P., Matacz, A., and Potters, M. Leverage effect in financial markets: The retarded volatility model. *Physical review letters 87*, 22 (2001), 228701.

[11] Brigo, D. Probability-free models in option pricing: statistically indistinguishable dynamics and historical vs implied volatility. *arXiv preprint arXiv:1904.01889* (2019).

[12] Buehler, H., Gonon, L., Teichmann, J., and Wood, B. Deep hedging. *Quantitative Finance 19*, 8 (2019), 1271–1291.

[13] Buehler, H., Horvath, B., Lyons, T., Perez Arribas, I., and Wood, B. A data-driven market simulator for small data environments. *Available at SSRN 3632431* (2020).

[14] Chevyrev, I., and Oberhauser, H. Signature moments to characterize laws of stochastic processes. *arXiv preprint arXiv:1810.10971* (2018).

[15] Chwialkowski, K., Strathmann, H., and Gretton, A. A kernel test of goodness of fit. In *International conference on machine learning* (2016), PMLR, pp. 2606–2615.

[16] Cont, R. Empirical properties of asset returns: stylized facts and statistical issues.

[17] Cont, R., and Ben Hamida, S. Recovering volatility from option prices by evolutionary optimization.

[18] Cuchiero, C., Khosrawi, W., and Teichmann, J. A generative adversarial network approach to calibration of local stochastic volatility models. *Risks 8*, 4 (2020), 101.

[19] Duembgen, M., and Rogers, L. Estimate nothing. *Quantitative Finance 14*, 12 (2014), 2065–2072.

[20] Flint, G., Hambly, B., and Lyons, T. Discretely sampled signals and the rough hoff process. *Stochastic Processes and their Applications 126*, 9 (2016), 2593–2614.

[21] Fortuin, V., Baranchuk, D., Rätsch, G., and Mandt, S. Gp-vae: Deep probabilistic time series imputation. In *International Conference on Artificial Intelligence and Statistics* (2020), PMLR, pp. 1651–1661.

[22] Foster, J., Lyons, T., and Oberhauser, H. An optimal polynomial approximation of brownian motion. *SIAM Journal on Numerical Analysis 58*, 3 (2020), 1393–1421.

[23] Freeman, J. Probability metrics and the stability of stochastic models. *Journal of the Operational Research Society 43*, 9 (1993), 923–923.

[24] Friz, P. K., and Hairer, M. *A course on rough paths*. Springer, 2020.

[25] Gatheral, J., Jaisson, T., and Rosenbaum, M. Volatility is rough. *Quantitative Finance 18*, 6 (2018), 933–949.

[26] Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *The Journal of Machine Learning Research 13*, 1 (2012), 723–773.

[27] Gretton, A., Fukumizu, K., Harchaoui, Z., and Sriperumbudur, B. K. A fast, consistent kernel two-sample test. In *NIPS* (2009), vol. 23, pp. 673–681.

[28] Henry-Labordere, P. Generative models for financial data. *Available at SSRN 3408007* (2019).

[29] Henry-Labordere, P. (martingale) optimal transport and anomaly detection with neural networks: A primal-dual algorithm. *Available at SSRN 3370910* (2019).

[30] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework.

[31] Horvath, B., Muguruza, A., and Tomas, M. Deep learning volatility. *Available at SSRN 3322085* (2019).

[32] Jitkrittum, W., Szabó, Z., Chwialkowski, K., and Gretton, A. Interpretable distribution features with maximum testing power. *arXiv preprint arXiv:1605.06796* (2016).

[33] Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196* (2017).

[34] Kidger, P., Foster, J., Li, X., Oberhauser, H., and Lyons, T. Neural sdes as infinite-dimensional gans. *arXiv preprint arXiv:2102.03657* (2021).

[35] Kidger, P., and Lyons, T. Signatory: differentiable computations of the signature and logsignature transforms, on both cpu and gpu. *arXiv preprint arXiv:2001.00706* (2020).

[36] Kidger, P., Morrill, J., Foster, J., and Lyons, T. Neural controlled differential equations for irregular time series. *arXiv preprint arXiv:2005.08926* (2020).

[37] Kingma, D. P., and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[38] Kingma, D. P., and Welling, M. An introduction to variational autoencoders. *arXiv preprint arXiv:1906.02691* (2019).

[39] Király, F. J., and Oberhauser, H. Kernels for sequentially ordered data. *Journal of Machine Learning Research 20*, 31 (2019), 1–45.

[40] Kondratyev, A., Schwarz, C., and Horvath, B. Data anonymisation, outlier detection and fighting overfitting with restricted boltzmann machines. *Outlier Detection and Fighting Overfitting with Restricted Boltzmann Machines (January 27, 2020)* (2020).

[41] Leonarduzzi, R., Rochette, G., Bouchaud, J.-P., and Mallat, S. Maximum-entropy scattering models for financial time series. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), IEEE, pp. 5496–5500.

[42] Levin, D., Lyons, T., and Ni, H. Learning from the past, predicting the statistics for the future, learning an evolving system. *arXiv preprint arXiv:1309.0260* (2013).

[43] Liao, S., Lyons, T., Yang, W., and Ni, H. Learning stochastic differential equations using rnn with log signature features. *arXiv preprint arXiv:1908.08286* (2019).

[44] Lyons, T. Rough paths, signatures and the modelling of functions on streams. *arXiv preprint arXiv:1405.4537* (2014).

[45] Lyons, T. J. Differential equations driven by rough signals. *Revista Matemática Iberoamericana 14*, 2 (1998), 215–310.

[46] Lyons, T. J., Caruana, M., and Lévy, T. *Differential equations driven by rough paths*. Springer, 2007.

[47] Mirza, M., and Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[48] Reizenstein, J., and Graham, B. The iisignature library: efficient calculation of iterated-integral signatures and log signatures. *arXiv preprint arXiv:1802.08252* (2018).

[49] Reutenauer, C. Free lie algebras. In *Handbook of algebra*, vol. 3. Elsevier, 2003, pp. 887–903.

[50] Ruf, J., and Wang, W. Neural networks for option pricing and hedging: a literature review. *Journal of Computational Finance, Forthcoming* (2020).

[51] Sejdinovic, D., Sriperumbudur, B., Gretton, A., and Fukumizu, K. Equivalence of distance-based and rkhs-based statistics in hypothesis testing. *The Annals of Statistics* (2013), 2263–2291.

[52] Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems 28* (2015), 3483–3491.

[53] Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Schölkopf, B., and Lanckriet, G. R. Hilbert space embeddings and metrics on probability measures. *The Journal of Machine Learning Research 11* (2010), 1517–1561.

[54] Toth, C., Bonnier, P., and Oberhauser, H. Seq2tens: An efficient representation of sequences by low-rank tensor projections. *arXiv preprint arXiv:2006.07027* (2020).

[55] Toth, C., and Oberhauser, H. Variational gaussian processes with signature covariances. *arXiv preprint arXiv:1906.08215* (2019).

[56] Toth, C., and Oberhauser, H. Bayesian learning from sequential data using gaussian processes with signature covariances. In *International Conference on Machine Learning* (2020), PMLR, pp. 9548–9560.

[57] Wiese, M., Knobloch, R., Korn, R., and Kretschmer, P. Quant gans: Deep generation of financial time series. *Quantitative Finance 20*, 9 (2020), 1419–1440.

[58] Xu, T., Wenliang, L. K., Munn, M., and Acciaio, B. Cot-gan: Generating sequential data via causal optimal transport. *arXiv preprint arXiv:2006.08571* (2020).

[59] Zhang, K., Zhong, G., Dong, J., Wang, S., and Wang, Y. Stock market prediction based on generative adversarial network. *Procedia computer science 147* (2019), 400–406.