Signature-based model recognition

in financial time series

I.J. Kooijman

Technische Universiteit Delft

TU Delft

Signature-based model recognition

in financial time series

by

I.J. Kooijman

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday May 25, 2021 at 16:00 PM.

Student number:	4365925	
Project duration:	September 1, 2020 – May 25, 2021	
Thesis committee:	Prof. dr. ir. C.W. Oosterlee,	TU Delft/Universiteit Utrecht, supervisor
	Dr. A.A. Tsvetkov,	ING Bank, daily supervisor
	Dr. C. Cantia,	ING Bank, daily supervisor
	Dr. R.J. Fokkink	TU Delft
	Prof. Dr. Ir. C. Vuik	TU Delft
	K. Andersson, MSc.	CWI

An electronic version of this thesis is available at http://repository.tudelft.nl/.
Cover image adapted from http://freepik.com.

TU Delft

Preface

Before you lies the thesis “*Signature-based model recognition in financial time series*”. This thesis entails my efforts in identifying the generative stochastic process of an observed (financial) time series by using the signature transformation. It is written in order to successfully complete the master program Applied Mathematics at the Delft University of Technology. The research is conducted in collaboration with ING Bank.

I am thankful for the opportunity to collaborate with the Model Development department within ING. I would like to thank my daily supervisors Artem Tsvetkov and Catalin Cantia from ING for their support, ideas and guidance, which helped me a lot in the past nine months. Additionally, I would like to thank Cornelis Oosterlee and Kristoffer Andersson who supervised me from TU Delft/CWI for their suggestions, feedback and kind conversations. Because of the current Covid-19 situations, our meetings were online, but despite the distance I felt supported in my research, for which I am grateful.

Furthermore, I would like to thank my family and friends for their unwavering support, enthusiasm, optimism and coffee breaks. Working on a thesis in times of Covid-19 was challenging, so their support was very valuable to me, as were the conversations with my fellow TU Delft graduates. Specifically I would like to thank my boyfriend for his kindness and his willingness to listen to my breakthroughs and challenges in the past nine months.

I am glad you are taking the time to read my thesis and I wish you a pleasant reading.

I.J. Kooijman
Delft, May 2021

Abstract

In financial mathematics, stochastic processes are regularly used to describe observed financial indicators such as stocks, options, futures or interest rates. Identifying the underlying dynamics of observed financial time series is crucial in risk management, as it greatly affects pricing and hedging strategies. The large number of available stochastic processes make selecting the most suitable stochastic process a non-trivial problem. Additionally, realisations of stochastic processes are elements of the path space which is infinite-dimensional and non-locally compact. Given these observations, we find that model selection methods from classical statistics, such as distribution metrics, are inadequate. In this thesis a signature-based model recognition method is proposed. The goal of this model is to select the most suitable stochastic process to describe an observed financial time series. Signatures are transformations from a path to an infinite-length sequence of properties of that path, which makes signatures a highly interesting approach to construct input features which can be used by a machine learning model. In our evaluation, we use the aforementioned methodology to distinguishing between various classification settings of Arithmetic Brownian Motion, Geometric Brownian Motion and the exponential jump diffusion process, both in a binary and multi-class classification setting. This evaluation shows that the proposed method can adequately distinguish between the same model with different parameters, models with and without jumps and models with different jump sizes and jump intensities.

Contents

1 Introduction

The analysis of financial time series is a field that has been extensively studied in the past years. Financial time series represent the historical evolution of different financial indicators like interest rates, stock prices or indices, foreign exchange rates, volatility curves and many more. These time series can be modelled through stochastic processes and a multitude of these processes were introduced over the last 50 years, such as Heston's model, the Hull-White model or the jump diffusion process (see Oosterlee et al. [41]). The introduction of these new models created a new problem of choosing the most appropriate model when observing a financial time series. This problem is often referred to as the *model selection problem* and is not unique to the financial field, but extends to the entire statistical field.

Selecting the most suitable model is vital for financial institutions like banks, insurance companies and pension funds. Each financial institution has a risk department whose purpose is to identify and manage potential risks for the company. These departments use the aforementioned mathematical models to model all types of financial risks, ranging from market risk (i.e. the risk of loss due to movement in the market), credit risk (i.e. the risk of loss due to default of the counter party), interest risk (i.e. risk of loss due to movement in the interest rate) to many more types of risk. In order to manage these risks, financial institutions hedge their positions through the trade of financial derivatives like options or futures. Finding the fair price of these financial products is essential, as they have a great impact on the hedging strategy. The price of these derivatives depends on the underlying assets, which is often modelled through a stochastic process. Therefore the choice of stochastic process is important, it influences the pricing equations and thereby the hedging strategy of the banks. Selecting the most suitable process to model the observed financial time series is a repeating challenge for the risk management departments.

A significant complication in selecting a suitable stochastic process derives from the fact that time series - or *paths* - sampled from a stochastic process are elements of the path space. The path space is the space that contains all possible paths in a certain time frame and is infinite dimensional and non-locally compact according to [10]. These properties cause complications when making inferences on the path space. Traditionally - for time series that belong to spaces that are finite-dimensional and compact - empirical distribution metrics are used to find the most suitable distribution. One can think of popular metrics like the Kolmogorov-Smirnov statistic, the Anderson Darling statistic or the Cramér–von Mises statistic (see [44]). Although these distributional metrics for marginals can be extended to (finite) multivariate distributions, the extension to the infinite dimensional path space is not straightforward, as mentioned in [14]. This issue greatly complicates the matter of selecting the correct model.

In summary, the problem can be formulated as shown below.

Problem. *Let $\{t_1, t_2, \ldots, t_n\}$ be discrete time points on which we observe path path $\omega_D = \{D_{t_1}; D_{t_2}; \ldots; D_{t_n}\}$. Given that ω_D is sampled from the continuous data generating process D_t, identify which class of parametric models M_i with parameter set θ^i from a predetermined set of models $\mathbb{M} = \{M_1(\theta^1); M_2(\theta^2); \ldots; M_k(\theta^k)\}$ is likely to have generated the path ω_D.*

Remark 1. *It is essential to remark that in this thesis the focus is put not only on distinguishing between paths generated by different models, but also between paths generated by the same model with different parameter sets θ^i.*

The choice of possible model $M_i \in \mathbb{M}$ is broad, examples are Brownian Motion, Heston Model, the jump diffusion process, Ornstein–Uhlenbeck etc.

In the remainder of this thesis, a transformation from path space to an alternative space is proposed in order to enhance the handling of the model selection problem. This transformation is called the *signature transformation* and transforms the path to a sequence of iterated integrals over the path, called the *signature*. This signature is a vector of infinite length, where every element of the vector is related to statistical properties of the path.

Before we take an in-depth look at how we can use signatures in solving the model selection problem, we first describe the stochastic fundamentals that are needed during the rest of this work in chapter 2. We introduce the problem in more detail in chapter 3, followed by a short overview of related approaches, the introduction of the signature and mathematical considerations of which the reader needs to be aware. Chapter 4 provides an overview of the methodology and used machine learning methods in this thesis. These machine learning methods are used to classify path ω_D to its likely generative process. The empirical results of this classification are presented in chapter 5. We conclude this thesis in chapter 6 and present our views on further research in this interesting research domain.

2 Preliminaries

Before describing the problem and its difficulties in more detail in chapter 3, this chapter describes mathematical concepts needed to fully define the problem. The set of models $\mathbb{M} = \{M_1(\theta^1); M_2(\theta^2); \dots; M_k(\theta^k)\}$ from which we would like to select the most appropriate model are taken to be *stochastic processes*. A stochastic process is a collection of random variables usually indexed by a time component. For financial applications, researchers are interested in the movements of the market, also called the *dynamics* of the market. Whereas in standard calculus movements are often described differentiable Ordinary Differential Equations (ODEs) or Partial Differential Equations (PDEs), stochastic processes are often driven by objects that are nowhere differentiable. Therefore the dynamics of the markets are often described Stochastic Differential Equations (SDEs), which will be introduced in this chapter. Furthermore, the three main stochastic models chosen to focus on in this thesis are given.

As described in the introduction, the goal in this thesis is to identify the generative process of an observed time series out of a set of stochastic processes. Let $(\Omega, \mathcal{F}_t, \mathbb{P})$ be a probability space where Ω denotes the space of possible outcomes, $(\mathcal{F}_t)_{t\in(0,T]}$ denotes a natural filtration, $\mathbb{P}$ denotes the market probability measure and $t \in (0, T]$ denotes time. The dynamics of stochastic process $(X(t))_{t>0}$ can be described through its stochastic differential equation.

$$\mathrm{d}X(t) = \bar{\mu}(t, X(t))\mathrm{d}t + \bar{\sigma}(t, X(t))\mathrm{d}W(t) \quad \text{for } t \in (0, T], \;\; X(0) = x_0. \tag{2.1}$$

Here $x_0 \in \mathbb{R}$, $\bar{\mu} : (0, T] \times \mathbb{R}^n \to \mathbb{R}^n$ is often referred to as the *drift* coefficient, i.e. the general direction of the process and $\bar{\sigma} : (0, T] \times \mathbb{R}^n \to \mathbb{R}^{n\times d}$ is referred to as the *volatility* coefficient, i.e. a diffusion term interpreted as the uncertainty. Notice that both $\bar{\mu}(t, X(t))$ and $\bar{\sigma}(t, X(t))$ are $\mathcal{F}_t$-measurable since $(X(t))_{t>0}$ is $\mathcal{F}_t$-measurable.

Equation (2.1) is interpreted as:

$$X(t) = x_0 + \int_0^t \bar{\mu}(s, X(s))\mathrm{d}s + \int_0^t \bar{\sigma}(s, X(s))\mathrm{d}W(s). \tag{2.2}$$

It is noteworthy to mention that the solution to equation (2.2) does not always exist, see [40] for details.

There exist many stochastic processes to describe financial processes. Any stochastic process can be included in the set of models $\mathbb{M}$, the only requirement being that paths of that model can be generated, either directly through the solution of (2.2) or - if the solution can not be expressed analytically - through numerical approximations like the Euler or Milstein scheme (see [41]). In this thesis, the focus is put on the following stochastic processes:

- Arithmetic Brownian Motion
- Geometric Brownian Motion

- Exponential jump diffusion process

These three processes are chosen because they are frequently used by practitioners in the financial field and since the density function of the discrete variants of Arithmetic Brownian Motion and Geometric Brownian Motion is easily obtained, which is useful in further analysis later in the thesis.

Before going into the definitions of the specific stochastic processes mentioned above, we look into the definition of a *Wiener process*.

Definition 2.1 (Wiener process, [41]). *A Wiener process $W = (W(t))_{t>0}$ is a stochastic process characterised by the following properties:*

1. *$W(t_0) = 0$ (t_0 is the starting time of the process).*
2. *$(W(t))_{t>0}$ is almost surely continuous in t.*
3. *$(W(t))_{t>0}$ has independent increments.*
4. *The increments are normally distributed, i.e. for all $s, t \in \mathbb{R}^+ : W(t) - W(s) \sim N(0, |t-s|)$.*

Additionally, Itô's lemma is needed. The version of Itô's lemma written below is cited from [43].

Theorem 2.1 (Itô's lemma, [41]). *Suppose a process $(X(t))_{t>0}$ follows the Itô dynamics,*

$$dX(t) = \bar{\mu}(t, X(t))dt + \bar{\sigma}(t, X(t))dW(t), \quad \text{with } X(t_0) = X_0,$$

where drift coefficient $\bar{\mu}(t, X(t))$ and volatility coefficient $\bar{\sigma}(t, X(t))$ satisfy the standard Lipschitz conditions on the growth of these functions.
Let $g : [0, T] \times \mathbb{R} \to \mathbb{R}$ be a function of $X = X(t)$ and time t, with continuous partial derivatives $\partial g/\partial x$, $\partial^2 g/\partial x^2$, $\partial g/\partial t$. Let $Y = (Y(t))_{t>0}$ be defined by $Y(t) := g(t, X(t))$. Then it holds that:

$$dY(t) = \left(\frac{\partial g}{\partial t} + \bar{\mu}(t, X)\frac{\partial g}{\partial x} + \frac{1}{2}\frac{\partial^2 g}{\partial x^2}\bar{\sigma}^2(t, X)\right)dt + \frac{\partial g}{\partial x}\bar{\sigma}(t, X)dW(t). \tag{2.3}$$

Proof. The formal prove is given in [43]. □

We will now look into the aforementioned stochastic processes and their properties.

2.1. Arithmetic Brownian Motion

Arithmetic Brownian Motion (ABM) is a well-known adapted stochastic process $X^A(t)$, governed by

$$\mathrm{d}X^A(t) = \mu \mathrm{d}t + \sigma \mathrm{d}W^{\mathbb{P}}(t).$$

Here $\mu \in \mathbb{R}$ denotes the drift coefficient, $\sigma \in \mathbb{R}^+$ denotes the volatility coefficient and $(W^{\mathbb{P}}(t))_{t>0}$ denotes the Wiener process described in definition 2.1, under the market measure. The SDE can easily be solved by integration:

$$\begin{aligned}
\int_{t_0}^{t} \mathrm{d}X^A(t) &= \int_{t_0}^{t} \mu \mathrm{d}t + \sigma \int_{t_0}^{t} \mathrm{d}W^{\mathbb{P}}(t), \\
X^A(t) - X^A(t_0) &= \mu(t - t_0) + \sigma(W^{\mathbb{P}}(t) - W^{\mathbb{P}}(t_0)), \\
X^A(t) &= X^A(t_0) + \mu(t - t_0) + \sigma(W^{\mathbb{P}}(t) - W^{\mathbb{P}}(t_0)). \qquad (2.4)
\end{aligned}$$

Notice that X^A is normally distributed since $X^A(t_0)$ and $\mu(t - t_0)$ are deterministic and $W^{\mathbb{P}}(t) - W^{\mathbb{P}}(t_0)$ is normally distributed per definition 2.1. Since we have

$$\begin{aligned}
\mathbb{E}^{\mathbb{P}}[X^A(t)] &= X^A(t_0) + \mu(t - t_0) + \sigma\mathbb{E}[W^{\mathbb{P}}(t) - W^{\mathbb{P}}(t_0)] = X^A(t_0) + \mu(t - t_0), \\
\mathbb{V}\mathrm{ar}^{\mathbb{P}}(X^A(t)) &= \mathbb{V}\mathrm{ar}^{\mathbb{P}}(\sigma(W^{\mathbb{P}}(t) - W^{\mathbb{P}}(t_0))) = \sigma^2(t - t_0),
\end{aligned}$$

we see that

$$X^A(t) \sim N\Big(X^A(t_0) + \mu(t - t_0),\ \sigma^2(t - t_0)\Big). \tag{2.5}$$

Furthermore, it is relevant to note that ABM has independent increments, due to the third property of the definition of the Wiener process.

Arithmetic Brownian Motion was one of the earliest processes used to model asset prices because of its relative simplicity; the density function of random variable $X(t)$ is known for all $t > 0$ and the increments of $(X(t))_{t>0}$ are independent. It was introduced by Louis Bachelier in 1900 ([1]) and is chosen as the underlying model in the Bachelier pricing model, used to price financial derivatives [41]. It can give rise to negative asset prices, which is an unwanted characteristic and caused a decrease in usage. However, recently it has gained popularity again in the modelling of prices of interest-based derivatives as a result of the negative interest rates observed in the markets, see [26]. An impression of the behaviour of the process is shown in figure 2.1.

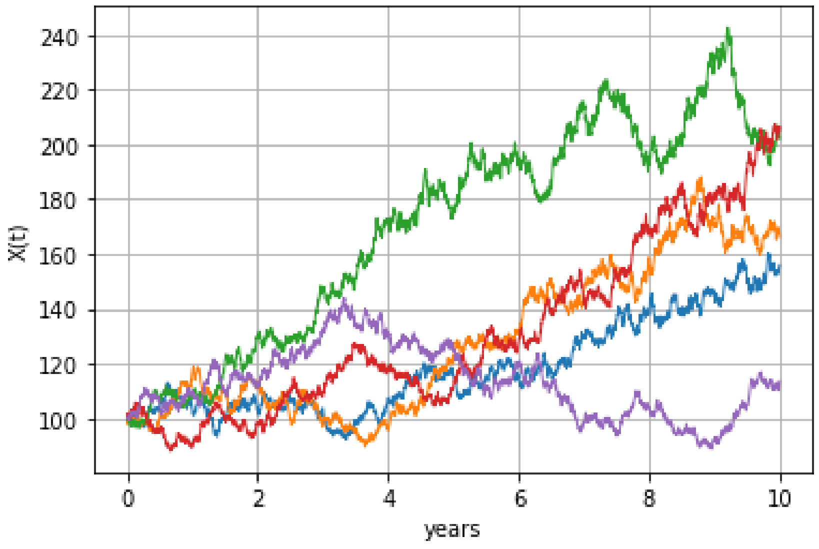


Figure 2.1: Five ABM paths, generated with parameters $X(t_0) = 100, \mu = 0.05, \sigma = 0.1$ and $T = 10$.

Parameter estimation
An important concern in practice is, once the model M_i is chosen, to estimate the parameter $theta_i$. For ABM we have $\theta = (\mu, \sigma)$. The estimation of the parameters is known as calibration of the process and is important for practical purposes like forecasting. Since we are analysing a process under the market measure $\mathbb{P}$ and μ, σ are assumed to be constant over time, this estimation can be done by calibrating to historical asset prices. A big advantage of ABM is that the distribution of $X(t)$ is known and its increments are independent, which means that μ, σ can be estimated by Maximum Likelihood Estimators (MLE). This is not always the case for other stochastic processes. The following results are based on analysis in [41].

In order to use MLE, we need the conditional distribution of $X^A(t)$ given its past. Let t denote the current time and let $t + \Delta t$ denote some time in the future. Then, similarly as before, we have that

$$\begin{aligned}\mathbb{E}^{\mathbb{P}}[X^A(t+\Delta t)|\mathcal{F}(t)] &= \mathbb{E}^{\mathbb{P}}[X^A(t) + \mu(t+\Delta t - t) + \sigma(W^{\mathbb{P}}(t+\Delta t) - W^{\mathbb{P}}(t))|\mathcal{F}(t)] \\ &= X^A(t) + \mu\Delta t,\end{aligned}$$

where $\mathbb{E}^{\mathbb{P}}[X^A(t)|\mathcal{F}(t)] = X^A(t)$ because $(X^A(t))_{t>0}$ was assumed to be $\mathcal{F}(t)$-measurable and $\mathbb{E}^{\mathbb{P}}[W^{\mathbb{P}}(t+\Delta t) - W^{\mathbb{P}}(t))|\mathcal{F}(t)] = 0$ by property 3 in definition 2.1. Furthermore, we have

$$\begin{aligned}\mathbb{V}\mathrm{ar}^{\mathbb{P}}(X^A(t+\Delta t)|\mathcal{F}(t)) &= \mathbb{V}\mathrm{ar}^{\mathbb{P}}(X^A(t) + \mu(t+\Delta t - t) + \sigma(W^{\mathbb{P}}(t+\Delta t) - W^{\mathbb{P}}(t))|\mathcal{F}(t)) \\ &= \mathbb{V}\mathrm{ar}^{\mathbb{P}}(X^A(t)|\mathcal{F}(t)) + \mathbb{V}\mathrm{ar}^{\mathbb{P}}(\sigma(W^{\mathbb{P}}(t+\Delta t) - W^{\mathbb{P}}(t))|\mathcal{F}(t)).\end{aligned}$$

Naturally $\mathbb{V}\mathrm{ar}^{\mathbb{P}}(X^A(t)|\mathcal{F}(t)) = 0$. Since $W^{\mathbb{P}}(t+\Delta t) - W^{\mathbb{P}}(t) \sim N(0, \Delta t)$ according to property 4 in definition 2.1, we have that

$$\mathbb{V}\text{ar}^{\mathbb{P}}(X^A(t+\Delta t)|\mathcal{F}(t)) = \sigma^2\Delta t.$$

Therefore we arrive at

$$X^A(t+\Delta t)|X^A(t) \sim N(X^A(t)+\mu\Delta t, \sigma^2\Delta t). \tag{2.6}$$

Now that the conditional distribution of $X^A(t)$ is known, the likelihood function can be computed. Let $X(t_0), \dots, X(t_m)$ denote *discrete* historical asset prices. Due to the independent increments of $(X(t))_{t>0}$, we can write the likelihood function as

$$L(\mu,\sigma^2|X^A(t_0),\dots,X^A(t_m)) = \prod_{k=0}^{m-1} f_{X^A(t_{k+1})|X^A(t_k)}(X^A(t_{k+1})).$$

Since the distribution of $X^A(t_{k+1})|X^A(t_k)$ is known and given in equation (2.6), we have

$$L(\mu,\sigma^2|X^A(t_0),\dots,X^A(t_m)) = \prod_{k=0}^{m-1} \frac{1}{\sqrt{2\pi\sigma^2\Delta t}} \exp\Big(-\frac{1}{2}\Big(\frac{X^A(t_{k+1})-X^A(t_k)-\mu\Delta t}{\sigma\sqrt{\Delta t}}\Big)^2\Big).$$

In order to find $\hat{\mu},\hat{\sigma}$ such that L is maximised, we take the logarithm to simplify the maximisation:

$$\begin{aligned}\log L(\mu,\sigma^2|X^A(t_0),\dots,X^A(t_m)) &= \log\Big(\frac{1}{\sqrt{2\pi\sigma^2\Delta t}}\Big)^m + \sum_{k=0}^{m-1}\Big(-\frac{1}{2}\Big(\frac{X^A(t_{k+1})-X^A(t_k)-\mu\Delta t}{\sigma\sqrt{\Delta t}}\Big)^2\Big)\\ &= -\frac{m}{2}\log(2\pi\sigma^2\Delta t) + \sum_{k=0}^{m-1}\Big(-\frac{1}{2}\Big(\frac{X^A(t_{k+1})-X^A(t_k)-\mu\Delta t}{\sigma\sqrt{\Delta t}}\Big)^2\Big)\end{aligned}$$

Taking the derivative w.r.t μ and σ^2 and setting it equal to zero gives the Maximum Likelihood Estimators:

$$\hat{\mu} = \frac{1}{m\Delta t}(X^A(t_m)-X^A(t_0)), \;\; \hat{\sigma}^2 = \frac{1}{m\Delta t}\sum_{k=1}^{m-1}(X^A(t_{k+1})-X^A(t_k)-\hat{\mu}\Delta t)^2. \tag{2.7}$$

Remark 2. *Notice that the availability of the density function of $X^A(t)$ and the independence of the increments of $(X^A(t))_{t>0}$ allow the estimation of μ,σ through the MLE based on discrete, historical observations. However, many processes do not have an analytically available density function, nor independent increments. The calibration of such processes can therefore be challenging and time consuming since new derivations must be made for every individual stochastic process. Therefore a model agnostic calibration method, as proposed later in this thesis, can be beneficial.*

Quadratic variation
A quantity which will play a role later in the thesis is the quadratic variation of a process.

Definition 2.2 (Quadratic variation). *Let $(X(t))_{t>0}$ be a stochastic process defined on probability space $(\Omega,\mathcal{F}_t,\mathbb{P})$. Let $\mathcal{P}=\{t_0,\dots,t_n\}$ be a discretisation of the interval $[0,T]$ and $||\mathcal{P}||$ denotes the mesh, i.e.*

$$||\mathcal{P}|| := \max_{i=0}^{n-1}\; t_{i+1}-t_i.$$

The quadratic variation of $(X(t))_{t>0}$ is given by

$$QV(X) := \lim_{||\mathcal{P}||\to 0}\sum_{i=0}^{n-1}\big(X(t_{i+1})-X(t_i)\big)^2.$$

For ABM, this quadratic variation can be computed analytically. Using equation (2.4), we have

$$\begin{aligned} QV(X^A) &= \lim_{||\mathcal{P}||\to 0} \sum_{i=0}^{n-1} \big(X^A(t_0) + \mu(t_{i+1} - t_0) + \sigma(W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_0)) \\ &\quad - X^A(t_0) - \mu(t_i - t_0) - \sigma(W^{\mathbb{P}}(t_i) - W^{\mathbb{P}}(t_0))\big)^2 \\ &= \lim_{||\mathcal{P}||\to 0} \sum_{i=0}^{n-1} \big(\mu(t_{i+1} - t_i) + \sigma(W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i))\big)^2 \\ &= \lim_{||\mathcal{P}||\to 0} \sum_{i=0}^{n-1} \mu^2(t_{i+1} - t_i)^2 + \sigma^2(W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i))^2 + 2\mu\sigma(t_{i+1} - t_1)(W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i)). \end{aligned}$$

Since we have $||\mathcal{P}|| \to 0$, certain terms in the expression above will disappear. Notice that

$$\lim_{||\mathcal{P}||\to 0} \sum_{i=0}^{n-1} \mu^2(t_{i+1} - t_i)^2 = 0,$$

since $(t_{i+1} - t_i)^2 \to 0$ faster than $t_{i+1} - t_i \to 0$ when $||\mathcal{P}|| \to 0$. Furthermore, we have

$$\begin{aligned} \mathbb{E}^{\mathbb{P}}[\sum_{i=0}^{n-1} (t_{i+1} - t_1)(W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i))] &= \sum_{i=0}^{n-1} (t_{i+1} - t_1)\mathbb{E}^{\mathbb{P}}[W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i)] = 0 \\ \mathbb{V}\text{ar}^{\mathbb{P}}\big(\sum_{i=0}^{n-1} (t_{i+1} - t_1)(W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i))\big) &\overset{\text{i.i.d}}{=} \sum_{i=0}^{n-1} (t_{i+1} - t_1)^2 \mathbb{V}\text{ar}^{\mathbb{P}}\big(W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i)\big) = \sum_{i=0}^{n-1} (t_{i+1} - t_i)^3, \end{aligned}$$

so $\lim_{||\mathcal{P}||\to 0} \sum_{i=0}^{n-1} (t_{i+1} - t_1)(W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i)) \to 0$ since its expectation equals zero and its variance tends to 0 as $||\mathcal{P}|| \to 0$. Therefore we arrive at

$$QV(X^A) = \lim_{||\mathcal{P}||\to 0} \sum_{i=0}^{n-1} \sigma^2(W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i))^2.$$

Since $W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i) \sim N(0, t_{i+1} - t_i)$, we know that

$$\begin{aligned} \mathbb{E}^{\mathbb{P}}[\sum_{i=0}^{n-1} (W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i))^2] &= \sum_{i=0}^{n-1} \mathbb{E}^{\mathbb{P}}[(W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i))^2] = \sum_{i=0}^{n-1} t_{i+1} - t_i = T, \\ \mathbb{V}\text{ar}^{\mathbb{P}}\big(\sum_{i=0}^{n-1} (W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i))^2\big) &\overset{\text{i.i.d}}{=} \sum_{i=0}^{n-1} \mathbb{V}\text{ar}^{\mathbb{P}}\big((W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i))^2\big) \\ &= \sum_{i=0}^{n-1} \mathbb{E}^{\mathbb{P}}[(W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i))^4] - \big(\mathbb{E}^{\mathbb{P}}[(W^{\mathbb{P}}(t_{i+1}) - W^{\mathbb{P}}(t_i))^2]\big)^2 \\ &= \sum_{i=0}^{n-1} 3(t_{i+1} - t_i)^2 - (t_{i+1} - t_i)^2 \\ &= \sum_{i=0}^{n-1} 2(t_{i+1} - t_i)^2. \end{aligned} \tag{2.8}$$

Similar as before we see that the variance in (2.8) tends to 0 as $||\mathcal{P}|| \to 0$, from which we can conclude that

$$QV(X^A) = \sigma^2 T. \tag{2.9}$$

This result implies that if the quadratic variation is known, so is the volitility and vice versa. This is a helpful result which is used in chapter 3.

2.2. Geometric Brownian Motion

Geometric Brownian Motion (GBM) is similar to Arithmetic Brownian Motion; the logarithm of a Geometric Brownian Motion is equal to Arithmetic Brownian Motion. Its SDE is given below:

$$\mathrm{d}X^G(t) = (\mu + \frac{1}{2}\sigma^2)X^G(t)\mathrm{d}t + \sigma X^G(t)\mathrm{d}W^{\mathbb{P}}(t).$$

Applying Itô's lemma (theorem 2.1) with $\bar{\mu}(t, X^G) = (\mu + \frac{1}{2}\sigma^2)X^G(t)$, $\bar{\sigma}(t, X^G) = \sigma X^G(t)$ and $g(t, X^G) = \log X^G$ results in $\partial g/\partial X^G = 1/x$, $\partial^2 g/\partial X^2 = -1/x^2$ and $\partial g/\partial t = 0$. Inserting this in equation (2.3) gives

$$\begin{aligned} \mathrm{d}Y(t) &= \Big(0 + (\mu + \frac{1}{2}\sigma^2)X^G(t)\frac{1}{X^G(t)} + \frac{1}{2}\frac{-1}{(X^G(t))^2}(\sigma X^G(t))^2\Big)\mathrm{d}t + \frac{1}{X^G(t)}\sigma X^G(t)\mathrm{d}W(t) \\ &= \mu\mathrm{d}t + \sigma\mathrm{d}W(t). \end{aligned}$$

Integration gives

$$\begin{aligned} \int_{t_0}^{t} \mathrm{d}Y(t) &= \int_{t_0}^{t} \mu\mathrm{d}t + \int_{t_0}^{t} \sigma\mathrm{d}W(t), \\ Y(t) &= Y(t_0) + \mu(t - t_0) + \sigma(W(t) - W(t_0)). \end{aligned}$$

Recall that $Y(t) = g(t, X^G) = \log X^G$. This results in

$$X^G(t) = X^G(t_0)e^{\mu(t-t_0)+\sigma(W^{\mathbb{P}}(t)-W^{\mathbb{P}}(t_0))}.$$

Notice that $X^G(t) = \exp(X^A(t))$. Since the distribution of $X^A(t)$ is known and given in equation (2.5), we have that $X^G(t)$ is lognormally distributed, i.e.

$$X^G(t) \sim \text{lognormal}(X^A(t_0) + \mu(t - t_0), \sigma^2(t - t_0))$$

As is indicated above, Geometric Brownian Motion and Arithmetic Brownian Motion are clearly related. GBM is an adaptation of ABM which has the advantage that its generated asset paths do not give rise to negative values, whereas paths generated by ABM do. Additionally, since the process is multiplicative instead of additive like ABM, the impact of the coefficient μ, σ is relative to the current asset value. This is also often observed in the market. It is therefore a popular stochastic model and its dynamics are used as the underlying dynamics in the Black-Scholes model, which is - like the Bachelier model - also a model used to price financial derivatives and is considered the standard baseline model in financial engineering, see [41]. An impression of the behaviour of the paths is shown in figure 2.2.

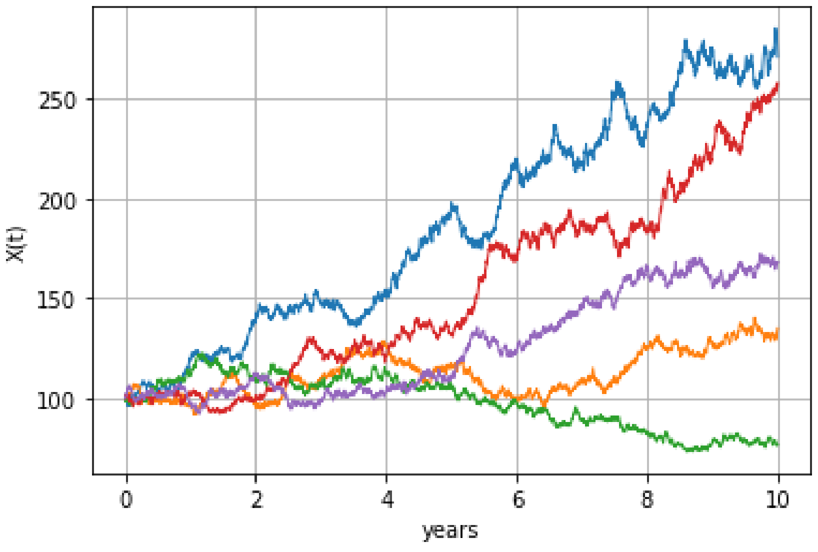


Figure 2.2: Five GBM paths, generated with parameters $X(t_0) = 100, \mu = 0.05, \sigma = 0.1$ and $T = 10$.

Parameter estimation
Since GBM is essentially the exponent of ABM, the parameter estimation is very similar. For historical observations $X^G(t_0), \dots, X^G(t_m)$ which are assumed to follow a GBM, the coefficients μ, σ can be computed by realising that $\log X^G(t_0), \dots, \log X^G(t_0)$ are historical observations assumed to follow an ABM and using the MLE as desribed in (2.7).

Quadratic variation
Next we compute quadratic variation of a GBM. Unlike ABM, the quadratic variation of GBM is not deterministic and depends on the the process $(X^G(t))_{t>0}$. To shorten the derivation, we will use the integral notation of the quadratic variation, equivalent to definition 2.2.

$$
\begin{aligned}
QV(X^G) &= \int_0^T \left((dX^G)^2\right. \\
&= \int_0^T \left((\mu + \frac{1}{2}\sigma^2)X^G \mathrm{d}t + \sigma X^G \mathrm{d}W^{\mathbb{P}}(t)\right)^2 \\
&= \int_0^T \left((\mu + \frac{1}{2}\sigma^2)^2 (X^G)^2 (\mathrm{d}t)^2 + \sigma^2 (X^G)^2 (\mathrm{d}W^{\mathbb{P}}(t))^2 + 2\sigma(\mu + \frac{1}{2}\sigma^2 (X^G)^2 \mathrm{d}t\, \mathrm{d}W^{\mathbb{P}}(t).\right.
\end{aligned}
$$

Similar as before in the discrete setting, we have that $(\mathrm{d}t)^2 \to 0$, $\mathrm{d}t\mathrm{d}W^{\mathbb{P}}(t) \to 0$ and $(\mathrm{d}W^{\mathbb{P}}(t))^2 \to \mathrm{d}t$. This results in

$$
QV(X^G) = \int_0^T \sigma^2 (X^G(t))^2 \mathrm{d}t. \tag{2.10}
$$

This means that the quadratic variation of GBM still depends on the values of the process. Nevertheless, equation 2.10 shows that also for GBM there exists a relationship between the volatility coefficient and the quadratic variation. Having information about the quadratic variation of a path is therefore useful in determining the volatility parameter. This result is helpful in chapter 3.

2.3. Exponential jump diffusion process

Exponential jump diffusion processes are processes that contain jumps in the generated paths. It is an adaptation of the GBM where the arrival of jumps is governed by a Poisson process $(X_P(t))_{t>0}$ with jump intensity $\lambda > 0$ and the size J of the jump is governed by a certain probability distribution chosen by the user. For completeness, the definition of a Poisson process is given in definition 2.3.

Definition 2.3 (Poisson process, [41]). *A Poisson process $(X_P(t))_{t>0}$ with parameter $\lambda > 0$ is an integer-valued stochastic process with the following properties:*

1. $X_P(0) = 0$.
2. *Let $t_0 = 0 < t_1 < \dots < t_n = T$. For all $i \in \{1, \dots, n\}$, increments $X_P(t_{i+1}) - X_P(t_i)$ are independent random variables.*
3. *For $s \geq 0,\ t > 0$ and integers $k \geq 0$, the increments have the Poisson distribution:*

$$
\mathbb{P}(X_P(s+t) - X_P(s) = k) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}.
$$

First, the SDE of ABM with jumps $(X^J(t))_{t>0}$ is given:

$$
\mathrm{d}X^J(t) = \mu \mathrm{d}t + \sigma \mathrm{d}W^{\mathbb{P}}(t) + J \mathrm{d}X_P(t).
$$

This SDE can be solved by integration, resulting in

$$
X^J(t) = X^J(t_0) + \mu(t - t_0) + \sigma(W^{\mathbb{P}}(t) - W^{\mathbb{P}}(t_0)) + \sum_{i=1}^{X_P(t)} J_i.
$$

Since in this section the exponential jump diffusion process is considered, the exponent of $X^J(t)$ called $X^{EJ}(t)$ is taken:

$$X^{EJ}(t) = X^{EJ}(t_0)e^{\mu(t-t_0)+\sigma(W^{\mathbb{P}}(t)-W^{\mathbb{P}}(t_0))}\prod_{i=1}^{X_P(t)} e^{J_i}.$$

Notice that due to the presence of jumps, $(X^{EJ}(t))_{t>0}$ is no longer a continuous process. The distribution of jump size J can be chosen by the user. Two popular choices are given below.

- *Classical Merton's model* ([37]): the jump size J is assumed to follow a normal distribution, i.e. $J \sim N(\mu_J, \sigma_J^2)$.
- *Non-symmetrical double exponential* (Kou's model, [31]): the jump size J is assumed to follow a distribution F_J with density function

$$f_J(x) = p_1\alpha_1 e^{-\alpha_1 x}\mathbb{1}_{\{x\geq 0\}} + p_2\alpha_2 e^{\alpha_2 x}\mathbb{1}_{\{x<0\}}.$$

In this thesis the choice is made to analyse the classical Merton's model with $J \sim N(0, \sigma_J^2)$.

Jump diffusion processes are extensions of the ABM and GBM given in the previous paragraphs. The main advantage of jump diffusion processes compared to ABM and GBM is that they can generate paths containing jumps. Jumps are often observed in the asset prices in the market, especially in volatile times like the 2008 financial crisis or the start of the Covid crisis in 2020. Additionally, according to [41], the jump diffusion process generally generates paths of which the increments (often called *returns*) have distributions with heavy tails. This, too, is often observed in the financial markets and is something that ABM and GBM to not exhibit. A disadvantage of jump processes is that jumps are not "tradable quantities". Therefore, when the jump diffusion process is chosen as an underlying process to price financial derivatives, there does not exist a perfect hedging strategy that perfectly replicates the value of the financial derivative. Nevertheless, it is a popular stochastic model used in practice.

An impression of the behaviour of the process is given in figure 2.3.

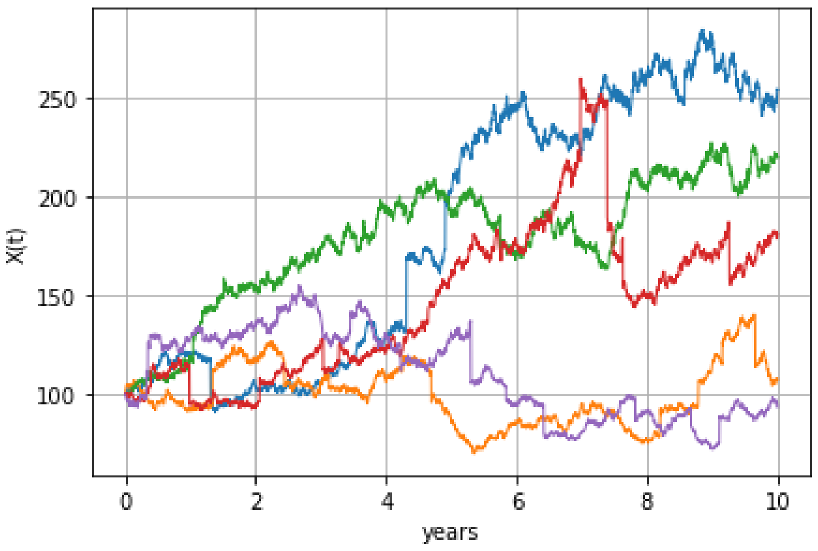


Figure 2.3: Five exponential jump diffusion paths, generated with parameters $X(t_0) = 100, \mu = 0.05, \sigma = 0.1, T = 10, \lambda = 1, \mu_J = 0$ and $\sigma_J = 0.1$.

3
Problem setting

This chapter presents a overview of the problem setting, starting with further background information on the problem in section 3.1. In section 3.2 a short overview is given of other research fields that are related to this problem, after which we introduce in section 3.3 the signature and its properties. Section 3.4 provides the proposed solution and the chapter concludes with some important considerations about the maximum classification accuracy in section 3.5.

3.1. The problem

As touched upon in chapter 1, the goal of this thesis is to identify the generative stochastic process of an observed financial time series. Correctly identifying the stochastic process is very beneficial, as stochastic processes can be used for forecasting the asset, pricing of derivatives based on the asset and thereby influencing the hedging strategies. For this reason the analysis in this thesis is focused on classifying time series to their generative stochastic process.

Identifying the generative process is problematic when this process is continuous in time. In general, the modelling of time series is done in two ways:

1. Discrete models,
2. Continuous models.

Discrete models
Discrete models are effective tools in the regression and forecasting of time series. Examples of popular discrete models are generalised linear models, AR(I)MA, (G)ARCH etc. Every model is a set of joint probability distributions and assumes that the time series of interest can be viewed as a sequence of random variables of which the true joint probability distribution is supposed to be in this set of joint probabilities. The objective is to find the joint distribution function that has the highest likelihood of having generated the observed financial time series. Note that this is similar to the objective of this thesis. The quality of the model for the given time series can then be assessed through distribution metrics like Kolmogorov-Smirnov, Anderson-Darling etc. or goodness-of-fit criteria like Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC), see [44].

Discrete models are effective tools in forecasting and regression and selecting the correct discrete model provides its own challenges. However, they are typically not used as underlying models in pricing financial derivatives, as they do not describe the dynamics but rather the distributions of financial quantities. As mentioned before, we focus on the models underlying pricing equations, we will therefore not further analyse discrete models.

Continuous models
Please recall that a mathematical introduction of the continuous models is given in chapter 2. As mentioned in chapter 1, stochastic processes are used to model the underlying asset when pricing financial

derivatives. For example when pricing options, using ABM to model the underlying asset results in the Bachelier pricing model, using GBM to model the underlying asset results in the famous Black-Scholes pricing model and using the exponential jump diffusion process to model the underlying asset results in the adapted Black-Scholes pricing model with jumps, see [41]. Where discrete models describe discrete situations (every value in an observed time series is assumed to be a realisation of a random variable), continuous models are continuous. The observed financial time series is thought of as a discrete realisation of the continuous process.

The models $M_i(\theta^i) \in \mathbb{M}$ are assumed to be continuous stochastic processes. The result of this is that the time series - or *paths* - generated by $M_i(\theta^i)$ are elements of the *path space*.

Definition 3.1 (Path space)**.** *The path space is the space of continuous functions, denoted as $C([0,T],\mathbb{R}^d)$, i.e.*

$$C([0,T],\mathbb{R}^d) := \{f : [0,T] \to \mathbb{R}^d | f \text{ is continuous}\}.$$

It is infinite dimensional and not locally[1] *compact*[2] *[14].*

Remark 3. *Notice that the exponential jump diffusion process is discontinuous at the times when jumps occur and is continuous everywhere else. Therefore it is in theory not an element of the path space.*

Deciding which stochastic process is most suitable based on an observed time series is hindered by the infinite dimensionality and non-locally compactness of the path space. The statistical measures used to distinguish between discrete models are generally defined over marginal distributions and can be extended to (finite) multivariate distributions for the purpose of distinguishing between discrete models when modelling time series. However, extending them to the infinite dimensional stochastic processes is not possible, according to [14]. Therefore, the wide variety of model selection measures from classical statistics cannot be used to distinguish between paths generated by different continuous stochastic processes, as these measures require finite distributions.

3.2. Related approaches

As mentioned in section 3.1, resorting to the classic statistical measures like the distribution metrics used to distinguish between discrete models do not work when choosing the most suitable continuous model. However, there exist other approaches in other research fields related to the problem discussed in this thesis that deserve our attention.

Time series classification

Time series classification is the field which attempts to classify multiple time series in two or more categories, which is also the purpose of this thesis. In the literature the topic of time series classification is an active research field. According to [2], until 2016 more than 100 algorithms had been proposed to solve this problem, this number is even larger today. Applications of time series classification range from medical (assessing whether an ECG displays anomalies [28]) to cultural (classifying types of music [38]) to financial, the latter being the category of interest in this thesis.

Generally, the methods used to approach the problem can be categorised in two categories:

- **Feature-based methods**: feature-based methods are methods that classify time series based on similar features. The idea behind this is that if time series have similar features, they must belong to the same class. This gives rise to the challenge of selecting features, which might differ per purpose. Examples of basic features of a time series are expected value, variance, but also seasonality, starting point of the time series etc. The importance of each feature depends on the situation and type of time series and are often determined by the user. Several approaches have been evaluated, of which [2] gives an overview.

[1]A space C is locally compact if for every $c \in C$ there exists an open set O and a compact set K, such that $c \in O \subset K$, see [11].

[2]A space C is compact if every open cover of C has a finite subcover. The Heine-Borel theorem provides a more commonly used equivalent:

$$C \text{ is compact} \Leftrightarrow C \text{ is bounded and closed,}$$

see [11].

- **Distance-based methods:** distance-based methods are methods that classify time series based on a measure of distance between the time series, for instance the Euclidean distance. Time series that are considered 'close' in the specific metric are classified to the same category. The main challenge of this approach is finding the correct measure, which differs per situation.

Often feature-based and distance-based methods are also referred to in machine learning as supervised and unsupervised learning, respectively.

Though a lot of research has been done on the problem of classifying time series, the research is often not applicable to the problem considered in this thesis. In many cases the research is based on paths being similar to each other, either in shape or having a small distance. This is different from the problem in this thesis, where even paths generated by the same model can have rather different shapes. An example of two paths generated by the same stochastic model is given in figure 3.1.

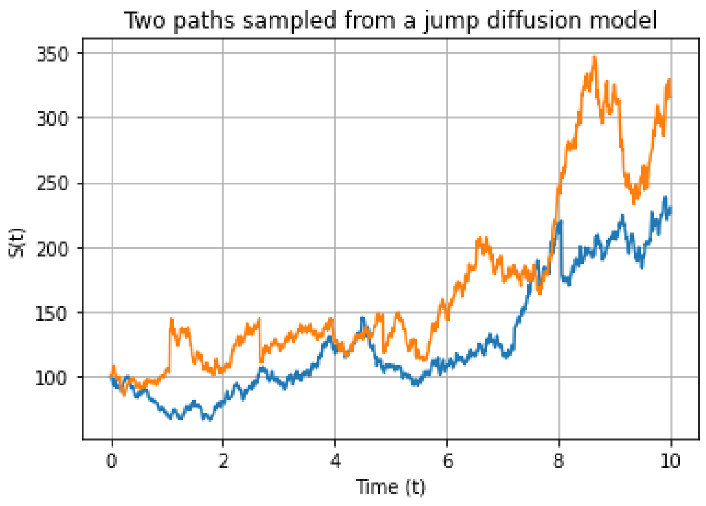


Figure 3.1: Two realisations of the jump diffusion model.

Figure 3.1 shows two realisations or paths that have a different trajectory, despite being generated by the same underlying process. Extracting features from these paths is futile, as is looking at distance measures to compare distance between the paths.

Synthetic market data generation
A problem closely linked to the classification of paths generated by stochastic processes is the field of synthetic market data generation. Synthetic market data generation is the topic of generating artificial market data (often time series) which is indistinguishable from real market data. One of the reasons this topic is relevant for this thesis is that, in order to generate synthetic data, the relevant information of financial time series must be captured. This relevant information is useful when attempting to classify paths generated by several stochastic processes.

Synthetic market data generation is a relatively new and active research field. Many financial institutions are interested in generating artificial market data in order to train their neural networks on a larger set of data. Additionally, the use of real market data for training algorithms can be limited by privacy issues. Since synthetic market data is not limited by privacy issue, this makes the use of artificial data more interesting.

A popular method to generate artificial data is the General Adverserial network (GAN), introduced by Goodfellow in [18]. A GAN consists of two neural networks, one generates the data (the Generator), the other attempts to distinguish real samples from samples from the samples generated by the Generator (the Adverserial). Adaptations of the GAN have been proposed to make the model better suited for market data generation, such as the restricted Bolzmann machine (RBM) [29] which deals with the small amount of available real market data to train the GAN on and the Quant GAN [45] which attempts to capture long-range dependencies such as the presence of volatility clusters. Also other methods have been proposed, such as adaptations of Variational Auto-encoders (VAE) in [10] or agent-based methods in [39].

All methods mentioned above attempt to extract the relevant information of the financial time series in order to generate new time series that are similar. Of these methods, we would like to highlight the one proposed in [10]. Most proposed methods train the neural network directly on the market data, but in [10] the market data is first transformed to its *signature*, after which the neural network is trained on this transformed input. The signature transformation is described as being an effective method to encode financial data streams parsimoniously and efficiently. For this reason, the next section dives deeper into this topic.

3.3. Signatures

As mentioned in section 3.1, inferring the underlying stochastic process directly from a path is disadvantageous as a consequence of the infinite dimensional and non-locally compact path space. A transformation from this space to a lower dimensional space eases this inference, as suggested by [10]. The results obtained from using the signature transformation in synthetic market data generation indicate that the signature transformation could be this desired transformation to a lower dimensional space. Signatures are useful transformations of a path to an infinite sequence of descriptive statistics. We are using continuous paths (at least for ABM and GBM), but in the approximation of the signatures we use a quadrature rule to approximate the integrals, and a truncation to have finite dimensions. This section introduces the concept of signatures in more detail, starting with definitions of interest in paragraph 3.3.1, followed by relevant properties in paragraph 3.3.2. Paragraph 3.3.3 gives the mathematical motivation supporting the choice to use the signature transformation. The section concludes with a description of transformations that can be applied to the path in order to capture specific characteristics of interest to the user in paragraph 3.3.4.

The concept of signatures stems from the rough path theory. Rough path theory is the research field which analyses rough paths, developed by Terry Lyons in the 1990s [34]. Rough paths are solutions to differential equations driven by highly irregular processes like for example a Wiener process. A signature transformation is used to map rough paths to a lower dimensional sequence of iterated integrals of the path which are related to its statistical characteristics. This also has many applications in machine learning, where the elements of the signature are often used as input for a regression or classification algorithm.

An efficient overview of signatures and its properties can be found in [13]. In [19] signature transformations are used to transform handwritten Chinese characters to signatures, which are then used as input for a Convolutional Neural Network (CNN). This approach gave a test error of 3.58%, compared with 5.61% for a traditional CNN. [30] uses signatures to detect bipolar disorders and revealed a previously unobserved indicator of treatment response. In [35] a novel method to price exotic options in a model-free way is proposed by looking at the implied expected signature of the market dynamics. Furthermore, the authors in [10] use signature transformations of market data as input for a VAE to generate market data, as mentioned in the previous paragraph. Using signatures allowed the authors to train the network on fewer data points. This is beneficial since market data is not widely available. In [22] the authors use signatures to classify financial data streams. In [13] a compact overview is given of other research in this field.

The papers mentioned above all use signatures as a transformation to a capture the relevant information of the path and have generated successful results. This indicates that using signatures is a promising solution to the problem observed in this thesis.

3.3.1. Definitions

Some definitions of interest described by [10] are given below.

Definition 3.2 (Signature of a path). *Let $X : [0,T] \to \mathbb{R}^d$ be a continuous path where $d \in \mathbb{N}$ denotes the dimension of the path. Then the signature of X is defined by the sequence of iterated integrals given* *by*

$$S(X)^{\leq\infty}_T := (1, S(X)^1_T, \dots, S(X)^n_T, \dots),$$

where

$$S(X)^n_T := \int_{0<t_1<\cdots<t_k<T} dX_{t_1} \otimes \cdots \otimes dX_{t_k} \in (\mathbb{R}^d)^{\otimes n}.$$

where $\otimes$ *denotes the tensor product and* $t_i \in [0,T]$ *denotes time. Similarly, given* $N \in \mathbb{N}$*, the truncated signature of order* N *is defined by*

$$S(X)^{\leq N}_T := (1, S(X)^1_T, \dots, S(X)^N_T)$$

Remark 4. *If the path is a Brownian motion, the integral is defined in the Itô sense as defined in chapter 2 [32].*

Remark 5. *If the path is of bounded variation*[3] *the above integral can be replaced by the Riemann-Stieltjes integral [10].*

Remark 6. *Notice that the observed time series are discrete paths, instead of continuous paths. The signature can only be computed over continuous paths, so the choice is made to connect the data points piece-linearly. More details about this is given in chapter 4.*

Example 3.1 and 3.2 are given for clarification of these definitions.

Example 3.1. *Let* $X = \{(X^1(t))_{t>0}, (X^2(t))_{t>0}\}$*. The signature of path* X *is given by*

$$S(X) = (1, S^{(1)}_{[0,T]}, S^{(2)}_{[0,T]}, S^{(1,1)}_{[0,T]}, S^{(1,2)}_{[0,T]}, S^{(2,1)}_{[0,T]}, S^{(2,2)}_{[0,T]}, S^{(1,1,1)}_{[0,T]}, \dots)$$

with

$$\begin{aligned}
S(X)^{(1)}_{[0,T]} &:= \int_{0<t<T} dX^1(t) \\
S(X)^{(2)}_{[0,T]} &:= \int_{0<t<T} dX^2(t) \\
S(X)^{(1,1)}_{[0,T]} &:= \int_{0<t_2<T} \int_{0<t_1<t_2} dX^1(t_1) dX^1(t_2) \\
S(X)^{(1,2)}_{[0,T]} &:= \int_{0<t_2<T} \int_{0<t_1<t_2} dX^1(t_1) dX^2(t_2) \\
S(X)^{(2,1)}_{[0,T]} &:= \int_{0<t_2<T} \int_{0<t_1<t_2} dX^2(t_1) dX^1(t_2) \\
S(X)^{(2,2)}_{[0,T]} &:= \int_{0<t_2<T} \int_{0<t_1<t_2} dX^2(t_1) dX^2(t_2) \\
S(X)^{(1,1,1)}_{[0,T]} &:= \int_{0<t_3<T} \int_{0<t_2<t_3} \int_{0<t_1<t_2} dX^1(t_1) dX^1(t_2) d^1X^1(t_3) \\
&\vdots
\end{aligned}$$

Example 3.2. *Let* $X = \{(X^1(t))_{t>0}, (X^2(t))_{t>0}, (X^3(t))_{t>0}\}$*. The signature of path* X *is given by*

$$S(X) = (1, S^{(1)}_{[0,T]}, S^{(2)}_{[0,T]}, S^{(3)}_{[0,T]}, S^{(1,1)}_{[0,T}, S^{(1,2)}_{[0,T]}, S^{(1,3)}_{[0,T]}, S^{(2,1)}_{[0,T]}, S^{(2,2)}_{[0,T]}, S^{(2,3)}_{[0,T]}, \dots).$$

Intuition: Fourier transform

In a broad sense, one could compare the signature transformation to the Fourier transformation. Fourier transforms are used to solve partial differential equations (PDEs). The Fourier transform is applied to

[3]Recall that a path $X : [0,T] \to \mathbb{R}^d$ has bounded variation when

$$\sup_{P \in \mathcal{P}} \sum_{i=1}^{n_P} |X(t_{i+1}) - X(t_i)| < +\infty$$

where $\mathcal{P} = \{x_1, \dots, x_{n_P} | P$ is a partition of $[0,T]$ satisfying $x_i \leq x_{i+1}$ for $1 \leq i \leq n_P\}$

the PDE, the transformed PDE is then solved in the Fourier domain, after which the solution is transformed back by using the inverse Fourier transform. Using the signature transformation follows this same principle. Distinguishing between stochastic processes is difficult, so the signature transformation is applied to change the problem to distinguishing between signatures.

Next to the signature, there exists also the log-signature. Similar to the signature, the log-signature is also an infinite sequence of elements related to statistical properties of the path. Every element of the log-signature is a linear combination of elements of the signature. Definition 3.3 gives a mathematical definition of the log-signature.

Definition 3.3 (Log-signature of a path)**.** *Let $X : [0,T] \to \mathbb{R}^d$ be a path such that its signature $S(X)^{<\infty}_{[0,T]}$ is well-defined. The log-signature is then defined by*

$$\begin{aligned}\log S(X)_{[0,T]} &:= -S(X)_{[0,T]} + \frac{1}{2}(S(X)_{[0,T]})^{\otimes 2} - \frac{1}{3}(S(X)_{[0,T]})^{\otimes 3} + \dots \\ &+ (-1)^n \frac{1}{n}(S(X)_{[0,T]})^{\otimes n} + \dots,\end{aligned}$$

which can be shown to be well-defined [10]. In other words, the log-signature is the formal power series of $S(X)$.

Remark 7. *$S(X)$ is a sequence of iterated integrals, making $\log S(X)$ a sequence as well.*

In order to give some examples about the computation of the log-signature, the *Lie bracket* is introduced.

Definition 3.4 (Lie bracket)**.** *The Lie bracket is an operator such that for X, Y the following holds:*

$$[X,Y] = XY - YX. \tag{3.1}$$

To give a bit more intuition for the log-signature, observe another example below.

Example 3.3. *Suppose $X = \{(X^1(t))_{t>0}, (X^2(t))_{t>0}\}$. For simplicity, the notation $S(X)_{[0,T]}$ is abbreviated by S. The signature is given in example 3.1. Its log-signature is given by*

$$\log S = (S^{(1)}, S^{(2)}, \frac{1}{2!}S^{[1,2]}, -\frac{1}{3!}S^{[1,[1,2]]}, -\frac{1}{3!}S^{[2,[1,2]]}, \frac{1}{4!}S^{[1,[1,[1,2]]]}, \frac{1}{4!}S^{[1,[2,[1,2]]]}, \frac{1}{4!}S^{[2,[1,[1,2]]]}, \frac{1}{4!}S^{[2,[2,[1,2]]]}, \dots)$$

Using definition 3.4 the Lie brackets can be expanded to allow the elements of the log-signature to be written in terms of the elements of the signature observed in example 3.1.

$$\begin{aligned}S^{[1,2]} &= S^{(1,2)} - S^{(2,1)} \\ S^{[1,[1,2]]} &= S^{(1,[1,2])} - S^{([1,2],1)} \\ &= S^{(1,(1,2)-(2,1))} - S^{((1,2)-(2,1),1)} \\ &= S^{(1,1,2)} - S^{(1,2,1)} - (S^{(1,2,1)} - S^{(2,1,1)}) \\ &= S^{(1,1,2)} - 2S^{(1,2,1)} + S^{(2,1,1)} \\ S^{[2,[1,2]]} &= -S^{(2,2,1)} + 2S^{(2,1,2)} - S^{(1,2,2)} \\ S^{[1,[1,[1,2]]]} &= S^{(1,1,1,2)} - 3S^{(1,1,2,1)} + 3S^{(1,2,1,1)} - S^{(2,1,1,1)} \\ &\vdots\end{aligned}$$

Example 3.4. *This example shows how the signature and log-signature can be expressed in terms of a path X. Let $X = \{(X^1(t))_{t>0}, (X^2(t))_{t>0}\}$. Simplifying notation and truncating at level $L = 2$, the (log-)signature can be expressed as follows:*

$$\begin{aligned}S(X) &= (1, S^{(1)}, S^{(2)}, S^{(1,1)}, S^{(1,2)}, S^{(2,1)}, S^{(2,2)}) \\ &= (1, \Delta X^1, \Delta X^2, \frac{1}{2}(\Delta X^1)^2, S^{(1,2)}, S^{(2,1)}, \frac{1}{2}(\Delta X^2)^2) \\ \log S(X) &= (S^{(1)}, S^{(2)}, \frac{1}{2}S^{[1,2]}) \\ &= (\Delta X^1, \Delta X^2, \frac{1}{2}(S^{(1,2)} - S^{(2,1)}))\end{aligned}$$

Here $\Delta X^i = X^i(T) - X^i(0)$, i.e. the last term minus the first term. $S^{(1,2)}$ and $S^{(2,1)}$ cannot be expressed easily directly in terms of the path, but do have a geometrical interpretation, see figure 3.2.

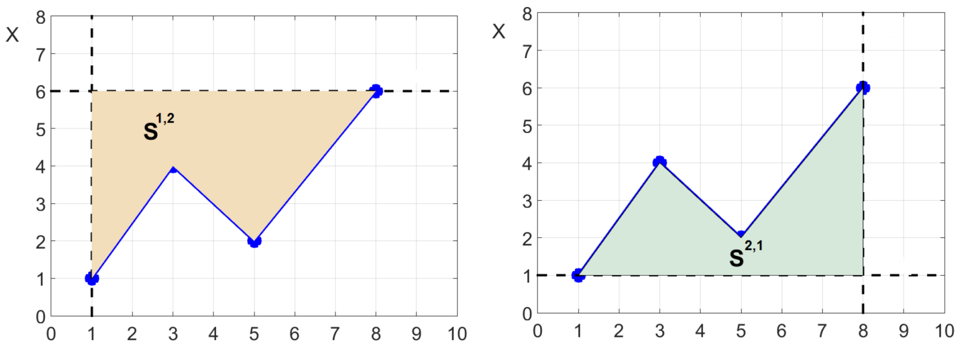


Figure 3.2: A geometrical representation of $S^{(1,2)}$ and $S^{(2,1)}$, taken from [13].

As can be seen from figure 3.2, $S^{(1,2)}$ and $S^{(2,1)}$ denote the area above an below a path, respectively.

Signature and log-signature in practice
In practice, both signatures and log-signatures are used in the available literature, though more literature is available on signatures than on log-signatures. Both transformations have their own advantages and disadvantages. A clear disadvantage for signatures is that signature are elements of the tensor space, which is not a linear space. This means that small perturbations in signatures might have large consequences in the corresponding path in the path space. The log-signature spans a linear space, according to [10]. This means that perturbations have less impact, which is the reason that log-signatures are very popular in synthetic market data generation, where this is an important property. Additionally, as can be seen in example 3.4, the log-signature sequence has a lower dimension than the signature sequence for the same truncation level. Lower-dimensional input is beneficial when using machine learning, which suffers from the curse of dimensionality.

An advantage of signatures over log-signatures is that signatures span an algebra (see section 3.3.3 for more details), which in short ensures that classifying based on signatures is possible, whereas it is more difficult to do this based on paths. The log-signatures do not span an algebra, which means that it does not have this advantage.

In summary, for the purpose of using (log-)signatures to classify between paths, the literature does not provide us with a clear preference for one or the other. Since more literature is available on signatures and they guarantee that classification is possible, the signatures are selected as transformation in this thesis. Section 5.2.3 gives some empirical insights in the difference between using signatures and log-signatures in several classification settings.

3.3.2. Properties

The signature has many properties, of which [13] gives an extensive overview. This section describes the properties of interest to this thesis.

Theorem 3.1 (Signatures)**.** *Some relevant properties of signatures are listed below.*

1. *Signatures are independent of starting point, i.e. paths $X(t)$ and $\tilde{X}(t) = X(t) + a$ (with $a \in \mathbb{R}^d$ and d the dimension of the path) have equal signatures.*

2. *Signatures are invariant under time reparametrisations, i.e. for $X(t)$ and $X(\phi(t))$, where $\phi : [0,T] \to [0,T]$ is a surjective, continuous, non-decreasing reparametrisation, it holds that $S(X(t)) = S(X(\phi(t)))$.*

3. *For path $X(t)$ with $X : [0,T] \to \mathbb{R}^d$ and its time-reversal path $\overleftarrow{X}(t) = X(T-t)$ we have $S(X) \otimes S(\overleftarrow{X}) = \mathbf{1}$. These paths are called 'tree-like paths'. The signatures can therefore not be used to distinguish between a path concatenated with its time reversal path or path $X \equiv 1$.*
4. *For all paths excluding the paths described in 1, 2, and 3 the corresponding signature is one-to-one, i.e. both the transformation from path to signature and its reverse are unique.*

Before addressing the proof of property 3.1, an example of a tree-like path is given in figure 3.3.

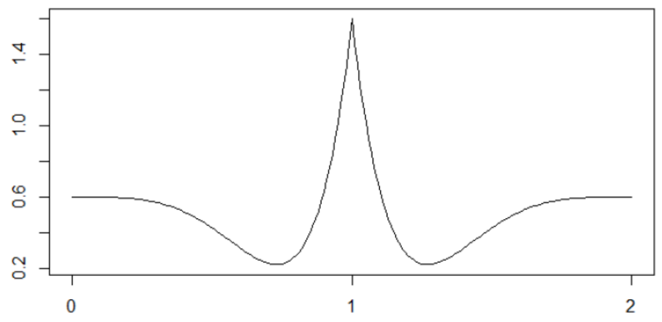


Figure 3.3: An example of a tree-like path by concatenation of a path with its time reversal path, taken from [16].

Figure 3.3 shows an example of a path $X(t),\ t \in [0,1]$ concatenated with its time reversal. Paths of this type do not have unique signatures.

Proof. The proof is based on the work of [13], [16] and [23].

1. Let $X : [0,T] \mapsto \mathbb{R}^d$ be a path and let $\tilde{X}(t) = X(t) + a$ with $a \in \mathbb{R}^d$ a constant vector with equal elements. Then

$$\tilde{X}(t) = X(t) + a \Leftrightarrow \mathrm{d}\tilde{X}(t) = \mathrm{d}X(t) + 0$$

This leads to

$$\mathrm{d}\tilde{X}(t) = \mathrm{d}X(t) \tag{3.2}$$

Since the signature consists of elements which are defined as iterative integrals and equation (3.2) holds, one can conclude that

$$S(X(t)) = S(X(t) + a)$$

2. Define paths $X, Y : [0,T] \to \mathbb{R}$ and let $\tilde{X}, \tilde{Y} : [0,T] \to \mathbb{R}$ such that $\tilde{X}(t) = X(\phi(t))$ and $\tilde{Y}(t) = Y(\phi(t))$. Assume for simplicity that ϕ is smooth and the derivatives of X exist. Applying the chain rule gives

$$\begin{aligned}\frac{\mathrm{d}\tilde{X}(t)}{\mathrm{d}t} &= \frac{\mathrm{d}\tilde{X}(t)}{\mathrm{d}\phi}\frac{\mathrm{d}\phi}{\mathrm{d}t} \\ &= \frac{\mathrm{d}X(\phi(t))}{\mathrm{d}\phi}\frac{\mathrm{d}\phi}{\mathrm{d}t}\end{aligned}$$

This results in

$$\int_0^T \tilde{Y}(t)\mathrm{d}\tilde{X}(t) = \int_0^T Y(\phi(t))\Big(\frac{\mathrm{d}X(\phi(t))}{\mathrm{d}\phi}\frac{\mathrm{d}\phi}{\mathrm{d}t}\Big)\mathrm{d}t = \int_0^T Y(a)\mathrm{d}X(a) \tag{3.3}$$

with $a := \phi(t)$. This means that an integral over reparametrisations of paths are equal to the integral over the original paths. Since the signature consists of a sequence of iterative integrated integrals and the integrals over the original paths and their time reparametrisations are equal according to (3.3), one can conclude that

$$S(X(t)) = S(X(\phi(t))) \tag{3.4}$$

3. The full proof of this property is quite involved, so only an intuition is given here. The full proof can be found in theorem 2.3.1 in [16]. Let $Z = X * \overleftarrow{X}$ where $*$ denotes the concatenation operator. As mentioned in the start of this paragraph, rough paths are the solution to differential equations driven by irregular processes like the Wiener process. So for an arbitrary map $f : V \to W$ and path $Y(t) : [0,T] \to V$ where V, W are Banach spaces, observe

$$dY(t) = f(Y(t))dX(t), \;\; Y(0) = a, \; Y(T) = b, \tag{3.5}$$

with $a, b \in \mathbb{R}$. Notice that the solution $Y(t)$ of equation (3.5) does not change when the equation changes to

$$\mathrm{d}Y(t) = f(Y(t))\mathrm{d}\overleftarrow{X}(t), \;\; Y(0) = b, \; Y(T) = a.$$

Notice that any solution $Y(t)$ of

$$\mathrm{d}Y(t) = f(Y(t))\mathrm{d}Z(t) \;\; Y(0) = a \tag{3.6}$$

will satisfy $Y(2T) = a$ for any function f. Let $\pi_N : V \to W$ be the canonical projection, i.e. a projection that projects infinite sequences to N-dimensional sequences:

$$\pi_N : (a_0, a_1, \dots, a_N, a_{N+1}, \dots) \to (a_0, a_1, \dots, a_N).$$

It is proven in lemma 2.3.2 of [16] that choosing $f = \pi_N$ gives that the solution of equation (3.6) is equal to $S(Z)^N_{[0,T]}$ (the signature of path Z, truncated at level N) and that this solution is $\underline{\text{unique}}$. However, notice that $Y(t) = \mathbf{1}$ is also a solution to equation (3.6) if $f = \pi_N$. For this reason we have for any $N > 0$

$$S(Z)^N_{[0,T]} = \mathbf{1}$$

and therefore

$$S(Z)_{[0,T]} = S(X) \otimes S(\overleftarrow{X}) = \mathbf{1}.$$

4. The proof of this statement is very involved and is the main result of [23]. In short, the authors prove the opposite of the theorem, i.e. paths X, Y are tree-like if and only if $S(X) = S(Y)$.

$\square$

3.3.3. Mathematical motivation for using signatures

Now that the (log-)signature is defined, we look at the motivation behind the signature transformation. The purpose of this thesis is to identify the stochastic process that is most likely to have generated an observed time series. The question that needs to be answered is whether or not signatures are a transformation that is descriptive enough that we can base our inferences on signatures, instead of directly on the paths. This motivation builds upon existing knowledge from [16].

In the field of financial engineering, signatures are used for the purpose of market data generation because of their ability to capture the statistical properties of the time series. Therefore elements of the signature relate to the mean, variance, skewness, kurtosis and higher order moments. Other approaches in market data generation make use of statistics, market generators (and mathematical models) which are designed to replicate often shown features in financial time series - referred to as *stylised facts* - such as heavy tails, volatility clustering and the leverage effect (negative return between volatility and asset returns). According to [10], training market generators directly on real financial time series has been done, but has proven difficult due to the fact that those time series (or paths) belong to the path space, see definition 3.1.

The fact that the path space is infinite dimensional and non-locally compact causes trouble when attempting to make inferences directly from paths. The inference we would like to make in this thesis is *classification*. Let $X \in \mathbb{R}^d$ denote a d-dimensional path with class label $y \in \{0, 1\}$ (in case of binary classification). The function $L : X \to y$ is the classification function, also called the classification boundary. Since $X \in C([0,T], \mathbb{R}^d)$ where C is infinite dimensional and non-locally compact, attempting to approximate $L(X)$ through machine learning techniques can be challenging.

$$\begin{tikzcd}
X \arrow[r, "L"] \arrow[d] & y \\
S(X) \arrow[ru, "\tilde{L}"'] &
\end{tikzcd}$$

Figure 3.4: Schematic overview of classification problem [46].

Figure 3.4 shows a schematic overview of the classification problem. Instead of attempting to approximate $L : X \to y$, the focus is put on approximating $\tilde{L} : S(X) \to y$. The question remains whether signatures are descriptive enough such that a combination of elements of the signature sequence will lead to a reasonable approximation of $\tilde{L}$. One can compare this to the approximation of smooth functions, where it is known through Taylor's theorem that every smooth function can (at least locally) be uniformly close approximated by a linear combination of monomials. In this situation, one can wonder if signatures can act as the "monomials of the path space".

Fortunately, this is possible. The reason for this is that signatures span an algebra. Recall that the definition of an algebra is the following:

Definition 3.5 (Algebra, [11]). *An algebra is a vector space A on which there is defined a multiplication $(f, g) \mapsto fg$ (from $A \times A$ into A) satisfying*

1. $(fg)h = f(gh)$*, for all* $f, g, h \in A$
2. $f(g + h) = fg + fh$, $(f + g)h = fh + gh\ \forall f, g, h \in A$
3. $\alpha(fg) = (\alpha f)g = f(\alpha g)$*, for all* $f, g \in A$ *and for all scalars* α*.*

For signatures, the multiplication is defined as the shuffle product. This product ensures that the product of two terms $S(X)^{(i_1,\dots,i_k)}S(X)^{(j_1,\dots,j_n)}$ can always be expressed as a sum of signatures.

Definition 3.6 (Set of all shuffles, [13]). *Let $I = (i_1, \dots, i_k)$ and $J = (j_1, \dots, j_n)$ be two sets of indices and write $(r_1, \dots, r_{n+k}) = (i_1, \dots, i_k, j_1, \dots, j_n)$. Then the set of all shuffles of indices I and J is denoted by $I \sqcup J$, with*

$$I \sqcup J = \{(r_{\sigma(1)}, \dots, r_{\sigma(n+k)}) | \sigma \in Sh(n, k)\}.$$

Here $\sigma(1) < \dots \sigma(n)$ and $\sigma(n+1) < \dots \sigma(n+k)$ denote permutations on the set $\{1, \dots, n+k\}$ and $Sh(n, k)$ is the set of all permutations.

Definition 3.7 (Shuffle product). *Let $I = (i_1, \dots, i_k)$ and $J = (j_1, \dots, j_n)$ be two sets of indices. The shuffle product between two signatures is then defined as*

$$S(X)^I S(X)^J = \sum_{K \in I \sqcup J} S(X)^K.$$

[13]

An example to clarify the shuffle product is given below.

Example 3.5. *Consider a two-dimensional path $X : [a, b] \to \mathbb{R}^2$. Then the shuffle product implies that*

$$\begin{aligned}
S(X)^{(1)}S^{(2)} &= S(X)^{(1,2)} + S(X)^{(2,1)}, \\
S(X)^{(1,2)}S^{(1)} &= 2S(X)^{(1,1,2)} + S(X)^{(1,2,1)}, \\
S(X)^{(1,2)}S^{(2)} &= 2S(X)^{(1,2,2)} + S(X)^{(2,1,2)}.
\end{aligned}$$

Lemma 3.1. *The space of signatures endowed with the shuffle product is an algebra.*

Proof. To show that this is true, we need to show that the three requirements in definition 3.5 are satisfied. Let $S(X)^I, S(X)^J, S(X)^K$ be tree signatures with I, J, K defined similar to definition 3.7.

1. We see that:

$$(S(X)^I S(X)^J)S(X)^K = (\sum_{L\in I\sqcup J} S(X)^L)S(X)^K = \sum_{L\in I\sqcup J\sqcup K} S(X)^L = S(X)^I(\sum_{L\in J\sqcup K} S(X)^L) = S(X)^I(S(X)^J S(X)^K).$$

2. We have:

$$S(X)^I S(X)^J + S(X)^I S(X)^K = \sum_{L\in I\sqcup J} S(X)^L + \sum_{M\in I\sqcup K} S(X)^M = S(X)^I(S(X)^J + S(X)^K)$$

3. We have:

$$\alpha(S(X)^I S(X)^J) = \alpha \sum_{L\in I\sqcup J} S(X)^L = \sum_{L\in I\sqcup J} \alpha S(X)^L = (\alpha S(X)^I)S(X)^J$$

The shuffle product respects the demands in definition 3.5, so the signatures span an algebra. □

To prove that signatures are descriptive enough such that a linear combination of them can uniformly approximate $\tilde{L}$, we use the well-known Stone-Weierstrass theorem.

Lemma 3.2 (Stone-Weierstrass, [11])**.** *Let X be a compact space, A a sub-algebra of $C(X)$. If A separates points* [4] *in X and vanished at no point* [5] *of X, then A is dense in $C(X)$.*

We can now state and proof the main result of this section:

Theorem 3.2. *Let D be a compact subset of path space $C([0,T],\mathbb{R}^d)$ containing paths that have the same starting point, equal time parametrisation and are not tree-like. Let $\tilde{L} : D \to \mathbb{R}$. Then for every $\epsilon > 0$, there exists $\omega \in \mathbb{R}^k$ such that for all $X \in D$:*

$$|\tilde{L}(X) - \langle \omega, S(X)\rangle| \leq \epsilon.$$

Here k is the number of signature elements in the sequence and $\langle\cdot\rangle$ denotes the inner product.

Proof. To prove this theorem, we need to apply the Stone-Weierstrass theorem. Consider

$$A = \text{span}\{f : X \mapsto \langle \omega, S(X)\rangle\}.$$

It indicates that A is a linear subspace of $C(D,\mathbb{R})$. In order to use the Stone-Weierstrass theorem, we need to show that A separate points and vanish at no point in D. The first property holds since the fourth property of theorem 3.1 ensures that each path has an unique signature. The second property holds since the first element of every signature sequence is equal to one, see definition 3.2. Therefore $S(X)$ is never equal to the vector containing only zeros, i.e. A vanishes nowhere in D. Applying the Stone-Weierstrass theorem tells us that A is dense in $C(D,\mathbb{R})$. Define

$$\overline{A} := A \cup \{\lim_{n\to\infty} a_n | a_n \in A \text{ for all } n \in \mathbb{N}\}.$$

Since A is dense in $C(D,\mathbb{R})$, by definition $\overline{A} = C(D,\mathbb{R})$ and from which directly follows that

$$|\tilde{L}(X) - \langle \omega, S(X)\rangle| \leq \epsilon.$$

□

Theorem 3.2 implies that a linear combination of signatures exists that can be uniformly close to the function $\tilde{L}$ that is being approximated. Therefore $\tilde{L}$ can be found through a combination of the elements of $S(X)$, whereas L cannot be found through a combination of X. This justifies the choice of the signature transformation in this thesis.

[4] A separates points in X if given $x \neq y$ there is some function $f \in A$ such that $f(x) \neq f(y)$.

[5] A vanishes at no point of X if given $x \in X$ there is some $f \in A$ such that $f(x) \neq 0$.

3.3.4. Transformations

Until now we have only discussed applying the signature transformation directly on the observed path X. However, there are a number of transformations that can be applied to the path before the signature transformation is taken. This is mostly done for numerical benefits, as these transformations can be used to explicitly capture certain characteristics of the path. The optimal type of transformation differs per situation, since it depends on the interests of the user. In this section the popular *lead-lag transformation* is introduced and some background is given on how the setting changes when no transformation is used, as opposed to when the lead-lag transformation is used.

No transformation
When no transformation is used, the path is two-dimensional:

$$X = \{t, X(t)\},$$

with

$$\begin{aligned} t &= (t_1, t_2, \dots, t_N), \\ X(t) &= (X_1, X_2, \dots, X_N), \end{aligned}$$

where X_i is short notation for $X(t_i)$.The signature and log signature truncated at level 2 are given below:

$$\begin{aligned} S(X) &= (S^{(1)}, S^{(2)}, S^{(1,1)}, S^{(1,2)}, S^{(2,1)}, S^{(2,2)}) \\ &= (t_N - t_1, X_N - X_1, \frac{1}{2}(t_N - t_1)^2, S^{(1,2)}, S^{(2,2)}, \frac{1}{2}(X_N - X_1)^2), \qquad (3.7) \\ \log S(X) &= (S^{(1)}, S^{(2)}, S^{[1,2]}) \\ &= (t_N - t_1, X_N - X_1, \frac{1}{2}(S^{(1,2)} - S^{(2,1)})). \end{aligned}$$

Lead-lag transform
The lead-lag transform is a popular transformation, since it explicitly captures the volatility of a path. The volatility is often a parameter of interest in financial time series, making this transformation quite interesting. The lead-lag transformation of a path X introduces a lead and lag component of a path, increasing the dimension of the path by one.

Definition 3.8 (Lead-lag transform, [22]). *Let $X = \{X_1, X_2, \dots, X_N\}$ be a one-dimensional path of length N. Its lead-lag transformation is given by*

$$X^{lead-lag} = \{X^{lead}, X^{lag}\}$$

where

$$X_j^{lead} = \begin{cases} X_i & \text{if } j = 2i \\ X_i & \text{if } j = 2i - 1 \end{cases}$$
$$X_j^{lag} = \begin{cases} X_i & \text{if } j = 2i \\ X_i & \text{if } j = 2i + 1 \end{cases}$$

.

The path is transformed to a lead and a lag component, where the lag component follows the lead component. To make this definition more intuitive, an example is shown in figure 3.5.

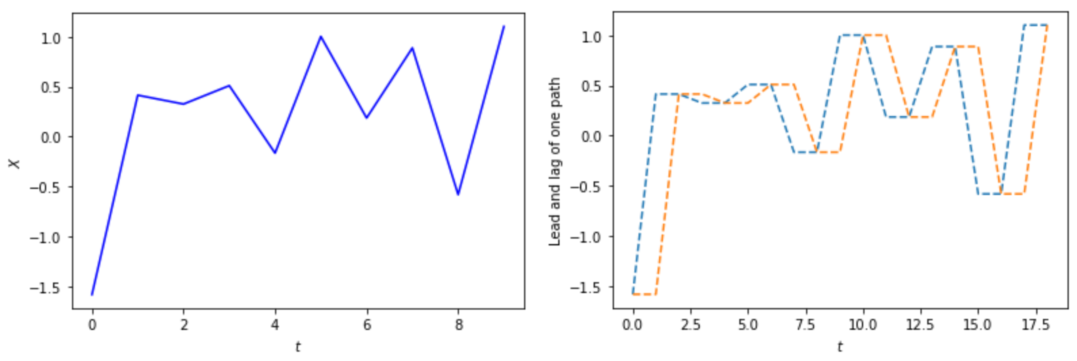


Figure 3.5: Left: one path. Right: lead-lag transform where with lead (blue) and lag (orange) of path on the left.

The path $X = \{t, X(t)\}$ is transformed to $X^{\text{lead-lag}} = \{t, X^{\text{lead}}(t), X^{\text{lag}}(t)\}$. Since the parameter t does not add information relevant to the distinction between paths generated by different stochastic processes, it can be dropped and the path $X^{\text{lead-lag}}$ can be plotted as in figure 3.6:

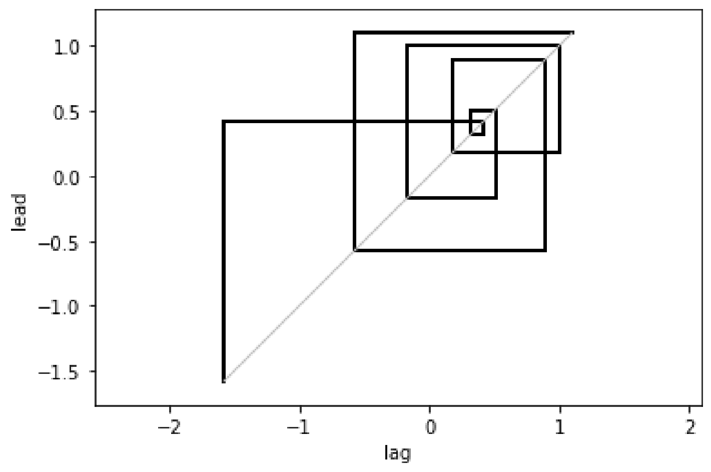


Figure 3.6: A plot of the lead-lag transformation of the path shown in figure 3.5.

Figure 3.6 shows the lead-lag transformation of the path shown in figure 3.5. Indeed, notice that the starting point (-1.5, -1.5) and final point (1.1, 1.1) of the lead-lag in figure 3.6 path indeed correspond to the starting point (0, -1.5) and final point (9, 1.1) of the original path in figure 3.5.

The (log-)signature (truncated at level 2) are then defined as:

$$\begin{aligned} X &= \{X^{lead}, X^{lag}\}, && (3.8)\\ S(X) &= (1, S^{(1)}, S^{(2)}, S^{(1,1)}, S^{(1,2)}, S^{(2,1)}, S^{(2,2)}) \\ &= (1, X_N^{lead} - X_1^{lead}, X_N^{lag} - X_1^{lag}, \frac{1}{2}(X_N^{lead} - X_1^{lead})^2, S^{(1,2)}, S^{(2,2)}, \frac{1}{2}(X_N^{lag} - X_1^{lag})^2), && (3.9)\\ \log S(X) &= (S^{(1)}, S^{(2)}, S^{[1,2]}) \\ &= (X_N^{lead} - X_1^{lead}, X_N^{lag} - X_1^{lag}, \frac{1}{2}(S^{(1,2)} - S^{(2,1)})). \end{aligned}$$

Notice from definition 3.8 and figure 3.5 that

$$X_1 = X_1^{\text{lead}} = X_1^{\text{lag}},$$
$$X_N = X_N^{\text{lead}} = X_N^{\text{lag}}.$$

Combining this with equations (3.7) and (3.9) we see that

$$S^{(2)}(X) = S^{(2)}(X^{\text{lead-lag}}) = X_N - X_1.$$

Recall from equation (2.7) that this is exactly equal to the MLE of the drift coefficient of ABM, up to a multiplicative constant. Also for GBM the quantity $X_N - X_1$ appears, through taking the exponential. This means that the signature captures the drift for ABM and GBM similarly to the manner in which the MLE is computed in classical statistics.

We explain how the volatility is captured by the lead-lag transform through example 3.6.

Example 3.6. *This example is based on an example in [13]. Observe a path* $X = \{(X^1(t))_{t>0}, (X^2(t))_{t>0}\} = \{t, (X(t))_{t>0}\}$ *with*

$$\begin{aligned} t &= [0,1,2,3] \\ X(t) &= [X(0), X(1), X(2), X(3)] \\ &= [1,4,2,6] \end{aligned}$$

The path and its lead-lag transform are shown in figure 3.7.

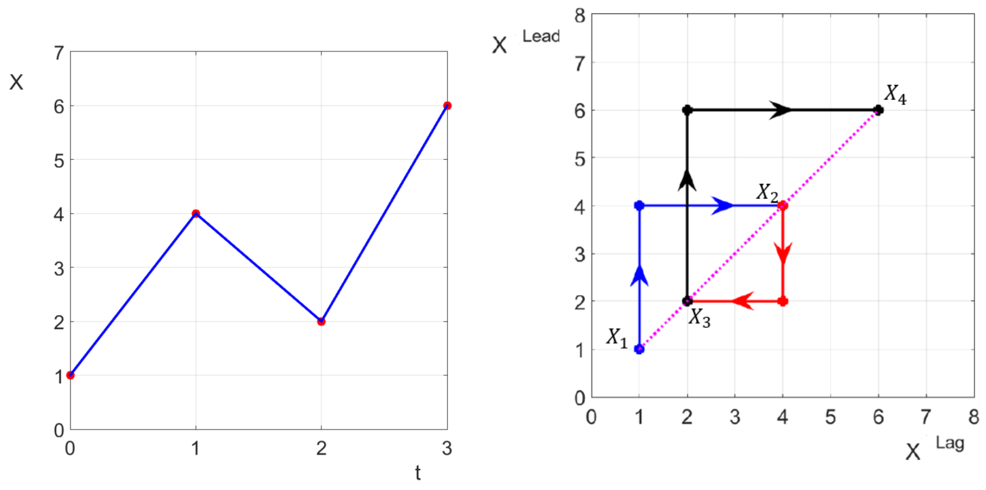


Figure 3.7: A path (left) and its lead-lag transform (right), taken from [13].

Let A denote the total area enclosed by the blue, red and black triangle in figure 3.7, i.e.

$$\begin{aligned} A &= \frac{1}{2}[(X(1)-X(0))^2 + (X(2)-X(1))^2 + (X(3)-X(2))^2] \\ &= \sum_{t=0}^{2}(X(t+1)-X(t))^2 \end{aligned}$$

Recall from definition 2.2 that the quadratic variation (QV) of a path X is given by $QV(X) := \sum_{t=0}^{N}(X(t+1)-X(t))^2$, so in general we have for a path $X(t)$:

$$A_{\text{lead-lag}} = \frac{1}{2}QV(X)$$

Recall from example 3.4 that $S^{(1,2)}$ and $S^{(2,1)}$ denote the area above and below the path, respectively. This means that for $X^{\text{lead-lag}}$ we have that $S^{(1,2)} - S^{(2,1)}$ denotes the area enclosed by the triangles in figure 3.7, i.e.

$$S^{(1,2)}(X^{\text{lead-lag}}) - S^{(2,1)}(X^{\text{lead-lag}}) = \frac{1}{2}QV(X)$$

For this reason the signature is able to capture the quadratic variation. Recall in chapter 2 how the quadratic variation is related to the volatility coefficient of the stochastic processes. Then, by capturing the quadratic variation, the signature is also able to capture the information about the volatility. This is beneficial because of the importance the parameter plays in practice. Therefore, in this thesis the lead-lag transform is used as a standard transformation in the further analysis of the problem.

3.4. Proposed solution

Recall the problem as stated in chapter 1. The proposed solution in this thesis is to randomly generate paths $\omega_{M_i(\theta^i)}$ from the set of parametric models $\mathbb{M}$ and label each generated path by its model name. Using this very large (generated) data set, a classification model can be trained to distinguish paths generated by different stochastic processes. After this, the trained classifier can be used to classify ω_D into one of the classes $M_j(\theta^j)$ or, alternatively, produce the likelihood of each model considered $\{M_1(\theta^1); M_2(\theta^2); \ldots; M_k(\theta^k)\}$.

As seen in section 3.1, a naive approach is to train the classifier directly on the generated paths. Unfortunately, this does not work. A reason for this is the infinite dimensionality of the path space. Learning a classification function on an infinite dimensional input space is challenging and would - in theory - require an infinite number of input samples. Additionally, as mentioned in section 3.2, both feature-based and distanced-based classifiers do not perform satisfactory on this type of input. As explained in section 3.3, signatures could provide a solution. Based on the results observed in available literature, transforming paths to their signatures and training the classifier to distinguish signatures instead of paths seems to be promising. This avoids the issue of the difficult path space by transforming the problem to the less difficult signature space. A schematic overview of the proposed solution can be seen in figure 3.8.

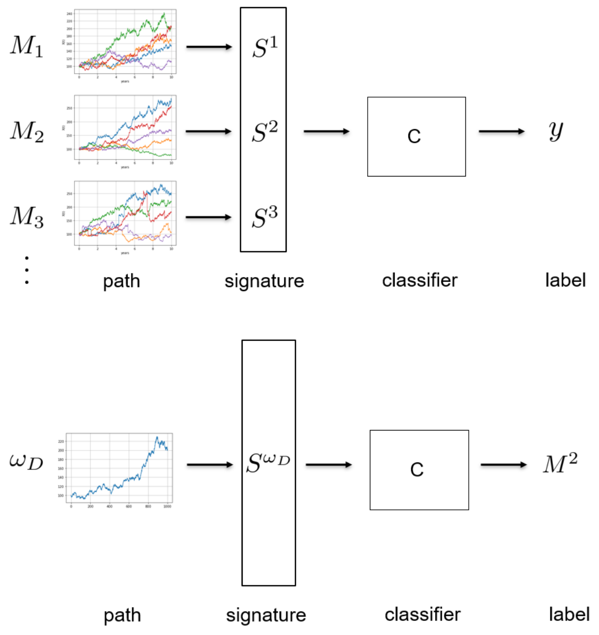


Figure 3.8: Schematic overview of the proposed solution. Up: training of the classifier, down: inference of the generative model of the observed time series.

The advantage of this proposed solution is that it is *model agnostic*, i.e. it can be applied to any type of model which can generate paths. Usually every model considered in the set of models has their own assumptions about the distribution of the samples and the temporal dependence between the elements of the time series. When using an empirical distribution metrics - a popular method to detect from which distribution a sample has been drawn in finite dimensional settings - these assumptions often complicate this discovery. These complications are circumvented by using this model agnostic proposed solution, since they no longer play a role there. This means that there is a wide range of situations where this can be used, making it a flexible and general solution.

3.5. Maximum classification accuracy

An important consideration before attempting to classify observed paths to their generative stochastic process is the question whether there is a theoretical upper bound to the classification accuracy when classifying between a number of stochastic processes. To clarify this statement, we give the following illustrative example.

Example 3.7. *Observe stochastic processes $(X(t))_{t>0}$ and $(Y(t))_{t>0}$, both following the law as described in equation (2.1). Assume they have the same volatility $\bar{\sigma}(t, X(t)) = \sigma$, but have unequal drift terms:*

$$\begin{aligned}\bar{\mu}(t, X(t)) &= \mu_x,\\ \bar{\mu}(t, Y(t)) &= \mu_y,\end{aligned}$$

where $\mu_1, \mu_2 \in \mathbb{R}$ and $\mu_x \neq \mu_y$. When attempting to classify a generated sample to belong to either process $(X(t))_{t>0}$ or $(Y(t))_{t>0}$, one easily realises that the maximum classification accuracy heavily depends on the difference between the two drift terms. For example, when $\mu_x = -10$ and $\mu_y = 10$, the classification is significantly easier than when $\mu_x = -0.01$ and $\mu_y = 0.01$. A 100% classification accuracy is therefore only possible in trivial cases.

It is important to realise that there is a theoretical best classification accuracy, which depends on the parametric model and the specific parameter choice.

To compute this theoretical best classification accuracy, the optimal classification rule is used. To simplify the setting, only a binary classification rule is given, but its extension to multi-class classification is straightforward. Let $X(t)$ and $Y(t)$ be two known random variables and let $Z(t)$ be a random variable of which either $Z(t) \overset{\mathrm{d}}{=} X(t)$ or $Z(t) \overset{\mathrm{d}}{=} Y(t)$ for a fixed $t > 0$. Observe one sample $z(t)$ from $Z(t)$.The optimal classification rule is given by

$$Z(t) \sim \begin{cases} X(t) \text{ if } f_{X(t)}(z(t)) \geq f_{Y(t)}(z(t)), \\ Y(t) \text{ otherwise.} \end{cases} \tag{3.10}$$

Here $f_{X(t)}(\cdot), f_{Y(t)}(\cdot)$ denote the density functions of $X(t), Y(t)$, respectively, at fixed time t. Notice that the classification rule above simply classifies a sample $z(t)$ based on which random variable is *more likely* to have generated it.

Unfortunately, only a select number of stochastic processes have analytically available density functions. In the remainder of this section, for several stochastic processes of which the density functions are known the theoretical maximum (binary) classification accuracy will be analytically computed. These will be compared to the empirical classification accuracy obtained by implementing the proposed solution given in section 3.4.

3.5.1. Discrete ABM: different drifts

The first process analysed is an ABM with different drifts. The attention is restricted to discrete ABM. Let $\xi_k \overset{i.i.d}{\sim} N(0,1)$ for all $k \in \mathbb{N}$.

$$\begin{aligned}X_{k+1} &= X_k + \mu\Delta t + \sigma\sqrt{\Delta t}\xi_k,\\ X_{k+1}|X_k &\sim N(X_k + \mu\Delta t, \sigma^2\Delta t).\end{aligned}$$

Here $\mu \in \mathbb{R}$ and $\sigma \in \mathbb{R}^+$.

Simplified case: one time step

To avoid a multivariate distribution, observe a path with just one time step. Assuming $X_0 = x_0$ is known, this gives us $X_1 = x_0 + \mu\Delta t + \sigma\sqrt{\Delta t}\xi \sim N(x_0 + \mu\Delta t, \sigma^2\Delta t)$.

For simplicity assume that $x_0 = 0$. Let $X \sim N(\mu_x\Delta t, \sigma^2\Delta t)$ and $Y \sim N(\mu_y\Delta t, \sigma^2\Delta t)$ where $\mu_y > \mu_x$. Assume either $Z \overset{d}{=} X$ or $Z \overset{d}{=} Y$, i.e. Z has either μ_x or μ_y as drift. The goal is to find out which drift corresponds to Z.

Given one observation z from Z, the optimal classification is of the shape as given in equation (3.10). In other words, Z is classified as having the same distribution as X if the sample is *more likely* to be a sample from X than a sample from Y.

Naturally, this classification rule does not always classify the sample correctly. Assuming that $Z \sim X$, the probability of correct classification is given by:

$$\begin{aligned}\mathbb{P}(f_X(z) \geq f_Y(z)|Z \sim X) &= \mathbb{P}\Big(\frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{1}{2}(\frac{Z-\mu_x\Delta t}{\sigma})^2} \geq \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{1}{2}(\frac{Z-\mu_y\Delta t}{\sigma})^2}\Big|Z \sim X\Big) \\ &= \mathbb{P}\Big(e^{-\frac{1}{2}(\frac{Z-\mu_x\Delta t}{\sigma})^2} \geq e^{-\frac{1}{2}(\frac{Z-\mu_y\Delta t}{\sigma})^2}\Big|Z \sim X\Big).\end{aligned}$$

Notice that taking the logarithm on both sides does not influence the inequality sign, since the logarithm is an increasing function. This gives:

$$\begin{aligned}\mathbb{P}(f_X(z) \geq f_Y(z)\Big|Z \sim X) &= \mathbb{P}\Big(-\frac{1}{2}(\frac{Z-\mu_x\Delta t}{\sigma})^2 \geq -\frac{1}{2}(\frac{Z-\mu_y\Delta t}{\sigma})^2\Big|Z \sim X\Big) \\ &= \mathbb{P}(Z\Delta t(-2\mu_x + 2\mu_y) \leq \Delta t^2(\mu_y^2 - \mu_x^2)|Z \sim X) \\ &= \mathbb{P}\Big(Z \leq \frac{\Delta t}{2}(\mu_x + \mu_y)|Z \sim X\Big) \\ &= F_X\Big(\frac{1}{2}(\mu_x + \mu_y)\Big). \qquad (3.11)\end{aligned}$$

Here $F_X(\cdot)$ is the cumulative distribution function of X. Since its known that $X \sim N(\mu_x\Delta t, \sigma^2\Delta t)$, $F_X(\cdot)$ is known and the value in (3.11) can be computed. Notice that a similar calculation gives

$$\begin{aligned}\mathbb{P}(f_X(z) \geq f_Y(z)|Z \sim Y) &= F_Y\Big(\frac{1}{2}(\mu_x + \mu_y)\Big), \\ \mathbb{P}(f_X(z) < f_Y(z)|Z \sim X) &= 1 - F_X\Big(\frac{1}{2}(\mu_x + \mu_y)\Big), \\ \mathbb{P}(f_X(z) < f_Y(z)|Z \sim Y) &= 1 - F_Y\Big(\frac{1}{2}(\mu_x + \mu_y)\Big),\end{aligned}$$

which represent the probability of misclassifying a sample from Y, misclassifying a sample from X and correctly classifying a sample from Y, respectively.

This means that the total probability of correct classification is given by

$$\mathbb{P}(\text{Correct classification}) = \mathbb{P}(f_X(z) \geq f_Y(z)|Z \sim X)\mathbb{P}(Z \sim X) + \mathbb{P}(f_X(z) < f_Y(z)|Z \sim Y)\mathbb{P}(Z \sim Y).$$

To simplify the notation, the event C is introduced as

$$C^{\text{A, drift}} := \{z : z \text{ is correctly classified}\}.$$

This results in

$$\mathbb{P}(C^{\text{A, drift}}) := \mathbb{P}(\text{Correct classification}). \qquad (3.12)$$

Since from both processes an equal number of paths is sampled, we set $\mathbb{P}(Z \sim X) = \mathbb{P}(Z \sim Y) = \frac{1}{2}$. This results in the following probability of correct classification:

$$\begin{aligned}\mathbb{P}(C^{\text{A, drift}}) &= \frac{1}{2}\mathbb{P}(f_X(z) \geq f_Y(z)|Z \sim X) + \frac{1}{2}\mathbb{P}(f_X(z) < f_Y(z)|Z \sim Y) \\ &= \frac{1}{2}\Big(F_X\Big(\frac{1}{2}(\mu_x + \mu_y)\Big) + 1 - F_Y\Big(\frac{1}{2}(\mu_x + \mu_y)\Big)\Big) \\ &= \frac{1}{2} + \frac{1}{2}\Big(F_X\Big(\frac{1}{2}(\mu_x + \mu_y)\Big) - F_Y\Big(\frac{1}{2}(\mu_x + \mu_y)\Big)\Big).\end{aligned}$$

This result is intuitive, since the probability to correctly classify z depends on $\frac{1}{2}(\mu_x+\mu_y)$, which is exactly halfway between the two peaks of the density functions, as is indicated in figure 3.9:

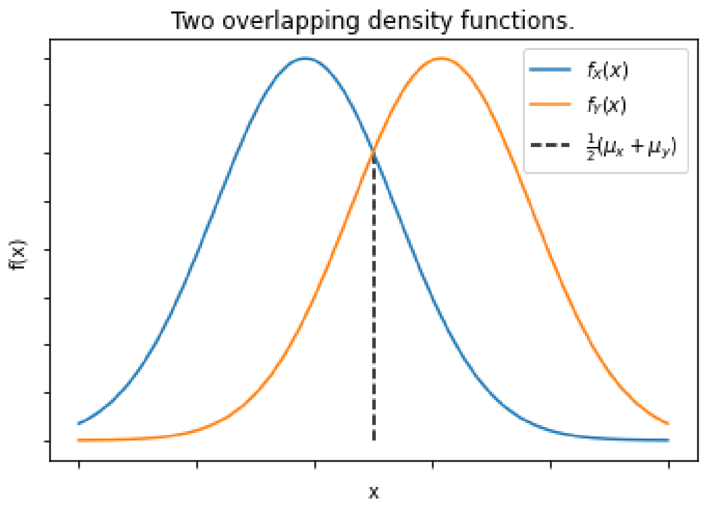


Figure 3.9: Density functions with different mean. The value $\frac{1}{2}(\mu_x+\mu_y)$ is located exactly halfway the two peaks.

Using classification boundary $\frac{1}{2}(\mu_x+\mu_y)$ maximises the probability of correct classification, as is indicated by Figure 3.9. It is the optimal classification boundary; choosing a higher or lower classification boundary decreases the probability of correct classification.

Multiple time steps
Now assume a path following ABM with n time steps, i.e. observe a sample $z = (z_1,\dots,z_n)$ from $Z = (Z_1,\dots,Z_n)$. Let $\xi_i \sim N(0,1)$ for all $i \in \{1,\dots,n\}$. Define the increments of Z as

$$\begin{aligned}\hat{Z}_i &= Z_i - Z_{i-1}\\ &= Z_{i-1} + \mu\Delta t + \sigma\sqrt{\Delta t}\xi_i - Z_{i-1}\\ &= \mu\Delta t + \sigma\sqrt{\Delta t}\xi_k && (3.13)\\ &\sim N(\mu\Delta t, \sigma^2\Delta t). && (3.14)\end{aligned}$$

Notice that the increments are independent and identically distributed (i.i.d.).
Let $X = (X_1,\dots,X_n)$ and $(Y_1,\dots,Y_n)$ follow an ABM with volatility σ and drifts μ_x and μ_y respectively. Given one observation z from Z, the optimal classification rule is

$$Z \sim \begin{cases} X \text{ if } f_X(z_1,\dots,z_n) \geq f_Y(z_1,\dots,z_n),\\ Y \text{ otherwise.}\end{cases}$$

Notice that this is equivalent to

$$Z \sim \begin{cases} X \text{ if } f_{\hat{X}}(\hat{z}_1,\dots,\hat{z}_n) \geq f_{\hat{Y}}(\hat{z}_1,\dots,\hat{z}_n).\\ Y \text{ otherwise.}\end{cases}$$

where $\hat{X} = (\hat{X}_1,\dots,\hat{X}_n)$, $\hat{Y} = (\hat{Y}_1,\dots,\hat{Y}_n)$ and $\hat{X}_i, \hat{Y}_i$ denote the increments of X and Y respectively.

Since $\hat{Z}_1,\dots,\hat{Z}_n$ are i.i.d distributed ,this gives us:

$$\begin{aligned}\mathbb{P}\big(f_{\hat{X}}(\hat{z_1},\dots,\hat{z_n}) \geq f_{\hat{Y}}(\hat{z_1},\dots,\hat{z_n})|Z\sim X\big) &= \mathbb{P}\Big(\prod_{i=1}^{n} f_{\hat{X}}(\hat{z_i}) \geq \prod_{i=1}^{n} f_{\hat{Y}}(\hat{z_i})|Z\sim X\Big)\\ &= \mathbb{P}\Bigg(\frac{1}{(2\pi\sigma^2\Delta t)^{\frac{n}{2}}} e^{-\frac{1}{2\sigma^2\Delta t}\sum_{i=1}^{n}(\hat{z}_i-\mu_x\Delta t)^2} \geq \frac{1}{(2\pi\sigma^2\Delta t)^{\frac{n}{2}}} e^{-\frac{1}{2\sigma^2\Delta t}\sum_{i=1}^{n}(\hat{z}_i-\mu_y\Delta t)^2} |Z\sim X\Bigg).\end{aligned}$$

Taking the logarithm on both sides and rearranging gives:

$$\begin{aligned}
&\mathbb{P}\Big(f_{\hat{X}}(\hat{z_1},\dots,\hat{z_n}) \geq f_{\hat{Y}}(\hat{z_1},\dots,\hat{z_n})|Z\sim X\Big)\\
&\quad = \mathbb{P}\Bigg(\sum_{i=1}^{n}(\hat{Z}_i-\mu_x\Delta t)^2 \leq \sum_{i=1}^{n}(\hat{Z}_i-\mu_y\Delta t)^2|Z\sim X\Bigg)\\
&\quad = \mathbb{P}\Bigg(\sum_{i=1}^{n} -2\mu_x\Delta t\hat{Z}_i + 2\mu_y\Delta t\hat{Z}_i + (\mu_x\Delta t)^2 - (\mu_y\Delta t)^2 \leq 0\Big|Z\sim X\Bigg)\\
&\quad = \mathbb{P}\Big(-2\mu_x\Delta t\sum_{i=1}^{n}\hat{Z}_i + 2\mu_y\Delta t\sum_{i=1}^{n}\hat{Z}_i + n(\mu_x\Delta t)^2 - n(\mu_y\Delta t)^2 \leq 0\Big|Z\sim X\Big)\\
&\quad = \mathbb{P}\Big(2\Delta t(\mu_y-\mu_x)\sum_{i=1}^{n}\hat{Z}_i \leq n(\mu_y\Delta t)^2 - n(\mu_x\Delta t)^2|Z\sim X\Big)\\
&\quad = \mathbb{P}\Bigg(\sum_{i=1}^{n}\hat{Z}_i \leq \frac{n\Delta t(\mu_y^2-\mu_x^2)}{2(\mu_y-\mu_x)}\Big|Z\sim X\Bigg)\\
&\quad = \mathbb{P}\Bigg(\sum_{i=1}^{n}Z_i - Z_{i-1} \leq \frac{n\Delta t}{2}(\mu_y+\mu_x)\Big|Z\sim X\Bigg).
\end{aligned}$$

Notice that $\sum_{i=1}^{n} Z_i - Z_{i-1} = Z_n - Z_0$, since it is a telescoping sum. Additionally $Z_0 = z_0$ is known and $n\Delta t = T$, where T is the time horizon of the path. This results in

$$\begin{aligned}
\mathbb{P}(f_{\hat{X}}(\hat{z_1},\dots,\hat{z_n}) \geq f_{\hat{Y}}(\hat{z_1},\dots,\hat{z_n})|Z\sim X) &= \mathbb{P}\Big(Z_n \leq z_0 + \frac{T}{2}(\mu_x+\mu_y)|Z\sim X\Big)\\
&= F_{X_n}\Big(z_0 + \frac{T}{2}(\mu_x+\mu_y)\Big)\\
&= F_{X_n}\Big(x_0 + \frac{T}{2}(\mu_x+\mu_y)\Big) \text{ (since } z_0 = x_0\text{)}.
\end{aligned}$$

Here $F_{X_n}(\cdot)$ denotes the cumulative distribution function of X_n.

Remark 8. *The result above shows that only the first and last element of the path are relevant for the classification of this specific time series, all elements in between are not needed.*

Shifting distribution
Observe again

$$\mathbb{P}(f_{\hat{X}}(\hat{z}_1,\dots,\hat{z}_n) \geq f_{\hat{Y}}(\hat{z}_1,\dots,\hat{z}_n)|Z\sim X) = F_{X_n}\Big(x_0 + \frac{T}{2}(\mu_x+\mu_y)\Big). \tag{3.15}$$

The manner in which this probability is presented can be misleading, since equation (3.15) might give the impression that the probability of correct classification increases when, for instance, x_0 increases. However, this reasoning is incorrect since μ_x and x_0 appear in the argument, but are also parameters of the distribution of X_n, as can be seen below:

$$\begin{aligned}
X_n &= X_{n-1} + \mu_x\Delta t + \sigma\sqrt{\Delta t}\xi_{n-1} = X_{n-2} + 2\mu_x\Delta t + \sigma\sqrt{\Delta t}(\xi_{n-1}+\xi_{n-2})\\
&= X_0 + n\mu_x\Delta t + \sigma\sqrt{\Delta t}\sum_{i=1}^{n}\xi_i \sim N(x_0 + n\mu_x\Delta t, n\sigma^2\Delta t)\\
&= N(x_0+\mu_x T, \sigma^2 T).
\end{aligned}$$

Therefore, random variable K_x is now defined as

$$K_x := \frac{X_n - x_0 - \mu_x T}{\sigma\sqrt{T}}.$$

Notice that $K_x \sim N(0,1)$. This leads to

$$\begin{aligned}\mathbb{P}(f_{\hat{X}}(\hat{z}_1,\dots,\hat{z}_n) \geq f_{\hat{Y}}(\hat{z}_1,\dots,\hat{z}_n)|Z \sim X) &= F_{X_n}\Big(x_0 + \frac{T}{2}(\mu_x+\mu_y)\Big) \\ &= F_{K_x}\left(\frac{x_0 + \frac{T}{2}(\mu_x+\mu_y) - x_0 - \mu_x T}{\sigma\sqrt{T}}\right) \\ &= F_{K_x}\left(\frac{\sqrt{T}}{2\sigma}(\mu_y-\mu_x)\right).\end{aligned}$$

From this result three interesting facts can be derived:

1. As the time length of the path increases, so does the probability of correctly classifying a sample from X.
2. As the difference in drifts increases, so does the probability of correctly classifying a sample from X.
3. As the volatility increases, the probability of correctly classifying a sample from X decreases.

Similar calculations give

$$\begin{aligned}\mathbb{P}(f_{\hat{X}}(\hat{z}_1,\dots,\hat{z}_n) \geq f_{\hat{Y}}(\hat{z}_1,\dots,\hat{z}_n)|Z \sim Y) &= F_{Y_n}\Big(y_0 + \frac{T}{2}(\mu_x+\mu_y)\Big) \quad = \quad F_{K_y}\left(\frac{\sqrt{T}}{2\sigma}(\mu_x-\mu_y)\right), \\ \mathbb{P}(f_{\hat{X}}(\hat{z}_1,\dots,\hat{z}_n) < f_{\hat{Y}}(\hat{z}_1,\dots,\hat{z}_n)|Z \sim X) &= 1 - F_{K_x}\left(\frac{\sqrt{T}}{2\sigma}(\mu_y-\mu_x)\right), \\ \mathbb{P}(f_{\hat{X}}(\hat{z}_1,\dots,\hat{z}_n) < f_{\hat{Y}}(\hat{z}_1,\dots,\hat{z}_n)|Z \sim Y) &= 1 - F_{K_y}\left(\frac{\sqrt{T}}{2\sigma}(\mu_x-\mu_y)\right),\end{aligned}$$

where

$$K_y := \frac{Y_n - y_0 - \mu_y T}{\sigma\sqrt{T}}.$$

Recall from equation (3.12) that $\mathbb{P}(C^{\text{A, drift}}) = \mathbb{P}(\text{Correctly classification})$. Again setting $\mathbb{P}(Z \sim X) = \mathbb{P}(Z \sim Y) = \frac{1}{2}$ leads to

$$\begin{aligned}&\mathbb{P}(C^{\text{A, drift}}) \\ &\quad = \frac{1}{2}\mathbb{P}(f_{\hat{X}}(\hat{z_1},\dots,\hat{z_n}) \geq f_{\hat{Y}}(\hat{z_1},\dots,\hat{z_n})|Z \sim X) + \frac{1}{2}\mathbb{P}(f_{\hat{X}}(\hat{z_1},\dots,\hat{z_n}) < f_{\hat{Y}}(\hat{z_1},\dots,\hat{z_n})|Z \sim Y) \\ &\quad = \frac{1}{2}\Big(F_{K_x}\Big(\frac{\sqrt{T}}{2\sigma}(\mu_y-\mu_x)\Big) + 1 - F_{K_y}\Big(\frac{\sqrt{T}}{2\sigma}(\mu_x-\mu_y)\Big)\Big) \\ &\quad = \frac{1}{2} + \frac{1}{2}\Big(\Phi\Big(\frac{\sqrt{T}}{2\sigma}(\mu_y-\mu_x)\Big) - \Phi\Big(\frac{\sqrt{T}}{2\sigma}(\mu_x-\mu_y)\Big)\Big).\end{aligned}$$

Here Φ is a common notation for the cumulative distribution function of a standard normal random variable, since $K_x, K_y \sim N(0,1)$. This gives an analytic expression for the theoretical maximum accuracy that can be obtained when classifying between paths of n time steps following a ABM with equal volatility and different drifts.

Empirical results

The results above are compared with empirical findings using the proposed solution explained in section 3.4. The precise structure of the empirical classification process is revealed in chapter 4, this section merely serves as an initial justification of the proposed method. Suppose $X_0 = Y_0 = Z_0 = 100$, $\sigma = 0.2$ and $n = 1000$, i.e. a path with 1000 time steps. 10,000 paths are generated per model and a random forest classifier is used. The data is split in a 70% training set and a 30% testing set. In table 3.1 the theoretical maximum classification accuracy is compared with the accuracy obtained by the signature method proposed in this thesis. The displayed empirical accuracy is computed over the test set.

μ_x	μ_y	Theoretical maximum accuracy	Obtained empirical accuracy
0.1	0.15	0.654	0.623
0.1	0.2	0.785	0.771
0.1	0.25	0.882	0.878

Table 3.1: Theoretical and empirical classification accuracy for classification between ABMs with different drifts.

Notice that the theoretical and empirical classification accuracies are similar. This indicates that the theory and practice are in agreement. The values chosen for μ_x, μ_y are values that are commonly observed in financial time series. Table 3.1 shows that distinguishing between ABMs with different drifts will be challenging - especially for small differences between μ_x and μ_y - as the maximum possible classification accuracy is quite low.

3.5.2. Discrete ABM: different volatilities

The situation is similar to the setting in the previous paragraph, only the difference is in the volatility parameters instead of the drift parameters. Assume a path following a discrete ABM with n time steps, i.e. observe a sample $z = (z_1, \dots, z_n)$ from $Z = (Z_1, \dots, Z_n)$.

Again let $X = (X_1, \dots, X_n)$ and $Y = (Y_1, \dots, Y_n)$ follow a discrete ABM with equal drift μ and volatility σ_x and σ_y respectively, with $\sigma_y > \sigma_x$. Let $\hat{X}, \hat{Y}, \hat{Z}$ denote the increments of the discrete processes X, Y, Z, respectively in the same manner as in equation (3.13). Recall from equation (3.14) that $\hat{Z}_i \overset{i.i.d}{\sim} N(\mu\Delta t, \sigma^2\Delta t)$.

In the same manner as before, the probability of correctly classifying a sample from X is described below:

$$
\begin{aligned}
&\mathbb{P}\Big(f_{\hat{X}}(\hat{z}_1, \dots, \hat{z}_n) \geq f_{\hat{Y}}(\hat{z}_1, \dots, \hat{z}_n) | Z \sim X\Big) \\
&\qquad = \mathbb{P}\Big(\prod_{i=1}^{n} f_{\hat{X}}(\hat{z}_i) \geq \prod_{i=1}^{n} f_{\hat{Y}}(\hat{z}_i) \Big| Z \sim X\Big) \\
&\qquad = \mathbb{P}\Big(\Big(\frac{1}{\sqrt{2\pi\sigma_x^2\Delta t}}\Big)^n e^{-\frac{1}{2\sigma_x^2\Delta t}\sum_{i=1}^{n}(\hat{Z}_i - \mu\Delta t)^2} \geq \Big(\frac{1}{\sqrt{2\pi\sigma_y^2\Delta t}}\Big)^n e^{-\frac{1}{2\sigma_y^2\Delta t}\sum_{i=1}^{n}(\hat{Z}_i - \mu\Delta t)^2} \Big| Z \sim X\Big) \\
&\qquad = \mathbb{P}\Big(\Big(\frac{1}{\sigma_x}\Big)^n e^{-\frac{1}{2\sigma_x^2\Delta t}\sum_{i=1}^{n}(\hat{Z}_i - \mu\Delta t)^2} \geq \Big(\frac{1}{\sigma_y}\Big)^n e^{-\frac{1}{2\sigma_y^2\Delta t}\sum_{i=1}^{n}(\hat{Z}_i - \mu\Delta t)^2} \Big| Z \sim X\Big).
\end{aligned}
$$

Taking the logarithm of the argument gives:

$$
\begin{aligned}
&\mathbb{P}\Big(f_{\hat{X}}(\hat{z}_1, \dots, \hat{z}_n) \geq f_{\hat{Y}}(\hat{z}_1, \dots, \hat{z}_n) \Big| Z \sim X\Big) \\
&\qquad = \mathbb{P}\Big(-n\log(\sigma_x) - \frac{1}{2\sigma_x^2\Delta t}\sum_{i=1}^{n}(\hat{Z}_i - \mu\Delta t)^2 \geq -n\log(\sigma_y) - \frac{1}{2\sigma_y^2\Delta t}\sum_{i=1}^{n}(\hat{Z}_i - \mu\Delta t)^2 \Big| Z \sim X\Big) \\
&\qquad = \mathbb{P}\Big(\frac{1}{2\Delta t}\Big(\frac{1}{\sigma_y^2} - \frac{1}{\sigma_x^2}\Big)\sum_{i=1}^{n}(\hat{Z}_i - \mu\Delta t)^2 \geq n\log(\frac{\sigma_x}{\sigma_y}) \Big| Z \sim X\Big) \\
&\qquad = \mathbb{P}\Big(\sum_{i=1}^{n}(\hat{Z}_i - \mu\Delta t)^2 \leq \frac{n\log(\frac{\sigma_x}{\sigma_y})}{\frac{1}{2\Delta t}\Big(\frac{1}{\sigma_y^2} - \frac{1}{\sigma_x^2}\Big)} \Big| Z \sim X\Big).
\end{aligned}
$$

Notice that the inequality sign flips in the last step, since both sides are divided by $\frac{1}{2\Delta t}\Big(\frac{1}{\sigma_y^2} - \frac{1}{\sigma_x^2}\Big) < 0$, because $\sigma_y > \sigma_x$, by assumption. We see that

$$\begin{aligned}\mathbb{P}(f_{\hat{X}}(\hat{z}_1,\dots,\hat{z}_n) \geq f_{\hat{Y}}(\hat{z}_1,\dots,\hat{z}_n)\Big|Z \sim X) &= \mathbb{P}\Bigg(\sum_{i=1}^{n}(\hat{Z}_i - \mu\Delta t)^2 \leq \frac{2n\Delta t \log(\frac{\sigma_x}{\sigma_y})}{\left(\frac{1}{\sigma_y^2} - \frac{1}{\sigma_x^2}\right)}\Big|Z \sim X\Bigg) \\ &= \mathbb{P}\Bigg(\sum_{i=1}^{n}(\hat{X}_i - \mu\Delta t)^2 \leq \frac{2n\Delta t \log(\frac{\sigma_x}{\sigma_y})}{\left(\frac{\sigma_x^2-\sigma_y^2}{\sigma_x^2\sigma_y^2}\right)}\Bigg).\end{aligned}$$

Since $\hat{X}_i \overset{i.i.d}{\sim} N(\mu\Delta t, \sigma_x^2\Delta t)$, it follows that

$$A_i := \hat{X}_i - \mu\Delta t \overset{i.i.d}{\sim} N(0, \sigma_x^2\Delta t).$$

Since every A_i is normally distributed and independent, it follows that

$$\begin{aligned}\frac{A_i}{\sigma_x\sqrt{\Delta t}} &\overset{i.i.d.}{\sim} N(0,1), \\ \sum_{i=1}^{n}\left(\frac{A_i}{\sigma_x\sqrt{\Delta t}}\right)^2 &\sim \chi_n^2, \quad \frac{\sum_{i=1}^{n} A_i^2}{\sigma_x^2\Delta t} \sim \chi_n^2.\end{aligned}$$

This implies that

$$\begin{aligned}\mathbb{P}\Big(f_{\hat{X}}(\hat{z}_1,\dots,\hat{z}_n) \geq f_{\hat{Y}}(\hat{z}_1,\dots,\hat{z}_n)|Z \sim X) &= \mathbb{P}\Bigg(\sum_{i=1}^{n}(\hat{X}_i - \mu\Delta t)^2 \leq \frac{2n\Delta t \log(\frac{\sigma_x}{\sigma_y})}{\left(\frac{\sigma_x^2-\sigma_y^2}{\sigma_x^2\sigma_y^2}\right)}\Bigg) \\ &= \mathbb{P}\Bigg(\frac{\sum_{i=1}^{n} A_i^2}{\sigma_x^2\Delta t} \leq \frac{1}{\sigma_x^2\Delta t}\frac{2n\Delta t \log(\frac{\sigma_x}{\sigma_y})}{\left(\frac{\sigma_x^2-\sigma_y^2}{\sigma_x^2\sigma_y^2}\right)}\Bigg) \\ &= \mathbb{P}\Bigg(\frac{\sum_{i=1}^{n} A_i^2}{\sigma_x^2\Delta t} \leq \frac{2n\sigma_y^2 \log(\frac{\sigma_x}{\sigma_y})}{\sigma_x^2 - \sigma_y^2}\Bigg).\end{aligned}$$

Remark 9. *Notice that the probability of correctly classifying a sample from X following an ABM where the volatility parameters differ depends on all elements $\{1,\dots,n\}$ of the time series. This is in contrast with the classification between processes with different drifts, where the probability only depends on the first and last element of the observed sample.*

A similar derivation gives:

$$\begin{aligned}\mathbb{P}\Big(f_{\hat{X}}(\hat{z}_1,\dots,\hat{z}_n) \geq f_{\hat{Y}}(\hat{z}_1,\dots,\hat{z}_n)|Z \sim Y) &= \mathbb{P}\Bigg(\frac{\sum_{i=1}^{n} B_i^2}{\sigma_y^2\Delta t} \leq \frac{2n\sigma_x^2 \log(\frac{\sigma_x}{\sigma_y})}{\sigma_x^2 - \sigma_y^2}\Bigg), \\ \mathbb{P}\Big(f_{\hat{X}}(\hat{z}_1,\dots,\hat{z}_n) < f_{\hat{Y}}(\hat{z}_1,\dots,\hat{z}_n)|Z \sim X) &= 1 - \mathbb{P}\Bigg(\frac{\sum_{i=1}^{n} A_i^2}{\sigma_x^2\Delta t} \leq \frac{2n\sigma_y^2 \log(\frac{\sigma_x}{\sigma_y})}{\sigma_x^2 - \sigma_y^2}\Bigg), \\ \mathbb{P}\Big(f_{\hat{X}}(\hat{z}_1,\dots,\hat{z}_n) < f_{\hat{Y}}(\hat{z}_1,\dots,\hat{z}_n)|Z \sim Y) &= 1 - \mathbb{P}\Bigg(\frac{\sum_{i=1}^{n} B_i^2}{\sigma_y^2\Delta t} \leq \frac{2n\sigma_x^2 \log(\frac{\sigma_x}{\sigma_y})}{\sigma_x^2 - \sigma_y^2}\Bigg),\end{aligned}$$

where $B_i := \hat{Y}_i - \mu$ and $\frac{\sum_{i=1}^{n} B_i^2}{\sigma_y^2\Delta t} \sim \chi_n^2$.

Recalling equation (3.12) and setting again $\mathbb{P}(Z \sim X) = \mathbb{P}(Z \sim Y) = \frac{1}{2}$ results in:

$$\begin{aligned}
&\mathbb{P}(C^{\text{A, vol.}}) \\
&= \frac{1}{2}\mathbb{P}(f_{\hat{X}}(\hat{z_1}, \dots, \hat{z_n}) \geq f_{\hat{Y}}(\hat{z_1}, \dots, \hat{z_n}) | Z \sim X) + \frac{1}{2}\mathbb{P}(f_{\hat{X}}(\hat{z_1}, \dots, \hat{z_n}) < f_{\hat{Y}}(\hat{z_1}, \dots, \hat{z_n}) | Z \sim Y) \\
&= \frac{1}{2}\mathbb{P}\Big(\frac{\sum_{i=1}^{n} A_i^2}{\sigma_x^2 \Delta t} \leq \frac{2n\sigma_y^2 \log(\frac{\sigma_x}{\sigma_y})}{\sigma_x^2 - \sigma_y^2}\Big) + \frac{1}{2}\Big(1 - \mathbb{P}\Big(\frac{\sum_{i=1}^{n} B_i^2}{\sigma_y^2 \Delta t} \leq \frac{2n\sigma_x^2 \log(\frac{\sigma_x}{\sigma_y})}{\sigma_x^2 - \sigma_y^2}\Big)\Big) \\
&= \frac{1}{2} + \frac{1}{2}\Big(\chi^2\Big(\frac{2n\sigma_y^2 \log(\frac{\sigma_x}{\sigma_y})}{\sigma_x^2 - \sigma_y^2}; n\Big) - \chi^2\Big(\frac{2n\sigma_x^2 \log(\frac{\sigma_x}{\sigma_y})}{\sigma_x^2 - \sigma_y^2}; n\Big)\Big).
\end{aligned}$$

Here $\chi^2(x; n)$ denotes the cumulative distribution function of the χ^2 distribution with argument x and n degrees of freedom.

Empirical results
Again the computed theoretical maximum accuracy is compared to the empirical results obtained using the proposed solution as described in section 3.4. As mentioned before, the precise structure of the empirical classification process is revealed in chapter 4, this section merely serves as an initial justification of the proposed method. Suppose $X_0 = Y_0 = Z_0 = 100$, $\mu = 0.1$ and $n = 1000$, i.e. a path with 1000 time steps. 10,000 paths are generated per model and a random forest classifier is used. The data is split in a 70% training set and a 30% testing set. In table 3.2 the theoretical maximum classification accuracy is compared with the accuracy obtained by the signature method proposed in this thesis. The displayed empirical accuracy is computed over the test set.

σ_x	σ_y	Theoretical maximum accuracy	Obtained empirical accuracy
0.1	0.105	0.8623	0.800
0.1	0.11	0.9834	0.955
0.1	0.2	1.0	1.0
0.1	0.3	1.0	1.0
0.1	0.4	1.0	1.0

Table 3.2: Theoretical and empirical classification accuracy for classification between ABMs with different volatility.

Again the empirical accuracy is similar to the theoretical maximum accuracy. Table 3.2 shows that the theoretical maximum accuracy is close to 1 for most parameter settings and only declines when the difference between σ_x and σ_y gets smaller than 0.01. When comparing the results in table 3.2 to the results listed in table 3.1, we immediately see that distinguishing ABMs with different volatilities is significantly easier than distinguishing between ABMs with different drifts, for the parameter choices in this thesis.

3.5.3. Discrete GBM: different drifts

The next process analysed is discrete Geometric Brownian Motion (GBM). GBM is a often used stochastic process in the simulation of financial variables and therefore deserves attention in this thesis. Let again $\xi_k \overset{i.i.d.}{\sim} N(0, 1)$ for all $k \in \{1, \dots, n\}$. The discrete GBM U_k is represented by

$$U_{k+1} = U_k e^{\mu \Delta t + \sigma \sqrt{\Delta t} \xi_k}. \tag{3.16}$$

Define the increments of U as

$$\begin{aligned}
\hat{U}_{k+1} &= \frac{U_{k+1}}{U_k} \\
&= e^{\mu \Delta t + \sigma \sqrt{\Delta t} \xi_k} \\
&\overset{i.i.d.}{\sim} \text{lognormal}(\mu \Delta t, \sigma^2 \Delta t).
\end{aligned} \tag{3.17}$$

Now assume a path following GBM with n time steps, i.e. observe a sample $u = (u_1, \dots, u_n)$ from $U = (U_1, \dots, U_n)$. Let $S = (S_1, \dots, S_n)$ and $R = (R_1, \dots, R_n)$ follow a GBM with volatility σ and drift $\mu_x \Delta t$

and $\mu_y \Delta t$ respectively. Given one observation u from U, the optimal classification rule is given by

$$U \sim \begin{cases} S \text{ if } f_S(u_1, \dots, u_n) \geq f_R(u_1, \dots, u_n), \\ R \text{ otherwise.} \end{cases}$$

Notice that this is equivalent to

$$U \sim \begin{cases} S \text{ if } f_{\hat{S}}(\hat{u}_1, \dots, \hat{u}_n) \geq f_{\hat{R}}(\hat{u}_1, \dots, \hat{u}_n), \\ R \text{ otherwise.} \end{cases}$$

where $\hat{S} = (\hat{S}_1, \dots, \hat{S}_n)$, $\hat{R} = (\hat{R}_1, \dots, \hat{R}_n)$ are defined in a similar way as in equation (3.17). Recall that in equation (3.13) we defined $\hat{Z}_{k+1} = \mu \Delta t + \sigma \sqrt{\Delta t} \xi_k$. Notice that for all increments $\hat{U}_{k+1}$ the following holds:

$$\hat{U}_{k+1} = e^{\mu \Delta t + \sigma \sqrt{\Delta t} \xi_k} = e^{\hat{Z}_{k+1}}.$$

So equivalently,

$$\log(\hat{U}_{k+1}) = \hat{Z}_{k+1}. \tag{3.18}$$

Since in a similar way $\hat{X}_{k+1} = \mu_x \Delta t + \sigma \sqrt{\Delta t} \xi_k$ and $\hat{Y}_{k+1} = \mu_y \Delta t + \sigma \sqrt{\Delta t} \xi_k$ is defined, the following holds:

$$\log(\hat{S}_{k+1}) = \hat{X}_{k+1}, \ \log(\hat{R}_{k+1}) = \hat{Y}_{k+1}.$$

In the same way as in the sections 3.5.1 and 3.5.2, the goal is to compute the probability of correctly classifying a sample from S:

$$\mathbb{P}(f_{\hat{S}}(\hat{u}_1, \dots, \hat{u}_n) \geq f_{\hat{R}}(\hat{u}_1, \dots, \hat{u}_n) | U \sim S). \tag{3.19}$$

Notice that

$$\begin{aligned} F_{\hat{S}}(\hat{s}) &= \mathbb{P}(\hat{S} \leq \hat{s}) = \mathbb{P}(e^{\hat{X}} \leq \hat{s}) \\ &= F_{\hat{X}}(\log(\hat{s})). \end{aligned}$$

In a similar way $F_{\hat{R}}(\hat{r}) = F_{\hat{Y}}(\log(\hat{r}))$. This implies that

$$f_{\hat{S}}(\hat{s}) = f_{\hat{X}}(\log(\hat{s})), \ f_{\hat{R}}(\hat{R}) = f_{\hat{Y}}(\log(\hat{r})). \tag{3.20}$$

This is useful, since $f_{\hat{X}}, f_{\hat{Y}}$ are known and given by equation (3.14). Combining this with equation (3.19) gives

$$\mathbb{P}(f_{\hat{S}}(\hat{u}_1, \dots, \hat{u}_n) \geq f_{\hat{R}}(\hat{u}_1, \dots, \hat{u}_n) | U \sim S) = \mathbb{P}(f_{\hat{X}}(\log(\hat{u}_1), \dots, \log(\hat{u}_n)) \geq f_{\hat{Y}}(\log(\hat{u}_1), \dots, \log(\hat{u}_n)) | U \sim S). \tag{3.21}$$

Using equation (3.18) gives

$$\mathbb{P}(f_{\hat{S}}(\hat{u}_1, \dots, \hat{u}_n) \geq f_{\hat{R}}(\hat{u}_1, \dots, \hat{u}_n) | U \sim S) = \mathbb{P}(f_{\hat{X}}(\hat{z}_1), \dots, \hat{z}_n) \geq f_{\hat{Y}}(\hat{z}_1, \dots, \hat{z}_n) | U \sim S).$$

Since additionally $U \sim S \Leftrightarrow Z \sim X$ we arrive at

$$\mathbb{P}(f_{\hat{S}}(\hat{u}_1, \dots, \hat{u}_n) \geq f_{\hat{R}}(\hat{u}_1, \dots, \hat{u}_n) | U \sim S) = \mathbb{P}(f_{\hat{X}}(\hat{z}_1), \dots, \hat{z}_n) \geq f_{\hat{Y}}(\hat{z}_1, \dots, \hat{z}_n) | Z \sim X). \tag{3.22}$$

In the same manner, notice that

$$\begin{aligned} \mathbb{P}(f_{\hat{S}}(\hat{u}_1, \dots, \hat{u}_n) \geq f_{\hat{R}}(\hat{u}_1, \dots, \hat{u}_n) | U \sim R) &= \mathbb{P}(f_{\hat{X}}(\hat{z}_1), \dots, \hat{z}_n) \geq f_{\hat{Y}}(\hat{z}_1, \dots, \hat{z}_n) | Z \sim Y), \\ \mathbb{P}(f_{\hat{S}}(\hat{u}_1, \dots, \hat{u}_n) < f_{\hat{R}}(\hat{u}_1, \dots, \hat{u}_n) | U \sim S) &= \mathbb{P}(f_{\hat{X}}(\hat{z}_1), \dots, \hat{z}_n) < f_{\hat{Y}}(\hat{z}_1, \dots, \hat{z}_n) | Z \sim X), \\ \mathbb{P}(f_{\hat{S}}(\hat{u}_1, \dots, \hat{u}_n) < f_{\hat{R}}(\hat{u}_1, \dots, \hat{u}_n) | U \sim R) &= \mathbb{P}(f_{\hat{X}}(\hat{z}_1), \dots, \hat{z}_n) < f_{\hat{Y}}(\hat{z}_1, \dots, \hat{z}_n) | Z \sim Y). \end{aligned}$$

This is a convenient result, since these probabilities are known and given in section 3.5.1. Recalling equation (3.12), this implies that

$$\begin{aligned}
\mathbb{P}(C^{\text{G, drift}}) &= \mathbb{P}(f_{\hat{S}}(\hat{u}_1, \dots, \hat{u}_n) \geq f_{\hat{R}}(\hat{u}_1, \dots, \hat{u}_n) | U \sim S) + \\
&\quad \mathbb{P}(f_{\hat{S}}(\hat{u}_1, \dots, \hat{u}_n) < f_{\hat{R}}(\hat{u}_1, \dots, \hat{u}_n) | U \sim R) \\
&= \mathbb{P}(f_{\hat{X}}(\hat{z}_1), \dots, \hat{z}_n) \geq f_{\hat{Y}}(\hat{z}_1, \dots, \hat{z}_n) | Z \sim X) + \\
&\quad \mathbb{P}(f_{\hat{X}}(\hat{z}_1), \dots, \hat{z}_n) < f_{\hat{Y}}(\hat{z}_1, \dots, \hat{z}_n) | Z \sim Y) \\
&= \mathbb{P}(C^{\text{A, drift}}).
\end{aligned}$$

Empirical results
Again the computed theoretical maximum accuracy is compared to the empirical results obtained using the proposed solution as explained in section 3.4. As mentioned before, the precise structure of the empirical classification process is revealed in chapter 4, this section merely serves as an initial justification of the proposed method. Suppose $X_0 = Y_0 = Z_0 = 100$, $\sigma = 0.2$ and $n = 1000$, i.e. a path with 1000 time steps. 10,000 paths are generated per model and a random forest classifier is used. The data is split in a 70% training set and a 30% testing set. In table 3.3 the theoretical maximum classification accuracy is compared with the accuracy obtained by the signature method proposed in this thesis. The displayed empirical accuracy is computed over the test set.

μ_x	μ_y	Theoretical maximum accuracy	Obtained empirical accuracy
0.1	0.15	0.654	0.629
0.1	0.2	0.785	0.772
0.1	0.25	0.882	0.877

Table 3.3: Theoretical and empirical classification accuracy for GBM with different drift.

Again the results show a close approximation of the theoretical maximum accuracy and are indeed similar to those in table 3.1, showing that distinguishing between GBMs with different drifts is complicated when the difference between the drifts is small.

3.5.4. Discrete GBM: different volatilities

Let again $\xi_k \overset{i.i.d.}{\sim} N(0,1)$ for all $k \in \{1, \dots, n\}$ and U_k follow a GBM represented by equation (3.16). Now assume a path following GBM with n time steps, i.e. observe a sample $u = (u_1, \dots, u_n)$ from $U = (U_1, \dots, U_n)$. Let $S = (S_1, \dots, S_n)$ and $T = (T_1, \dots, T_n)$ follow an GBM with equal drift μ and volatility σ_x, σ_y respectively.

Notice that with a similar derivation as in section 3.5.3, one arrives at

$$\mathbb{P}(C^{\text{G, vol.}}) = \mathbb{P}(C^{\text{G, vol.}}).$$

Empirical results
In this section we compare the results above with our empirical findings, using the same setting as in tables 3.1- 3.3.

σ_x	σ_y	Theoretical maximum accuracy	Obtained empirical accuracy
0.1	0.105	0.862	0.822
0.1	0.11	0.983	0.961
0.1	0.2	1.0	1.0
0.1	0.3	1.0	1.0
0.1	0.4	1.0	1.0

Table 3.4: Theoretical and empirical classification accuracy for classification between GBM with different volatility.

Again a close relation between the empirical accuracy and the maximum possible accuracy is observed and we see that distinguishing between GBMs with different volatilities results in higher classification accuracies than distinguishing GBMs with different drifts as shown in table 3.3.

3.5.5. Conclusion

In summary, the maximum theoretical binary classification accuracy for ABM and GBM with different drifts or different volatilities is given by:

$$
\begin{aligned}
\mathbb{P}(C^{\text{A, drift}}) &= \frac{1}{2} + \frac{1}{2}\Big(\Phi\Big(\frac{\sqrt{T}}{2\sigma}(\mu_y - \mu_x)\Big) - \Phi\Big(\frac{\sqrt{T}}{2\sigma}(\mu_x - \mu_y)\Big)\Big), && (3.23)\\
\mathbb{P}(C^{\text{A, vol.}}) &= \frac{1}{2} + \frac{1}{2}\Big(\chi^2_n\Big(\frac{2n\sigma_y^2 \log(\frac{\sigma_x}{\sigma_y})}{\sigma_x^2 - \sigma_y^2}; n\Big) - \chi^2_n\Big(\frac{2n\sigma_x^2 \log(\frac{\sigma_x}{\sigma_y})}{\sigma_x^2 - \sigma_y^2}; n\Big)\Big), && (3.24)\\
\mathbb{P}(C^{\text{G, drift}}) &= \mathbb{P}(C^{\text{A, drift}}), && (3.25)\\
\mathbb{P}(C^{\text{G, vol.}}) &= \mathbb{P}(C^{\text{A, vol.}}), && (3.26)
\end{aligned}
$$

where $\Phi(\cdot)$ denotes the cdf of the standard normal distribution and $\chi^2_n(\cdot)$ denotes the cdf of the χ^2 distribution with n degrees of freedom.

An important conclusion based on sections 3.5.1 - 3.5.4 is that it is significantly easier to distinguish between ABMs or GBMs with different volatilities than it is to distinguish between ABMs or GBMs with different drifts. It is essential to keep this in mind when assessing the results obtained in chapter 5.

4 Methodology

In this chapter, the methods used for the classification of the signature of a path are described, along with some mathematical properties of the methods. Additionally, the pre-processing and dimension reduction techniques are described.

In practice, multiple actions are needed in order to find the signature of a path. As described in [13], the workflow can be summarised as

$$data \overset{I}{\rightarrow} path \overset{II}{\rightarrow} signature\ of\ path \overset{III}{\rightarrow} features\ of\ path$$

I Since financial data streams consist of discrete elements and signatures are computed on continuous paths, an adjustment to the data must be made in order to apply the transformation. Though many ways exist to transform discrete elements to a continuous path, the most straightforward way is to connect the elements linearly, creating a piece-wise linear path.

II The signature is computed by integrating multiple times along the piece-wise linear path.

III Not all features are equally important, the most relevant information is stored in the first 4 elements. A possibility is to use lasso regression or Principle Component Analysis (PCA) to perform feature selection and dimension reduction.

4.1. Pre-processing

Before the data can be used as input for the classifier, it needs to be pre-processed. Section 4.1.1 explains the case when there are two models between which the classifier has to differentiate. The extension to multiple models is mostly straightforward, but issues that arise are briefly mentioned and dealt with in section 4.1.2.

4.1.1. Binary classification

First, the paths have to be generated. Generate N paths per model, where every path consists of M time steps. This gives a data set which can be stored in a matrix $X \in \mathbb{R}^{2N\times M}$. Attach to every path the label of the model it belongs to: either 0 or 1 and save this in the vector y. The classifier is a certain function $f : X \mapsto y$ where

$$X = \begin{bmatrix} X_1^1 & X_2^1 & \dots & X_M^1 \\ X_1^2 & X_2^2 & \dots & X_M^2 \\ \vdots & \ddots & \ddots & \vdots \\ X_1^N & X_2^N & \dots & X_M^N \\ X_1^{N+1} & X_2^{N+1} & \dots & X_M^{N+1} \\ X_1^{N+2} & X_2^{N+2} & \dots & X_M^{N+2} \\ \vdots & \ddots & \ddots & \vdots \\ X_1^{2N} & X_2^{2N} & \dots & X_M^{2N} \end{bmatrix}, \quad y = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}.$$

Hereafter the following steps are performed:

- Perform a lead-lag transformation on X (optional).
- Transform the paths to their signatures or log signatures, depending on which is more suitable for the situation. In `Python`, the package `esig` [36] is available that performs this transformation.
- Shuffle the paths (row-wise).
- Standardise the features (column-wise, optional).
- Split the matrix (row-wise) in a training part (70%) and a test part (30%).

4.1.2. Multi-class classification

The extension to multi-class classification is straightforward, except for the label encoding of the output, where several approaches are available. When dealing with multiple classes, one can choose the output (labels) to either be label-encoded or one-hot encoded. The illustration below explains the difference between the two encodings.

0
1
2
1
2
0
⋮

1	0	0
0	1	0
0	0	1
0	1	0
0	0	1
1	0	0
⋮	⋮	⋮

Table 4.1: Suppose there are three classes. On the left is an example of label-encoding, every sample is assigned a class label out of $\{0, 1, 2\}$. On the right is an example of one-hot encoding. It has three columns - one per class - where per row the number 1 is placed in the column of the correct class, all other elements of the row equal zero.

The underlying algorithm works the same, regardless of which encoding is used. However, the advantage of using one-hot encoding is that it is a simple technique to quickly get an intuition on the confidence of the classifier of the class of the sample. As an example, compare two potential outputs of a classifier in table 4.2.

0.99	0.005	0.005
0.10	0.85	0.05
0.03	0.07	0.90
⋮	⋮	⋮

0.36	0.30	0.34
0.25	0.40	0.35
0.30	0.27	0.43
⋮	⋮	⋮

Table 4.2: Two potential outputs of a classifier distinguishing between three classes. The column number of the maximum per row is the class as which the sample is classified. Both give exactly the same number of correct classifications, but the output on the left is more certain than output on the right. Obtaining this information about the classification is useful in further analysis.

In other words, one-hot encoding allows for output which can be interpreted as similar to a ‘probability distribution’ over the classes, which is interesting information. It is relevant to notice that this information about the confidence of the prediction is also available for label-encoded output by going back one processing step in the classifier (see 4.3 for an overview of the classifiers), but using one-hot encoding the information is immediately visible in the output. For this reason, the output is one-hot encoded in this thesis.

4.2. Feature selection and feature extraction

Once the data has been pre-processed and transformed to signatures, the training of the classifier can begin. One of the problems that immediately becomes apparent, is that the truncation level of the signature needs to be determined.

As mentioned before, the untruncated signature completely determines the path. Truncating this sequence therefore inescapably leads to a loss of information, indicating that a high truncation level might be preferable. This is not entirely the case, though, since a higher truncation level leads to more elements in the signature sequence, which means that the classifier has a higher dimensional feature input. The signature elements with the highest impact are located at the beginning of the signature sequence [10]. Intuitively, this is similar to Taylor expansion, where the terms in the start of the expansion have a greater impact than later terms. Dimensionality reduction techniques can be used to combine features (feature extraction) or remove features (feature selection).

4.2.1. Feature selection

Even though according to [10] the most relevant characteristics of the path are at the start of the signature sequence, it is desirable to have an indication of which features are important in the classification between paths generated by different models, so that the level of truncation can be chosen in such a way that these important features are included.

The topic of feature selection has been extensively studied over the years; [12] gives a valuable overview of the different methods to achieve feature selection. According to [12], three types of feature selection methods exist:

1. **Filter methods:** filter methods rank the features from most relevant to least relevant based on a certain relevance criterion.

2. **Wrapper methods:** wrapper methods go over all subsets of features to determine the optimal classification accuracy. The idea behind this method is that irrelevant features introduce noise and therefore reduce the classification accuracy. Going over all subsets of features quickly becomes NP-hard, so often special heuristics are used, see [12].

3. **Embedded methods:** embedded methods include feature selection as part of the training process.

Since wrapper methods are computationally expensive and embedded methods embed feature selection in the classification process and therefore do not explicitly give the important features, the focus is put on filter methods.

Several filter methods have been tried in this thesis. Think of the χ^2-test to test the statistical significance of the feature in relation to the class label, Wald z-statistic, which has as null-hypothesis that the coefficient of the feature equal zero, Lasso regression etc. However, for all methods the results were ambiguous. Running the classifier multiple times led to different features being identified as important, for all classifiers mentioned in section 4.3.

The reason that the filter methods mentioned above failed to indicate the important features is because of the *multicollinearity* of the features. Multicollinearity occurs when two or more features are highly linearly related, see definition 4.1.

Definition 4.1 (Multicollinearity). *Variables $X_1, \dots, X_n$ are said to be perfectly multicollinear if there exist $\lambda_0, \dots, \lambda_n \in \mathbb{R}$ such that*

$$\lambda_0 + \lambda_1 X_{1i} + \cdots + \lambda_n X_{ni} = c, \quad (c \in \mathbb{R})$$

holds, for every i^{th} observation of the j^{th} variable X_{ji}.

An easy way to check for multicollinearity is to compute the correlation between every pair of features. In general, one speaks of multicollinearity if the correlation between the features is higher than 0.7. In the figure below the correlation between the features is shown, where the signatures are used as features.

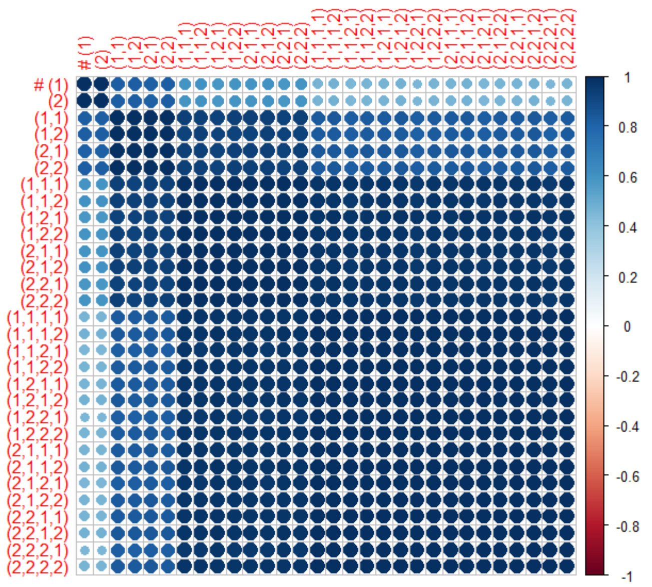


Figure 4.1: Correlation between the signature elements truncated at level 4 of a GBM. The colour indicates the correlation between the two signature elements. Correlation plots of other type of processes look similar.

As can be seen in figure 4.1, many features are almost perfectly correlated.

Multicollinearity among features does not impact the classification accuracy, but it makes the identification of important features complex. The reason for this can best be explained through example 4.1.

Example 4.1. *Let $X_1, X_2 \in \mathbb{R}$ be two features used to predict the output $Y \in \mathbb{R}$ through a simple, linear model:*

$$Y = w_1X_1 + w_2X_2.$$

Here w_1, w_2 can be interpreted as the weights of the features, with $0 \leq w_1, w_2 \leq 1$ and $w_1 + w_2 = 1$. Assume X_1 and X_2 are perfectly (Pearson) correlated with $\rho(X_1, X_2) = 1$. It indicates that one can express X_1 as $X_1 = \alpha X_2 + \beta$. For simplicity, assume $\alpha = 1, \beta = 0$, so $X_1 = X_2$. When determining whether X_1 or X_2 is more important for determining the output Y, one could look at the weights. However, in this case the weights do not give any information. The weight combinations $\{0, 1\}, \{1, 0\}, \{\frac{1}{2}, \frac{1}{2}\}$ all give the same value output Y since $\rho(X_1, X_2) = 1$. It is not possible to determine whether X_1 or X_2 is more important.

The problem described in example 4.1 extends in the same manner to the feature selection problem in the classification process. High multicollinearity of the features is undesirable, since it makes the identification of important features unfeasible.

Feature selection is therefore not useful in this case to reduce the dimensionality of the input. Therefore the next paragraph looks into the concept of *feature extraction*.

4.2.2. Feature extraction

Feature extraction methods do not focus on selecting important features, but rather combine features together to reduce dimensionality while keeping the most important information. [21] gives a profound, though slightly outdated, overview of the possible methods. Many methods exist, ranging from classical methods like Principle Component Analysis (PCA) or Singular Value Decomposition (SVD) to deep learning methods where extracting features from the data is done with neural networks. For the purpose of this thesis a simple, fast feature extraction method is preferable, since the goal is to reduce the dimensionality for a as high as possible truncation level of the signature. Since PCA is simple, fast and works well on multicollinear data [21], this method is chosen.

Principle Component Analysis (PCA)
PCA is a non-parametric, statistical technique often used to reduce the dimensionality of a data set consisting of many interrelated variables, while preserving as much variance as possible. This is done by projecting the features on its *principle components*. These principle components are uncorrelated and ordered from components with most variance to least variance. The following explanation is based on [27].

Consider an $N \times M$ matrix X of input data, where every row represents one observation and the columns represent the features.Let $\mathbf{x}_i \in \mathbb{R}^M$ be one observation. Suppose the features are closely correlated. Figure 4.2 gives an example for $M = 2$.

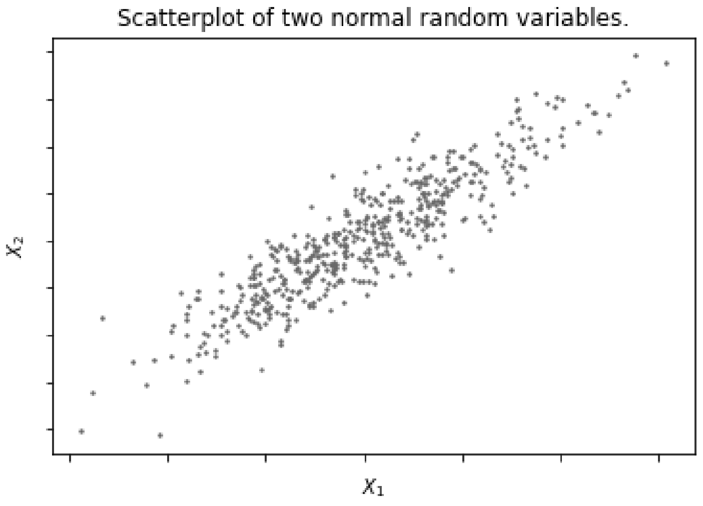


Figure 4.2: Scatterplot of two correlated random variables.

The first step is to look for the direction in the coordinate system for which the variance is maximum, i.e. look for $\alpha_1^T\mathbf{x}_i$ with $\alpha_1 = (\alpha_{11}, \ldots, \alpha_{1M})$ such that the variance of $\alpha_1^T\mathbf{x}_i$ over observations i is maximised. The next step is to look for $\alpha_2^T\mathbf{x}_i$ where $\rho(\alpha_1^T\mathbf{x}_i, \alpha_2^T\mathbf{x}_i) \approx 0$ where the variance is maximised over observations i. Here ρ denotes the correlation. Continuing this results in M linear functions $\alpha_j x$ which are uncorrelated. In this way, a coordination transformation has been made to a space where the features are uncorrelated, where $\alpha_j x$ is called the j^{th} principle component.

An example of the principle components of the data set in figure 4.2 is given in figure 4.3.

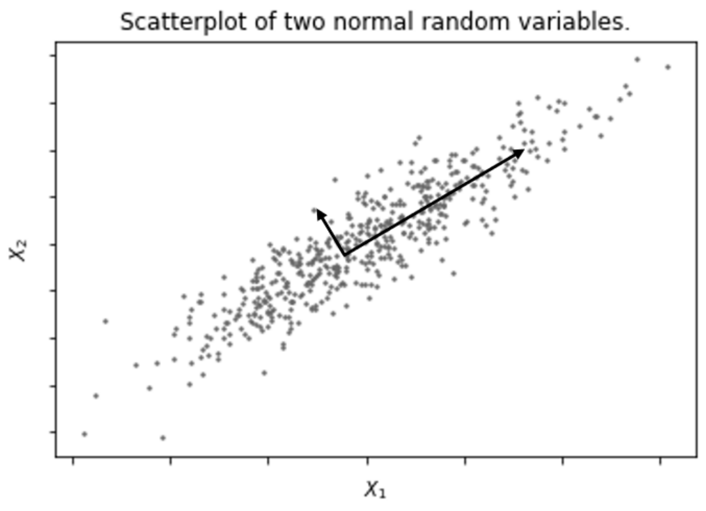


Figure 4.3: Coordination transformation from the X_1, X_2 axes to the new axes (called principle components).

The dimension reduction takes place by removing the last few principle components. Since the components are ordered by variance, these components contain the least variance and therefore the least information about the data set. For further information about PCA, the reader is referred to [27].

Remark 10. *In this thesis, PCA is only used as a pre-processing technique for the random forests, not for the neural network. The reason for this is that neural networks are able to handle multicollinear and high-dimensional input data, so using PCA in pre-processing is not necessary.*

4.3. Classification

Now that the data is processed and the dimensionality-reduction techniques are introduced, the classification can start. In this thesis two classifiers are used:

- **Random forests:** this method is included because it is fast to implement and train.
- **Neural network:** this method gives for many classification settings the best accuracy, but is slow to train and requires a considerable amount of hyper-parameter tuning.

Both classifiers have known advantages and disadvantages. Random forests is a machine learning technique for which is known that it works well for sparse data and for input data of which the dimension is moderate. When the dimension is high, dimensionality reduction techniques like the previously mentioned PCA need to be applied on the input data before the random forest algorithm can be applied. Additionally, for data which is not multicollinear, it can often be used to determine the feature importance, which is helpful for interpreting the classification problem. Unfortunately, as mentioned in section 4.2, the elements of the signature are multicollinear, therefore the advantage does not hold within this thesis. Compared to the neural network, it requires less training data. Its main advantage is that it is a fast method, both to implement and to train and that it generally performs well when compared to other machine learning techniques [17].

Neural networks are known to perform well when a lot of training data is available. Since the training data in this thesis is generated by the user, as much training data as required can be generated, which means that the neural network can be used. Furthermore, neural networks tend not to suffer from the earlier mentioned ‘curse of dimensionality’ [4], meaning that the level of truncation of the signature could optionally be chosen higher than for random forest.

4.3.1. Random forests

Random forests are a combination of decision trees. Originally proposed by Ho in 1995 in [24] and extended by Breiman in 2001 [5], it quickly gained popularity because of its flexibility (it can be used for both classification and regression problems), its practicality (it is a fast method with few hyper-parameters) and the advantage that the default hyper-parameter setting often already give quite good results. In this thesis the popular package `sklearn` in `Python` is used for the implementation.

A random forest is a collection of independent tree-structured classifiers that each get a vote on the classification. A schematic representation is shown in the figure 4.4.

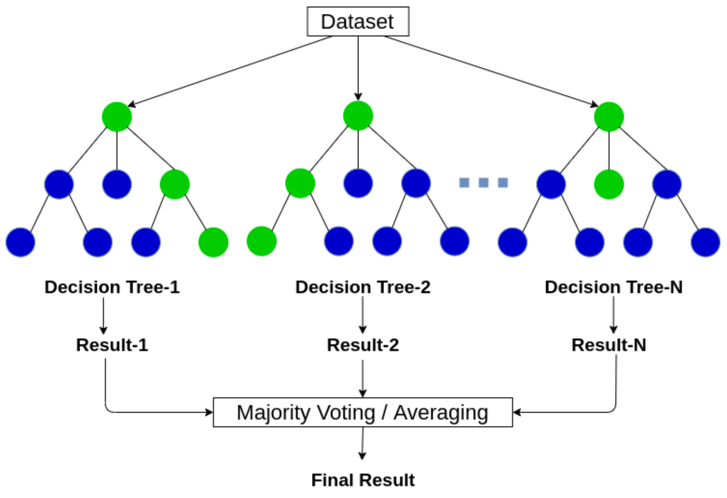


Figure 4.4: Schematic representation of a random forest [42].

From the input, the algorithm forms bootstrapped clusters (drawn with replacement), which serves as input for the individual trees. Note that the individual trees are therefore trained independently from each other. The advantage of this is that it corrects for decision trees' habit of overfitting to their training set.

A decision tree is a training algorithm that takes as input observations about an event and carries the observations to the event's target value (i.e. class label in this case). It is build up from nodes and branches, where at every node one of the features of the data is evaluated in order to split the input observations in the training process. The idea is that the input can be grouped into classes by observing the feature values. This grouping is done on the notion that observations with similar feature values should be classified as being from the same class.

The decision tree works from the top down, where at every level a feature is chosen that best splits the set of observations. A decision is made to which direction to go based on a metric. Gini impurity is an often used metric for this.

The gini impurity is a metric proposed by Breiman et al in [6] to measure the quality of a split in data, in order to find the best possible split.

Definition 4.2 (Gini impurity)**.** *Recall that S is the matrix containing the signatures of every path and let $p_i = p(i|S)$ be the proportion of signatures belonging to class i, where $i \in \{1, ..., k\}$ is the class label. The gini impurity is defined as*

$$I_{gini}(S) = \sum_{i=1}^{k} p_i(1 - p_i).$$

It measures how frequently a random sample from S would be classified incorrectly if it was randomly labelled according to the distribution of labels in the subset. The Gini impurity obtains its minimum value zero when all elements of the subset are from the same class.

Summarising, the algorithm looks like this:

Algorithm 1 Random forest classification

```
1: Create clusters of the input data by bootstrapping from the data (drawing with samples with replacement). Nodes are created such that the mean squared error in the resulting branches is minimized.
2: The random forest algorithm generates a significant amount of these trees, resulting in a ”forest”. Note that each tree is different since the training data for each tree are chosen at random.
3: The validation set is used to evaluate how well the fitted model performs. The data from the validation set is put into the trees and the predictions from all trees are averaged for a final random forest prediction.
```

By splitting the data based on these features we try to minimise the variance in the resulting branches. For this purpose the quality of every node is assessed by calculating how much the node reduces the mean squared error compared to the original branch. A node in a tree is the point where the path splits into two branches.

A random forest model consists of a large number of individual decision trees. Each individual tree in the random forest gives a prediction for a certain data set and the final result of the random forest model is the average of all these predictions.

4.3.2. Artificial Neural Networks

This section gives a brief introduction to Artificial Neural Networks (ANNs), explaining the basic structure and purpose. For a complete overview, the reader is referred to [4], on which this section is based. This section describes the basic neural network called the ’feed-forward neural network’. Note that many adaptations of the feed-forward neural network exist, of which [3] gives a brief literature review. The implementation of the neural network is done with `keras`, which is a package in `Python`.

Architecture

ANNs are deep learning methods which have gained massive popularity in the past 20 years, as they can be used for a broad range of applications in regression and classification. An ANN consists of multiple connected *layers* where only the input and output layer are visible to the user. The other layers in between the input and output layer are not visible and are therefore called the *hidden layers*. Every layer consists of a number of *neurons* N_i.

Let $x_1, ..., x_D$ denote the input. A neural network with one hidden layer, containing one neuron N is depicted below.

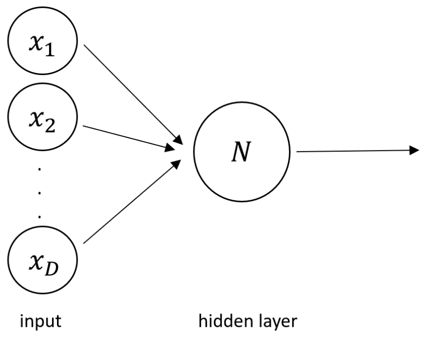


Figure 4.5: Schematic representation of a neural network with one hidden layer, containing one neuron.

Within neuron N two actions are performed:

- Linear combination with weights w_j: $k := \sum_{i=j}^{D} w_j x_j + w_0$. w_0 is commonly referred to as the *bias*.
- Activation function $h(k)$ where $h : \mathbb{R} \to [a,b]$. The activation function is typically chosen to be non-linear and in such a way that $[a,b]$ is either $[-1,1]$ or $[0,1]$. Additionally, $h(k)$ is required to be differentiable with respect to k.

A neural network is essentially a network of several layers consisting of neurons, where every neuron in layer i is connected to every neuron in the adjacent layer $i+1$ through weights $w_j^{i+1}, j \in \{1, ..., l_{i+1}\}$. Here l_{i+1} denotes the number of nodes in layer $i+1$. Commonly the number of hidden layers is referred to as the *width* of the network and the number of neurons per layer is referred to as the *depth* of the network. A schematic representation is depicted in figure 4.6.

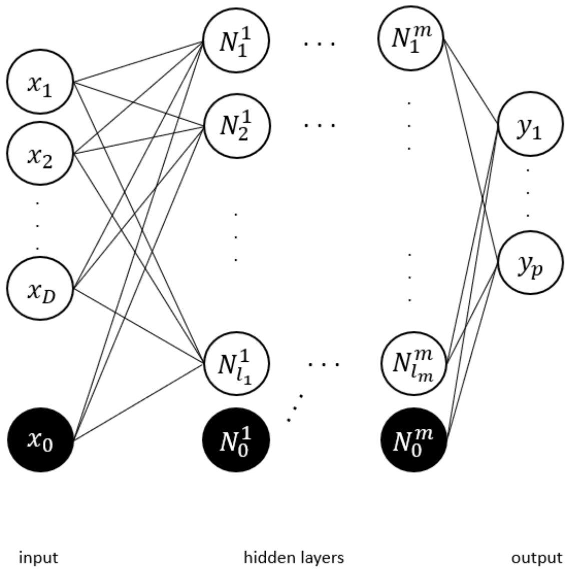


Figure 4.6: Schematic overview of a feed-forward neural network with m hidden layers. Layer N^i has l_i nodes per layer. The black neurons $x_0, N_0^1 \dots N_0^m$ are commonly set equal to 1 and are multiplied by the bias w_0^i such that $N_0^i w_0^i = w_0^i$.

Intuitively one can picture the information passing through the network from the input layer to the output layer, hence the name *feed-forward neural network*.

The neural network used in this thesis is a feed-forward network with the specifications as listed below.

- Number of hidden layers: 3.
- Number of neurons per hidden layer: 1024, 512, and 256 respectively.
- Number of neurons in input layer: D. This is required to be equal to the dimension of the input and therefore depends on the level of truncation of the signature sequence.
- Number of neurons in the output layer: p. This is equal to the number of classes between which one attempts to classify.
- Value of bias neurons: 1.

The architecture is schematically represented in figure 4.7.

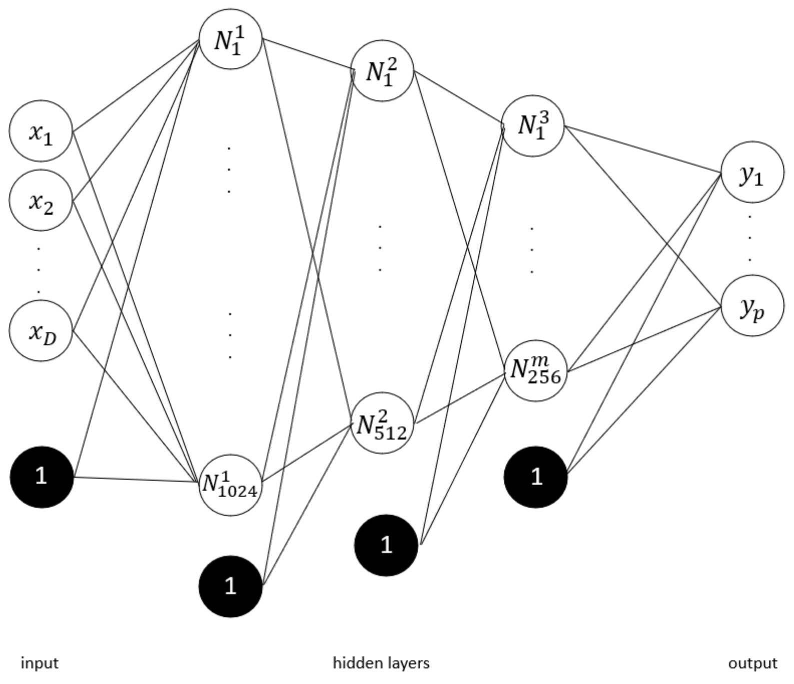


Figure 4.7: Schematic representation of the ANN used in this thesis.

Pre-processing: no PCA
Recall from remark 10 that PCA is not used as pre-processing technique when using the neural network. The reason for this is that, according to [15], multicollinearity does not have a big impact on neural networks, since the final output of the neural network is a combination of many combinations of the (non-linear) activation functions. This means that the original (multicollinear) input data is subjected to so many non-linear transformations, that the multicollinearity does not play an important role on the weights.

Choice of activation function
The activation function $h(\cdot)$ is chosen by the user. As mentioned before, $h(k)$ needs to be differentiable with respect to k (in order to perform backward propagation in the training phase, see next paragraph). Additionally, a non-linear activation function ensures that non-linear relations between the input and output can be learned. The activation functions do not need to be identical for every layer, but common practice is to have the same activation function for all hidden layers and optionally a different activation function for the output layer. The choice of activation function depends on the purpose of network and the type of data. For a rigorous overview of commonly used functions, the reader is referred to [4].

The activation function chosen for the hidden layers in the neural network used in this thesis is $h(x) = \tanh(x)$ with

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

Notice that this is indeed a differentiable and non-linear function. The activation function for the output layer is standard in classification, where the sigmoid function $h(x) = \Sigma(x)$ with

$$\Sigma(x) = \frac{1}{1 + e^{-x}}$$

is chosen for binary classification and the softmax function $h(\mathbf{x}) = \text{softmax}(\mathbf{x})$ with

$$\text{softmax}(\mathbf{x}) = \frac{e^{x_i}}{\sum_{j=1}^{p} e^{x_j}} \quad \text{for } i \in \{1, \dots, p\}$$

is chosen for multi-class classification with p classes. The softmax function is chosen because it normalises the input of the output layer to values in [0,1] which sum up to 1, meaning that this can be interpreted as a probability distribution over the classes. When analysing the classification accuracy, the predicted label is chosen for which the probability is maximal, i.e.

$$\arg \max_{i \in \{1,\dots,p\}} y_i.$$

Training: backpropagation
A crucial detail overlooked until now is the value of the weights $\mathbf{w}^i = [w_0^i, \dots w_{l_i}^i]$. Naturally before one can pass the input $x_1, \dots, x_D$ through the network, one needs to know the weights with which the values are multiplied in the linear combination within each neuron. These weights are determined through the training process. As is common in statistical learning, one splits the data in a training set and a test set. For the training set several instances of input $x_1, \dots, x_D$ and target output $t_1, \dots, t_p$ are known and a method called *backpropagation* is used to iteratively determine the weights of the network which minimise a specific loss function $L(\mathbf{w}^i)$ set by the user for all $\mathbf{w}^i$. This loss function measures the difference between the output $y_1, \dots, y_p$ of the network and the actual (known) target values $t_1, \dots, t_p$. The choice of loss functions is broad and again depends on the purpose of the network, but it is required to be a smooth, continuous function of $\mathbf{w}$. For classification the cross entropy loss function is regularly used [4]. The cross entropy loss function to be minimised is defined as:

$$CE = -\sum_{i=1}^{p} t_i \log(y_i).$$

Recall that $t_i \in \{0,1\}$ is the target label and $y_i \in [0,1]$ is the output of the softmax function, as that is done in the last layer of the network. Ideally, y_i is close to 1 when $t_i = 1$ and y_i is close to 0 when $t_i = 0$. Notice that CE indeed attains its minimum when $y_i = t_i$ for all $i \in \{1, \dots, p\}$.

Backpropagation is a method used in deep learning to determine the weights in the network. Intuitively it starts from the error measured by the loss function at the output of the network and passes through the network to its input. To illustrate the concept, notice that when a small step is made from weights $\mathbf{w}$ to $\mathbf{w} + \Delta\mathbf{w}$ where $\Delta\mathbf{w}$ is the update of the weight vector (defined differently for every choice of optimisation algorithm), the loss changes with $\delta L \approx \Delta\mathbf{w}^T \cdot \nabla L(\mathbf{w})$, where ∇ is used to denote the gradient. $\nabla L(\mathbf{w})$ points in the direction where the increase of the loss function is greatest, meaning that moving in the direction of $-\nabla L(\mathbf{w})$ will decrease the loss. Notice that this means that the loss is minimal for $\mathbf{w}$ such that

$$\nabla L(\mathbf{w}) = 0. \tag{4.1}$$

The goal is to find $\mathbf{w}$ such that the gradient vanishes, but this is hindered by the typically non-linear dependence of the loss function on the weights. An analytical expression for $\mathbf{w}$ such that (4.1) holds is seldom available, so practitioners resort to numerical methods to find this $\mathbf{w}$. A wide variety of methods are available, but most methods start with an initialisation $\mathbf{w}^0$ and move through the weight space by

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} + \Delta\mathbf{w}^{\tau}, \tag{4.2}$$

where τ denotes the iteration step and Δ is often referred to as the *learning rate*.

Naturally the above is merely a short introduction to the complex training process of the neural network. For more details on this, the reader is referred to [4].

Hyper-parameter settings
Neural networks have many hyper-parameters, which makes it challenging to find the best performing set of hyper-parameters. An extra challenge in this thesis is that the network should be suitable to be used for many different classification settings. Usually, when learning data, networks are trained on one type of data and the hyper-parameters can be tuned to that type of data. In this thesis, we look at several different classification settings such as classifying between different drifts, classifying between different volatilities, classifying on the presence of jumps etc. It is not necessarily true that in each classification setting the same hyper-parameter setting will be the optimal setting. However,

since hyper-parameter tuning is time-consuming and since we would like to have a network which performs stable regardless of the models included in $\mathbb{M}$, we will search for the set of hyper-parameters that perform best in the classification settings observed in this thesis, as opposed to finding the best hyper-parameter settings for every situation individually.

Batches
Recall equation (4.2) and suppose we have one data input $\mathbf{x} = (x_1, \ldots, x_D)$. $\mathbf{x}$ is passed through the network and results in one observed value of $L(\mathbf{w})$. The weights are then updated in such a way that for the observed input $\mathbf{x}$ the loss decreases. However, introducing the training data like this, one by one, can result in unstable training, since the updated weights are decided on only one input data point. An alternative idea is to introduce the data to the network in larger batches B of k data points: $B = (\mathbf{x}_i, \ldots, \mathbf{x}_{i+k})$. This ensures that we have k observations of the loss instead of one, which allows the weights to be updated in such a way that the average loss observed over all k input values is reduced. In general a larger batch size results in a better updating of the weights and therefore in a more stable training process. However, the batch size is limited by the memory of your computer. Practitioners suggest using a batch size " between 1 and a few hundreds" [8]. In this thesis a batch size of 1000 has proven to be work well in most settings.

Number of epochs
The number of epochs is the number of times the full training data set is passed through the network. There is no analytical way to determine the best number of epochs, but the rule of thumb is to choose the smallest number of epochs for which the training accuracy stops increasing. However, in this thesis many different classification settings are observed. There is no single smallest number of epochs such that the training accuracy stops increasing for all classification settings analysed. However, choosing a number of epochs higher than the optimal number does not hurt the classification accuracy either, it merely increases the computation time. Figure 4.8 shows how the number of epochs influences the training accuracy for two different classification settings.

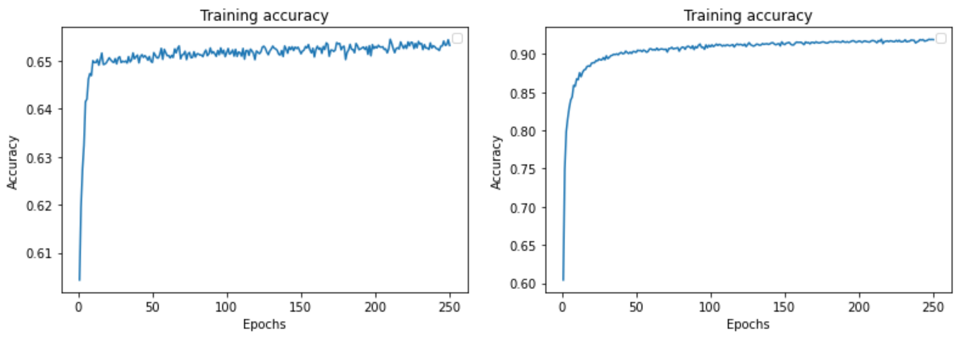


Figure 4.8: Influence of the number of epochs on the training accuracy. Left: classification between different drifts in GBM. Right: classification between different jump sizes in an exponential jump diffusion process.

Figure 4.8 shows that when classifying between different drifts in GBM (left picture), there does not seem to be an significant increase in accuracy after around 50 epochs. However, when classifying between different jump sizes in the jump diffusion process (right picture), the training accuracy is still increasing after 50 epochs (notice the different scale on the y-axis) and would suggest 150 or even 200 epochs to be sure. We would like to choose one single number of epochs for all settings to simplify the situation. The number of epochs is chosen as the maximum of the optimal number of epochs observed in the classification settings, which in this thesis was 200 epochs.

Dropout
Because neural networks typically have many parameters (i.e. weights), they are prone to *overfitting*. Overfitting occurs when models are too much tailored to the training data that they fail to make inferences on unseen test data. To circumvent this problem, a popular technique for neural networks is to use *dropout*. Dropout is a simple, but very effective technique, where during the training per layer a certain percentage of the output of the neurons is ignored or "dropped out". The neurons whose output

are ignored are chosen randomly and differ per every batch. In this way, the training of the network is hindered slightly, preventing overfitting of the network. The dropout percentage can differ per layer, though practitioners suggests to use a dropout of close to 0 for the input and output layer and a dropout of around 0.5 for the hidden layers [7]. Through empirical testing and observing the train- and testing error, we found that a dropout percentage of 0.3 for hidden layers and 0.0 for the visible layers works best in most settings tried in this thesis, and therefore these are the values chosen for the dropout.

Learning rate
Recall equation (4.2). Δ was defined as the *learning rate*, which denotes by how much the weights are updated after every iteration. Choosing the learning rate is a tight balance between computational speed and convergence, as illustrated in figure 4.9.

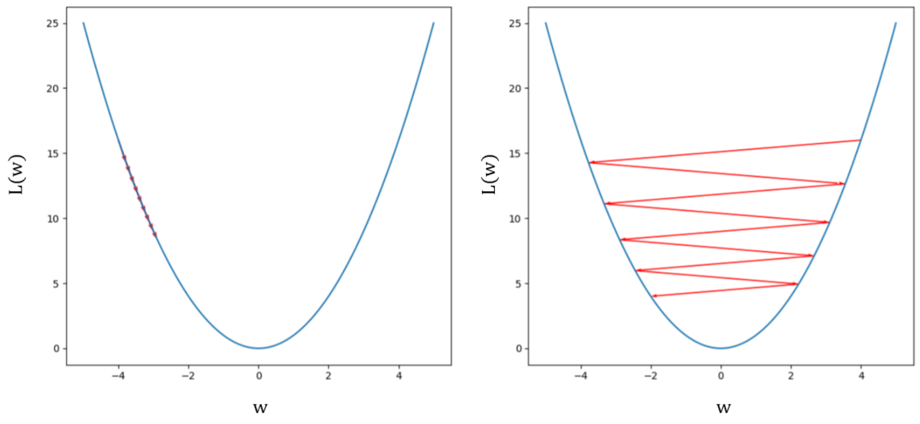


Figure 4.9: An illustration of how the learning rate influences the training. Left: low learning rate, right: high learning rate. Figure taken from [20].

Figure 4.9 illustrates that when attempting to find the weights that minimises the loss function, choosing a too high learning rate can lead to divergence (right figure) and choosing a too low learning rate will lead to convergence, but will become computationally intensive since more epochs are needed to lead to the same result. Typically the learning rate is chosen between $0.1 - 0.001$ [9]. In this thesis, a learning rate of 0.001 has proven not to lead to a diverging training process, while the computational time was still acceptable.

Universal approximation of the target function
An important concern that needs to be addressed before using a neural network is whether the neural network is descriptive enough to find the possibly non-linear relation between the input and the target. Fortunately, this concern is answered by the *universal approximation theorem*. The classical form of the universal approximation theorem provided in [25] states that a neural network with one hidden layer of arbitrary depth can approximate any relation between the input and output uniformly close. This theorem was extended by [33] which claims that a network of bounded depth and arbitrary width can also uniformly close approximate the relation between the input and the output. Though the framework with a finite depth and a finite width has not been proven to being able to do the same, these theorems - combined with the notion that computationally large networks of thousands of neurons or thousands of hidden layers are possible - provide a solid basis to promote the usage of neural networks. As mentioned at the start of this section, in this thesis the neural network equals or outperforms the other two classifiers in every setting. The universal approximation theorems are the likely reason for this.

5 Experiments

In this chapter the results of the several experiments conducted in this thesis are discussed. Recall that the purpose in this thesis is to classify observed time series to their likely generative stochastic process. As indicated before, the collection of stochastic processes is broad and the choice was made to focus on ABM, GBM and the exponential jump diffusion process. Since ABM and GBM are very similar and analytical results on ABM are available in section 3.5, in this chapter we focus on empirical results from:

- Geometric Brownian Motion,
- Exponential jump diffusion process.

These are processes following laws as described in chapter 2. In this chapter we aim to classify between these models, but also to classify between paths generated by the same model with different parameter settings, as this is useful for calibration purposes.

The chapter starts with an overview of the default settings used in the experiments throughout the chapter in section 5.1, followed by the results of several binary classification settings in section 5.2 and results of multi-class classification in section 5.3.

5.1. Settings

In the consecutive sections, the following settings are used unless specified otherwise.

- The data set is split in a 70% training set and a 30% test set. The classification accuracy listed are computed over the test set.
- All paths have 1000 equidistant time points and the total time horizon equals $T = 10$.
- All paths have equal starting point $X_0 = 100$.
- The signature transformation is used. Section 5.2.3 investigates the difference in classification accuracy between signatures and log signatures.
- The signature is truncated at level $L = 4$. Section 5.2.4 investigates the difference in classification accuracy between signatures truncated at different levels.
- The lead-lag transformation is used.
- When using random forests, PCA is applied using 25 components. When using the neural network, no PCA is applied and the network is trained directly on the signatures.
- The random forest is uses data sets of 10,000 paths per model. The neural network uses 100,000 paths per model.
- The labels of the paths are one-hot encoded, see section 4.1.2.

For a schematic overview of the classification setting, the reader is referred back to figure 3.8. Notice that the choice is made to apply the lead-lag transformation as the standard setting. This transformation (recall paragraph 3.3.4) captures the volatility parameter explicitly and has proven to work best in most experiments conducted in this chapter. In section 5.2.3 the effect of the lead-lag transformation on the classification accuracy is examined.

Computational note
Ideally, we would like to try a large number of different classification problems to give an complete overview of the efficiency of the signature method. Though in this chapter many classification problems are observed, we are limited in the number of experiments by the computational power available. For every experiment, many paths for all $M_i(\theta^i) \in \mathbb{M}$ need to be generated, which all need to be transformed to their lead-lag equivalent, over which the signature is computed. This means that many pre-processing computations are done before the input enters the classifier. Especially the computation of the signature is time consuming. The random forest classifier does not need a lot of training data, in this thesis it was found that the classification accuracy over the test set does not increase if the number of training samples is extended to more than 10,000 per model. The computational issue for this classifier is therefore minor. However, the neural network needs much more training data, usually around 100,000 paths per model at least. This has as consequence that analysing, for example, the influence of level of truncation on the classification accuracy can be problematic when using a neural network. For this reason, those inferences are all done by using the random forest in classification problems where the difference in accuracy between random forest and the neural network is negligible.

5.2. Binary classification

The first setting analysed is binary classification, i.e. classifying between two models. Binary classification is more straightforward to analyse than multi-class classification, so in addition to the results of classification between different stochastic processes in subsection 5.2.1, this section will also discuss how different combinations of parameters and models influence the classification accuracy.

5.2.1. Relevant classification problems

Many different binary classification settings can be considered, this section describes those settings that are most relevant in practice, see chapter 2.

Two GBM processes with different parameters
The first classification setting concerns two GBM processes $X^1(t), X^2(t)$ with different parameters, following the dynamics:

$$\begin{aligned} \mathrm{d}X^1(t) &= \mu_1 X^1(t)\mathrm{d}t + \sigma_1 X^1(t)\mathrm{d}W(t) \\ \mathrm{d}X^2(t) &= \mu_2 X^2(t)\mathrm{d}t + \sigma_2 X^2(t)\mathrm{d}W(t) \end{aligned} \tag{5.1}$$

Though $X^1(t), X^2(t)$ are assumed to follow the same model, performing this classification can help practitioners in the calibration of the parameters μ and σ.

Table 5.1 shows the classification accuracy computed for various values of μ_1, μ_2, σ_1 and σ_2. To give an impression of what the paths of the chosen parameters in table 5.1 look like, several paths for two settings are plotted in figure 5.1. More sample paths examples can be found in Appendix A.1.

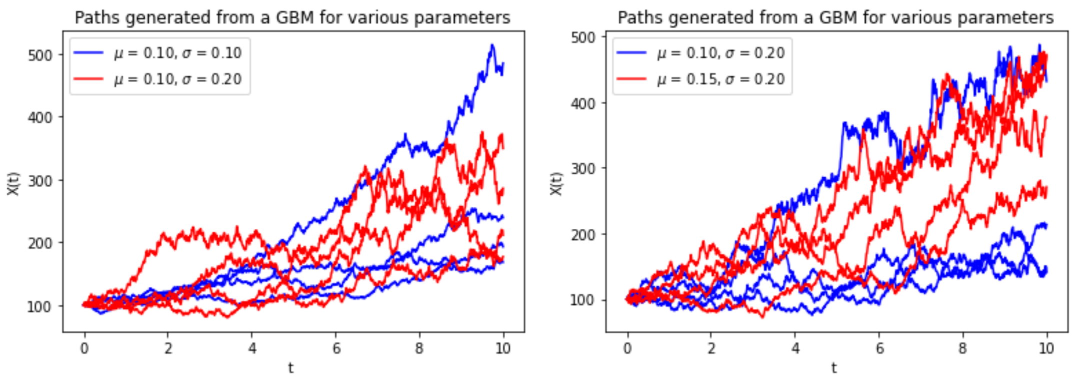


Figure 5.1: Overview of the classification between paths generated by two GBMs. Left: paths with different volatilities. Right: paths with different drifts.

Using the setting as described in 5.1, table 5.1 lists the classification accuracies for various parameter values.

Parameters				Accuracy		
μ_1	μ_2	σ_1	σ_2	Theoretical max. accuracy	Random forest	ANN
0.1	0.1	0.1	0.105	0.862	0.800 (0.026)	0.788 (0.025)
0.1	0.1	0.1	0.11	0.983	0.955 (0.015)	0.973 (0.018)
0.1	0.1	0.1	0.2	1.000	1.0 (0.000)	1.0 (0.001)
0.1	0.1	0.1	0.3	1.000	1.0 (0.000)	1.0 (0.001)
0.1	0.1	0.1	0.4	1.000	1.0 (0.000)	1.0 (0.000)
0.1	0.15	0.2	0.2	0.654	0.629 (0.025)	0.651 (0.025)
0.1	0.20	0.2	0.2	0.785	0.772 (0.021)	0.779 (0.017)
0.1	0.25	0.2	0.2	0.882	0.877 (0.013)	0.881 (0.016)

Table 5.1: Percentage of paths correctly classified for two GBM processes with different parameters. The random forest classifier has been trained once on 10,000 paths per model and tested 50 times on test sets of 600 paths. The neural network has been trained once on 100,000 paths per model and tested on the same 50 test sets of 600 paths per test. The results for random forest and the neural network are presented as mean accuracy over the 50 test sets, followed by the sample standard deviation of the accuracy in between brackets.

Notice that the first three rows in table 5.1 relate to GBMs with different volatilities and the last three rows relate to GBMs with different drifts. The results are the same as discussed previously in tables 3.3 and 3.4 and we see again that classification between different volatilities is significantly more straightforward than classifying between different drifts. Based on table 5.1, we can conclude that with the specific parameter settings chosen in section 5.1, distinction between GBMs with different drifts is reasonable when $|\mu_2 - \mu_1| > 0.1$ and distinction between GBMs with different volatilities is reasonable when $|\sigma_2 - \sigma_1| > 0.005$.

Both the random forest classifier and the neural network perform satisfactory, reporting classification accuracies that are close to the theoretical maximum. This demonstrates that, for this specific situation, using the signature method is successful in distinguishing between different models. However, in practice we would recommend using the theoretical optimal classification rule in this specific classification problem. This classification rule, as explained in section 3.5, attains the maximum possible classification accuracy and is computationally cheaper, as the GBM paths and their signature transformation do not need to be generated [1]

A GBM and an exponential jump diffusion process

In the next classification problem, paths from a GBM and an exponential jump diffusion process are classified. Unlike the previous setting, there is no analytical result available about the theoretical maximum

[1]Notice that the optimal classification rule is not suitable in more general classification problems, for example testing whether a distribution contains jumps (see next experiment). It is only suitable for calibration purposes.

classification accuracy. This classification problem is relevant, since it allows the user to determine whether the observed path stems from a GBM or from the exponential jump diffusion process, both of which are popular processes in practice. Consider the following setting:

$$\begin{aligned} \mathrm{d}X^1(t) &= 0.1X^1(t)\mathrm{d}t + \sigma_1 X^1(t)\mathrm{d}W(t), \\ \mathrm{d}X^2(t) &= 0.1X^2(t)\mathrm{d}t + \sigma_2 X^2(t)\mathrm{d}W(t) + JX^2(t)\mathrm{d}\chi(t), \end{aligned} \tag{5.2}$$

where $\chi(t)$ denotes the Poisson process (independent of the Wiener process) with intensity λ and let $J \sim N(0, \sigma_J)$.

Remark 11. *The parameters $\sigma_1, \sigma_2, \sigma_J$ and λ need to be chosen carefully in order to obtain a meaningful classification. Notice that $X^2(t)$ follows dynamics very similar to $X^1(t)$, the only difference being a term added to represent the jumps. Adding this extra term introduces extra overall volatility in the paths. We have already seen in table 5.1 that classifying between paths with different volatilities results in high classification accuracies. To avoid classification based on the overall volatility of the path, the following requirement is set:*

$$Var(X^1(t)) = Var(X^2(t)) \tag{5.3}$$

This ensures that the classification is purely based on the presence of jumps in the paths, instead of on the increased volatility.

Table 5.2 shows the classification results for parameters chosen such that (5.3) holds.

Parameters				Classifier	
λ	σ_1	σ_2	σ_J	Random forest	ANN
0.01	0.2	0.175	0.1	0.853 (0.019)	0.798 (0.0)
0.02	0.2	0.13	0.1	0.959 (0.011)	0.973 (0.0)
0.03	0.2	0.08	0.1	0.977 (0.007)	0.985 (0.0)
0.04	0.2	0.05	0.1	0.981 (0.008)	0.928 (0.0)
0.05	0.2	0.0	0.1	0.992 (0.004)	0.985 (0.0)

Table 5.2: Percentage of paths correctly classified for one GBM process and one jump diffusion process. The intensity λ increases and σ_2 is decreased in order to meet the demand in equation (5.3). The random forest classifier has been trained once on 10,000 paths per model and tested 50 times on test sets of 600 paths. The neural network has been trained once on 100,000 paths per model and tested on the same 50 test sets of 600 paths per test. The results for random forest and the neural network are presented as mean accuracy over the 50 test sets, followed by the sample standard deviation of the accuracy in between brackets.

Figure 5.2 shows several paths for one of the classification settings in table 5.2 to give an impression. Examples of sample paths from the other classification problems can be found in Appendix A.1.

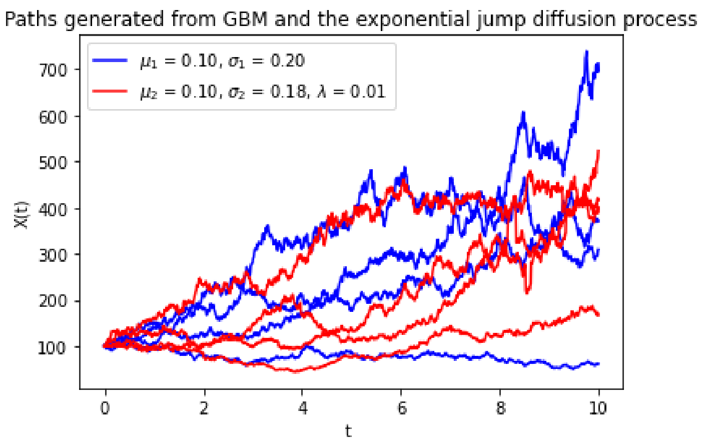


Figure 5.2: Paths generated by a GBM (blue) and an exponential jump diffusion process (red).

Table 5.2 shows the classification accuracy for an increasing intensity, i.e. an increasing expected number of jumps per path. The distribution of the jump size, governed by σ_J, remains equal, the overall volatility σ_2 is decreased in order to compensate for the increased intensity. Table 5.2 indicates that as the number of expected jumps per path increases, so does the classification accuracy. The results are intriguing, since they imply that the signatures are able to capture the difference between GBM and the exponential jump diffusion process through the presence or absence of jumps. This is an important result, since - though it was known that signatures can capture higher order statistics of a path - these relations could not be made explicit analytically. The results of table 5.2 imply that the presence of jumps (as governed by the exponential jump diffusion process) is one of the statistics that the signature can describe and that the signature method is suitable for distinguishing between paths generated by GBM and paths generated by the exponential jump diffusion process.

There is no clear preference for using a random forest or a neural network based on the classification accuracies in table 5.2. However, since random forest is faster in the training process and requires less training data, we would recommend to use the random forest in this classification setting.

Two exponential jump diffusion processes
In the next classification problem two exponential jump diffusion processes are compared with different intensities and jump sizes. Again no analytical maximum classification accuracy is available for this situation. Let

$$\begin{aligned} \mathrm{d}X^1(t) &= 0.1X^1(t)\mathrm{d}t + 0.2X^1(t)\mathrm{d}W(t) + J_1X^1(t)\mathrm{d}\chi^1(t), \\ \mathrm{d}X^2(t) &= 0.1X^2(t)\mathrm{d}t + 0.2X^2(t)\mathrm{d}W(t) + J_1X^2(t)\mathrm{d}\chi^2(t), \end{aligned} \tag{5.4}$$

where $J_1 \sim N(0, \sigma_{J_1})$, $J_2 \sim N(0, \sigma_{J_2})$, $\chi^1(t)$ is a Poisson process with intensity λ_1 and $\chi^2(t)$ is a Poisson process with intensity λ_2. Similar to remark 11, the parameters $\lambda_1, \lambda_2, \sigma_{J_1}$ and σ_{J_2} need to be chosen such that equation (5.3) holds. In table 5.3 the classification results are shown for increasing jump size in $X^1(t)$ and increasing intensity in $X^2(t)$.

Parameters				Classifier	
λ_1	λ_2	σ_{J_1}	σ_{J_2}	Random forest	ANN
0.01	0.02	0.15	0.1	0.603 (0.024)	0.658 (0.0)
0.01	0.041	0.2	0.1	0.719 (0.021)	0.830 (0.0)
0.01	0.082	0.3	0.1	0.853 (0.019)	0.913 (0.0)
0.01	0.103	0.4	0.1	0.844 (0.016)	0.928 (0.0)

Table 5.3: Percentage of paths correctly classified between two jump diffusion processes. The intensities λ_1, λ_2 and volatilities σ_1, σ_{J_2} are chosen order to meet the demand in equation (5.3). The random forest classifier has been trained once on 10,000 paths per model and tested 50 times on test sets of 600 paths. The neural network has been trained once on 100,000 paths per model and tested on the same 50 test sets of 600 paths per test. The results for random forest and the neural network are presented as mean accuracy over the 50 test sets, followed by the sample standard deviation of the accuracy in between brackets.

Figure 5.3 shows several paths for one of the classification problems in table 5.3 to give an impression. Examples of sample paths from the other classification problems can be found in Appendix A.1.

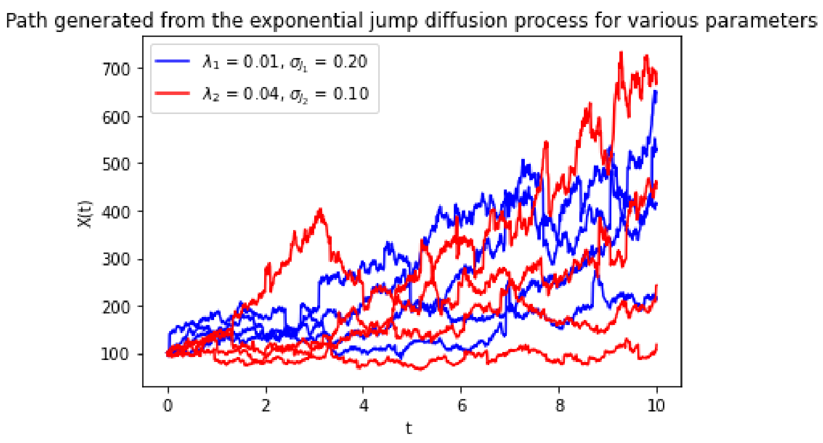


Figure 5.3: Paths generated by the exponential jump diffusion process for various parameters.

Table 5.3 shows that the classification accuracy increases as the jump size increases in $X^1(t)$ and the expected number of jumps per path increases in $X^2(t)$. This is in line with the expectation, as intuitively one expects that a path with a larger number of small jumps is more easily distinguishable from a path with higher (absolute) jump sizes. This result indicates that not only is the signature able to capture the whether jumps of the type as introduced by the jump diffusion process are present, but is also able to capture the intensity and size of the jumps. This means that the signature method can be used to give information about the parameters λ and σ_J when fitting an exponential jump diffusion process.

Where in the previous two settings we saw an equal performance of both classifiers, we see in table 5.3 that the neural network outperforms the random forest. Though the signature captures information about the jump intensity and the jump size, the relation between these features and the output is likely of such a non-linear type that cannot be captured by the random forest. The neural network, as a consequence of the universal approximation theorem (see section 4.3), can capture any type of relation between the input and the output. This explains the difference in observed performance between the two classifiers.

5.2.2. Increasing drift classification accuracy

In practice, the drift parameter μ plays a prominent role in the modelling of financial time series. Especially when the time series spans many years, selecting an accurate drift parameter can make a big difference in the movement of the time series and thereby influences the inference obtained from using the model. The classification accuracy of the proposed signature method provided in table 5.1 - though very close to the theoretical maximum - might not present the user with sufficient confidence in the selected drift parameter.

Increasing the accuracy within the setting of table 5.1 is barely possible, the theoretical maximum is almost attained. However, equations (3.23) and (3.25) show the relation between the classification accuracy and the parameters T, σ and $\mu_2 - \mu_1$. According to these equations, the probability of correct classification increases with $\mu_2 - \mu_1$ (as table 5.1 empirically confirms), increases as $\sqrt{T}$ increases and increases as $\frac{1}{\sigma}$ increases. Though in practice both σ and T are determined by the observed time series one wishes to classify, it is worth investigating to what extend they influence the classification accuracy and which settings are ideal for obtaining a high classification accuracy when determining the drift parameter.

Influence of the time horizon

The purpose of this experiment is to visualise the relation between the time horizon T and the classification accuracy when classifying between paths with different drifts. This experiment focuses on GBM with different drifts, but the situation for ABM or jump diffusion processes with different drifts is very similar. Let $X^1(t), X^2(t)$ be two GBMs as described in equation (5.1) with equal volatility $\sigma_1 = \sigma_2 = 0.2$ and drifts $\mu_1 \neq \mu_2$ with $\mu_2 > \mu_1$. Figure 5.4 shows the classification accuracy where $\Delta\mu := \mu_2 - \mu_1$.

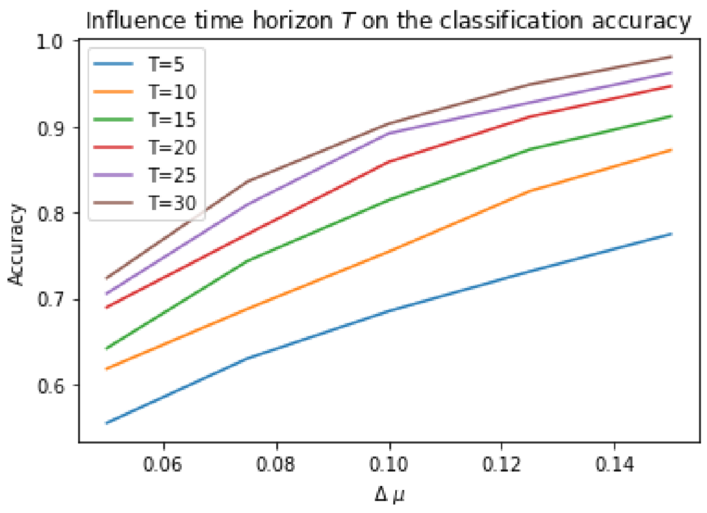


Figure 5.4: Accuracy binary GBM classification between different drifts with fixed $\sigma = 0.2$ using random forest.

Figure 5.4 shows that indeed the classification accuracy increases as T increases; there appears to be a linear translation upwards as T increases. As T increases, the positive effect of this increase on the accuracy grows smaller. Recall equation (3.25) and notice that for $T \to \infty$ we have

$$\Phi\Big(\frac{\sqrt{T}}{2\sigma}(\mu_2 - \mu_1)\Big) \to 1,$$
$$\Phi\Big(\frac{\sqrt{T}}{2\sigma}(\mu_1 - \mu_2)\Big) \to 0,$$

because by assumption $\mu_2 > \mu_1$. This results in

$$\mathbb{P}(C^{\text{GBM, drift}}) \to 1.$$

In other words, the probability of correct classification tends to 1 as $T \to \infty$. Similarly, for $T = 0$ we have

$$\mathbb{P}(C^{\text{GBM, drift}}) \to \frac{1}{2}.$$

A classification accuracy of $\frac{1}{2}$ is the worst possible result, since it signifies that the classifier is no more accurate than a simple flip of a coin.

It is up to the user of the signature method to determine what level of accuracy is required and whether the time horizon of the observed path is sufficiently long to obtain this accuracy. The information in figure 5.4 can be useful in this determination.

Influence of the volatility
Increasing the volatility strains the classification between paths with different drifts, as the increase in 'noise' makes it difficult to distinguish the signal. Let the setting be the same as in the previous paragraph, but with fixed $T = 10$. Figure 5.5 demonstrates the influence of the volatility on the classification accuracy.

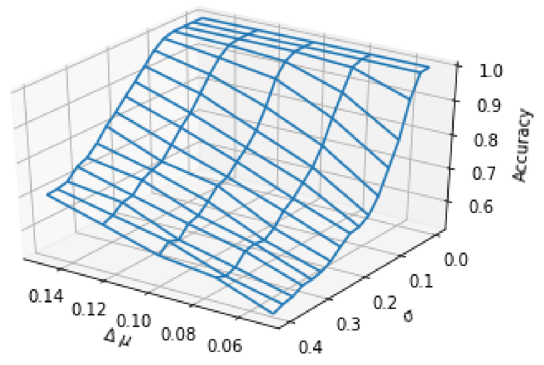


Figure 5.5: Accuracy binary GBM classification between different drifts with fixed $T = 10$ using random forest.

Figure 5.5 shows that the volatility has a profound influence on the classification accuracy. Especially the combination of small $\Delta\mu$ and high σ severely complicates the classifying process. It is again up to the user to determine if the setting in the observed path results in a sufficient accuracy, where he or she can make use of the information in figure 5.5.

5.2.3. Impact of transformations

Recall that in this thesis the choice is made to use signatures instead of log-signatures (see section 3.3.1) and to use the lead-lag transformation instead of no transformation (see section 3.3.4). This section analyses what the effect is of these choices on the classification accuracy on the relevant classification problems discussed in section 5.2.1.

Signature vs. log-signature

As mentioned in section 3.3.1, there is no clear preference from the literature between the usage of signatures or log-signatures, as both have their advantages and disadvantages. Log-signatures span a linear space and capture paths in a more parsimonious way than signatures, but they do not form an algebra whereas signatures do (see section 3.3.3). Therefore they cannot offer the guarantee that the classification function can be uniformly closely be approximated by a combination of log-signature elements, whereas this is possible for the signature elements.

In this section we empirically investigate the influence of using the signature or log-signature transformation on the classification accuracy of the three classification settings as described in section 5.2.1. Both the signature and log-signature are truncated at $L = 4$ and are computed over the same lead-lag transformation of the paths. For both settings, a random forest classifier is used for classification. The first setting we observe is the classification between two GBMs with different parameters, as described by equation (5.1). The results are shown in table 5.4.

Parameters				Accuracy	
μ_1	μ_2	σ_1	σ_2	$S(X)$	$\log S(X)$
0.1	0.1	0.1	0.2	1.0 (0.000)	1.0 (0.001)
0.1	0.1	0.1	0.3	1.0 (0.000)	1.0 (0.001)
0.1	0.1	0.1	0.4	1.0 (0.000)	1.0 (0.000)
0.1	0.15	0.2	0.2	0.629 (0.025)	0.605 (0.027)
0.1	0.20	0.2	0.2	0.772 (0.021)	0.755 (0.025)
0.1	0.25	0.2	0.2	0.877 (0.013)	0.876 (0.014)

Table 5.4: Percentage of paths correctly classified for two GBM processes with different parameters. The difference between using signatures and using log-signatures is presented.

Table 5.4 shows that there is no significant difference in accuracy between using the signature or the log-signature transformation. This means that, despite the fact that the log-signature does not span an algebra, its elements are descriptive enough to distinguish between GBMs with different drifts or volatilities. This is consistent with the findings in section 3.3.4, which shows that the drift and volatility are related to the second-order terms of both the signature and the log-signature.

The next analysed classification setting is the classification between paths with and without jumps (see equation (5.2)). Table 5.5 shows the classification accuracy where the parameters are equal to the parameters used in table 5.2 in order to generate paths with overall equal variance.

Parameters				Accuracy	
λ	σ_1	σ_2	σ_J	$S(X)$	$\log S(X)$
0.01	0.2	0.175	0.1	0.853 (0.019)	0.586 (0.022)
0.02	0.2	0.13	0.1	0.959 (0.011)	0.737 (0.018)
0.03	0.2	0.08	0.1	0.977 (0.007)	0.788 (0.017)
0.04	0.2	0.05	0.1	0.981 (0.008)	0.796 (0.018)
0.05	0.2	0.0	0.1	0.992 (0.004)	0.805 (0.021)

Table 5.5: Percentage of paths correctly classified for one GBM process and one exponential jump diffusion process. The difference between using signatures and using log-signatures is presented.

Table 5.5 shows that there is a significant difference between using signatures or log-signatures; using signatures results in higher classification accuracies for all parameters observed. This implies that jumps within the jump diffusion process are not completely captured by a combination of the log-signature elements, whereas they are captured by the signature elements. This is a consequence of the fact that log-signatures do not span an algebra. The results of the classification between two exponential jump diffusion processes with the parameters as in table 5.3 are given in Appendix A.4 and show a similar result: using the signature transformation leads to higher accuracies than using the log-signature transformation. This indicates that log-signatures are less suitable for handling classification problems which include distinguishing between paths generated by the exponential jump diffusion process.

Lead-lag transformation
In section the definition of the lead-lag transformation is given, which is often used in combination with the signature transformation. In the start of this chapter we revealed that the lead-lag transformation works best in the classification settings used in this thesis, but this section provides a deeper insight in how the transformation affect the classification accuracy and the results show that the application of the lead-lag transformation substantially improves the accuracy of the classification.

The signature can be expressed in terms of the path as shown in equations (3.7) and (3.9). Let again $X^1(t), X^2(t)$ follow two GBMs with parameters as given in table 5.6. This table provides the classification accuracy for when different transformations are applied.

Parameters				Accuracy	
μ_1	μ_2	σ_1	σ_2	No transformation	Lead-lag transformation
0.1	0.1	0.1	0.105	0.789 (0.019)	0.800 (0.026)
0.1	0.1	0.1	0.11	0.809 (0.021)	0.995 (0.015)
0.1	0.1	0.1	0.2	0.910 (0.014)	1.0 (0.000)
0.1	0.1	0.1	0.3	0.953 (0.012)	1.0 (0.000)
0.1	0.1	0.1	0.4	0.979 (0.006)	1.0 (0.000)
0.1	0.15	0.2	0.2	0.612 (0.023)	0.629 (0.025)
0.1	0.20	0.2	0.2	0.766 (0.026)	0.772 (0.021)
0.1	0.25	0.2	0.2	0.877 (0.014)	0.877 (0.013)

Table 5.6: Percentage of paths correctly classified for several different transformations, using a random forest.

The influence of the lead-lag transformation on the other classification problems discussed in section 5.2.1 is similar and the results are added in Appendix A.2. As mentioned in section 3.3.4, the lead-lag transformation allows the signature to capture the volatility parameter explicitly. Table 5.6 indeed shows that the classification accuracy is higher when the lead-lag transformation is used when classifying between processes with different volatilities. The classification accuracy when classifying between different drifts only depends on the first and last value of the path (see equation (3.15)) and since these values are equal for the case of the lead-lag transformation as for applying no transformation, these results remain (almost) equal. Since the volatility is an important parameter, also in other processes like the exponential jump diffusion process, this transformation is recommended when making inferences about observed paths.

5.2.4. Level of truncation

The level of truncation L is an important parameter, since it essentially controls the amount of 'information' that is captured from an observed path. From an information perspective, selecting L as high as possible is beneficial, since theoretically more information should lead to a higher classification accuracy. However, choosing L too high leads to practical issues, as the length of the signature sequence increases rapidly when one level higher is taken, resulting in a high dimensional input which can reduce the classification accuracy (also known as the 'curse of dimensionality'). In this section we consider the influence of the level of truncation on the classification accuracy.

The optimal level of truncation differs per classification setting. For example, we know from section 3.3.4 that $S^{(1)}(X)$, $S^{(2)}(X)$ relate to the drift and $S^{(1,2)}(X)$, $S^{(2,1)}(X)$ to the volatility when the lead-lag transformation is applied. This means that when we are classifying between GBMs with different drifts or different volatilities, truncating at $L = 2$ will likely already provide enough information to distinguish between the processes. Let $X^1(t), X^2(t)$ described in equation (5.1) and choose the same parameters as in table 5.1. Table 5.7 shows the influence of the level of truncation on the classification between GBMs with different drifts or different volatilities.

Parameters				Accuracy			
μ_1	μ_2	σ_1	σ_2	$L = 2$	$L = 3$	$L = 4$	$L = 5$
0.1	0.1	0.1	0.2	0.996 (0.003)	1.0 (0.000)	1.0 (0.000)	1.0 (0.000)
0.1	0.1	0.1	0.3	0.999 (0.001)	1.0 (0.001)	1.0 (0.000)	1.0 (0.000)
0.1	0.1	0.1	0.4	1.0 (0.001)	1.0 (0.000)	1.0 (0.000)	1.0 (0.000)
0.1	0.15	0.2	0.2	0.597 (0.028)	0.617 (0.023)	0.629 (0.025)	0.630 (0.021)
0.1	0.20	0.2	0.2	0.755 (0.025)	0.763 (0.021)	0.772 (0.021)	0.774 (0.022)
0.1	0.25	0.2	0.2	0.860 (0.017)	0.876 (0.018)	0.877 (0.013)	0.877 (0.017)

Table 5.7: Percentage of paths correctly classified for two GBM processes with different parameters where the signature is truncated at various levels.The results were generated using a random forest. PCA was applied to reduce the 'curse of dimensionality'.

Indeed, table 5.7 shows that the classification accuracy only slightly increases as the level of truncation increases when classifying between different drifts and remains constant when classifying between different volatilities. However, the computational cost increases significantly as the level of truncation increases. Since $S(X)^{L=k} \in \mathbb{R}^{2^{k+1}-1}$ we have that $S(X)^{L=2} \in \mathbb{R}^7$, $S(X)^{L=3} \in \mathbb{R}^{15}$, $S(X)^{L=4} \in \mathbb{R}^{31}$ and $S(X)^{L=5} \in \mathbb{R}^{63}$. Therefore, for this specific setting, the truncation level can be chosen lower than the standard level $L = 4$ used in this thesis.

The situation is different when models containing jumps are included. Table 5.8 shows the classification accuracy when classifying between a GBM process and an exponential jump diffusion process, where the parameters are equal to the parameters in table 5.2.

Parameters				Accuracy			
λ	σ_1	σ_2	σ_J	$L = 2$	$L = 3$	$L = 4$	$L = 5$
0.01	0.2	0.175	0.1	0.512 (0.025)	0.746 (0.019)	0.853 (0.019)	0.890 (0.015)
0.02	0.2	0.13	0.1	0.534 (0.025)	0.879 (0.016)	0.959 (0.011)	0.987 (0.005)
0.03	0.2	0.08	0.1	0.545 (0.021)	0.832 (0.017)	0.977 (0.007)	0.983 (0.003)
0.04	0.2	0.05	0.1	0.540 (0.018)	0.942 (0.012)	0.981 (0.008)	0.990 (0.005)
0.05	0.2	0.0	0.1	0.561 (0.022)	0.815 (0.013)	0.992 (0.004)	0.983 (0.005)

Table 5.8: Percentage of paths correctly classified for one GBM process and one jump diffusion process where the signature is truncated at different levels. The results were generated using a random forest and PCA was applied to reduce the 'curse of dimensionality'.

Table 5.8 indeed shows that the classification accuracy increases as L increases. This is consistent with our expectation, as truncating the signature at a higher level includes more information about the process. The table also shows that truncating the signature at $L = 2$ reduces the classification accuracies. This is coherent with the analysis of the signature elements of $S(X)^{L=2}$ in section 3.3.4, which shows that the elements are related to the drift and volatility, but contain no information about possible jumps. This means that we expect to be unable to distinguish between the two processes, resulting in a classification accuracy around 0.5. We see that the accuracy increases for $L = 3$ and $L = 4$, indicating that jumps in paths are captured by a combination of 3^{rd} and 4^{th} order elements of the signature sequence. Unfortunately, due to the high collinearity of the features (see section 4.1), it is not possible to identify these signature elements. Furthermore, table 5.8 shows using signatures of $L = 5$ does not increase the accuracy significantly, so it suffices to use signatures truncated at $L = 4$ in this specific classification setting.

Remark 12. *Though table 5.8 indeed shows accuracies around 0.5 for $L = 2$, it also shows that the classification accuracy increases slightly as the jump intensity λ increases. This is likely a result of the choice of parameters in order to satisfy equation (5.3). The parameters are chosen in such a way that (5.3) holds in theory, but in practice 10,000 paths are generated, meaning that the sample variance can always differ a little bit.*

Distinguishing between two exponential jump diffusion processes with different jump intensities and jump sizes (see equation (5.4)) shows the same type of results as listed in table 5.8. These results are included in Appendix A.3.

5.2.5. Influence of the number of time steps on signature

In this thesis, paths with 1000 time steps are used when computing the signature. This paragraph illustrates the effect of the number of time steps on the value of the signature. We generate one GBM path (see equation (2.2)) of 1,000,000 time steps with $\mu = 0.1, \sigma = 0.1$ and all other parameters as described in 5.1. The number of time steps is varied by sub-sampling from the original path in order to make a valid comparison. E.g. for a path of 10 time steps, the 1st, 10,001th, 20,001th etc. time step of the original path are taken. Figure 5.6 shows the paths and its sub-sampled path.

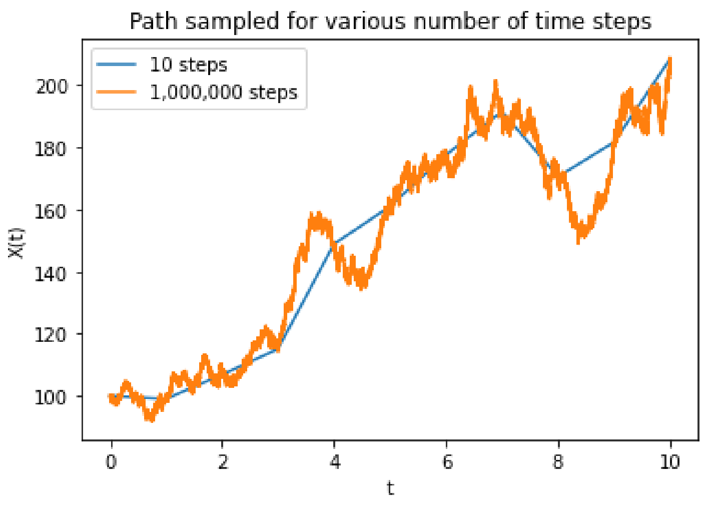


Figure 5.6: A path plotted with 10 and 1,000,000 time steps.

Recall that the signature is a vector of elements as described in example 3.1. Table 5.9 lists several of these elements of the path represented in figure 5.6. The total signature truncated at level 3 is given in Appendix A.5.

N	$S^{(1,2)}$	$S^{(2,1)}$	$S^{(1,1,2)}$	$S^{(2,1,1)}$
10	7447	4283	313 596	142 298
100	7289	4440	310 540	156 233
1000	6958	4771	281 972	163 579
10 000	7020	4709	285 124	160 001
100 000	7046	4683	286 936	158 984
1 000 000	7041	4668	286 595	159 195

Table 5.9: Value of the signature for N number of time steps, rounded at whole numbers.

From the table in Appendix A.5 a number of interesting observations can be made:

- $S^{(1)}, S^{(2)}, S^{(1,1)}, S^{(2,2)}, S^{(1,1,1)}$ and $S^{(2,2,2)}$ stay the same for every N. Recall from section 3.3.4 that $S^{(1)}, S^{(2)}$ are defined as $X_N - X_1$ which remain equal for every value of N since we are sub-sampling from the same sequence. Additionally, $S^{(1,1)} = \frac{1}{2}(S^{(1)})^2$, $S^{(2,2)} = \frac{1}{2}(S^{(2)})^2$ and similarly $S^{(1,1,1)} = \frac{1}{3!}(S^{(1)})^3$, $S^{(2,2,2)} = \frac{1}{3!}(S^{(2)})^3$.

- The other elements of the signature sequence get better approximated as the time step decreases, which is consistent with what we would expect.

Table 5.10 gives more information about the convergence of the elements listed in table 5.9 by looking at the decrease in increments percentage. For this purpose, write $S^{(I)}(N)$ with $I \in \{(1,2),(2,1),(1,1,2),(2,1,1)\}$ as a function of N and define d as the percentage difference between two consecutive values , i.e.

$$d(S^{(I)}) := \frac{|S^{(I)}(N) - S^{(I)}(N/10)|}{S^{(I)}(N/10)} \cdot 100.$$

Table 5.10 lists the percentage difference computed over the values in table 5.9.

N	$d(S^{(1,2)})$	$d(S^{(2,1)})$	$d(S^{(1,1,2)})$	$d(S^{(2,1,1)})$
10	-	-	-	-
100	2.0%	3.7%	0.97%	9.8%
1000	4.5%	7.4%	9.2%	4.7%
10,000	0.88%	1.2%	1.1%	2.2%
100,000	0.37%	0.55%	0.64%	0.63%
1,000,000	0.07%	0.03%	0.11%	0.13%

Table 5.10: Percentage difference in signature element for various numbers of time steps.

Table 5.10 shows that the difference between the signature elements decreases as N increases and that the signature gets better approximated. Based on table 5.10 either $N = 10,000$ or $N = 100,000$ would give a good numerical approximation of the signature.

However, in this thesis the choice is made to use paths of 1000 time steps. There are two main reasons for this choice:

1. **Computational power:** generating paths, their lead-lag transformation and their signature transformation is already quite a costly process. Since especially the neural network needs about 100,000 - 200,000 samples in order to be trained properly, generating the signatures becomes exceedingly computationally expensive as N increases.

2. **Classification accuracy:** the classification accuracy of the signature method is very close to the theoretical maximum classification accuracy for those cases where the theoretical maximum could be computed (see section 3.5). Increasing N leads to a more accurate numerical approximation of the actual signature, but since the accuracy for $N = 1000$ already almost attains the theoretical upper bound, increasing N in hopes of a higher classification accuracy is futile.

5.3. Multi-class classification

In the previous section, we focused on binary classification and how certain changes in the classification setting influence the classification accuracy. However, in practice it is very useful to be able to distinguish between multiple stochastic processes at the same time. In this section, we extend the binary classification settings as observed in the previous setting to a multi-class classification setting.

5.3.1. GBM with different parameters

Section 5.2.1 showed the accuracy for the classification between two GBM processes with several different parameter settings, so $\mathbb{M} = \{M(\theta^1), M(\theta^2)\}$ with $\theta = (\mu, \sigma)$. In this section we analyse how the classification accuracy is impacted when p parameters settings are analysed, i.e. $\mathbb{M} = \{M(\theta^1), \dots, M(\theta^p)\}$.

Different volatility

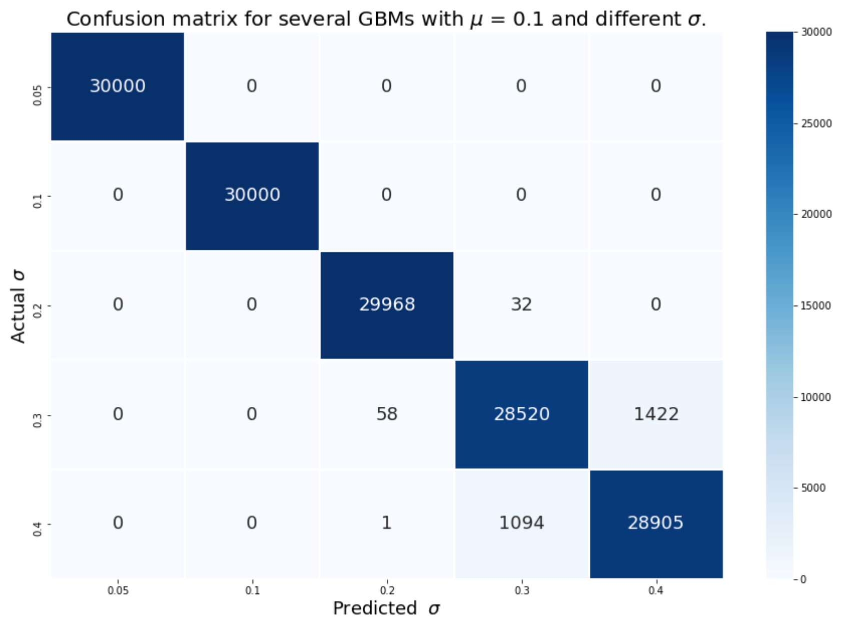
The first setting analysed is GBM with constant drift $\mu = 0.1$ and different volatility $\sigma \in \{0.05, 0.1, 0.2, 0.3, 0.4\}$. Where for binary classification the overall classification accuracy gave a good representation of the fit of the machine learning model, the classification accuracy for multi-class classification is split per class and listed in table 5.11.

σ	Accuracy per class
0.05	1.0
0.1	1.0
0.2	0.999
0.3	0.951
0.4	0.964
Overall	0.983

Table 5.11: Percentage of paths correctly classified for five GBMs with equal drift $\mu = 0.1$ and different volatilities. The classification accuracy is split per class. This result was generated with a neural network.

Table 5.11 shows high classification accuracies, indicating that using the signature method for calibration of the volatility parameter in GBM is feasible. This is consistent with the results obtained for binary classification, which also showed high classification accuracies for GBMs with different volatilities. Table 5.11 also illustrates that samples with $\sigma = 0.3$ or $\sigma = 0.4$ are slightly fewer times correctly classified, though the difference in accuracy is small.

It is relevant to further analyse the misclassified samples. If a sample is misclassified, it is interesting to know whether it was misclassified to a class with a volatility close to the actual volatility of the sample or not. For this purpose, we use the *confusion matrix*. The confusion matrix shows per class the number of samples classified to that class and their actual class. As mentioned in section 5.1, from every model 100,000 paths are generated of which 30% are set aside to use for testing, meaning that every class contains 30,000 testing samples. Figure 5.7 shows the confusion matrix of the results listed in table 5.11.



Actual σ \ Predicted σ	0.05	0.1	0.2	0.3	0.4
0.05	30000	0	0	0	0
0.1	0	30000	0	0	0
0.2	0	0	29968	32	0
0.3	0	0	58	28520	1422
0.4	0	0	1	1094	28905

Figure 5.7: Confusion matrix for several GBMs with equal drift $\mu = 0.1$ and different volatilities, using a neural network.

Figure 5.7 shows that the misclassified samples are indeed samples with $\sigma \in \{0.2, 0.3, 0.4\}$ as indicated in table 5.11, but it also shows how they are misclassified. The diagonal of the matrix represents the number of correctly classified samples per class, off-diagonal elements of the matrix represent misclassified samples. Since there are 30,000 samples per class, the sum of every row equals 30,000. The confusion matrix shows that the samples that are misclassified are indeed classified to a class with a value of σ close to the actual value of σ.

Different drift

The next problem analysed is GBM with constant volatility $\sigma = 0.2$, but different drift $\mu \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$. The results for binary classification in table 5.1 show that classification between paths with different drifts is challenging, since it is not always possible to distinguish between paths generated from different models (see section 3.5). This same concept also holds for multi-class classification, of which the results are presented in table 5.12.

μ	Accuracy per class
0.05	0.672
0.10	0.291
0.15	0.275
0.20	0.321
0.25	0.668
Overall	0.445

Table 5.12: Percentage of paths correctly classified for five GBMs with equal volatility $\sigma = 0.2$ and different drifts. The classification accuracy is split per class. This result was generated with a neural network.

Table 5.12 shows an overall classification accuracy of 44.5%, significantly lower than the overall classification for the classification between different volatilities, which is in line with the expectation. The classification accuracy of the first and last class are significantly higher. To further analyse this, the confusion matrix is given in figure 5.8.

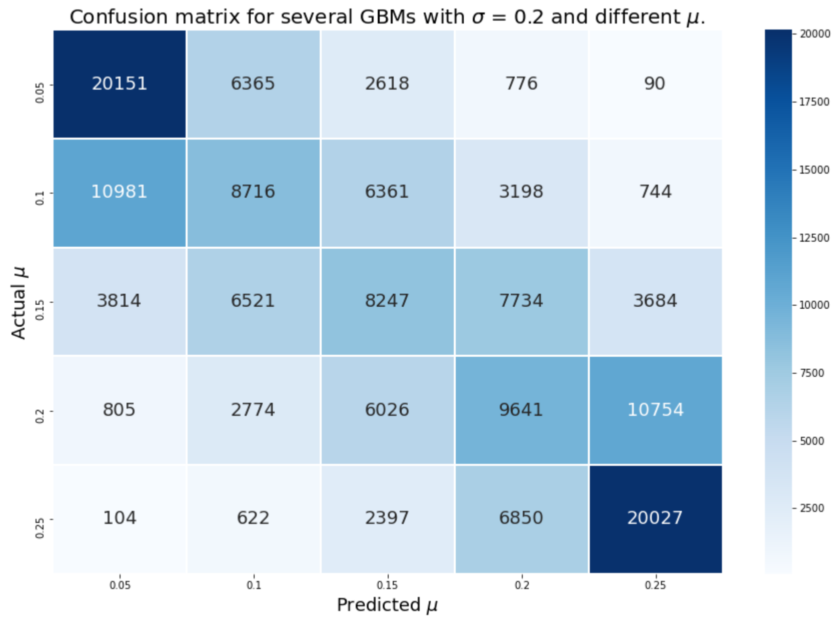
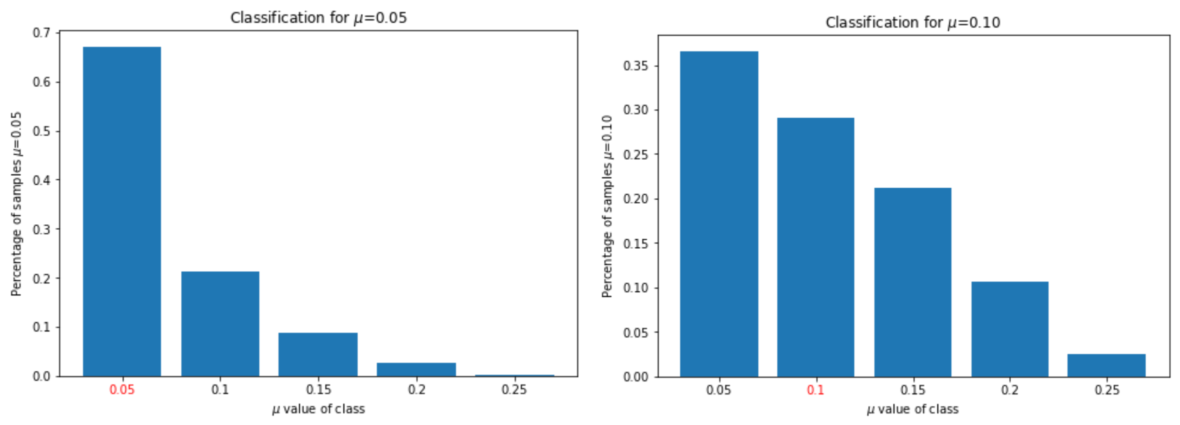


Figure 5.8: Confusion matrix for several GBMs with equal volatility $\sigma = 0.2$ and different drifts, using a neural network.

The confusion matrix shows how the samples are classified, notice again that all rows sum up to 30,000. Though the classification accuracy is low, the confusion matrix shows that when samples are misclassified, they are classified to classes close to their actual class. To give more insight in this, figure 5.9 shows a histogram representation of the rows of the confusion matrix in figure 5.8.



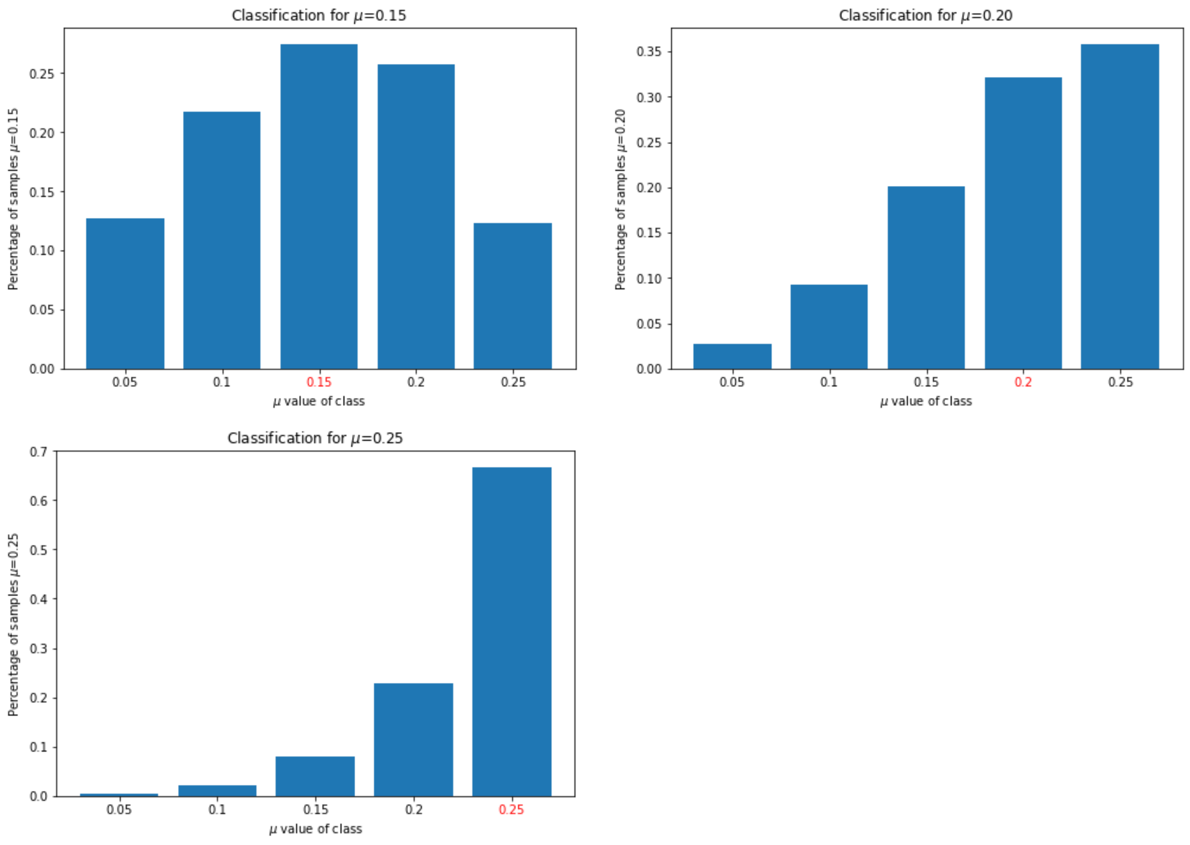


Figure 5.9: Histogram per class indicating the predicted class of the samples. The red value on the x-axis denotes the actual value of the samples.

Figure 5.9 indeed shows that misclassified samples are classified to a class close to the actual class. Especially for $\mu = 0.15$ we see that the misclassified samples are symmetrically distributed around the true class. For the other classes, especially $\mu = 0.1$ and $\mu = 0.2$, we see that this is not the case: the misclassified samples are not symmetrically distributed around the true class. The reason for this is the limited number of classes. For example for the class with $\mu = 0.2$, there is only one class with a higher drift. Therefore the samples which according to the neural network have $\mu > 0.2$ are all classified in the class with $\mu = 0.25$, since there is no class available with a higher drift. This explains the asymmetric shape of the histograms for classes in the lower or higher part of the range in drifts.

Different drift and different volatility

For practical purposes it is also interesting to observe how the classification accuracy is impacted when both μ and σ are different. Since in this situation the class labels can no longer be represented by μ or σ, we set the class labels as indicated below. Recall that $\theta = (\mu, \sigma)$.

- $\theta^1 := (0.1, 0.2)$
- $\theta^2 := (0.15, 0.2)$
- $\theta^3 := (0.2, 0.2)$
- $\theta^4 := (0.25, 0.2)$
- $\theta^5 := (0.1, 0.1)$
- $\theta^6 := (0.1, 0.3)$
- $\theta^7 := (0.1, 0.4)$

Notice that $\theta^1 : \theta^4$ contain different drifts and $\theta^5 : \theta^7$ contain different volatility. Classification between classes $\theta^1 : \theta^7$ results in the accuracies as listed in table 5.13.

θ	Accuracy per class
θ^1	0.664
θ^2	0.291
θ^3	0.275
θ^4	0.352
θ^5	1.0
θ^6	0.963
θ^7	0.952
Overall	0.642

Table 5.13: Percentage of paths correctly classified for five GBMs with different drift and volatility. The classification accuracy is split per class. This result was generated with a neural network.

The confusion matrix in figure 5.10 gives more insight in the classification accuracies listed in table 5.13.

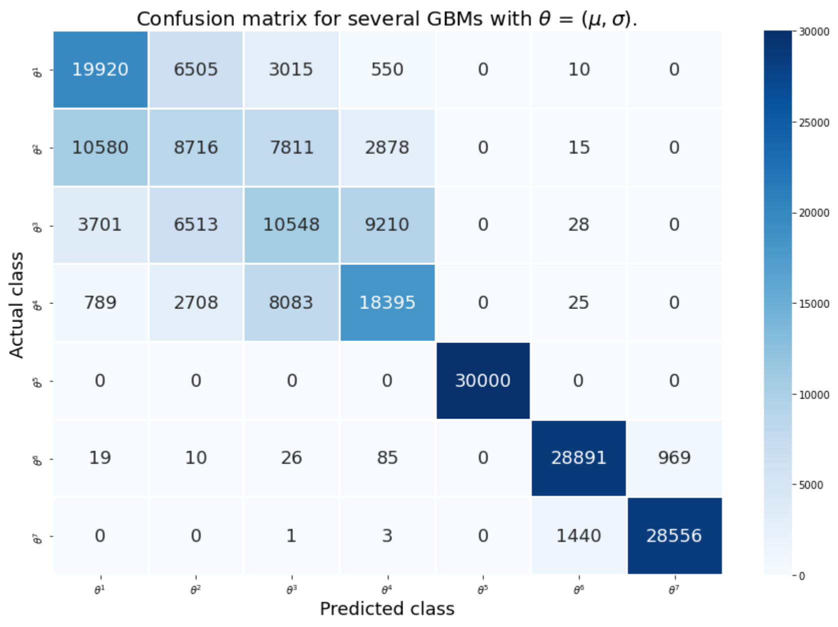


Figure 5.10: Confusion matrix for classification between several GBMs with different drift and different volatility, generated using a neural network.

Recall that $\theta^1 : \theta^4$ are classes with different drifts. The confusion matrix again shows that distinguishing between drifts remains difficult, but it also shows that again the misclassified samples are classified to a class close to their actual class, and it also shows that the classification accuracy of samples with different volatilities remains high, it is not impacted by the inclusion of samples with different drifts.

6 Conclusion and discussion

In this chapter a summary is given of the signature-based method proposed in this thesis, followed by a discussion of the implications and limitations of the method. This thesis concludes with a section on topics for further research.

6.1. Concluding summary

This research aimed to identify the stochastic process that has most likely generated an observed financial time series $\omega_D = \{D_{t_1}, D_{t_2}, \dots, D_{t_n}\}$ out of a set of stochastic models $\mathbb{M} = \{M_1(\theta^1), \dots, M_k(\theta^k)\}$ where M_i denotes the stochastic model and θ^i denotes the parameters. Though methods are available to achieve this when M_i are assumed to be discrete-time stochastic processes, these methods cannot easily be extended to continuous time stochastic processes. The reason for this is that the time series - or *paths* - generated by these models are elements of the path space, which is infinite-dimensional and non-locally compact, making classification based on the paths directly infeasible.

In this thesis a signature-based classification method is proposed. The signature transformation transforms the paths to the signature space. A signature of a path is a infinite-length sequence, where every element of this sequence is an iterated integral over the path. In this thesis it is shown how these iterated integrals are related to well-known properties of paths like the drift and the volatility. Higher order elements of the signature are related to the skewness, kurtosis etc. In general, the most relevant statistical properties are located at the start of the signature sequence and the full infinite-length signature is known to capture all relevant information about a path. This thesis affirms the proof given in [16] that signatures span an algebra and that therefore any function on the signature space can be uniformly closely approximated through a combination of the elements of the signature. This means that the elements of the signature are suitable to use as input in machine learning methods for classification. In practice the signature is truncated in order to obtain a finite sequence.

Three stochastic processes were chosen to distinguish between: Arithmetic Brownian Motion (ABM), Geometric Brownian Motion (GBM) and the exponential jump diffusion process. ABM and GBM have analytically available density functions, which are used in this thesis to compute the maximum possible classification accuracy when distinguishing between models with different drifts or models with different volatilities. Two machine learning methods are used to classify between different stochastic processes by training on their signatures: random forest and neural network. Random forest is used in combination with PCA to reduce the dimensionality and the multicollinearity of the signatures. It is a fast method and requires few training samples. The neural network is trained directly on the signatures. It is a slow method and requires many training samples, but equals or outperforms the random forest for every classification setting in this thesis.

The proposed signature-based classification method is tested in binary and multi-class classification. In binary classification this research illustrated that classification between ABM or GBM with different parameters using the signature-based method results in classification accuracies which are very close to the theoretical maximum classification accuracy. This research showed that classification between paths with different volatilities lead to higher accuracies than classification between paths with different drifts and analysed how the classification accuracy depends on other parameters like the time horizon and the number of steps of the path. Furthermore, using the proposed method allowed for classification between models with and without jumps. This indicates that signatures are able to effectively capture information about jumps in paths. The effect of various choices of the classification setting were examined, like the impact of the level of truncation on the signature and the number of time steps of the path on the classification accuracy. Most notably, the research revealed that increasing the time horizon or decreasing the volatility could increase the classification accuracy when attempting to classify between paths with different drift. Multi-class classification was conducted on several GBM models with different parameters. The experiments indicated that the signature-based method is able to classify between paths with different volatilities, but the classification accuracies for paths with different drifts declined.

6.2. Discussion

This section reflects on the contribution of this thesis and discusses the results obtained. The signature method is a model agnostic method, indicating that the method can be used to distinguish between any type of stochastic process from which generated samples are available. It is therefore a flexible method and can be used in a wide range of practical applications in finance, but also in other application areas. However, we are aware that the research has limitations, which we discuss below.

The effectiveness of the method is limited by the similarity of the models between which one attempts to classify. Using the proposed method to distinguish between highly similar models can result in low accuracy scores. We provided examples of this limitation by experimenting with classification between GBMs with different drift. Deciding which stochastic process is most suitable is more difficult in these situations.

Additionally, efficiency can become problematic as the signatures on which the classifier is trained need to be generated. This is a slow process, because it involves many steps like generating the path, computing the lead-lag transformation and computing the signature over the lead-lag transformation. Since each new stochastic process requires a fixed number of signatures to be generated for training, including more of these processes results in the need for more signatures. This process can be parallelised in order to speed up the process, but nevertheless it is making the classification time-consuming. Therefore this method is not recommended for situations were computational resources are scarce.

Furthermore, the hyper-parameters of the classifiers used in this thesis have been calibrated to work well on average on the classification problems discussed. Though the performance of both classifiers is observed to be stable over these classification problems, there is no guarantee that the chosen hyper-parameters generalise well to every individual classification problem. Calibrating the hyper-parameters per individual classification problem could potentially increase the classification accuracy slightly.

Lastly, the signatures used for training the classifier are computed over discrete paths that are connected piece-wise linearly. This means that the computed signatures are approximations of the true signatures of continuous paths. A consequence of this is that generation of paths with long time horizons and few time steps might lead to inaccurate approximations of the true signature.

6.3. Topics for further research

In this thesis the focus is put on the stochastic processes ABM, GBM and the exponential jump diffusion process, chosen because of their available analytical results and their popularity in practice. An interesting topic for further research would be to analyse the effectiveness of the signature-based classification methods on other type of stochastic processes. Examples are to classify between the Cox–Ingersoll–Ross (CIR) process and the Ornstein-Uhlenback (OU) process. Both are processes that are mean-reversion processes, i.e. processes that tend to reverse back to the mean. It is interesting to include these processes in $\mathbb{M}$, in order to analyse whether the signature can effectively capture the mean-reversion property of the paths. Another classification problem of interest is classification between stochastic volatility models. In the models introduced in this thesis, the volatility coefficient is assumed to be a constant. However, in practice many local volatility models exist, where the volatility is assumed to be a function depending on time, or even is assumed to follow a stochastic process. Since the results in this research indicate that the signature-based method is able to distinguish well between models with different (constant) volatility coefficients, it is worthwhile to analyse the performance of the signature-based classification methods in other models with non-constant volatility.

The signature is known to capture all statistical properties of a path, but the interpretation of the elements of the signature is not known explicitly. It is known which elements capture the drift and volatility, but especially the higher order terms of the signature are still unexplained. As shown in this thesis, the signature is able to capture the presence and size of jumps, as we are able to distinguish between stochastic processes with different jump sizes and intensities and we are able to classify between stochastic processes with and without jumps. Identifying which signature elements are linked to the jumps would help the interpretation of the signature elements. However, as mentioned in this thesis, this is complicated by the multi-collinearity of the signature elements. Further research in identifying which signature elements are capturing specific properties despite this multi-collinearity will be benificial for the understanding of the signature transformation.

Bibliography

[1] L. Bachelier. Theory of speculation: The origins of modern finance. *Francia: Gauthier-Villars*, 1900.

[2] A. Bagnall, A. Bostrom, J. Large, and J. Lines. The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version, 2016.

[3] K. Benidis, S. S. Rangapuram, V. Flunkert, B. Wang, D. Maddix, C. Turkmen, J. Gasthaus, M. Bohlke-Schneider, D. Salinas, L. Stella, L. Callot, and T. Januschowski. Neural forecasting: Introduction and literature overview. 2020.

[4] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

[5] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[7] J. Brownlee. A gentle introduction to dropout for regularizing deep neural networks, December 2018. URL machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks. Blog post, accessed 28-04-2021.

[8] J. Brownlee. How to control the stability of training neural networks with the batch size, January 2019. URL machinelearningmastery.com/how-to-control-the-speed-and-stability-of-training-neural-networks-with-gradient-desc Blog post, accessed 28-04-2021.

[9] J. Brownlee. Understand the impact of learning rate on neural network performance, January 2019. URL machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/.

[10] H. Bühler, B. Horvath, T. Lyons, I. P. Arribas, and B. Wood. A data-driven market simulator for small data environments, 2020.

[11] N.L. Carothers. *Real Analysis*. Cambridge University Press, 2000. ISBN 9780521497565.

[12] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers Electrical Engineering*, 40(1):16–28, 2014. ISSN 0045-7906. URL sciencedirect.com/science/article/pii/S0045790613003066.

[13] I. Chevyrev and A. Kormilitzin. A primer on the signature method in machine learning. 2016.

[14] I. Chevyrev and H. Oberhauser. Signature moments to characterize laws of stochastic processes. 2018.

[15] R. D. De Veaux and L. H. Ungar. Multicollinearity: A tale of two nonparametric regressions. In *Selecting models from data*, pages 393–402. Springer, 1994.

[16] A. Fermanian. Signature and statistical learning, 10 2018. Master thesis at ETH Zürich.

[17] CFI for business. Random forest. URL corporatefinanceinstitute.com/resources/knowledge/other/random-forest. Accessed 18-04-2021.

[18] I. Goodfellow, J. Pouget-Abadie, Me. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 3, 06 2014.

[19] B. Graham. Sparse arrays of signatures for online character recognition. *arXiv preprint arXiv:1308.0371*, 2013.

[20] D. Grattarola. Deep feature extraction for sample-efficient reinforcement, October 2017. Master thesis at Politecnico di Milano.

[21] I. Guyon and A. Elisseeff. *An Introduction to Feature Extraction*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-35488-8.

[22] L. G. Gyurkó, T. Lyons, M. Kontkowski, and J. Field. Extracting information from the signature of a financial data stream. 2014.

[23] B. Hambly and T. Lyons. Uniqueness for the signature of a path of bounded variation and the reduced path group. *Annals of Mathematics*, pages 109–167, 2010.

[24] T. K. Ho. Random decision forests. 1:278–282 vol.1, 1995.

[25] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080.

[26] J. Hull. Interest rate models and negative rates, 2016. URL `fincad.com/blog/interest-rate-models-and-negative-rates`. Blog post, accessed 22-4-2021.

[27] I. T. Jolliffe. *Principle component analysis*. Springer, 2002.

[28] P. Kanani and M. Padole. Ecg heartbeat arrhythmia classification using time-series augmented signals and deep learning approach. *Procedia Computer Science*, 171:524 – 531, 2020. ISSN 1877-0509. URL `sciencedirect.com/science/article/pii/S1877050920310231`. Third International Conference on Computing and Network Communications (CoCoNet'19).

[29] A. Kondratyev and C. Schwarz. The market generator. *Econometrics: Econometric Statistical Methods - Special Topics eJournal*, 2019.

[30] A. B. Kormilitzin, K. E. A. Saunders, P. J. Harrison, J. R. Geddes, and T. J. Lyons. Application of the signature method to pattern recognition in the cequel clinical trial, 2016.

[31] S. G. Kou. A jump-diffusion model for option pricing. *Management science*, 48(8):1086–1101, 2002.

[32] D. Levin, T. Lyons, and H. Ni. Learning from the past, predicting the statistics for the future, learning an evolving system. 2016.

[33] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The expressive power of neural networks: A view from the width. 2017.

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

[35] T. Lyons, S. Nejad, and I. Perez Arribas. Numerical method for model-free pricing of exotic derivatives in discrete time using rough path signatures. *Applied Mathematical Finance*, 26(6):583–597, 2019.

[36] David Maxwell. `esig` package in python, 2017. URL `esig.readthedocs.io/_/downloads/en/latest/pdf/`. Version 0.6.

[37] R. C. Merton. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1):125 – 144, 1976. ISSN 0304-405X. URL `sciencedirect.com/science/article/pii/0304405X76900222`.

[38] F. Mörchen, I. Mierswa, and A. Ultsch. Understandable models of music collections based on exhaustive feature generation with temporal statistics. pages 882–891, 01 2006.

[39] N.Raman, J. L. Leidner, K. Vytelingum, and G. Horrell. Synthetic reality: Synthetic market data generation at scale using agent based modeling. 6 2019. URL `simudyne.com/wp-content/uploads/2019/06/Simudyne-Refinitiv-Paper.pdf`.

[40] B. Oksendal. *Stochastic Differential Equations (3rd Ed.): An Introduction with Applications*. Springer-Verlag, Berlin, Heidelberg, 1992. ISBN 3387533354.

[41] C.W. Oosterlee and L.A. Grzelak. *Mathematical Modeling And Computation In Finance: With Exercises And Python And Matlab Computer Codes*. World Scientific Publishing Company, 2019. ISBN 9781786347961. URL `quantfinancebook.com`.

[42] A. Sharma. Random forest figure. `analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm`, 5 2020. Accessed: 2021-01-21.

[43] S.E. Shreve. *Stochastic Calculus for finance II: continuous time models*. Springer, 2004.

[44] M. A. Stephens. Edf statistics for goodness of fit and some comparisons. *Journal of the American Statistical Association*, 69(347):730–737, 1974.

[45] M. Wiese, R. Knobloch, R. Korn, and P. Kretschmer. Quant gans: deep generation of financial time series. *Quantitative Finance*, 20(9):1419–1440, Apr 2020. ISSN 1469-7696.

[46] Shengyang Zhou. Signatures, rough paths and applications in machine learning, 6 2019. Master thesis at Utrecht University.

A

Appendix

A.1. Paths sampled from various processes

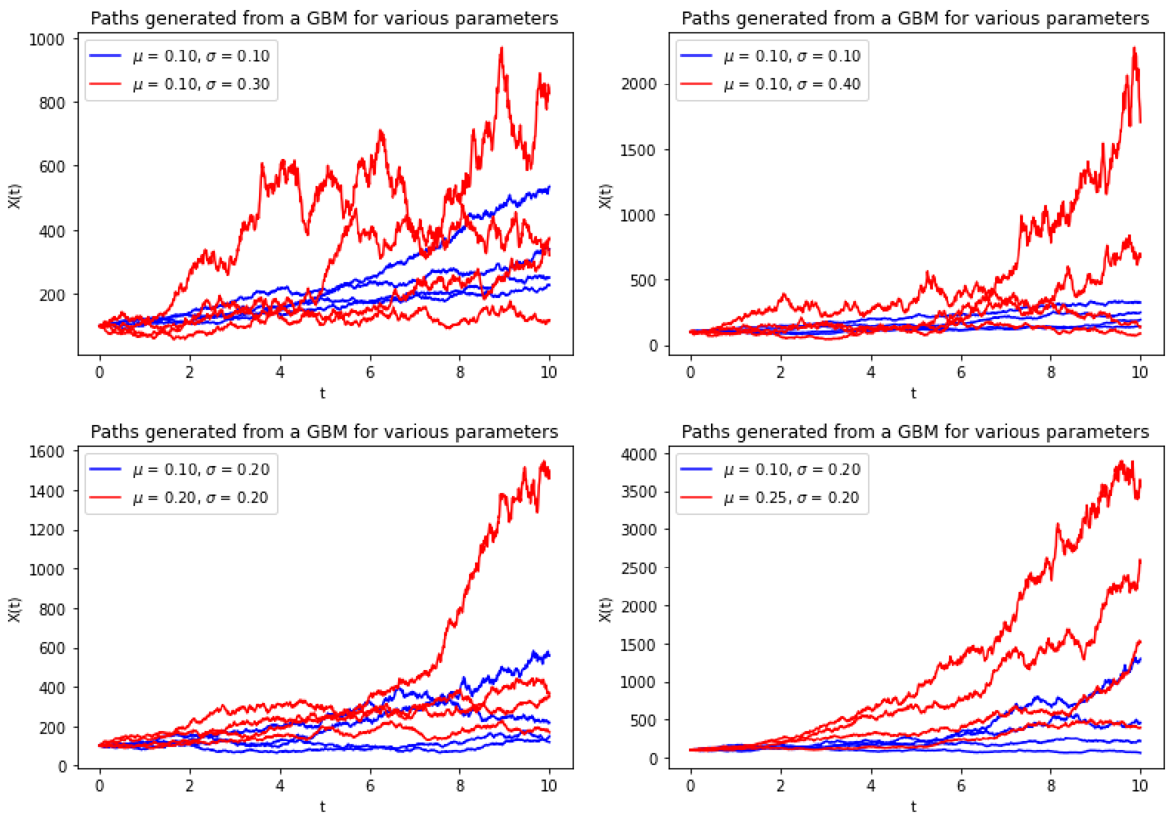


Figure A.1: Paths generated from a GBM for various parameters

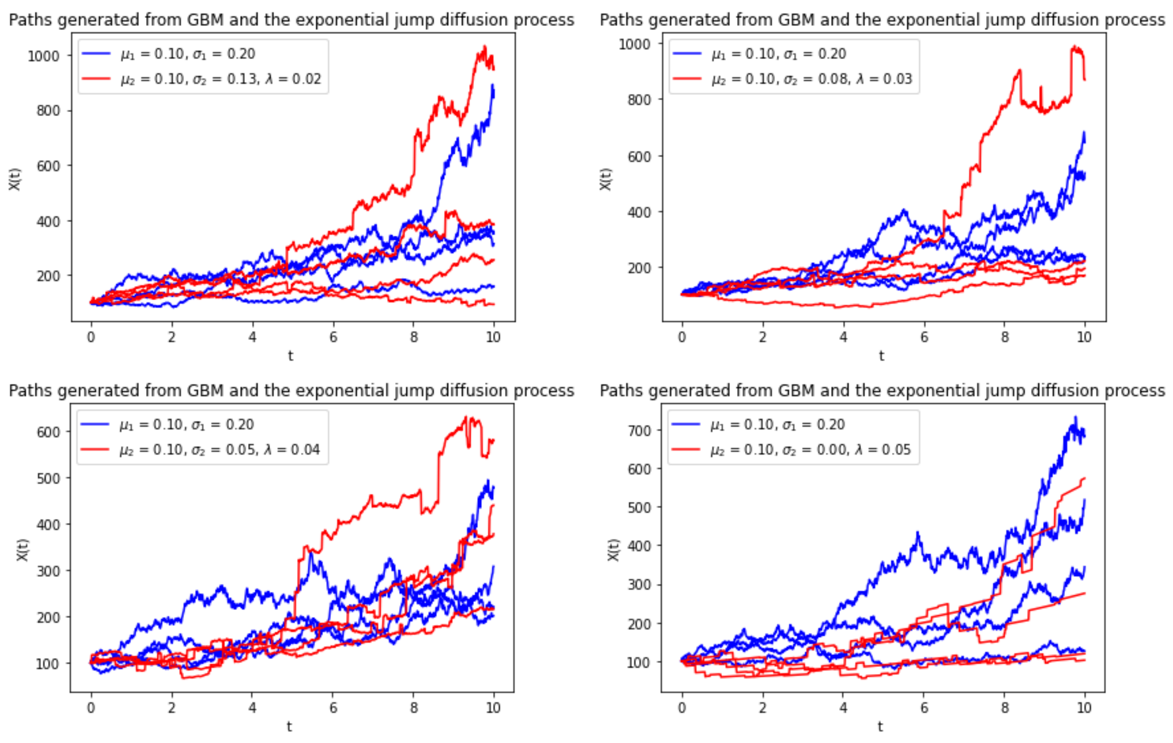


Figure A.2: Paths generated from a GBM (blue) and the exponential jump diffusion process (red).

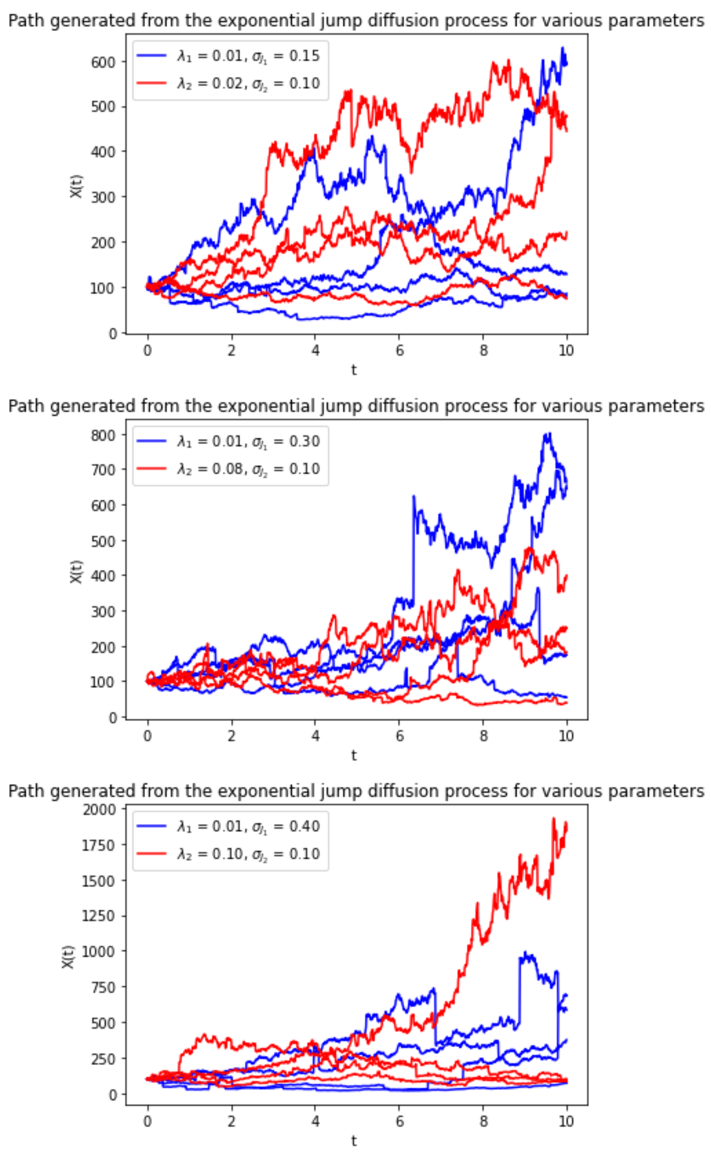


Figure A.3: Paths generated from the exponential jump diffusion process for various parameters.

A.2. Impact lead-lag transformation

Parameters				Classifier	
λ	σ_1	σ_2	σ_J	No transformation	Lead-lag transformation
0.01	0.2	0.175	0.1	0.504 (0.027)	0.853 (0.019)
0.02	0.2	0.13	0.1	0.517 (0.030)	0.959 (0.011)
0.03	0.2	0.08	0.1	0.520 (0.025)	0.977 (0.007)
0.04	0.2	0.05	0.1	0.515 (0.023)	0.981 (0.008)
0.05	0.2	0.0	0.1	0.562 (0.024)	0.992 (0.004)

Table A.1: Percentage of paths correctly classified for one GBM process and one jump diffusion process, using a random forest.

Parameters				Classifier	
λ_1	λ_2	σ_{J_1}	σ_{J_2}	No transformation	Lead-lag transformation
0.01	0.02	0.15	0.1	0.504 (0.027)	0.603 (0.024)
0.01	0.041	0.2	0.1	0.503 (0.025)	0.719 (0.021)
0.01	0.082	0.3	0.1	0.521 (0.028)	0.853 (0.019)
0.01	0.103	0.4	0.1	0.566 (0.022)	0.844 (0.016)

Table A.2: Percentage of paths correctly classified between two jump diffusion processes, using a random forest.

A.3. Level of truncation

Parameters				Accuracy			
λ_1	λ_2	σ_{J_1}	σ_{J_2}	$L = 2$	$L = 3$	$L = 4$	$L = 5$
0.01	0.02	0.15	0.1	0.505 (0.029)	0.554 (0.030)	0.603 (0.024)	0.592 (0.019)
0.01	0.041	0.2	0.1	0.515 (0.019)	0.644 (0.028)	0.719 (0.021)	0.763 (0.023)
0.01	0.082	0.3	0.1	0.514 (0.026)	0.761 (0.020)	0.853 (0.019)	0.843 (0.014)
0.01	0.103	0.4	0.1	0.523 (0.021)	0.727 (0.023)	0.844 (0.016)	0.858 (0.012)

Table A.3: Percentage of paths correctly classified between two jump diffusion processes where the signature is truncated at different levels. The results were generated using a random forest and PCA was applied to reduce the ‘curse of dimensionality’.

A.4. Signature vs. log-signature

Parameters				Accuracy	
λ_1	λ_2	σ_{J_1}	σ_{J_2}	$S(X)$	$\log S(X)$
0.01	0.02	0.15	0.1	0.603 (0.024)	0.522 (0.021)
0.01	0.041	0.2	0.1	0.719 (0.021)	0.595 (0.027)
0.01	0.082	0.3	0.1	0.853 (0.019)	0.715 (0.019)
0.01	0.103	0.4	0.1	0.844 (0.016)	0.771 (0.022)

Table A.4: Percentage of paths correctly classified between two jump diffusion processes. The difference between using signatures and using log-signatures is presented.

A.5. Number of steps

N	$S^{(1)}$	$S^{(2)}$	$S^{(1,1)}$	$S^{(1,2)}$	$S^{(2,1)}$	$S^{(2,2)}$	$S^{(1,1,1)}$	$S^{(1,1,2)}$	$S^{(1,2,1)}$	$S^{(1,2,2)}$	$S^{(2,1,1)}$	$S^{(2,1,2)}$	$S^{(2,2,1)}$	$S^{(2,2,2)}$
10	108	108	5865	7447	4283	5865	211 732	313 596	179 302	291 128	142298	224 238	119 829	211 732
100	108	108	5865	7289	4440	5865	211 732	310 540	168 422	270 036	156 233	249 430	115 729	211 732
1000	108	108	5865	6958	4771	5865	211 732	281 972	189 644	260 055	163 579	233 478	141 662	211 732
10 000	108	108	5865	7020	4709	5865	211 732	285 124	190 070	263 478	160 001	233 362	138 355	211 732
100 000	108	108	5865	7046	4683	5865	211 732	286 936	189 276	264 481	158 984	234 186	136 528	211 732
1 000 000	108	108	5865	7041	4688	5865	211 732	286 595	189 404	264 268	159 195	234 059	136 868	211 732

Table A.5: Value of the signature for various number of time steps, rounded at whole numbers.