DU_XINGYU_01520835.pdf

by Xingyu Du

Submission date: 04-Sep-2022 09:24AM (UTC+0100)
Submission ID: 185671025
File name: DU_XINGYU_01520835.pdf (2.2M)
Word count: 14365
Character count: 70781

Imperial College London

Imperial College London

Department of Mathematics

Predicting Economic Recessions Using Signature Kernels

Author: Xingyu Du (CID: 01520835)

A thesis submitted for the degree of

MSc in Mathematics and Finance, 2021-2022

Declaration

The work contained in this thesis is my own work unless otherwise stated.

Acknowledgements

First and foremost, I would like to thank my friends and family, it could be a tough year without their comfort and support. I really appreciate every call from my parents and grandparents, their encouragement was so warm and powerful that made me more positive and be able to overcome challenges encountered.

I would also like to express my deepest gratitude to my supervisor Dr. Thomas Cass for his professional and effective advice during the research project, which kept me in the appropriate progressing. It is truly my honour to work with him.

Abstract

In this paper we explore the use of signature kernels, which is one of the kernel methods based on support vector machine framework, in predicting US economic recessions. In particular, we tackle binary classification task of forecasting whether it was in recession period or not indicated by 0/1. We examine powers of Treasury term spreads and other macroeconomic variables including Excess Bond Premium(EBP) etc. to forecast recession periods. We also do the comparison of results between using RBF kernel only and using RBF kernel as static kernel in signature kernel framework, the results showed that indeed, signature kernel has improved the performance of solely using RBF kernel to predict in terms of comprehensive performance metrics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recession is not an unfamiliar word in the 21^{st} century, we have experienced several extreme global economy recession in the former two decades. Many studies tried to capture some features or indicators that highly correlated to this financial disaster. Among them, Michael Puglia and Adam Tucker (2020) gave comprehensive explanation that Treasury term spread, or inverted Treasury yield curve is a strong indicator of recession in the United States [4]. Indeed, in early 1980s, the term spread between the 3-month Treasury bill discount and 10-year on-the-run yield-to-maturity was even negative, in other words, the level of 10-year Treasury bill yield-to-maturity was lower than that of 3-month discount since the double dip recession occurred [5].

It is natural with arising of the issue of predicting recessions, and this issue was researched in a lot of previous studies by econometrists and mathematicians and is still one of the most concerned topic in present. This binary classification task has been studied using a variety of machine learning methods including probit method, XGBoost classifier, Random Forest, Neural Network, Support Vector Machine, etc. In particular, support vector machine is one of the most commonly used supervised machine learning methods because of their robustness and high accuracy in classification tasks. SVM's have been proved to be promising and successful in time series classification, especially for data with high dimensional feature space, i.e. data with a large number of features [6]. In the very beginning, SVM's were only useful for classifying linearly separable data, and the usage was fairly limited, researchers started to seek if there is any ways or methods that can be implemented on SVM's such that they are able to manage some more sophisticated problems. Therefore, fortunately, Bernhard Schölkopf and Alexander J. Smola(2001) has described a method of tracking non-linear features of data, in other words, we are enabled to classify non-linearly separable data, this procedure is known as kernel tricks [7], which brought SVM's to a higher extend.

In our paper, we are going to deal with sequential data, then there is a challenging task of designing appropriate kernel functions, otherwise, with referenced to [8], some kernel functions including the widely-used family of Dynamic Time Warping (DTW) kernels performed poor when classifying

time series. C. Salvi et al. therefore pointed out that a principal constraint where most machine learning model have to face is symmetry present in the data, this obstacle became more significant when confronting with sequential data, this is to encounter with much bigger, for example infinite-dimensional, group of symmetries [9]. Hence we are introducing another family of kernel functions, namely, signature kernels, based on the signature of a given path, which can be considered as inner product of two signatures, to deal with sequential data more efficiently.

This paper is structured as follows. In Chapter 2, we first give a plain description of data source and the reason why we chose them as components of feature set in our training model, afterwards, the cleaning and processing of which is stated as well. Chapter 3 covers the detailed theoretical background of our approach, starting from ideas behind support vector machines, followed by kernel tricks and some examples, the last part of this chapter would be the main technique in this paper, as suggested in our title, the mechanism of signature and signature kernels. Chapter 4 goes into detail about the implementation of the models and algorithms, in the following, we will state the results of how models are performing as well. Subsequently, elaborate explanation of potential reasons why some models performed better than others will be given. Finally, Chapter 5 concludes the paper, discussing some open questions and stating any further studies should be continued on the topic.

Chapter 2

Data

The primary purpose of this paper is to compare the performance of predictions made by signature kernels and the machine learning methods used in previous studies, hence we are going to use similar financial market and macroeconomic data as our feature set throughout the research. Therefore, the features we have chosen are fairly standard and the results using previous methods should be consistent with that in the past literature. Specifically, we used mainly consistent data with Michael Puglia and Adam Tucker (2020).

2.1 Features and Label

- The first feature we have chosen is term spreads, or yield curve slope(*Slope* hereafter), we chose the difference between monthly average of 10-year Treasury spot yield(on investment basis) and 3-month Treasury bill discount in the Federal Reserve's H.15 series.

- The second one is related to stock market, it is not surprise to choose S&P 500 index as our measure. To be consistent with Estrella and Mishkin (1998), we computed 3-month log difference of end-month values of the index [10], we will call this series as $SP500$ hereafter.

- Then we consider credit market as well, Excess Bond Premium(EBP hereafter) in Gilchrist and Zakrajsek (2012) was used to reflect the conditions [11], which started available from 1973 and is a monthly series.

- As for risk free interest rate, we use the end-of-month values of the effective federal funds rate (FF hereafter) as in Wright (2006) [12].

- Moreover, as in Puglia and Adam Tucker (2020), Federal Reserve Bank of Chicago's National Financial Conditions Index ($NFCI$ hereafter) is used to give more information about financial conditions that Excess Bond Premium provided [4].

- All the data series above can be concluded as financial market inputs, we will introduce two more series to augment our feature space, namely, a macroeconomic/business conditions

index and a survey-based measure of recession probability. The first one is Aruoba-Diebold-Scotti business conditions index (ADS hereafter), we use this to capture macroeconomic conditions.

- The survey measure we include in this paper is cumulative probability of recession over four quarters calculated from the Federal Reserve Bank of Philadelphia's Survey of Professional Forecasters (SPF hereafter). There are $RECESS1$ to $RECESS5$, i.e. five series of probabilities in total, and we are going to calculate the cumulated probability of recession assuming the probabilities of recession in quarters calculated from $RECESS2$ to $RECESS5$ are independent.

- Finally, we add another measure of term premium in our analysis. Again, to be consistent as in [4], we are going to use the 6-month percentage point change in the 5-year term premium of Adrian, Crump and Moench($ACM5D$ hereafter).

The recession indicator we use is as same as in most literature, i.e. National Bureau of Economic Research(NBER)-dated recessions indicator. Different form the assumptions made by Michael Puglia and Adam Tucker (2020) in their paper of twelve month period of prediction, we will include three, six, nine months as the period of prediction as well. Specifically, the indicator is defined as true or 1, if any of the following p (chosen between three, six, nine, or twelve) months falls into recession period, and false or 0 otherwise.

All data used is monthly, starting from January 1973, since EBP started available from that time, ended up with July 2022(595 months were included).
Figure 2.1 and 2.2 summarizes nine series of data, and shaded areas indicate recession periods.

2.2 Data Pre-processing

Cleaning the data is one of the crucial steps in ensuring data indeed makes sense when fed into model, because otherwise it may cause serious errors when fitting the model. For example, since SPF is quarterly data, stands for the cumulated probability of recession over four quarters calculated from the Federal Reserve Bank of Philadelphia's Survey of Professional Forecasters, while other data series are all monthly updated, hence we assume the probability is the same for three months in every single quarter recorded. Despite there being other methods including linear interpolation, we choose to use a constant value so that we do not create data points that do not exist by ourselves.

Also, there are some shifts we need to specify. As we have mentioned in the previous section, the recession indicator is defined as 1 if any of the following p (chosen between three, six, nine, or twelve) months fell in the NBER-dated recession periods [4]. In other words, we are going to model $Pr(NBER_{t+1,t+p} = 1)$ with different collections of features using signature kernels. Hence

Figure 2.1: List of data(1)

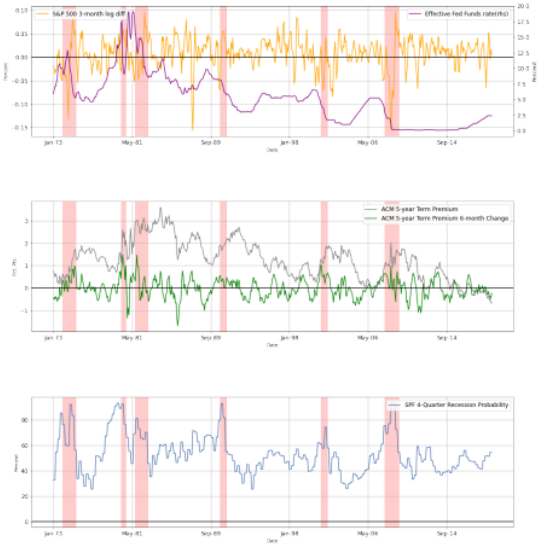

Figure 2.2: List of data(2)

we added p 1's before each recession to be consistent with this framework, which made the comparison of the results more convenient.

Further, as mentioned in [13], for SVM models, the data are often standardised in two ways, namely, $MinMax$ and $MeanVariance$. They are, doing some transformations on each datum point, for $MinMax$, based on extrema(minima and maxima) of training data in the way of

$$z_i = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}.$$

And for $MeanVariance$, we calculate the mean and standard deviation of the training data and do the transformation

$$z_i = \frac{x_i - \mu}{\sigma}$$

on each datum point. In our later implementation, we will present the comparison of the results of $None$, $MinMax$ and $MeanVariance$ standardised data, and we will see which method will perform better.

Last but not least, since our goal is to predict future recessions, and to compute signatures, the rolling window scheme is used, i.e. we will use a 5-month window to predict if the next month is in one year period before or in recession period.

Chapter 3

Methodology

In this Chapter, we first give detailed explanation of theoretical background about the methods we used to forecast. Starting from maximal margin classification, the basic idea of support vector machines, followed by the mechanics of support vector classifier and introduce kernel tricks, a transformation of data that allow work in high dimension.

Secondly, we are introducing concepts of signature and corresponding signature kernels as the kernel of support vector classifiers. Then we will give a proof that signature kernel is the solution of a linear second order, hyperbolic *partial differential equation* (PDE) if the inputs are continuously differentiable paths, as mentioned in [9], numerical approach is also provided and hence can be used in practical implementation.

3.1 Support Vector Machines for Classification

The support vector machine(SVM) model was first introduced in Vladimir N. Vapnik et al.(1992) and was a both powerful and straightforward model, based on ideas of geometrically solving classification or regression problems [14].

In this section, we start by describing separating hyperplanes, the main idea of SVM models. Followed by the kernel method, namely, transforming data using certain functions and thus can capture non-linearity features, several examples of common kernels will also be introduced.

3.1.1 Hard Margin Classification

We start by considering the classification problem with regular setup, where the input variables are continuous and linearly separable, and the output variable is binary:

$$\mathbf{x}^{(i)} = \left(x_1^{(i)}, \ldots, x_p^{(i)}\right) \in \mathbb{R}^p \quad \rightarrow \quad y^{(i)} \in \{0, 1\}, \quad i = 1, \ldots, N$$

Without loss of generality, we chose 0 and 1 as the two classes of our outcomes, to be consistent with recession period indicators. To visualise the general concept, we consider 2-dimensional case

at the very beginning, i.e. $\mathbf{x}^{(i)} = \left(x_1^{(i)}, x_2^{(i)}\right)$, and is sketched below (See in Figure 3.1 (Source: M. Barahona et al. 2020 [1])). Then the ideas of SVM are intuitively easy: find the hyperplane and use it to bisect the input space then divide the input data into two classes. In Figure 3.1, the blue line is what we are interested in, i.e. the hyperplane(it is a line for $\mathbf{x} \in \mathbb{R}^2$;it is a plane for $\mathbf{x} \in \mathbb{R}^3$). The two shaded areas in orange and red indicate the two classes of the points.

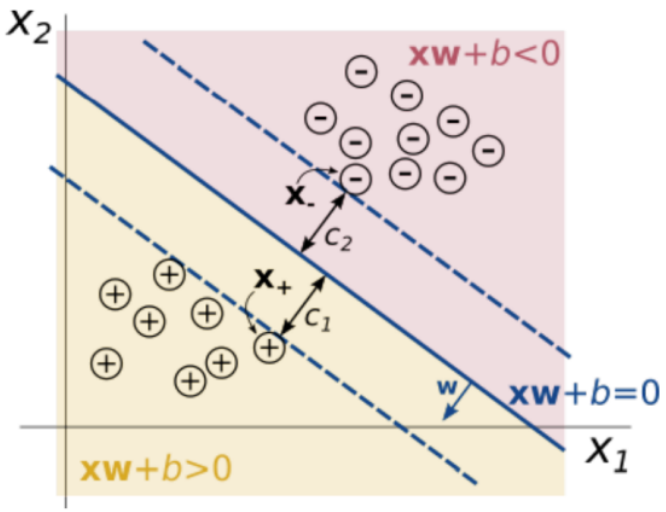

Figure 3.1: Sketch of a support vector machine with input dimension two and binary outcome

Then we want to generalise the case to higher dimensions, and to find the separating hyperplane in the form of

$$\{\mathbf{x} \in \mathbb{R}^p \mid \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}, \mathbf{w} \in \mathbb{R}^p, b \in \mathbb{R}, \tag{3.1.1}$$

where $\mathbf{w}$ is vertical normal to the hyperplane. This is to say that any point $\mathbf{x}^{(i)} = \left(x_1^{(i)}, \ldots, x_p^{(i)}\right)$ lying on the hyperplane satisfies equation 3.1.1. Our goal here is to obtain $\mathbf{w}$ and b, which define the hyperplane. Once we have determined both, a separating hyperplane is defined and we can have the following deciding rules of binary classes [1]:

$$\text{Given } \mathbf{x}, \text{ if } \begin{cases} \mathbf{x} \cdot \mathbf{w} + b \geq 0 & \text{then } \hat{y} = 1 \\ \mathbf{x} \cdot \mathbf{w} + b < 0 & \text{then } \hat{y} = 0 \end{cases} \tag{3.1.2}$$

In other words, we first the model on the training data set, and find the hyperplane based on that. And to classify the test data, what we only need to do is to compute $\mathbf{x} \cdot \mathbf{w} + b$ and to see whether it is negative or not. If the sign is negative, then the y corresponded to $\mathbf{x}$ is classified as 0, and vice versa. Another fact worth to mention is that, regardless of the sign, consider the absolute value of $\mathbf{x} \cdot \mathbf{w} + b$, if the value is far larger than 0, it indicates the classification is confident in this case. Otherwise, if the value is close to 0, it can cause some uncertainties due to it is not assured how close the point is to the separating hyperplane.

As we can see in Figure 3.1, once we determined a separating hyperplane(line in this case), we can have infinite lines such that bisects the input space theoretically as a result of infinitesimal translation. Therefore, it is sensible to find the 'widest street' possible through the data, i.e. to find the hyperplane that separates the points as far as possible, which means to maximise the distance between the hyperplane and the closest points of the two classes. The distance between the separating hyperplane and points in each class is called *geometrical margin*, where the minimum value of margin is called *margin*. It became natural for us to maximise margin and find the optimised hyperplane, namely, the *maximal margin hyperplane*. As we have seen in Figure 3.1, the hyperplane is 'supported' by x_+ and x_-, the normal vectors from the hyperplane to the closet points from the two classes(x_+ is for class 1 and x_- is for class 0), these are called *support vectors*, and that is also the reason of the name of this method.

Now the problem becomes finding the width of the street we mentioned above, the first step is to set up the problem mathematically. Similar to 3.1.2, we still use the notations in the Figure 3.1, and the problem can be written as:

$$\text{maximise } c_1, c_2 \in \mathbb{R}^+ \text{such that } \begin{cases} \mathbf{x}^{(i)} \cdot \mathbf{w} + b > c_1 & \text{if } y^{(i)} = 1 \\ \mathbf{x}^{(i)} \cdot \mathbf{w} + b < -c_2 & \text{if } y^{(i)} = 0 \end{cases} \tag{3.1.3}$$

If we substitute our $y^{(i)} \in \{0, 1\}$ in the condition 3.1.3, a more compact condition is achieved:

$$y^{(i)} \left(\mathbf{x}^{(i)} \cdot \mathbf{w} + b \right) \geq 0,$$

and it is intuitive to see that the separating hyperplane, should lie in the half-way between two classes of points, i.e. x_+ and x_- should have same length:

$$c_1 = c_2.$$

Substitute x_+ and x_- into the condition 3.1.3, we get:

$$\begin{cases} \mathbf{x}_+ \cdot \mathbf{w} + b = c_1 \\ \mathbf{x}_- \cdot \mathbf{w} + b = -c_1, \end{cases}$$

and the 'width of the street' can be defined as:

$$\text{width } = (\mathbf{x}_+ - \mathbf{x}_-) \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} = \frac{2c_1}{\|\mathbf{w}\|}$$

It worths mentioning that c_1 here is just a constant, in other words, a scaling number. Hence

without loss of generality, we can fix it as unity:

$$c_1 := 1$$

Finally, we have our cost function:

$$\text{width } = \frac{2}{\|\mathbf{w}\|} \tag{3.1.4}$$

This expression tells us to maximise the margin, it is the same as minimising length of $\mathbf{w}$, i.e. $\|\mathbf{w}\|$. We finalize the optimisation problem by considering expressions 3.1.3 and 3.1.4 as follows:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y^{(i)}\left(\mathbf{x}^{(i)} \cdot \mathbf{w} + b\right) \geq 1, \quad i = 1, \ldots, N \tag{3.1.5}$$

The problem in 3.1.5 is a convex quadratic optimization with linear constraints, also known as quadratic programming, and there are a bunch of methods mentioned solving this family of problems in the former literature.

3.1.2 Soft Margin Classification

In the above part, we considered data points that can be separable in d-dimensional space, however, when the data points got more noises or even started to overlap hence not lineally separable, hard margin classifier may not work in the sense of taking those overlaps into account. In a more realistic setting, the strict separating hyperplane does not exist [15], i.e. the case of *imperfect separation*. Therefore, in this part, we are going to extend our previous method to this more realistic scenario, namely, the *Soft Margin Classifier*.

Even if the data is linearly separable, some extreme cases happen, and is quite frequent. It sometimes would be better to allow some misclassifications, being trade-off with generalisation and sensibility of the model itself. For instance, there is an outlier existing in a certain class, and the margin of the separating hyperplane would be way smaller if the hyperplane fit the data perfectly. Therefore, it is better to allow this kind of misclassification in terms of robustness of the model and moreover, avoiding the occurrence of overfitting when classifying test data.

In order to deal with the imperfect separating case, we introduce a *penalty* term and its corresponding regularising parameter C. Figure 3.2, 3.3 and 3.4 give geometric interpretation of soft margin support vector machines for 3 different values of regularising parameter C. It is intuitive to see that with smaller C, the corresponding margin is larger, this is to say smaller C allows more misclassifications. And by contrast, the number of misclassified points is reduced with larger C, compromising the smaller margin it gives. We will go in details about why the thing it is mathematically in the later part.

We now give mathematical details of the setup of our soft margin classification problem. To allow

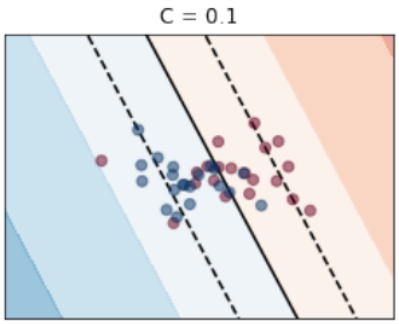

Figure 3.2: Sketch of a soft margin support vector machine with regularising parameter $C = 0.1$

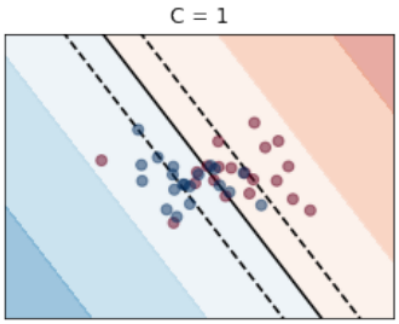

Figure 3.3: Sketch of a soft margin support vector machine with regularising parameter $C = 1$

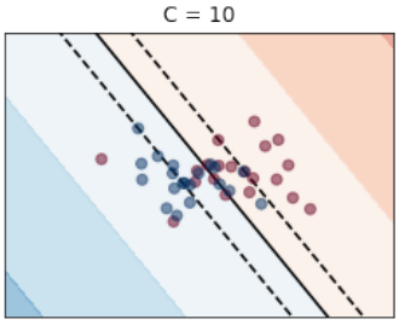

Figure 3.4: Sketch of a soft margin support vector machine with regularising parameter $C = 10$

possible violations of condition 3.1.5, Bennett and Mangasarian [16] introduced so-called *slack variables*, ξ_i:

$$\xi_i \geq 0, \text{ where } i = 1, \ldots, N \tag{3.1.6}$$

and from our previous constraints mentioned in 3.1.5, we have:

$$y^{(i)} \left(\mathbf{x}^{(i)} \cdot \mathbf{w} + b \right) \geq 1 - \xi_i, \quad i = 1, \ldots, N \tag{3.1.7}$$

Obviously, constraint 3.1.7 can be easily satisfied if we choose ξ_i large enough, but our target here is not to obtain the trivial solution with every ξ_i is large(close to 1). Thus, we choose the sum of each ξ_i, i.e. $\sum_{i=1}^{m} \xi_i$ to be penalized in the cost function. Combined constraints 3.1.6 and 3.1.7, the soft margin classification problem is done by solving, for some $C > 0$,

$$\min_{\mathbf{w}, \xi_1, \ldots, N} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \xi_i \quad \text{subject to} \quad \begin{cases} y^{(i)} \left(\mathbf{x}^{(i)} \cdot \mathbf{w} + b \right) \geq 1 - \xi_i, & i = 1, \ldots, N \\ \xi_i \geq 0, & i = 1, \ldots, N \end{cases} \tag{3.1.8}$$

Again, as we have done in Section 3.1.1, the same procedure(quadratic programming) is applied with a new parameter C, which helps us to tune how 'hard' the model is. This is to say that we can decide how much the sum of slack variables and our original cost function $\frac{1}{2}\|\mathbf{w}\|^2$ was taken in account respectively. Now it is intuitive why increasing C will give smaller margin, the sum of ξ_is, $\sum_{i=1}^{N} \xi_i$, played a more significant role in the decision procedure. Lastly, it worths to note that when $\xi_i = 0$, the point is correctly classified, when $\xi_i > 0$, the point lies within the margin, and $\xi_i > 1$ indicates an misclassification.

3.1.3 Multi-Dimensional Support Vector Machines

So far, the goal has been to find a linear hyperplane to separate data. In the above section, we gave mathematical details about classifying linearly separable(hard-margin) or quasi linearly separable(soft-margin) data sets on a 2-d plane, however, in reality, data sets in most cases we deal with are higher dimensional and non-linear as shown in Figure 3.5 (Source: M. Barahona et al. 2020 [1]) and Figure 3.6 (Source: G.J. James et al. 2013 [2]), and non-linear separating hyperplane would be more appropriate in these cases. Therefore, it is now valuable to introduce *Multi-Dimensional Support Vector Machines*, utilising a well-known *Kernel Method*, which enables us to lift our observation data to a higher, possibly infinite dimensional feature space or give non-linear hyperplane.

3.1.3.1 Kernel method

As mentioned above, the main idea of this kernel method is to enlarge our feature space towards higher dimensions in order to allow non-linear boundaries applied when separating data. We can fit this idea neatly into our previous method by thinking all the expressions included 'dot products', and we change it to so-called (non-linear) kernel functions harmoniously.

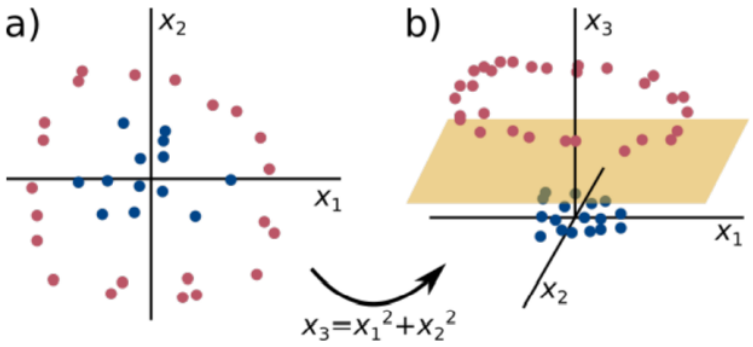

Figure 3.5: Illustration of kernel tricks. **a)** shows a set of binary samples that are not linear separable in $\mathbb{R}^2$. **b)** shows how kernel tricks work and the samples can be linearly separated by a plane in $\mathbb{R}^3$. Source: M. Barahona et al. 2020 [1]

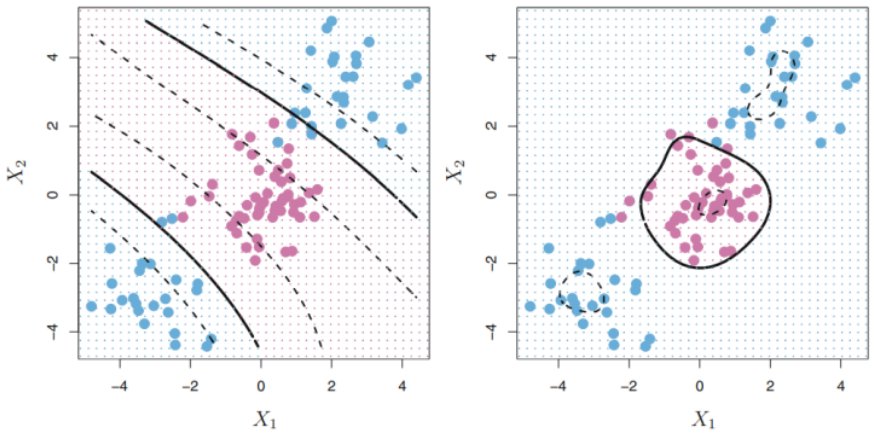

Figure 3.6: Non-linear kernels applied on non-linear data(Left Panel: Polynomial Kernel & Right Panel: Radial Kernel) Source: James et al. (2013) [2]

First let us define the term *dot product*. We already know that dot product of two d-vectors $\mathbf{a}$ and $\mathbf{b}$ is defined as $\langle a, b\rangle = \sum_{i=1}^{d} a_i b_i$, hence, for two p-dimensional observations x_i and x_j, the dot product is given by:

$$\langle x_i, x_j\rangle = \sum_{l=1}^{p} x_{il} x_{jl} \tag{3.1.9}$$

We then give the following proposition first [2].

Proposition 3.1.1. *The linear support vector classifier can be represented as:*

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i\rangle, \tag{3.1.10}$$

where $\alpha_1, \ldots, \alpha_n$ and β_0 are constants.

Now suppose every time the dot product in 3.1.9 appears in a calculation of the solution for the linear support vector classifier 3.1.10, we can generalise the dot product to some *kernel functions*, i.e. replace the dot product with a function of the form

$$k(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j}), \tag{3.1.11}$$

where $\phi(\cdot)$ is so-called *feature map*. We will illustrate the ideas of *kernel trick* by an example. First let us consider the following:

$$\text{Given} \quad \mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$$

$$\text{define}: \quad \boldsymbol{\phi}(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \end{pmatrix} \in \mathbb{R}^3 \quad \text{to be our feature map.}$$

The dot product of the feature map is:

$$\phi(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{y}) = x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 x_2 y_1 y_2,$$

and we have observed that:

$$\begin{aligned} (\mathbf{x} \cdot \mathbf{y})^2 &= (x_1 y_1 + x_2 y_2)^2 \\ &= x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 x_2 y_1 y_2 \\ &= \boldsymbol{\phi}(\mathbf{x}) \cdot \boldsymbol{\phi}(\mathbf{y}) \end{aligned}$$

This is in fact the mathematical explanation of what we have already seen in Figure 3.5, transforming 2-dimensional vectors to 3-dimensional ones. Hence the kernel function here is $K(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{x}, \boldsymbol{y}\rangle^2$, lifting our feature space from $d = 2$ to $D = 3$. This is not a great saving, while more complicated problems will be solved if we have defined a kernel function of dimension $D = 500$, say.

3.1.3.2 Common Kernels

Now we list some families of functions that usually be treated as kernel functions as defined in [17], together with their basic ideas.

Euclidean Kernel

$$K(x, y) = x^T \cdot y$$

Euclidean kernel is the most common, i.e. linear kernel, which was used mostly in comparing how close two observations are to each other via Pearson correlation.

Radial Basis Function(RBF) Kernel

$$K(x,\ y) = e^{-\gamma \|x-y\|^2}, \quad \gamma > 0$$

Radial basis function(RBF kernel) is one of the most popular kernels in implementing non-linear classifications. It adds a "bump" around each observation.

Polynomial Kernel

$$K(x,\ y) = (1 + x \cdot y)^d, \quad d \in \mathbb{N}$$

Polynomial kernel is the kernel function used in the example illustrated in Figure 3.5, the output depends on the directions of the input vectors due to the property of dot product, hence the kernel is directional.

Sigmoid Kernel

$$K(x, y) = \tanh(\beta(x \cdot y) + r), \quad \beta, r \in \mathbb{R}$$

For this family of kernel functions, it became special due to the Gram matrix may not be positively semi-definite in some cases of β and r chosen. However, in those cases of β and r chosen appropriately, i.e. the Gram matrix is computed positively semi-definite, it takes ideas from common activation functions of neuron networks.

3.2 The Signature Kernel

Learning sequential data using kernel methods has been rising in interests in recent years, hence we will be concerned with the setting where X is a subset of unparameterised paths of the form $X : I \to \mathbb{R}^d$, where $I = [a, b]$ is a closed (time) interval with $0 < a < b$, it is natural to model sequential data with this form, where sequential data is nowadays being produced and stored at an unprecedented rate. In this section, we are going to introduce the learning tool, signature kernel, which potentially enables us to handle irregularly sampled, multivariate time series. Some examples of applications will be included as well.

3.2.1 The Signature

Before defining the signature, which is pathwisely dependent, we will start by two concepts, namely, p-variation of a path, and formal polynomials power series over V, V a *Banach space*.

Definition 3.2.1. (p-variation). Let $p \geq 1$ be a real number and $X : [t_0, T] \to \mathbb{R}^m$ be a continuous path. The p-variation of X on any subinterval $[s, t] \subset [t_0, T]$ is the following quantity

$$\|X\|_{p,[s,t]} := \left(\sup_{\mathcal{D} \subset [s,t]} \sum_{t_i \in \mathcal{D}} \left\| X_{t_{i+1}} - X_{t_i} \right\|^p \right)^{1/p},$$

where the norm is any norm on $\mathbb{R}^m$, since they are all equivalent, and the supremum is taken over all partitions $\mathcal{D}$ of the interval $[s, t]$, i.e. over all increasing sequences of ordered indices such that $\mathcal{D} = \{s = k_0 < k_1 < \ldots < k_r = t\}$. And a path X is said to have finite p-variation on $[s, t]$ if $\exists$ $M > 0$ such that $\|X\|_{p,[s,t]} \leq M$, we denote by $C^{p-var}\left([t_0, T], \mathbb{R}^m\right)$ the space of continuous paths from $[t_0, T]$ to $\mathbb{R}^m$ which have finite p-variation.

Definition 3.2.2. Let V be a Banach space. The spaces of formal polynomials and formal power series over V are defined, separately, as

$$T(V) = \bigoplus_{k=0}^{\infty} V^{\otimes k} \quad \textit{and} \quad T((V)) = \prod_{k=0}^{\infty} V^{\otimes k},$$

where $\otimes$ denotes the (classical) tensor product of vector spaces.

Both $T(V)$ and $T((V))$ can be equipped with both operations addition $+$ and multiplication $\otimes$ for two elements $A = (a_0, a_1, \ldots)$ and $B = (b_0, b_1, \ldots)$ in the following way:

$$A + B = (a_0 + b_0, a_1 + b_1, \ldots),$$

$$A \otimes B = (c_0, c_1, c_2, \ldots), \quad \text{where} \quad V^{\otimes k} \ni c_k = \sum_{i=0}^{k} a_i \otimes b_{k-i} \quad \text{for all k} \geq 0.$$

Along with the natural action of $\mathbb{R}$ by $\lambda A = (\lambda a_0, \lambda a_1, \ldots)$ and unity $\mathbf{1} = (1, 0, 0, \ldots)$, $T((V))$ becomes a real, non-commutative algebra with unity, which is called *Tensor algebra*. With these two definitions, we now can define one of the key concepts through this paper, namely, the signature.

Definition 3.2.3. Let $I \subset \mathbb{R}$ be a compact interval, let V be a Banach space, and let $X : I \to V$ be a continuous path of finite p-variation with $p < 2$. For any $s, t \in I$ such that $s \leq t$, the signature $S(X)_{[s,t]} \in T((V))$ of the path X over the subinterval $[s, t]$ is defined as the following infinite collection of iterated integrals:

$$S(X)_{[s,t]} = \left(1, \int_{s<u_1<t} dX_{u_1}, \ldots, \int_{s<u_1<\cdots<u_k<t} \cdots \int_{u_1} \otimes \cdots \otimes dX_{u_k}, \ldots\right).$$

Furthermore, we would like to include some more discussion about the properties signature have and some relevant theorems of signature in the following section, and we are going to explore the reason why signature is a suitable feature map for data streams.

3.2.2 Properties of Signature

Firstly, we would like to introduce a fundamental theorem called *Stone-Weierstrass Theorem*, which is one of the most important theorem in our subsequent derivation of several properties of signature, it is interpreted as follows.

Theorem 3.2.4. *Suppose $\mathbb{R}[X]$ separates points, in that for any $x \neq y \in X$ there exists a polynomial $P \in \mathbb{R}[X]$ such that $P(x) \neq P(y)$. Then, $\mathbb{R}[X]$ is uniformly dense in $C(X)$, i.e. for any continuous function $f \in C(X)$ and for any $\epsilon > 0$, there exists a polynomial $P \in \mathbb{R}[X]$ such that*

$$\sup_{x \in X} |f(x) - P(x)| < \epsilon$$

This important theorem, or the mathematical structure, motivates the ideas about the approximation of learnable parameters during the training process and to optimise some loss functionals. Now we are going to highlight the following key properties of signatures that is relevant to the application(how signatures are able to be one of the valid feature maps) in this paper.

Invariance to reparameterizations

First, we would like to give a motivated example of this property, let us consider the reparameterization $\phi : [0,1] \to [0,1]$ given by $\phi(t) = t^2$ and the path $\gamma : [0,1] \to \mathbb{R}^2$ defined by $\gamma_t = (\gamma_t^x, \gamma_t^y)$ where $\gamma_t^x = \cos(10t)$ and $\gamma_t^y = \sin(3t)$. It is illustrated clearly in Figure 3.7 (source: C. Salvi (2022) [3]), such that the two contributes (γ^x, γ^y) of γ are reparametrized independently and is very different from one each other, however, as it in the right panel, the shape of the curve γ has not been affected at all. It is, in most cases, very hard to construct a such invariant reparameterization, fortunately, signature is equipped with this property, and is stated in the following lemma.

Lemma 3.2.5. ***(Reparameterization invariance).*** *Let $X \in C^{1-var}([t_0, T], \mathbb{R}^m)$ and let $[a,b]$ and $[c,d]$ be two subintervals of $[t_0, T]$. Let $\lambda : [c,d] \to [a,b]$ be a reparameterization. Then $S(X)_{[a,b]} = S(X \circ \lambda)_{[c,d]}$.*

With this lemma, we can easily cope with high dimensional sequential data using signature kernels, such to keep the time parameterization of the path.

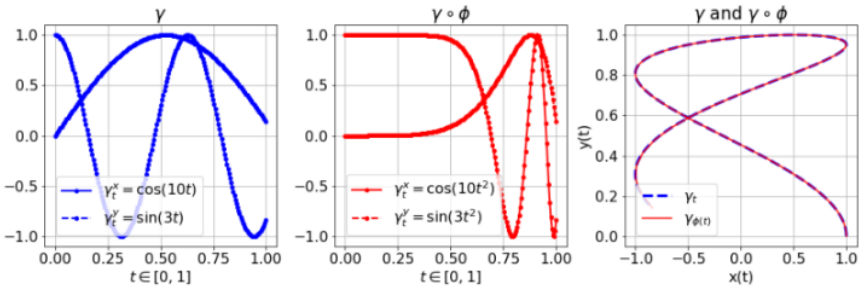

Figure 3.7: The shape of curves under reparameterization Source: C. Salvi (2022) [3]

Shuffle Identity

We have interpreted how tensor product $\otimes$ made the Banach space $T((\mathbb{R}^m))$ an algebra, now we would like to move to another product called *shuffle product*, denote by $\sqcup\!\sqcup$, this product makes the vector space $T((\mathbb{R}^m))$ an algebra successfully as well, which will turn out to be a fundamental tool for explaining how signatures are so triumphant in machine learning applications concerning time series or sequential data.

Now we give the formal mathematical definition of shuffle product:

Definition 3.2.6. (Shuffle Product). Let $i, j \in \mathbb{N}$. Given any two dual basis elements $e_{k_1}^* \otimes \ldots \otimes e_{k_i}^*$ and $e_{k_{i+1}}^* \otimes \ldots \otimes e_{k_{i+j}}^*$, their shuffle product $\sqcup\!\sqcup$ is defined as follows

$$\left(e_{k_1}^* \otimes \ldots \otimes e_{k_i}^*\right) \sqcup\!\sqcup \left(e_{k_{i+1}}^* \otimes \ldots \otimes e_{k_{i+j}}^*\right) = \sum_{\sigma \in \Sigma \begin{pmatrix} k_1, \ldots, k_i \\ k_{i+1}, \ldots, k_{i+j} \end{pmatrix}} e_{\sigma(k_1)}^* \otimes \ldots \otimes e_{\sigma(k_{i+j})}^*$$

where $\Sigma \begin{pmatrix} k_1, \ldots, k_i \\ k_{i+1}, \ldots, k_{i+j} \end{pmatrix}$ is the subset of the permutations of $\{k_1, \ldots, k_{i+j}\}$ satisfying $\sigma(k_1) < \ldots < \sigma(k_i)$ as well as $\sigma(k_{i+1}) < \ldots < \sigma(k_{i+j})$.

The key observation is the following result known as *shuffle identity*.

Definition 3.2.7. (Shuffle Identity) Let $i, j \in \mathbb{N}$ be two integers and $e_{k_1}^* \otimes \ldots \otimes e_{k_i}^*$ and $e_{k_{i+1}}^* \otimes \ldots \otimes e_{k_{i+j}}^*$ be any two dual basis elements. Then, for any path $X \in C^{1-var}([t_0, T], \mathbb{R}^m)$, the following relation holds

$$\left(e_{k_1}^* \otimes \ldots \otimes e_{k_i}^*, S(X)\right)\left(e_{k_{i+1}}^* \otimes \ldots \otimes e_{k_{i+j}}^*, S(X)\right) = \left(\left(e_{k_1}^* \otimes \ldots \otimes e_{k_i}^*\right) \sqcup\!\sqcup \left(e_{k_{i+1}}^* \otimes \ldots \otimes e_{k_{i+j}}^*\right), S(X)\right)$$

The shuffle identity here can be extended by linearity to any two linear functionals $\mathcal{L}_1, \mathcal{L}_2 \in T((\mathbb{R}^m))^*$: for any $X \in C^{1-var}([t_0, T], \mathbb{R}^m)$ one has that $(\mathcal{L}_1, S(X))(\mathcal{L}_2, S(X)) = (\mathcal{L}_1 \sqcup\!\sqcup \mathcal{L}_2, S(X))$. Therefore, the shuffle identity allows to construct a set of functionals acting on the range of the signature that can approximate arbitrarily well continuous functions on compact sets of paths, this

gives the following property which we are going to introduce called *universality*.

Universality

Theorem 3.2.8. ***(Universality).*** *Let $K \subset C^{1-var}\left([t_0, T], \mathbb{R}^m\right)$ be a compact subset of paths and let $C(K, \mathbb{R})$ denote the algebra (with pointwise multiplication) of continuous, real-valued functions on K. Consider the following set of functionals acting on the range of the signature of paths in K*

$$A_K = \left\{\mathcal{L} \circ S : K \to \mathbb{R} \mid \mathcal{L} \in T\left(\mathbb{R}^m\right)^*\right\}$$

Then, A_K is dense in $C(K, \mathbb{R})$.

Hence signature kernel is one of such called *universal kernel*, in other words, all the functions defined on some compact subsets Z of $\mathbb{R}^m$, can be estimated by linear functionals on the signature, or in plain text, linear combinations of signature. Undoubtedly, there are necessary and sufficient conditions to be satisfied for a kernel being universal, indeed, these conditions are provided by Charles A. Micchelli et al. (2006) [18], and intuitively, our signature kernel is one of which satisfied these conditions. In practice, the universality, or universal approximating property of linear functionals on the signature allows to learn some functions $f : K \to \mathbb{R}$ on a dataset K of time series with following procedure:

- extract *signature features* from the input data series,
- perform *linear regression* on such signature features.

Now we have some basic knowledge about the properties of signatures, we are able to deduce why signature has been our choice in this paper. Firstly, since the signature (Definition 3.2.3) is invariant to reparameterizations, as mentioned above, it works as a filter to X, removing infinite-dimensional group of symmetries [3]. Moreover, as the results mentioned in [19], with pointwise multiplication, linear functionals acting on the range of signature form an algebra and separate points. Therefore, by the Stone-Weierstrass Theorem, for any compact set C of continuous paths of bounded variation, the set of linear functionals on signatures is uniformly dense in $C(X)$, the set of continuous real-valued functions from the path X to $\mathbb{R}$, in other words, the universality property of signatures. With these two properties, signature becomes an ideal feature map for data streams [20].

3.2.3 PDE Method

We already defined signature in the above part, what we are going to define is our main method, signature kernel. Similar to the kernels we have introduced before, signature kernel is presented in the form of dot product as well. In the following parts, we will denote by $C^1(I, V)$ the space of continuously differentiable paths defined over an compact interval $I = [s, t]$ and with values on Banach space V. Now we give the definition of signature kernel.

Definition 3.2.9. Let $I = [u, u']$ and $J = [v, v']$ be two compact intervals, and let $x \in C^1(I, V)$ and $y \in C^1(J, V)$. The signature kernel $k_{x,y} : I \times J \to \mathbb{R}$ is defined as

$$k_{x,y}(s,t) = \langle S(x)_s, S(y)_t \rangle .$$

This is theoretical definition of signature kernel, however, this can not be used practically, hence we are going to introduce a *PDE approach*, relating the signature kernel to the solution of a linear hyperbolic partial differential equation(PDE) [9].

Theorem 3.2.10. *Let $I = [u, u']$ and $J = [v, v']$ be two compact intervals, and let $x \in C^1(I, V)$ and $y \in C^1(J, V)$. The signature kernel $k_{x,y}$ is a solution of the following linear, second order, hyperbolic PDE:*

$$\frac{\partial^2 k_{x,y}}{\partial s \partial t} = \langle \dot{x}_s, \dot{y}_t \rangle_V \, k_{x,y}, \quad k_{x,y}(u, \cdot) = k_{x,y}(\cdot, v) = 1, \tag{3.2.1}$$

where $\dot{x}_s = \frac{dx_p}{dp}\Big|_{p=s}, \dot{y}_t = \frac{dx_q}{dq}\Big|_{q=t}$ are the derivatives of x and y at time s and t, respectively.

The information in sequential data is often extracted in the form of complex data streams, with values in non-trivial *ambient spaces* [9]. A clever strategy of training would be to first *lift* the data corresponded or underlying ambient space to higher-dimensional, possibly infinite-dimensional, *feature space* through some *feature maps* on the static data(RBF, Sigmoid, etc.), this is the reason why the kernel here is called *static kernel*, and then consider the signature kernel of this lifted path as the final training kit [21]. It is then natural for us to ask a question about the practicability of this idea: can we compute this signature PDE kernel of this lifted path from the static kernel associated to the feature map being used. Király and Oberhauser (2019) [21] have proposed an algorithm to perform the procedure described, which we will deliver the explanation in the sense of Banach spaces and PDEs in the next part. However, before we study how to compute this kernel computationally, we would like to introduce a crucial property of signature called *factorial decay*, which helps to assure the convergence of some limiting processes [9].

Lemma 3.2.11. *For any path X of finite p-variation ($p > 1$), for any $k \geq 1$,*

$$\left\| \int_{s<u_1<\cdots<u_k<t} \cdots \int_{V^{\otimes k}} dx_{u_1} \otimes \cdots \otimes dx_{u_k} \right\|_{V^{\otimes k}} \leq \frac{\|x\|^k_{p,[s,t]}}{k!},$$

Now let us start to learn the procedure of how we are able to solve the PDE in Theorem 3.2.10 numerically. We first note that a kernel can be defined with a pair of maps from a set $\mathcal{X}$ to a Banach space E and its topological dual E^*. Denote this pair of maps by $\phi : \mathcal{X} \to E$ and $\psi : \mathcal{X} \to E^*$, the kernel then leads to a function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which is the natural pair between a Banach space and its dual:

$$\kappa(a, b) := (\phi(a), \psi(b))_E \quad \text{for all } a, b \in \mathcal{X}.$$

E here is assumed to be a Hilbert space as usual, this is to assure that ψ can be taken to be the composition $e \circ \phi$, where $e : E \to E^*$ is the canonical isomorphism interpreted in the *Riesz*

representation theorem. However, in the general framework which we often consider, the assumption of E being a Hilbert space is unnecessary. In the general case, for two paths $x : I \to \mathcal{X}$ and $y : J \to \mathcal{X}$, with $I = [u, u'], J = [v, v']$, can be lifted to paths on E and E^*, respectively, in the form of following:

$$X_s = \phi(x_s), \quad Y_t = \psi(y_t) \qquad \text{for all } s \in I, t \in J$$

Now if we make the assumption of continuity and bounded variation on the paths X and Y, then their signatures are well defined and belong to $T(E)$. By the results Terry Lyons (2007) presented, $T(E)$ is again a Banach space with $T^N(E)^* \cong T^N(E^*)$ for any $N \geq 1$ [19], which ensures the signature kernel starting with a kernel κ on $\mathcal{X}$ is well defined:

$$k_{X,Y}(s,t) = (S(\phi \circ x)_s, S(\psi \circ y)_t)_{T(E)} = (S(X)_s, S(Y)_t)_{T(E)}.$$

Now Theorem 3.2.10 can be applied if we, furthermore, assume X and Y belong to class C^1, and we have the following PDE:

$$\frac{\partial^2 k_{X,Y}}{\partial s \partial t} = \left(\dot{X}_s, \dot{Y}_t\right)_E k_{X,Y}, \quad k_{X,Y}(u,\cdot) = k_{X,Y}(\cdot, v) = 1. \tag{3.2.2}$$

The partial differential equation 3.2.2 can be solved numerically, i.e. expressed in terms of the static kernel and its corresponding underlying paths x and y solely in the following way, if we approximate the derivatives $\dot{X}_s, \dot{Y}_t$ with first order differences:

$$\begin{aligned}\frac{\partial^2 k_{X,Y}}{\partial s \partial t} &= ((X_s, Y_t)_E - (X_{s-1}, Y_t)_E - (X_s, Y_{t-1})_E + (X_{s-1}, Y_{t-1})_E)\, k_{X,Y} \\ &= (\kappa(x_s, y_t) - \kappa(x_{s-1}, x_t) - \kappa(x_s, y_{t-1}) + \kappa(x_{s-1}, y_{t-1}))\, k_{X,Y}.\end{aligned}$$

Using notations Király and Oberhauser (2016) have used [22], given two differentiable paths σ, τ, the truncated signature kernel at level M, $k^{\oplus}_{\leq M}(\sigma, \tau)$ satisfies the equation

$$k^{\oplus}_{\leq M}(\sigma,\tau)_{u,v} = 1 + \int_{(s_1,t_1)\in(0,u)\times(0,v)} \int_{(s_M,t_M)\in(0,s_{M-1})\times(0,t_{M-1})} \left(1 + \cdots + \int_{\sigma,\tau} (s_M, t_M)\right) d\kappa_{\sigma,\tau}(s_1, t_1),$$

where $d\kappa_{\sigma,\tau}(s,t) = k(\dot{x}_s, \dot{y}_t)\, dsdt$ and where k is a kernel on the ambient space of the paths. Note that it is in fact a recursive relation in the sense that the first integrand in the last equation is itself a truncated signature kernel which truncated at level $M-1$, hence we have the following recursive equation:

$$k^{\oplus}_{\leq M}(\sigma,\tau)_{u,v} = 1 + \int_{(s_1,t_1)\in(0,u)\times(0,v)} \int^{\oplus}_{\leq M-1} (\sigma,\tau)_{s_1,t_1} d\kappa_{\sigma,\tau}(s_1, t_1). \tag{3.2.3}$$

The untruncated signature kernel exists due to the former factorial decaying property (Lemma 3.2.11) of the signature, hence the limiting process in the equation 3.2.3, the term $k^{\oplus}_{\leq \infty}(\sigma,\tau)_{u,v}$ converges uniformly. In other words, the untruncated signature kernel is by taking the limit $M \to \infty$ in 3.2.3 since if two uniformly convergent sequences are equal at all finite levels M, they

are equal at the limit as well:

$$k^{\oplus}_{\leq\infty}(\sigma,\tau)_{u,v} = 1 + \int_{(s_1,t_1)\in(0,u)\times(0,v)} \int^{\oplus}_{\leq\infty} (\sigma,\tau)_{s_1,t_1} d\kappa_{\sigma,\tau}(s_1,t_1)\,. \tag{3.2.4}$$

And finally if we differentiate 3.2.4 both sides we obtain our original PDE in theorem 3.2.10.

3.3 Other Classification Methods

Last but not least, knowing whether our method is efficient or not, it is necessary to do some contrasts. Hence, in this paper, there are some other classical classification methods being included, including *Random Forest classifier*, *XGBoost classifier*, etc. These are introduced as base methods or the benchmarks to compare with the results obtained by our main method, i.e. signature kernels, and we will see the comparison of the results in terms of different performance metrics in next chapter. Lastly, we are going to compare the results between *RBF kernel* and *signature kernel* using RBF kernel as our static kernel.

Chapter 4

Implementation and Results

In this chapter, we would like to give a description of the methods mentioned in Chapter 3 before we going on to the details of the performance metrics and the numerical results we have obtained from different models and interpret their practical meaning in terms of the problem we are working on, i.e. economic recession prediction.

4.1 Implementation

All the pre-processing of data and implementations of models in the whole project were done so by $Python$, with $Numpy$, $Pandas$, $Scikit-learn$, $Tensorflow$ and $Sig-kernel$ (mentioned in Thomas Cass et al. (2021) [9]) libraries appropriately used. In the whole procedure, we first need to select appropriate families of models as our final comparing framework, i.e. choosing convincing feature or combination of features, to train and compare in terms of the correlations between each data series and their practical meaning to the indicator of economic recession, which coincides the way we are going to implement during the choosing procedure.

Secondly, trying not to overfit training data has been a crucial element in supervised learning as always, such that the model generalises well when applied under a realistic scenario. Models possibly fit very well and perform exceptionally up to mark on the data they have been trained on, they, however, in most cases, perform significantly worse when facing new incoming data, i.e. to make new predictions. Moreover, it is not enough to refine our model by including this new data point in the training set, occurrence of overfitting still exists. Therefore, we utilise a function named $train_test_split$ built in the $scikit-learn$ library, splitting our data into 7:3 ratio, i.e. take 30% of the data as test set, then shuffle it (although signature is path, here time dependent, we used a rolling window scheme, hence our training data is in the form of windows, shuffling will not affect the results) to overcome overfitting and make the model more robust. The detailed procedure would be firstly splitting whole data set into a *training set* and a *test set*, then performing same procedure on the training set to obtain a training set and a *validation set*. The benefits of going

through this process are: first we have a large set to train on, and we have our validation set to refine our model, i.e. to tune our hyperparameters, second we have our test set for final evaluations and results.

Last but not least, since the whole work is a task of supervised learning, we have labels, i.e. our binary indicator of recession period, and we need to adjust the number of our performance metrics instead of using *accuracy* as the only measure due to the sparseness of the label (that means even if we predicted all 0's the final predicting accuracy is around 85 percent). Hence, some other measures will be introduced, to cope with the situation, i.e. to mitigate the effects that sparseness brought.

4.1.1 Model Selection

Before we start to select best-in-class models, we first need to make ourselves clear about some terminologies, i.e. the concepts of a *model*, an *algorithm*, and a *classifier*. The three concepts can be difficult to distinguish, in which these three terms are often interchangeably used. Moreover, a bunch of conventions in the fields of economics and finance that does not coincide with that used in the fields from which machine learning methods are derived. Now we give our detailed explanation:

- ***Model*** Model stands for a specific feature or some combinations of features, which can possibly transformed, representing some underlying economic systems, for instance, in linear regression, we have our specific feature or variable standing for the specification, or during training some machine learning algorithms, model can be collection of features to which treated as one into it. Sometimes the name *hypothesis* is endowed as well.

- ***Algorithm*** Algorithm is sort of a strategy or the realization of our model or hypothesis and using given data observations to approximate some target functions. In the case of this paper, the target function is future recession probability or equivalently, the binary label mapping to a data point. Examples of algorithms for classification would include random forest, XGBoost, neural networks or any other machine learning methods. In practice, algorithms often exist in the form of software packages, so that researchers are able to apply models and data and to create classifiers.

- ***Classifier*** Classifier is our final result of applying models and data on to an algorithm of classification. In contrast to the former two relatively abstract concepts, classifiers themselves are more specific and they are the tangible machines that researchers work on. Also, if we would like to compare the performance of two models or two algorithms, we must have two estimated classifiers and the comparison must be made through samples classified by them.

In our paper, we study a bunch of nested families of models, which is what we will do in this section. Firstly, we denote the values of each feature at month t by $Slope_t$, $NFCI_t$, EBP_t, FF_t,

$SP500_t$, ADS_t, SPF_t, $ACM5D_t$. After this, as we mentioned in Chapter 2, we define our response indicator by $NBER_{t+1,t+p}$, i.e. takes value 1 if one of the following p months is contained in NBER-dated recession periods and is 0 otherwise. Finally, we denote $SIG()$ as our signature kernel algorithm over a combination of features, denote $RBF()$ as support vector machines with RBF kernel classifier algorithm over a combination of features, denote $SVM()$ as support vector machines (linear kernel) classifier algorithm over a combination of features, denote $RF()$ as random forest binary classifier algorithm over a combination of features, and denote $XGB()$ as an XGBoost binary classifier algorithm over a combination of features.

First, we choose univariate models in each of the eight features, choosing the best-in-class feature. In the case of the *Slope* model, the structure of the batch of the univariate models using different algorithms would be:

$$\begin{cases} Pr_{Signature\ Kernel}(NBER_{t+1,t+p} = 1) = SIG(Slope_t) \\ Pr_{RBF\ Kernel}(NBER_{t+1,t+p} = 1) = RBF(Slope_t) \\ Pr_{Support-vector\ Machine}(NBER_{t+1,t+p} = 1) = SVM(Slope_t) \\ Pr_{Random\ Forest}(NBER_{t+1,t+p} = 1) = RF(Slope_t) \\ Pr_{XGBoost}(NBER_{t+1,t+p} = 1) = XGB(Slope_t) \end{cases}$$

Secondly, we combine our slope feature with one of the rest seven features as our new bivariate model, choosing the best-in-class combinations. In the case of *Slope* and ADS model, the structure of the batch of the bivariate models using different algorithms would be:

$$\begin{cases} Pr_{Signature\ Kernel}(NBER_{t+1,t+p} = 1) = SIG(Slope_t,\ ADS_t) \\ Pr_{RBF\ Kernel}(NBER_{t+1,t+p} = 1) = RBF(Slope_t,\ ADS_t) \\ Pr_{Support-vector\ Machine}(NBER_{t+1,t+p} = 1) = SVM(Slope_t,\ ADS_t) \\ Pr_{Random\ Forest}(NBER_{t+1,t+p} = 1) = RF(Slope_t,\ ADS_t) \\ Pr_{XGBoost}(NBER_{t+1,t+p} = 1) = XGB(Slope_t,\ ADS_t) \end{cases}$$

And we continue this process recursively, so that we find best-in-class 3-, 4-, 5-, 6-, 7- and 8-feature models by combining the features occurred best-in-class models in the previous batch and the remaining features in terms of performances. It worths to mention that the majority of the models that combine some of the Slope, ADS, EBP, SP500, FF and SPF features have been studied previously in the literature.

4.1.2 K-fold Cross Validation

As previously mentioned, overfitting a model to the training data has significantly adverse impact on the generalisation of the model itself. The occurrence of the model being optimal or performing

extremely well on the training data only while performs deficient on a new incoming data point is what we do not hope to see, hence the following method is implemented. The method we chose to get rid of this phenomenon is called *K-fold Cross Validation*. In K-fold cross validation, we first split the shuffled training data into k *folds*, i.e. k distinct subsets with identical, or close to identical, size. The training process is run through k times, leaving out one of the k fold or subset as our validation set and running training process on the rest $k-1$ folds or subsets each time, then validate on the validation set. To evaluate this model as a whole, we first compute the performance metric of each of these models and then calculate the arithmetic mean from these k results as the final result. We give following the graphic interpretation of the k-fold cross validation procedure (Source: S. Raschka (2018) [23]).

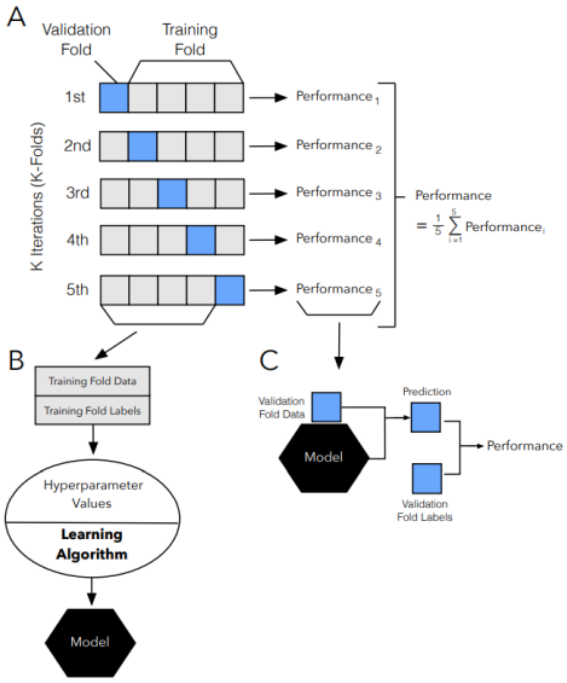

Figure 4.1: An illustration of the k-fold cross-validation procedure with $k = 5$

In this procedure, the number k, the number of folds split, is itself a hyperparameter, in other words, it can affect the performance of the model and hence, in our training, different values of k are included in our grid search and thus to find the most optimal model, where we will introduce in next part. Normally, k is taken as 5 for computationally ease, and for larger values of k, the training time would be a lot longer whereas any values of k smaller than 5 would have less data to train and thus *pessimistic bias* of performance metric would rise and in the meantime, the increase in *variance* arises due to the fact that it is more sensitive to the way of splitting data.

Another issue worths to mention is about the case when facing *imbalanced data*, in other words, there exists bias in the dataset of label towards one class over the other. The conventional means of splitting data in k-fold cross validation procedure might have some problems because the default setting is to treat data points as balance ones. In our case, the recession period (Class 1) rarely happens, hence is one of the cases of imbalance, and we are going to implement a modified version of k-fold cross validation called *StratifiedKFold* existing in *scikit-learn* library.

4.1.3 Hyperparameter Tuning

The *hyperparameters*, they stand for the tunable parameters that affect the results, and are what we mainly work on when tuning the model since these parameters are fixed in the beginning of training and cannot be learned during the training process. Hence, we must find the optimal values of these hyperparameters to give a best performing model. We use the standard method called *grid search* to find the best combination of hyperparameters. A cross validation is implemented for each grouping of the hyperparameter space and the best-in-class model is selected based on which model performs best according to some measures, and in our case, the measure, or the performance metric, is accuracy of each prediction made.

4.2 Evaluation

4.2.1 Performance Metrics

Confusion Matrix

When it comes to performance metrics, we would like to first give the concept of *confusion matrix*, which is an intuitive way of interpreting how the learning algorithms performs during the training process. Since our task is binary classification, the corresponding confusion matrix is a 2×2 matrix whose rows are the predicted results of the trained models and the columns are the actual classes of each instance.

		Actual Class		
		Positive	Negative	Total
Predicted Class	Positive	TP	FP	$TP + FP$
	Negative	FN	TN	$TN + FN$
	Total	$TP + FN$	$FP + TN$	N

Table 4.1: A Confusion Matrix

As presented in the table above, the elements in the confusion matrix can be described as follows:

- *True Positive (TP)* The number of instances correctly predicted as positive by the trained model.

- *False Positive (FP)* The number of instances incorrectly predicted as positive by the trained model, also known as Type I error.

- *True Negative (TN)* The number of instances correctly predicted as negative by the trained model.

- *False Negative (FN)* The number of instances incorrectly predicted as negative by the trained model, also known as Type II error.

The confusion matrix allows us to see exactly where the instance is classified and the existence of bias towards one class to another. However, when it comes to *accuracy*, which we will give more detailed discussion next, is considered as the performance metric in most of the models, one of mostly happened problem is that the performance of a model purely based on performance can lead to some "catastrophe" when encountered with imbalanced data. As we mentioned in the very beginning, our label, or the binary y-variable, is extraordinarily sparse (with majority of zeros), hence is a typical example of imbalance. This is not to say we are going to dispose of accuracy in the following work, but instead examining the confusion matrix carefully can be beneficial and productive in understanding how a model behaves.

Accuracy

Accuracy is one of the most natural performance metrics to consider, and is calculated by the number of instances correctly classified over the whole sample size. Mathematically, it is calculated as follows:

$$Accuracy \quad = \quad \frac{TP + TN}{TP + TN + FP + FN}$$

When cope with balanced data, accuracy is, undoubtedly, one of the most sensible and convincing measures on how a model is performing.

Precision & Recall

Precision can be understood as the proportion of the data in the positive class which were correctly classified, here we give the formal definition:

$$\text{Precision} \quad = \quad \frac{TP}{TP + FP}$$

Fundamentally, high precision refers to the high qualitative ability of one predictive model to accurately predict the positive instances in the positive class, i.e. among those data points that are classified as positive. In the context of our model we are going to build, a high precision would mean our predictive model is highly successful in predicting a recession period.

Another metric that usually goes hand in hand with precision is *recall*, which is computed similarly

to precision, and here we give its formal definition:

$$\text{Recall} \quad = \quad \frac{TP}{TP + FN}$$

Essentially, it is the ability of a model explicitly classifying the positive instances as positive. In terms of our problem, high recall means that our predictive model is able to capture the information or signal of recession, which is a promising metric in this sense.

Achieve either a high precision or a high recall is relatively not tough for a trivial model, this is because even if a model classify all the data points as positive, unsurprisingly, we will achieve a 100% recall with this "retarded" model. However, the model described above is not a model we want for certain, it is therefore not enough to assess a model based on purely precision or recall, and it is necessary to combine these two measures as a whole. Thereafter, we are going to introduce another performance metric called F_1 *score*, which will be looked at further in the following part.

F_1 Score

As we have discussed in the previous section, we are going to consider both precision and recall in the meantime, one of the measures to consider is F_1 *score*. From the documentation of scikit-learn metrics, the F_1 score can be interpreted as a harmonic mean of the precision and recall, where an F_1 score reaches its best value at 1 and worst score at 0 [24]. Here we give the mathematical definition of it:

$$F_1 \text{ Score } = 2 \times \frac{\text{Precision } \times \text{ Recall}}{\text{Precision } + \text{ Recall}}.$$

It is intuitive that the relative contribution of precision and recall to the F_1 score are equal by the definition. Another question may arise, is equal weights sufficiently generalised concerning all the situations? The answer is definitely not, and can we have a metric that enable us to distribute different weights on the respective metrics? The answer is fortunately yes. This metric is called F_β score, which is defined as:

$$F_\beta \text{ Score } = \left(1 + \beta^2\right) \times \frac{\text{Precision } \times \text{ Recall}}{(\beta^2 \times \text{ Precision }) + \text{ Recall}}$$

Different weightings of either precision or recall is sometimes essential depending on what kind of problems to be solved. In the F_β score, β is defined to be the weight attributed to precision, i.e. it is the scale by which we consider recall β times more important than precision.

4.2.2 Classical Classification Methods

We now present the results of stratified K-fold cross validation across all the the best performing models trained by signature kernels and RBF kernels respectively, along with their comparison. As mentioned in the previous section 4.1.2, we made the use of the splitting method of stratified k-fold in the process of cross validation to get over the problem of imbalance existing in our data, hence we

are still going to use accuracy in the test set, in other words, the out-of-sample mean accuracy, to score the models and the algorithms, which is sensible. Following the procedure we have interpreted in the previous section of model selection, we have made our choices of the following 'best-in-class' nested family models starting from 3-feature models with 6 months of prediction:

# Features	Model
3	Slope/ADS/ACM5D
6	Slope/ADS/NFCI/EBP/ACM5D/SP500
7	Slope/ADS/NFCI/EBP/ACM5D/SP500/USREC
8	Slope/ADS/NFCI/EBP/ACM5D/SP500/USREC/FF

Table 4.2: "Best-in-Class" Nested Family Models, where all the subsequent analysis of results are based on the classifiers trained from these models listed.

After an thorough search, we find there is no such an appropriate global benchmark to be compared for the time series performance metrics, therefore, for the aiming to make some comparisons, we have introduced the naive prediction, which is a vector all zeros, i.e. assume there is no future recessions since we are not experiencing recessions during most of the periods, to see if our tuned models do have some contributes to the forecasts and performs better. Table 4.3 shows the results of naive model, it worths to mention that we set F Score as 0 with both precision and recall being 0, this can be understood as the limiting performance of F Score when precision and recall are both approaching 0.

	Accuracy	Precision	Recall	F Score
Naive Model	0.851852	0	0	0

Table 4.3: Results of the naive model

Table 4.4 presents the performance scores from the best-in-class models trained by classical algorithms, and we will compare them to the naive model, to see how well our actively tuned models are performing quantitatively. By the first glance, we are able to conclude that all the models perform better than the given benchmark concerning all the aspects, which proves some evidence of feasibility of the choices of the features in the model and the algorithms we use. It is intuitive that 7-feature models perform best across the three algorithms, except for SVM, in terms of F Score, 3-feature model was bested at 72%, but this is not a far lead, though. Hence, we can conclude that, of our classical methods, the information embedded in 7 features was extracted most.

Another point worths to mention is that the difference between the performances of 3-, 6-, and 7-feature models are not significant across the three methods. This means that in fact, the three classical methods, do not manage to extract the information embedded in the other three or four data series efficiently, this can be caused by the simplicity of the methods themselves. And the results are going far worse when another feature *FF* being added in the model, especially for random forest, not only the accuracy has decreased to 87%, but the F Score has dropped

to 22% with about 50% gap from the 7-feature model. This phenomenon, in some way, provides evidence of the inefficiency of the classical methods in terms of reading or studying high dimensional information.

Model	Algorithm	Accuracy	Precision	Recall	F Score
Slope/ADS/ACM5D	Random Forest	0.888889	0.625	0.625	0.625
Slope/ADS/NFCI/EBP/ ACM5D/SP500	Random Forest	0.925926	1.0	0.5	0.666667
Slope/ADS/NFCI/EBP/ ACM5D/SP500/USREC	Random Forest	0.938272	1.0	0.583333	0.736842
Slope/ADS/NFCI/EBP/ ACM5D/SP500/USREC/FF	Random Forest	0.870370	1.0	0.125	0.222222
Slope/ADS/ACM5D	XGBoost	0.870370	0.565217	0.541667	0.553191
Slope/ADS/NFCI/EBP/ ACM5D/SP500	XGBoost	0.907407	0.736842	0.583333	0.651163
Slope/ADS/NFCI/EBP/ ACM5D/SP500/USREC	XGBoost	0.925926	0.875	0.583333	0.7
Slope/ADS/NFCI/EBP/ ACM5D/SP500/USREC/FF	XGBoost	0.901235	0.785714	0.458333	0.578947
Slope/ADS/ACM5D	SVM(linear kernel)	0.913580	0.692308	0.75	0.72
Slope/ADS/NFCI/EBP/ ACM5D/SP500	SVM(linear kernel)	0.919753	1.0	0.458333	0.628571
Slope/ADS/NFCI/EBP/ ACM5D/SP500/USREC	SVM(linear kernel)	0.913580	1.0	0.416667	0.588235
Slope/ADS/NFCI/EBP/ ACM5D/SP500/USREC/FF	SVM(linear kernel)	0.895062	0.818182	0.375	0.514286

Table 4.4: Mean out-of-sample results of the 'best-in-class' models using classical algorithms

Last but not least, we found the performances of such classical models is not stable during the training process, in other words, the variances of the performance metrics are relatively high and the mean results cannot reveal this shortage. In practice, it is very "dangerous" to use these models to predict recession, since they will confuse us when we are making decisions. Now we are going to give a hypothetical setting to interpret how this danger can be caused. Imagine we are investors and we are investing both bond and equity, or stocks as usual, and we are trying to predict the economic situation in the following months in order to decide the allocation of our assets. If we are employing one of the classical methods, the results or outcomes can be variant and the possibility of forecasting reversely is not trivial, and such wrong predictions can lead us to some fatal errors, for instance, we would invest more in growth stocks and less in valuable stocks and bond after

the outcome of our predictive model showing that the next investing period is not in recession. However, if the true situation is opposite to what we have predicted, we are going to experience huge losses and this is what we do not hope to see. Hence the shortages of these classical methods are obvious and we are going to introduce another two methods and focus on our main method, signature kernels.

4.2.3 Signature Kernel & RBF Kernel

Following the results obtained from the classical methods, we proceed onto the outcomes by employing signature kernels based on static kernel of RBF kernel and that of implementing RBF kernel only. First we note that 7-feature model seems performed best across the three classical algorithms, and we will see if this is the case as well for these two models. Now we are going to present the mean out-of-sample results of all best-in-class models.

Model	Algorithm	Accuracy	Precision	Recall	F Score
Slope/ADS/ACM5D	RBF Kernel	0.925287	0.84	0.7	0.763636
Slope/ADS/NFCI/EBP/ ACM5D/SP500	RBF Kernel	0.942529	0.64	0.941176	0.761905
Slope/ADS/NFCI/EBP/ ACM5D/SP500/USREC	RBF Kernel	0.919540	0.68	0.739130	0.708333
Slope/ADS/NFCI/EBP/ ACM5D/SP500/USREC/FF	RBF Kernel	0.925287	0.48	1.000000	0.648649
Slope/ADS/ACM5D	Signature Kernel	0.931034	0.52	1.0	0.684211
Slope/ADS/NFCI/EBP/ ACM5D/SP500	Signature Kernel	0.925287	0.56	0.875000	0.682927
Slope/ADS/NFCI/EBP/ ACM5D/SP500/USREC	Signature Kernel	0.948276	0.68	0.944444	0.790698
Slope/ADS/NFCI/EBP/ ACM5D/SP500/USREC/FF	Signature Kernel	0.948276	0.68	0.944444	0.790698

Table 4.5: Mean out-of-sample results of the 'best-in-class' models using RBF kernel and signature kernel

The first obvious point to draw out of Table 4.5 is that these two algorithms, in general, perform a lot better than the classical methods we have mentioned in the previous section in terms of every performance metric. Even the lowest accuracy of signature kernel has reached more than 92.5%, and this is almost the same as the best performed model across all the classical algorithms, RBF kernel works better than the previous methods as well, not as significant as signature kernel, though.

Now we move to the comparison between the two methods explained in this section. Immediately we can conclude that in terms of accuracy, signature kernel is better performing, across most choices of feature set(only for 6-feature model, the out-of-sample accuracy trained by RBF kernel is about 1.7% higher than that of signature kernel). This means we are able to identify the length of recession period more accurately using signature kernels and in practice, this means we are enabled to decide whether to buy or sell different assets more precisely and hence make more profits. As for the rest 3 metrics, especially F score, we can see that for signature kernel models, the performance is getting better along with more of the features, this means signature kernel models succeeded in identifying and extracting the information embedded in new relevant data series. While for RBF kernels, as the number of features increases, the out-of-sample performances of which become worse and worse, however, the in-sample performance of RBF kernels, as shown in Table 4.6, tend to be perfect as the number of features increases, in contrary to the out-of-sample performance, these such 'brilliant' results are pointing to the problem of overfitting existing in this method. Moreover, if we move our eyes onto the precision and recall, we can note that in terms of recall, signature kernels are bested in most of the cases(except for the 6-feature model, and this is the case for every metric here, so basically signature kernel does not perform well in this case), while as for precision, the situation is reversed. This means signature kernels are more conservative than RBF kernels while the capability of reading and capturing the signal of recession is stronger, and in practice, we definitely prefer the models being more conservative and faultless since almost every investor is adverse to risk or variance.

Model	Algorithm	Accuracy	Precision	Recall	F Score
Slope/ADS/ACM5D	RBF Kernel	0.963325	0.904762	0.915663	0.91018
Slope/ADS/NFCI/EBP/ ACM5D/SP500	RBF Kernel	1.0	1.0	1.0	1.0
Slope/ADS/NFCI/EBP/ ACM5D/SP500/USREC	RBF Kernel	1.0	1.0	1.0	1.0
Slope/ADS/NFCI/EBP/ ACM5D/SP500/USREC/FF	RBF Kernel	0.99511	0.988095	0.988095	0.988095

Table 4.6: Mean in-sample results of the 'best-in-class' models using RBF kernel

We have also plotted the probability curves of predictions made by both signature kernels and RBF kernels, we will show the cases of 6-, 7- and 8-feature models in Figure 4.2, 4.3 and 4.4, where the light grey shaded areas are the NBER-dated recession periods and the light blue shaded areas are the months of prediction before the dated recessions. It is clear that in the context of 6-feature model, both methods failed to predict the second period of recession(which is the great recession caused by COVID-19), while for 7- and 8-feature setting, signature kernels did predict the recession, showing that signature kernels are robust encountered with higher dimensional data

while RBF kernels predicted successfully only for 7-feature setting, this coincides with the case of best performing with seven features as we have concluded in the previous section for classical methods. For other months of prediction period, we have also plotted the cases, and these figures are presented in the appendix.

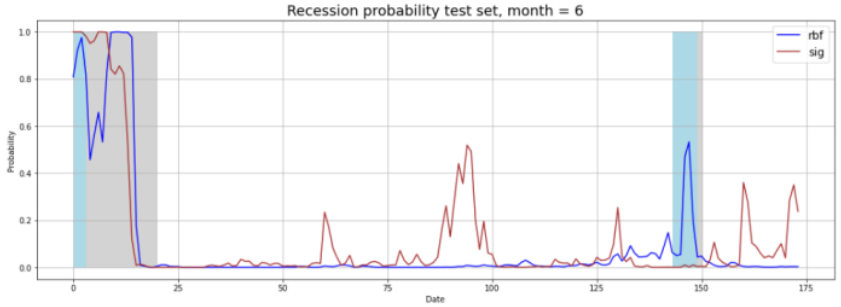

Figure 4.2: Probability curves predicted by RBF kernels and signature kernels(6-feature model)

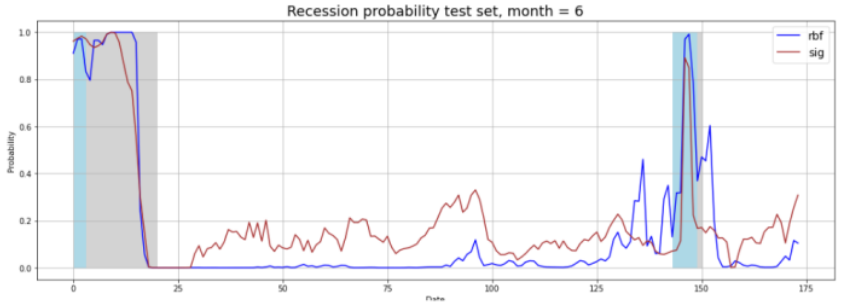

Figure 4.3: Probability curves predicted by RBF kernels and signature kernels(6-feature model)

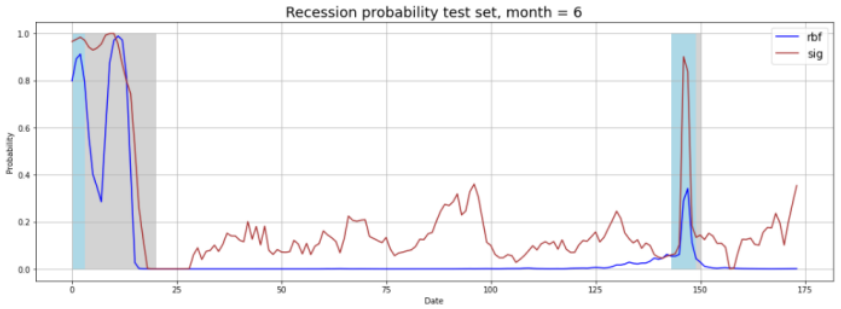

Figure 4.4: Probability curves predicted by RBF kernels and signature kernels(6-feature model)

A closer look at the plots reveals that in most cases, there are more peaks or so called noises existing in the periods of non-recession in the signature model with seven and eight features, with probability less than 50%, this means signature model is telling the investor to keep an eye on the potential recession in the future and prepare for it. In reality, with regard to the second recession

period, it is the great recession attributable to COVID-19, signature model is telling the possibility of occurrence of recession at the moment of start of the global pandemic. However, when it comes to RBF models, conditioned on seven and eight features as well, RBF models either give wrong signals of recession before and after the actual periods of recession(the second recession period in the test set) or even failed to capture the signal of the recession.

Lastly, it is interesting to mention the forecasts made by our signature kernel model, using the most recent data. We found the results are telling that, in the next period of prediction, we are in a relatively stable financial condition. The results predicted above are, in fact, fairly coincident with what we roughly see the overall trend of US economy.

Chapter 5

Conclusion

In this thesis, we have coped with the problem of predicting economic recession periods by employing signature kernels, which is the main method in this paper, at the same time as RBF kernel, support vector machine, XGBoost and random forest to be compared with reference to a set of relevant performance metrics. It is evident that signature kernel with static kernel set as RBF kernel performed consistently better than RBF kernel and all the other classical methods, achieving the highest mean out-of-sample accuracy of 95% with seven and eight feature set which was significantly better than the results of both RBF kernel and classical algorithms. Of the other methods, RBF kernel with the set of six features performed the best, with an accuracy achieving at 94%, beat all of the models trained by classical algorithms, this proved that, in some way, our choice of static kernel was appropriate as well. Overall, we have summed up several factors that can potentially explain the reason why the classical algorithms are performing worse than our signature kernels with static kernel set as RBF kernel, or even RBF kernels only:

- The model structures are too simple to capture the complicated market information and potential signals rooted in macroeconomic indexes and thus predict the trends of the whole economy.
- The grid search in the hyperparameter space of classical algorithms is too shallow, hence difficult to make precise decisions.
- The dimension of data is relatively high and is sequential, while the classical algorithms are not pathwise dependent, therefore it is very tough for them to get the following forecasts.

On the other hand, since signature depends on the path of the data, it is natural that signature kernel inherited this property(since signature kernel is simply the inner product of two signatures), thus it is ideal for sequential data problems. The results showed above, indeed, did not disappoint us with surprisingly high accuracy and F score. In particular, we found modified NBER-dated recession data, i.e. the labels we are going to predict, provided deep insight into the movements of itself, contributing remarkably to the overall results. Additionally, on the subject of 8-feature model, all the methods have given worse results than those of 7-feature models, except for signature

kernel, kept the outcomes at the same level, which means signature kernel is more robust than the others, where the other methods tend to overfit as the dimension and complexity of data growing up.

Upon reflection, there are four directions I would like to propose for further research and studies. The first direction comes to my mind is to seek a more efficient, intelligent grid search method to find the optimum hyperparameters since during the coding work, it took a lot of time looping every instance and computationally expensive. In a great number of machine learning problems, Bayesian optimisation has been chosen as the optimisation method due to its effectiveness, and thus it is an ideal starting point for a more efficient global optimisation method. Secondly, the dimension of feature set is relatively low(models with the highest dimension is 8), and some other macroeconomic data series, such as GDP growth rate, unemployment rate, inflation rate, etc. should be included in our search. Thirdly, there are some further investigation of comparing the methods other than simply splitting the dataset with some ratio and giving the performance metrics. To make the comparison more statistically significant, we can consider the ideas of nested analysis when training the dataset. In other words, we first employ the methods and train the models in the period of 1973(where the data series start from) to 2007, say, and set the test data as the period from 2007 to 2017, and we have only one recession period during this time. After training this, we have an accuracy, then we retrain the models using data from 1973 to 2017, with data from 2017 to 2022 to be tested, that implies the training set which aims to test the great recession aroused by COVID-19 will take 2009 recession into account. Last but not least, there is a hyperparameter called *dyadic order* during training signature kernels, which can be understood as how close the numerical results calculated from the hyperbolic PDE and the real signature kernel are, a higher dyadic order means they are closer. Due to time constraints, we have only tested the results with dyadic order zero and one. Given more time, it would be definitely beneficial to test the results with higher dyadic order which will improve the present results with high probability. These would be the next step in improving the current research proposed.

Appendix A

Plots of other months of prediction

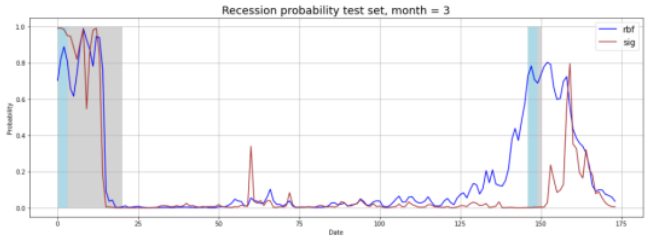

Figure A.1: Probability curves predicted by RBF kernels and signature kernels(3-feature model with 3 months of prediction)

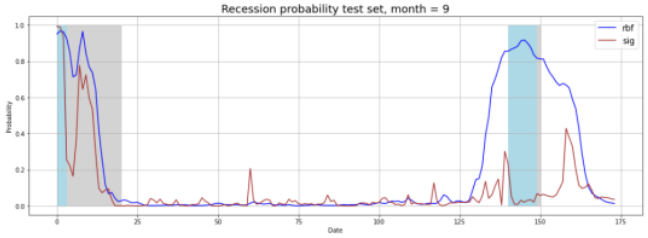

Figure A.2: Probability curves predicted by RBF kernels and signature kernels(3-feature model with 9 months of prediction)

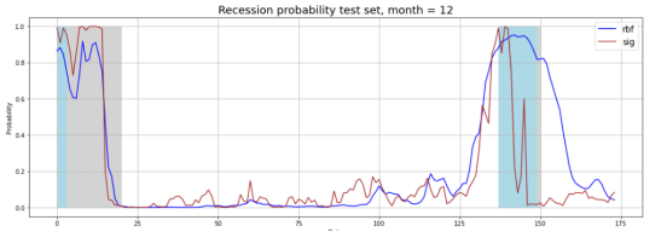

Figure A.3: Probability curves predicted by RBF kernels and signature kernels(3-feature model with 12 months of prediction)

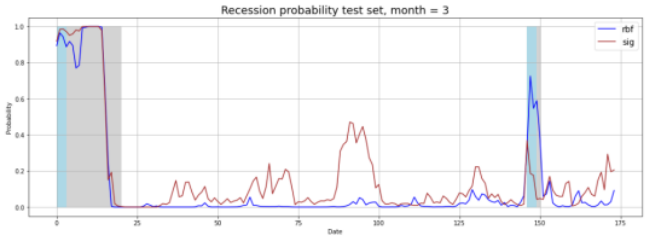

Figure A.4: Probability curves predicted by RBF kernels and signature kernels(6-feature model with 3 months of prediction)

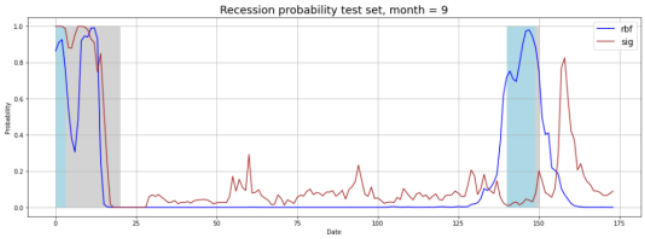

Figure A.5: Probability curves predicted by RBF kernels and signature kernels(6-feature model with 9 months of prediction)

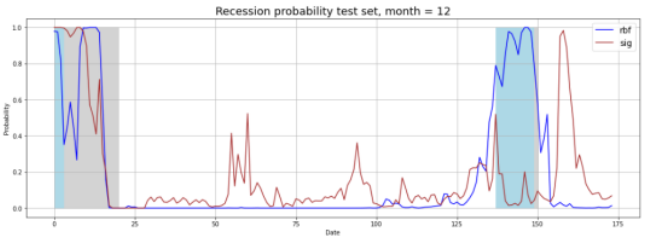

Figure A.6: Probability curves predicted by RBF kernels and signature kernels(6-feature model with 12 months of prediction)

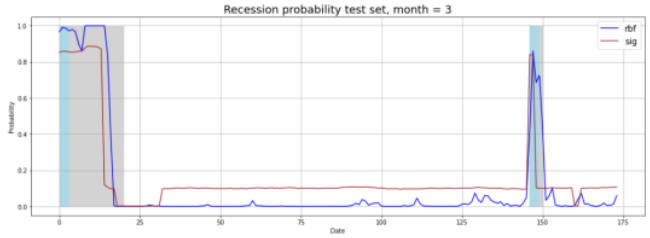

Figure A.7: Probability curves predicted by RBF kernels and signature kernels(7-feature model with 3 months of prediction)

Figure A.8: Probability curves predicted by RBF kernels and signature kernels(7-feature model with 9 months of prediction)

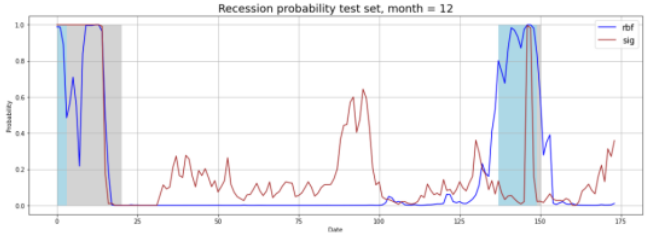

Figure A.9: Probability curves predicted by RBF kernels and signature kernels(7-feature model with 12 months of prediction)

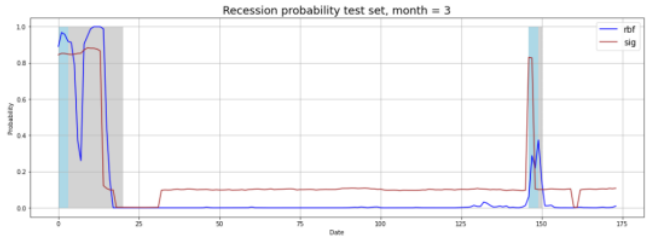

Figure A.10: Probability curves predicted by RBF kernels and signature kernels(8-feature model with 3 months of prediction)

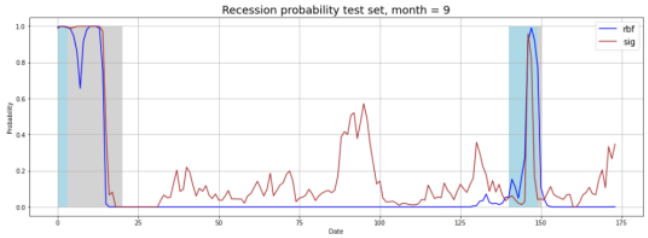

Figure A.11: Probability curves predicted by RBF kernels and signature kernels(8-feature model with 9 months of prediction)

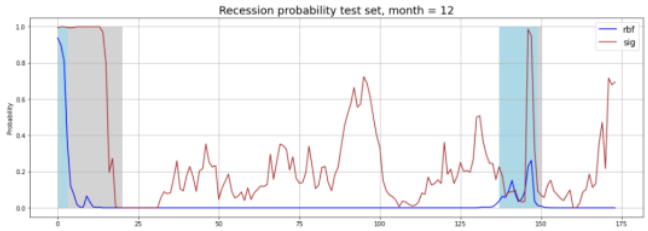

Figure A.12: Probability curves predicted by RBF kernels and signature kernels(8-feature model with 12 months of prediction)

Bibliography

[1] Mauricio Barahona, Florian Song, Felix Laumann, and Kevin Webster. Lecture notes in methods for data science, 2020.

[2] Gareth J. James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning : with applications in R.* Springer texts in statistics. Springer, New York, 2013.

[3] Cristopher Salvi. Lecture notes in rough paths in machine learning, 2022.

[4] Michael Puglia and Adam Tucker. Machine learning, the treasury yield curve and recession forecasting. *Finance and Economics Discussion Series 2020-038*, 2020.

[5] Ben L. Kyer and Gary E. Maggs. American experience with double dip recession in the early 1980's. *Springer*, 2016.

[6] R. Samsudin, A. Shabri, and P. Saad. A comparison of time series forecasting using support vector machine and artificial neural network model. *Journal of Applied Sciences*, 10:950–958, 2010.

[7] Bernhard Schölkopf and Alexander J. Smola. Kernels. In *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, pages 25–60. MIT Press, 2001.

[8] Marco Cuturi. Fast global alignment kernels. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 929–936, 2011.

[9] Cristopher Salvi, Thomas Cass, James Foster, Terry Lyons, and Weixin Yang. The signature kernel is the solution of a goursat pde. *SIAM Journal on Mathematics of Data Science*, 3(3):873–899, 2021.

[10] Arturo Estrella and Frederic S. Mishkin. Predicting u.s. recessions: Financial variables as leading indicators. *The Review of Economics and Statistics*, 80(1):45–61, 1998.

[11] Simon Gilchrist and Egon Zakrajšek. Credit spreads and business cycle fluctuations. *The American economic review*, 102(4):1692–1720, 2012.

[12] Jonathan H. Wright. The yield curve and predicting recessions. *Finance and economics discussion series*, 2006(7):1–19, 2006.

[13] Alex Graves. *Supervised sequence labelling with recurrent neural networks*. Studies in computational intelligence ; v. 385. Springer, Heidelberg ;, 2012.

[14] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 144–152, New York, NY, USA, 1992. Association for Computing Machinery.

[15] Bernhard Schölkopf and Alexander J. Smola. Pattern recognition. In *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, pages 189–226, 2001.

[16] Kristin P. Bennett and O. L. Mangasarian. Robust linear programming discrimination of two linearly inseparable sets. *Optimization methods and software*, 1(1):23–34, 1992.

[17] A Patle and D. S Chouhan. Svm kernel functions for classification. In *2013 International Conference on Advances in Technology and Engineering (ICATE)*, pages 1–9. IEEE, 2013.

[18] Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(12), 2006.

[19] Terry J. Lyons. Differential equations driven by rough paths. In *École d'Été de Probabilités de Saint-Flour, 1908*, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[20] Terry Lyons. Rough paths, signatures and the modelling of functions on streams. *arXiv preprint arXiv:1405.4537*, 2014.

[21] FJ Király and H Oberhauser. Kernels for sequentially ordered data. *Journal of Machine Learning Research*, 20, 2019.

[22] Franz J Király and Harald Oberhauser. Kernels for sequentially ordered data, 2016.

[23] Sebastian Raschka. Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808*, 2018.

[24] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score`.