ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Probabilistic Forecasting for Time Series Anomaly Detection

Master Thesis

Markus Chardonnet

Wednesday 7th June, 2023

Advisors: Florian Krach(1), Jakob Heiss(1), Mitch Gusat(2), Josef Teichmann(1)
Department of Mathematics, (1) ETH Zürich
Cloud Computing, (2) IBM Research Zürich

Abstract

This work introduces a new Multivariate time series Anomaly Detection method based on Probabilistic forecasting, the process of predicting the distribution of future values. It is based on the Path-Dependent Neural Jump ODE (PD-NJ-ODE) (Krach et al., 2022), a framework that is able to learn non-Markovian multidimensional stochastic processes forecasting, by learning on irregularly observed times series samples, and which theoretically converges to the optimal conditional expectation process. We take advantage of that framework to approximate the conditional distribution using learnt conditional moments. By employing the conditional distribution to automatically compute an anomaly score of time series observations, we implicitly provide an adaptive threshold on the forecasting error to raise anomalies. We exploit the ability to learn on irregularly observed time series in order to robustly learn forecasting with multiple forecast horizons. We design a simple yet interpretable anomaly detection module that raises per timestamp anomalies using the anomaly scores from multiple forecast horizons and timestamps. We demonstrate the effectiveness of our method on a tailor-made synthetic dataset, with multiple anomaly types.

Contents

Acknowledgement

First, I would also like to thank Prof. Josef Teichmann for giving me the opportunity to work with Stochastic Finance Group of ETH Zürich as well as Mitch Gusat, who allowed the collaboration with the Cloud Computing Group of IBM Research. Secondly, I would like to express my special thanks of gratitude to Florian Krach and Jakob Heiss for their guidance and support in completing my thesis. I would also like to sincerely thank my parents for supporting me during my master's thesis. Finally, I would like to thank my friends and roommates with which I had good times to free my mind.

Chapter 1

Introduction

The main Focus of this work is to approach the problem of Time Series Anomaly Detection with Probabilistic Forecasting. We aim at developing a method which is suited for the Application to Cloud System Data, but also extendable to other domains.

We propose a new approach for utilising forecasting towards anomaly detection. By forecasting, we mean the process of estimating future values conditioned on past observations, which is often referred in the literature as online forecasting (Krach et al., 2022). The idea behind this innovation is to account for the stochastic character of KPIs evolution through time, which arises from external factors such as human usage (technicians or customers), or internal factors such as measurement noise. We formalise this aspect with the predictive distribution. Instead of trying to predict future observations as in former forecasting methods, one aims at estimating the distribution of future observation. We claim that this novel approach is able to better asses the anomalousness of time series observations, as it can assess the likelihood of an observation. Furthermore, we utilise forecasting to its full ability by making several predictions with different time horizons for each observation.

Using the predicted distribution, we build an interpretable and flexible scoring scheme that assesses the anomalousness of time series observations. The scoring module can be configured to target different types of anomalies. Thanks to the simplicity of this method, one can perform multiple inferences with different scoring module configurations, without drastically increasing the time complexity.

Because of the lack of annotated data, we investigated leads to evaluate the performance of the designed method. For this purpose, we propose a synthetic data generating process which is configurable and incorporates some characteristics of the true data. Furthermore, we elaborate some leads

in order to incorporate different types of anomalies in our synthetic data. The resulting annotated datasets serve both for evaluating and tuning the designed method.

The main research questions we address are the following. Are we able to learn the expected evolution of Time Series throughout time, by accounting for the stochasticity? How can we use that knowledge to improve former Anomaly Detection methods? How can we design and tune the method to achieve targeted goals?

1.1 Project Description

1.1.1 Cloud Data Centers

Cloud and datacenters are large aggregations of computer systems (servers, storage, networks), each virtualized and monitored in real time by system characteristics, called key performance indicators (KPIs), producing a large number of data streams (our multivariate time series dataset). Anomaly detection (AD) in such systems is a challenging problem due to the high dimensionality of the sequential data involved and the lack of labels, hence unsupervised learning. The different data streams arising from different sensors / measures "show" relationships as they monitor parts of one system with *physical, virtual and logical* connections.

1.1.2 Data description

The data consists of multivariate time series originating from the real-time monitoring of cloud systems. The principal usage of these cloud systems is storage, and most of the monitored components have a role in the storage organisation. Systems are built in a hierarchical way and include components such as I/O groups, Pools, MDisks, Disks and Volumes. For each of those components, one monitors metrics such as Response times, Transfer Sizes, I/O rates. These metrics, and their respective time series are called KPIs. The time series consists in regular samples of the metrics with a common period of 5 minutes. Furthermore, time series are scaled to have values lying in the $[0, 1]$ range.

The multivariate time series are a concatenation of univariate time series monitors corresponding to individual KPIs. It is important to consider the data in a multivariate fashion as there are multiple interactions between KPIs of a system.

The data from a particular system can be requested for different hierarchy levels : the overall system level, pool levels, disk levels, etc.. The KPIs from higher components in the hierarchy are usually aggregations of the KPIs

originating from components underneath. Indeed, we have a rather similar set of metrics for each component allowing these aggregations.

1.1.3 Anomaly Detection procedure

In this section, we briefly describe the currently implemented anomaly detection procedure, which does not involve automatic detection.

Anomalies in the cloud systems are currently raised through tickets which have an opening and a closing date. These tickets originate from different sources. Some of them are opened by humans such as users when they notice that the system is not behaving as it should. In some cases, the tickets are opened by the customers when they observe an issue, or simply when the system is not behaving as suited. In other cases, tickets are opened automatically using simple procedural algorithms. For instance, an alert can be thrown when the value of a KPI exceeds a certain threshold.

When a ticket is opened, an expert on cloud systems will take over the issue. This basically consists in identifying the problem by looking at the description in the ticket, and the different KPI time series themselves. Based on this analysis, the expert will most probably locate the anomaly in the system hierarchy and find out which metric is anomalous. After proceeding to some maintenance, and making sure the issue has been treated, the ticket is closed. During this process, it sometimes happens that the system has to be shut down. Thus we are often left with KPI values being recorded as zeros.

Since this process is mostly handled by humans with all the constraints it implies, tickets can be seen as an imperfect labelling of the anomalies. First of all an anomaly has not a clear definition and rather depends on the appreciation of the user. Secondly, the tickets do not necessarily overlap the targeted anomaly because there is often delay between opening / closing and anomaly start / end respectively.

There is a natural way to view an anomaly in opposition with the normality. Indeed, we expect cloud systems to “behave as they shoul”, “be in accordance to their specifications”, “make the client happy”. However, it is very difficult to normalise the above mentioned criteria, as they mostly depend on human appreciation. Prior to the Anomaly Detection project, there were no particular tool designed for this intention, except manually set thresholds which the KPIs should not exceed. However, there has been no study of the particular shapes and characteristics of the targeted anomalies in cloud systems.

1.1.4 Project purpose

The purpose of this project is to develop an automatic anomaly detection algorithm based on the multivariate time series arising from cloud system monitoring. More specifically, these algorithms should most importantly be able to detect when an anomaly starts, but also its duration, its characteristics, the components or metrics involved. These can have multiple benefits both for the clients and for the provider (IBM).

Firstly, it can significantly reduce the anomaly detection and fixing procedure durations. Indeed, one expects an automatic tool to detect anomalies much quicker than humans would, as the latter depends on when a human will actually notice something. Furthermore, if the tool provides meaningful information about what it detected, it can also accelerate the maintenance. This aspect is particularly important as tickets often have duration ranging to days and even weeks.

The second and related purpose is that it can facilitate the job of people in charge of maintenance. Depending on the reliability of the anomaly detection tool, technicians could use it to focus directly on the events that are detected, rather than having an eye on everything. It can compensate potential elements that have not been noticed by humans.

Thirdly, it can be beneficial to give some updated information about the systems status to clients, and add value to the cloud providing service. For instance, it could provide information like the current state (anomaly / or normal), the expected fixing date when an anomaly is happening, the consequences on the usage, etc.

It is important to mention that for real case use, it is better suited for this application to have online algorithms. By online, we mean that the algorithm constantly runs, and has the ability to process directly new incoming inputs without having to go over previously gathered data.

1.1.5 Project Challenges

Here, we identify the main challenges of the whole project, and which are specific to the Cloud System data at IBM.

First of all, there is no labelled data per say, except for the tickets and their limits mentioned earlier. Thus, unsupervised anomaly detection methods are preferred.

Secondly, the quality of the data is not always perfect. Some KPIs, i.e time series coordinates happen to be missing when requesting the data, usually because those are unmeasured or have a significant amount of failed measurements. And even if the request of KPI is successful, it can happen that certain of its value are inconsistent. For instance, it has often been seen that

KPIs such as Response Times sometimes fall to zero, which is impossible and can be attributed to a measurement failure. Indeed, measurements can include errors and noise.

Thirdly, the whole data is high dimensional. Strictly speaking, there are thousands of KPIs per system if we count in all the different hierarchies of the system. However, a lot of these metrics are redundant in different system components and hierarchy levels. Furthermore, there is a set of key KPIs that are primarily targeted by the experts, when they do manual anomaly detection.

Fourthly, there is little knowledge and no documentation about the relationships and dependence between the different KPIs.

Finally, it is unclear whether the KPI values are taken regularly every 5 minutes measures, or whether these value are any aggregation of of measures within a 5 minutes interval. Since it is a rather significant interval with respect to the speed of evolution of a cloud system, we don't know exactly what happens during that time period.

In this thesis, we address the first two challenges, while the others are left for future work. On one hand, our unsupervised method, together with our synthetic data generation process allows to overcome the issue of unlabelled data. On the other hand, our modelling of the time series dynamics is compatible with irregular measurements.

1.2 Preliminaries

1.2.1 Notation and Denomination

- **(Multivariate) Time Series** will be denoted with bold lower case characters : $\mathbf{x} = (x_{t_1}, .., x_{t_n})$. They consist in several **observations** $x_{t_k} \in \mathbb{R}^d$, $k \in \{1..n\}$ appearing at specific **timestamps** t_k, which are ordered in time i.e $t_1 < .. < t_n$. Furthermore, we will denote $\mathbf{x}_{s:t} = (x_{t_k})_{s \leq t_k \leq t}$ a time series **subsequence** of observations appearing in the time interval $[s, t]$. $d \in \mathbb{N}$ denotes the **dimension** of the time series, and the coordinates $j = 1..d$ of $\mathbf{x}$ are noted as $\mathbf{x}_j = (x_{t_1,j}, .., x_{t_n,j})$, which are themselves univariate time series. Since the time series we are dealing with are mostly regular in time, one will sometimes simplify the time indexing $\mathbf{x} = (x_1, .., x_n)$, $\mathbf{x}_{s:t} = (x_k)_{s \leq k \leq t}$.
- Our study will involve modelling the data generating process with **Stochastic processes** which will be denoted with upper case characters $X = (X_t)_{t \in [0,T]}$, and $d_X \in \mathbb{N}$ will denote the dimension of the stochastic process, i.e $X_t \in \mathbb{R}^{d_X}$.

1.2.2 Abbreviation

- KPI : Key Performance Indicator, i.e the univariate time series coordinates in our dataset
- TS or MTS : (Multivariate) Time Series
- AD : Anomaly Detection
- NN, RNN, CNN : (Recurrent / Convolutional) Neural Network
- MLP : Multi Layer Perceptron
- SSM : State Space Model
- ODE, SDE : Ordinary / Stochastic Differential Equation

1.3 Overview of Anomaly Detection in Time Series

Technological progress has enabled real time and large scale monitoring of multiple systems and physical quantities. As a result, increasingly large amounts of time series data are collected, which requires but also enables the development of more powerful machine learning algorithms. Anomaly detection is an important task in time series processing taking advantage of these advances. Anomaly detection aims at identifying abnormal or rare events, and targets multiple purposes such as failure and fraud detection, predictive maintenance, prevention of accidents or damages. It has multiple applications in the environmental (temperature, precipitations, wind speed), industrial (manufacturing), financial (stock prices) or medical (brain activity) domains among others.

Time series analysis refers to numerous tasks involving the extraction of information in time-indexed data. The abundance of sensors often provides us with high dimensional multivariate time series, and analysing them requires to account for both temporal and coordinate-wise dependencies.

Anomaly detection can be regarded as a binary classification problem, aiming at distinguishing outliers from the rest of the data. Time series anomalies can be categorised into three different types; **points outliers**, **subsequence outliers**, **outlier time series**. Point outliers correspond to unusual points that occur at a particular instant, and that are anomalous compared to other values of the time series. Subsequence outliers denominate temporal sequence of points which are jointly anomalous. Note that the length of the subsequence is often a sensitive parameter. Finally, outliers time series are entire time series which are considered anomalous compared to other time series. The temporal character is important when dealing with time series as the time context is often important to analyse the anomalousness of instances (points or sequences). In the following, we will primarily focus on

subsequence outliers, and point outliers as a particular case of it, and we will simply refer to them as anomalies.

One recurring challenge is that we rarely have access to labelled data in time series anomaly detection. Therefore, unsupervised methods often prevail. There are multiple families of approaches involving classical (Blázquez-García et al., 2020) and deep learning (Choi et al., 2021; Zhao et al., 2022) algorithms. The paradigm behind unsupervised anomaly detection is to identify anomalies as **novelties**, i.e., sequences with shapes or characteristics which appear more rarely compared to other sequences. Most methods aim at identifying a reference of normality from the data at hand, and compare individual sequences to this reference.

In **discord detection methods**, the goal is to identify the most unusual subsequence, called the discord (Keogh et al., 2005). The principle is to compute the pairwise distance between any two subsequences, and to extract the one with the largest distance to the other subsequences. the Euclidean distance, or Dynamic Time Wrapping (Benkabou et al., 2018) are standard distance choices. In such methods, there is no reference of normality, nor any threshold as the decision for an instance to be an outlier relies explicitly on other instances. Most of such techniques require fixed sequences lengths, and use optimised algorithms to overcome the quadratic complexity.

The principle of **dissimilarity-based methods** is to compare a subsequence, or its representation to a reference of normality using a dissimilarity measure. A threshold is then applied to the result in order to discriminate normal from anomalous subsequences. Different approaches exist in order to infer the reference of normality. One of those is clustering; subsequences are clustered based on their similarity, and the reference of normality is taken as the centroid of clusters the subsequences belong to. Different clustering techniques can be applied such as K-means, one-class support vector machines (OCSCAN) (Manevitz and Yousef, 2001) or density-based clustering of application with noise (DBSCAN) (Ester et al., 1996). Another approach is to store most relevant subsequences in a dictionary, and to set the reference of normality to be the set of linear combinations of the subsequences in the dictionary (Carrera et al., 2016). Some researchers propose to directly compute a connectivity value for each subsequence that indicates how dissimilar it is from the other. This can be done by building a distance based graph whose nodes correspond to subsequences, and by applying a Markov Chain random walk to infer the connectivity value (Ren et al., 2017).

Frequency-based methods classify subsequences as anomalous when they do not appear as frequently as other (Keogh et al., 2002). Subsequences are usually discretized as it is unlikely that real-valued subsequences occur several times. Some of these methods also focus on the periodic occurrence of the subsequences, as a non anomalous subsequence might appear periodi-

cally in time (Rasheed and Alhajj, 2014).

Today, some of the most popular approaches are reconstruction-based and forecasting-based. In both cases, the approach consists in estimating time series subsequences or individual points, but they differ in the covariates they use for the estimation; reconstruction-based methods will exploit past, current, and future values while Forecasting-based methods restrict themselves to past values. Both methods have the advantage of being compatible with deep learning techniques. We will first briefly explain reconstruction-based methods, as the current most well-established pipeline at IBM is built on such a method. Then, we will emphasise on forecasting based techniques, and discuss their comparative advantages to the other methods.

1.3.1 Reconstruction based Methods

Reconstruction-based methods are very popular in time series anomaly detection as they rely on a simple principle and are easily extendable to multiple deep learning architectures.

The idea is to compress input time series sequences in a compact hidden representation, and then to reconstruct back the original input from this representation. Sequences labelled as anomalies are the ones for which the reconstructed input differs too much from the original. The model would therefore learn both the compression and reconstruction mappings with the objective to accurately reconstruct the input. As the hidden representation is more compact than the input dimension, the model should learn general representative mappings that reconstruct well most of the data. The intuition is that as anomalous sequences are different from normal sequences, and appear less frequently, the model shouldn't learn them as well as the others.

The most common general architecture for Reconstruction based Methods is the autoencoder (AE), which illustrates well the their principle. Consider d-dimensional input time series sequence of length n $\mathbf{x}^{(i)} = (x_{t_{1,i}}^{(i)}, .., x_{t_{n,i}}^{(i)}) \in \mathbb{R}^{n \times d}$. Furthermore, let $f_{encoder} : \mathbb{R}^{n \times d} \to \mathbb{R}^m$ and $f_{decoder} : \mathbb{R}^m \to \mathbb{R}^{n \times d}$ be two parameterised NNs, denominated **encoder** and **decoder** respectively. The architecture outputs

$$\hat{\mathbf{x}}^{(i)} = f_{decoder} \circ f_{encoder}(\mathbf{x}^{(i)})$$

The sequence is labelled as an anomaly if

$$D(\mathbf{x}^{(i)}, \hat{\mathbf{x}}^{(i)}) > \tau$$

where D is a particular distance function such as the mean absolute distance (L_1) or the euclidean distance (L_2) and τ is some threshold.

Different neural network types are well suited for implementing this encoder-decoder architecture on time series data such as recurrent neural networks (RNNs)(Malhotra et al., 2016), convolutional neural networks (CNNs) (Kieu et al., 2018), or transformers (Deng et al., 2021).

Another similar architecture which is popular for reconstruction-based anomaly detection is variational autoencoder (VAE) Su et al. (2019). Here, the model rather aims to learn the underlying distribution of the time series sequences. Learning VAE models usually involves Bayesian inference techniques, and consist in learning a posterior distribution from a prior and the data, through the optimisation of the evidence lower bound (ELBO) loss.

1.3.2 Forecasting based Methods

Forecasting is another very popular approach for time series anomaly detection. The literature of forecasting is very rich as this domain has a multitude of applications such as weather, stock prices (finance) or economic indicators prediction. We will start by explaining the principle of forecasting-based anomaly detection. We will then emphasise on graph-based forecasting methods and their relation with time series causality which led to an initial subject for the thesis. We will stress the advantages that we saw in forecasting-based methods compared to other anomaly detection approaches, and the intuition leading to probabilistic forecasting.

The principle is to forecast the values of an input time series after a specific timestamp t, based on the available observation up to time t. Then, one compares the forecast predictions with the true observations after t, and one labels the observations as anomalous if they differ from their prediction by more than a specific threshold. Intuitively, the model learns how to forecast time series from the dataset, and thus will better learn the dynamics that occur the most often. Therefore, time series sequences with different dynamics that occur more rarely, will result in larger forecasting errors.

To illustrate this, consider d-dimensional input time series sequence of length n $\mathbf{x}^{(i)} = (x^{(i)}_{t_{1,i}}, .., x^{(i)}_{t_{n,i}})$, where i is the index of the time series and t the timestamp that separates the past from the future. Let $f_{forecast}$ be a parameterised NN, called the forecasting function, which maps past to future observations :

$$\hat{x}^{(i)}_{t+\tau} = f_{forecast}(\{x^{(i)}_{t_{j,i}}\}_{t_{j,i} \leq t}) \tag{1.1}$$

Then, for $\tau > 0$, one compares observations with predictions, and as in reconstruction-based methods, one labels an anomaly if the forecasting error excesses a certain threshold τ :

$$\mathcal{D}(x^{(i)}_{t+\tau}, \hat{x}^{(i)}_{t+\tau}) > \tau, \tag{1.2}$$

where $\mathcal{D}$ is a distance function. Note that one can naturally compute the reconstruction error for time series sequences rather than for individual points.

Forecasting methods often need to be more flexible if they have to account for varying number of past observations. Therefore, they often rely on RNN-like architectures such as long short-Term memory (LSTM) (Hundman et al., 2018; Ding et al., 2019). By design, such architectures can only treat time series with regular time intervals between observations. One sensitive parameter is the **forecasting horizon** : the difference between the time for which we make the prediction and the last observations that were used for the prediction. The model has to specifically learn for the forecasting horizons it will use at inference.

Graph Dependent Forecasting Methods

Among forecasting-based methods, graph-based methods have received a particular attention these last years (Chen et al., 2021; Deng and Hooi, 2021; Zhao et al., 2020) and have shown promising results (Choi et al., 2021). The advantage of these methods arises in multivariate time series forecasting, as they explicitly model the dependencies of the different coordinates of a time series, which are often referred to as variables.

Specifically, given a d-dimensional time series $\mathbf{x} = (x_{t_1}, .., x_{t_n})$, the jth coordinate or variable of $\mathbf{x}$ refers to the univariate time series $(x_{t_1,j}, .., x_{t_n,j})$ where $x_{t_1,j}$ is the jth coordinate of x_{t_1}. The term variable comes from the fact that the different coordinates of a multivariate time series represent different measures that are dependent from one another, such as the KPIs in our application. Graph-based forecasting methods aim to learn an underlying directed graph that models these dependencies as temporal causal relationships. This intuitively means that for variable j, there is information present in the past of its parent variables $\mathcal{P}(j)$ about the future of j. If such causal relationships exist, then it makes sense to exploit them for forecasting, as these notions are closely related. Graph-based forecasting architectures restrict the information flow between variables to the edges of the causal graph.

To the best of our knowledge, no work has tried to give theoretical motivations for the advantage of using directed graphs towards multivariate time series forecasting or anomaly detection, even though these methods have demonstrated superiority on some benchmark datasets (Choi et al., 2021). One possibility is that restricting the information through the edges of a graph could yield some generalisation capabilities. Indeed, a model without such restrictions could learn relationships between variables that have no direct relation, and therefore generalise poorly to unseen data. It could also be the case that in order to predict a specific variable, some useful information is present in multiple other variables. In such a case, if the graph's

edges models the true relations of the variables, it could play the role of a regulariser and reduce the uncertainty of the predictions.

Usually, these graphs are learned in an end-to-end fashion together with the rest of the architecture. Learning these graphs is challenging, as they are characterised by their edges, and therefore not straightforward to incorporate in a learnable framework that uses backpropagation. One solution to overcome this issue is the Gumbel-softmax sampling (Chen et al., 2021; Löwe et al., 2020). These methods then propagate information through the graph using graph neural network architectures such as graph convolution Chen et al. (2021) or graph attention Deng and Hooi (2021); Zhao et al. (2020).

In cloud data there are probably causal relationships, or simply dependencies between KPIs that are intangible because of virtual, logical, physical relations between the different components of the system. Exploiting them could yield benefits when doing forecasting based Anomaly Detection, as they could be directly incorporated as a graph in such an architecture. A good knowledge of the cloud systems could allow to produce such graph, but requires advanced knowledge on the cloud systems architecture. Instead of this, we started to investigate time series causal discovery to see whether one can learn causal relationships between the KPIs. Apart from forecasting-based anomaly detection, one could also benefit from causal relations for Root Cause Analysis (RCA). The principle of RCA is to identify the causes of an anomaly, for instance the KPIs that reflect the origin of the anomaly. A causal structure together with a per-KPI anomaly detection tool could be used for RCA, by tracking the anomalies of KPIs through the causal structure. This work includes some work and experiments on causal discovery. However, this topic was not further pursued because one couldn't evaluate the causal discovery method. Indeed, there is no documentation on the virtual/physical connections of cloud systems, and resulting causal graphs could not properly be assessed by cloud experts. The reader can refer to the KPI Causal Relations Learning for an introduction to time series causal discovery.

1.4 Motivation for Forecasting-Based Approaches

We first discuss potential limits of the previously mentioned approaches, especially the reconstruction-based approach. First of all, for most of the approaches mentioned earlier, the algorithm usually requires the ingested time series sequences that are comparable, and thus have fixed lengths. Consequently, the length is a sensitive parameter. Also, the algorithm may become unusable in situations where we simply do not have long enough sequences to ingest. In order for the time series sequences to be comparable, they also need to have a temporal correspondence in the sense that time series

should have the same time intervals. Therefore, it is often assumed that the sequences have regular observations in time. This might be an issue if parts of the data are corrupted. Secondly, as most methods ingest fixed length sequences regardless of where they are located in time, the temporal context of the sequence is not taken into account. This is however an important aspect, as the abnormality of a sequence may depend on the time when it appears. For instance in cloud data, significantly higher traffic KPI values (such as response transfer size, response rate) than average are maybe not alarming during work times, but could be during the night. Reconstruction-based methods have the disadvantage that their reference of normality is less clearly defined (than for forecasting based approaches). Although such models should appropriately learn to reconstruct normal data of the dataset, it is unclear whether anomalies are not properly reconstructed as well. In other words, it might be the case that certain anomalies are properly projected in the underlying space and result in low reconstruction errors. Furthermore, the size of the latent space is an important hyperparameter, and it is difficult to evaluate it. Indeed, a bigger latent space will tend to give lower reconstruction errors, while a smaller latent space might be better to differentiate anomalies from normal sequences.

Here we try to give some motivations of using forecasting-based methods for anomaly detection. In this approach, the reference of normality for an ingested time series sequence or observation is explicitly and meaningfully defined, and consists in its expected value, based on past observation. This constitutes an advantage compared to other approaches that most often rely on the similarity with other time series sequences in the dataset. Indeed, the reference of normality is inferred specifically for the sequence at hand, and relies on the context of past observations. Furthermore, in the forecasting based approach, one aims to learn the underlying dynamic of time series. The model gets closer to the origin of the data, as it learns something about the data generating process. A forecasting model can also very easily be run online, i.e process directly new observations by comparing it with pre-computed predictions.

One aspect that is usually not taken into account by forecasting-based methods, is the stochasticity of the data generating process. Indeed, those methods directly try to infer predictions for future observations, while there might be uncertainty when solely relying on past observations. In other words, there might be several plausible trajectories compatible with the idea that the time series is non-anomalous. Comparing future observations with one possible prediction might therefore lead to large prediction errors, even if we don't have an anomaly. This motivates the introduction of a probabilistic formulation, where we instead of 1.1 want to model the predictive distribution of future observations, conditioned on past observations :

$$\hat{q}\left(x_{t+\tau}^{(i)} | \{x_{t_{j,i}}^{(i)}\}_{t_{j,i} \leq t}\right) \tag{1.3}$$

In addition, forecasting-based approaches rely on a sensitive parameter : the threshold τ applied to the prediction error. This parameter should depend on the forecasting horizon, that is the difference between the time of the observation we predict, and the last observation taken into account. Larger forecasting horizon should intuitively require a larger threshold, because prediction errors will tend to be higher the further we predict in the future. Also, it might be the case that their is more uncertainty in the prediction at certain periods of time, or depending on the previous observations. For instance in cloud systems, the dynamics of the KPI could be more uncertain in periods of higher usage, which depend on the time (hour in the day for instance) and on the previous measurements (which can indicate that the system is being intensively used). For this, a calibrated predictive distribution could take this aspect into account, for instance by showing a larger variance when the predictions are more uncertain.

Based on the previous discussion, we also consider making the forecasting model depend on the time of the observation (both past and future), that is replacing $x_{t_{j,i}}^{(i)}$ with $y_{t_{j,i}}^{(i)} = \left(x_{t_{j,i}}^{(i)}, t_{j,i}\right)$.

1.5 Outline

In chapter 2, we start by investigating the KPI data of cloud systems, and we propose a synthetic data generating process. In chapter 3, we explain the concept of probabilistic forecasting, and reveal two main approaches to model it. We pay a particular attention to the second approach and give the fundamental motivations for choosing it. In chapter 4, we reveal our general methodology and we propose a pipeline for multivariate anomaly detection. In chapter 5, we investigate the advantages and disadvantages of the elements we incorporated in in our forecasting module. We further show the learning schemes and inference examples of our pipeline. Finally, in chapter 6, we explore future research direction of the designed method.

Chapter 2

Cloud and Synthetic Data

In this chapter, we start by investigating the KPI cloud data and to extract useful knowledge for the design of the method. Then we justify the usage of synthetic data and mention a synthetic data generating tool for KPI data. Finally, we design a data generating process which will be used for our experiments.

2.1 KPI Cloud Data study

Several experiments have been conducted on the data in order to get a better understanding of its distribution. In this prior study, we did not have access to the targeted conditional distribution of the forecasting framework. We rather looked at the unconditional distribution.

After a first overview on the data distribution, we found that it is (for most of the considered metrics) non symmetric / skewed, as they have a heavier tail on the right (higher values). Thus, this suggests that the normal distribution is probably not well adapted when modelling this distribution. This is also justified with the type of data we have in mind. Although the times series are normalised, metrics such as response times and transfer sizes / rates take positive values and have no reason to be symmetric as they are naturally lower-bounded by 0, while not having a natural upper bound.

We also identify a peak mode at zero values, without other nearby values taken. These zeros correspond to unrecorded values, shut down systems, etc . Although zero values might be of interest when searching for anomalies, they do not fit with the normal behaviour of the system. Thus, it will be required to filter them out when learning the model on non-anomalous samples. Apart from the zero values, there are often two to three modes, with at least one significant peaky mode for smaller values, and one less significant more diffused mode for higher values. Otherwise, the density of the data

seems more or less to decrease after the first significant mode and to vanish way before 1 values (the data is normalised). The values often concentrate almost exclusively in the $[0, 0.5]$ interval.This suggests that KPIs more often take smaller values (because of the first mode), which we interpret as corresponding to periods where the system has less activity. The second mode thus probably corresponds to usual values when the system is performing some task(s). In this case the values have a higher variability.

We also investigated per hour and per weekday distributions. We remark that the distribution of some KPIs changes on a per hour basis. Especially, we often see a difference between day and night hours. We observe a similar behaviour when looking at the distribution of KPIs on weekday basis, where we primarily see differences between the weekends and the rest of the week. This phenomena suggests that there is some daily and weekly seasonality in the data. Thus it could be useful to incorporate this aspect in both the design of the synthetic data, and the anomaly detection algorithm.

As we are aiming to model a conditional distribution, we also investigated the time-difference distribution of KPI values. By time-difference, we mean the difference between the current and the previous values. We argue that this can give us valuable information on the conditional distribution as the conditional expectation of the current KPI value given the past will probably be near to the value of the last observation. The time-difference distribution is seem to be symmetrical and leptokurtic, which means it has fatter tails and a higher concentration around the mean. Thus, the normal distribution is probably not the best choice for modelling the data and one might prefer Student or Laplace distributions.

2.2 Synthetic Data Generation

The principal issue with the real data is that it is largely unlabelled, as mentioned earlier. Consequently, one big challenge resides in the evaluation of the AD algorithm that we build. Therefore, we consider the option of generating synthetic data in order to design an AD pipeline that fits suitable characteristics. Furthermore, since the characteristics of the anomalies we aim to detect are unknown, we would like to conduct an analysis of the specific anomaly types that our final method would be able to detect. In other words, our intention is to define properly what we are targeting, and which tools allow us to achieve these targets. Synthetic data have another advantage; they are clean, quickly accessible and easy to use when conducting experiments. Furthermore, synthetic data can be publicly shared and discussed while real world data is often subject to regulations and data privacy. After mentioning a relevant related work, we will attempt to design an synthetic data generating method that suits some properties identified

in the real dataset, and then propose methods to inject different types of anomalies.

TSAGen Framework

We start by mentioning TSAGen (Wang et al., 2022), a framework that has explicitly been designed for generating KPI data of network systems, and ingesting anomalies in it. This is interesting to us because the data type it targets is similar to our cloud KPI data. However, this method is restricted to the generation of univariate time series, and therefore does not lead to the property of dependency between multivariate time series coordinates. The authors design a modular and tractable tool so that the user can model specific data that fits its use case. Furthermore, the data and especially the anomalies are generated with diversity in order to match the reality.

The time series generation model consists of an additive model with different components; the seasonal, trend and noise components. Each component is characterised by different parameters that can be chosen by the user. Since one of the purposes is to generate time series that are similar one to another, but not exactly the same, they propose different ways for achieving this. The first is to apply random drift factors to the different parameters of trend, season and noise component. The second is specific to the generation of the seasonal component, and relies on random midpoint displacement fractal (RMDF). The principle is the following. One starts with an affine (time-indexed) path, and extracts the midpoint. This midpoint is displaced orthogonally to the path, where the displacement amplitude follows a Guassian distribution. This results in a path with two affine parts. This procedure is recursively applied to all the affine parts of the path. In order to generate different but similar seasons, the authors propose to generate a common path with a certain recursion depth. Then the final seasons are obtained by further applying some iterations of the same procedure, but independently for each season. The authors also propose different anomaly ingestion procedures, which they categorise into structural and non-structural anomalies. For structural anomalies, the idea is to distort the parameters of the trend, noise and seasonal components. For non-structural anomalies, they generate random anomaly patterns such as peaks which they simply inject into the pre-generated paths.

We first oriented our choice towards the TSAGen framework, as it proposes an interesting solution for generating diverse but similar time series. More specifically, we based our initial generation procedure on the proposed seasonal component to make the data somehow periodic. However, we identified some limitations to this method. The resulting paths given by the RMDF procedure are generated geometrically, and therefore not necessarily meet any smoothness assumptions. Furthermore, the shapes don't result

necessarily in functions of time in theory, as the coordinates of the control points are computed by considering both time and space dimensions as a regular 2D space. As mentioned earlier, they do not propose any lead for the multivariate case either.

2.2.1 Ornstein-Uhlenbeck based generator

The following describes a data generation process that attempts to tackle previously mentioned issues. Firstly we tried to incorporate elements observed in the real data. The real time series are the result of a real-time monitoring of metrics that evolve continuously in time. Therefore, we chose a model of stochastic process from which we can sample continuous paths. Because of the strong natural relations of different KPIs from one system, we want to model dependencies between different components of the multivariate time series. We also want to be able to reproduce the seasonality that exist in the true data. Secondly, we sought for a stochastic process model that matches some theoretical assumptions. Indeed, the process would preferably match the assumptions from the PD-NJ-ODE (see section 3.2) Framework in an attempt to guarantee the convergence of the models outputs to the true conditional moments. Thus, the process should have a form that allows the computation of conditional moments for validation.

Following the examples given in Krach et al. (2022), we oriented the choice toward a standard Ito diffusion.

Definition

The process we chose in order to generate the synthetic dataset is inspired from the Orstein-Uhlenbeck process (Vatiwutipong and Phewchean, 2019), formalised as an Ito Diffusion (see B.1). The difference resides in the mean that is not constant anymore. Instead, we chose a variable mean that is periodic in time. Formally, $X = (X_t)_{0 \leq t \leq T}$ is the d_X-dimensional process defined as the solution to the following SDE :

$$\begin{aligned} dX_t &= -\theta(X_t - m(t))dt + \sigma dW_t \\ X_0 &= x_0 \end{aligned} \tag{2.1}$$

where $m : [0, T] \to \mathbb{R}^{d_X}$ is a periodic continuous function of smallest period $\mathcal{T}$, W is a d_W-dimensional Brownian motion, $\theta \in \mathbb{R}^{d_X \times d_X}$ is positive definite and $\sigma \in \mathbb{R}^{d_X \times d_W}_{\geq 0}$.

Equation (2.1) admits a unique strong solution. Please refer to Appendix B.7 for the details about this statement.

We emphasise on the interesting properties of the process defined above for the generation of synthetic KPI data. First it is rather clear that the drift will

enforce the random paths to be attracted towards the periodic m function. This is especially true if the speed term, θ, is diagonal. In this case, each coordinate j of X will be solely influenced by m_j, depending on the strength of θ_j. Secondly, this process can encode dependencies between the coordinates, i.e the different KPIs in the real dataset. Indeed, the drift component can enforce some intrinsic dependencies. In other terms, coordinate j will be influenced by m_k, whenever $\theta_{kj} \neq 0$, where m_k shapes the dynamics of X_k. In addition, σ can encode some correlations between the different coordinates, since each coordinate of the Brownian Motion W_k will play the role of a confounder for each coordinate j of X such that $\sigma_{jk} > 0$. W could be regarded as factors that influence the value of several KPIs, where an example of factor could be a quantification of the current usage of the system. Note that in opposition to Wang et al. (2022), we did not add any trend component. This is because there is no particular reason that the average values of a KPI increase throughout time. Of course, this could be changed by adding a trend to m, which would therefore not be periodic anymore.

Distribution

The conditional distribution $X_t|\mathcal{A}_{\tau(t)}$ is Gaussian, with conditional expectation and variance :

$$\mathbb{E}[X_t|\mathcal{A}_{\tau(t)}] = e^{-\theta(t-\tau(t))}X_{\tau(t)} + \int_{\tau(t)}^{t} e^{-\theta(t-s)}\theta m(s)ds$$

and

$$\text{Cov}(X_t|\mathcal{A}_{\tau(t)}) = \int_{\tau(t)}^{t} e^{-\theta(t-s)}\sigma\sigma^T e^{-\theta^T(t-s)}ds \tag{2.2}$$

With those two terms and using the fact that the distribution is Gaussian, one can naturally derive higher conditional moments. The conditional distribution is derived in Appendix B.9.

2.3 Anomaly Ingestion

By taking inspiration of Goswami et al. (2023); Wang et al. (2022), we designed 7 anomaly types that are compatible with our synthetic data generation process. All these schemes are described for the univariate case. The first six types are sequence anomalies which means that they last for a certain time range. We denote this time range as (a, b) where $0 \leq a < b \leq T$. The last type is a punctual anomaly, i.e the anomaly occurs only at a certain time c.

1. **Deformation :** We deform the base shape of the data, which is given by the periodic m function. This means that for $t \in (a, b)$, the process is driven by another (randomly sampled) m^{anomaly} function.
2. **Diffusion :** We change the diffusion term of the process, i.e σ is replaced by $\sigma^{anomaly} > \sigma$ for $t \in (a, b)$.
3. **Noise :** We add a white noise on top of the process, i.e x_t is replaced by $x_t + \varepsilon_t$ with independent $\varepsilon_t \sim \mathcal{N}(0, \delta^{anomaly})$ for $t \in (a, b)$.
4. **Cutoff :** This type is similar to the deformation anomaly, but with m^{anomaly} being set to a random constant.
5. **Scale :** We replace m with $m^{\text{anomaly}} = sm$, $s \neq 1$ for $t \in (a, b)$.
6. **Trend :** We replace m with a trendy version of it $m^{\text{anomaly}} = m + tr(t - a)$ for $t \in (a, b)$ where tr is the trend slope.
7. **Spike :** We add a spiking value v at $t = c$.

Chapter 3

Probabilistic Forecasting

In this chapter, we investigate probabilistic forecasting, which constitute the main ingredient of our approach toward time series anomaly detection. First, we start by introducing the theory of state space models (SSM) and explaining the concept of filtering. Then, we emphasise on linear state space models, for which the filtering equations can efficiently be computed using the Kalman algorithm. We stress that SSMs are a popular approach for probabilistic forecasting on regular time series. We then introduce path-dependent neural jump ODE, a forecasting framework that is particular suited when the time series can be observed at any time (not necessarily regular TS), and which offers theoretical guarantees for learning to make optimal predictions. We will explain how this framework can be extended towards probabilistic forecasting; that is how to estimate the predictive distribution from predictive moments.

3.1 State Space Model based Forecasting

State space modelling is a basic approach that allows to model the predictive distribution of discrete regular (in time) time series data. We will first show the general formulation of state space models, before describing the particular case of linear SSM. We will discuss how to infer the predictive distribution from this theory, as well as how to learn an SSM. Finally we explain how the linear SSM can be applied in our setting, and the limits of this approach.

3.1.1 State Space Model

State space models (SSM) (Giordani et al., 2011) are a convenient tool for modeling dynamical systems under uncertainty. One assumes that the distribution of obervations only depends on a sequence of underlying latent states, and that the pair of both observations and states is Markovian. We

aim at modelling the dynamics of time series with a probabilistic approach. Therefore, we will regard time series as the realisations of a stochastic process.

Lets assume we are trying to model the evolution of a regular time series $\mathbf{x} = \{x_t\}_{t\in\mathbb{N}}$. We will denote by $\mathbf{z} = \{z_t\}_{t\in\mathbb{N}}$ the state time series of the SSM. We will simply denote by $\boldsymbol{\theta}$ every other parameter excluding $\mathbf{z}$.

The state space model assumes that there is an underlying Markovian process behind the data generation process of $\mathbf{x}$ and $\mathbf{z}$, i.e.

$$p(x_t|\mathbf{z}_{1:t}, \mathbf{x}_{1:t-1}; \boldsymbol{\theta}) = p(x_t|z_t; \boldsymbol{\theta}), \tag{3.1a}$$
$$p(z_{t+1}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}; \boldsymbol{\theta}) = p(z_{t+1}|z_t; \boldsymbol{\theta}). \tag{3.1b}$$

The process $\{(x_t, z_t)\}_{t\in\mathbb{N}}$ is Markovian as $p(z_{t+1}, x_{t+1}|\mathbf{z}_{1:t}, \mathbf{x}_{1:t}, \boldsymbol{\theta}) = p(z_{t+1}, x_{t+1}|z_t, x_t; \boldsymbol{\theta})$. We call (3.1a) and (3.1b) the **observation equation** and the **state transition equation** respectively.

In the following, we will see how to derive the filtering equations, as well as the likelihood of the observations. The former is used to compute the predictive distribution of the time series observations $p(x_{T:T+H}|\mathbf{x}_{1:T}; \boldsymbol{\theta})$ based on the model, while the latter can asses the likelihood of the observations, which in turn can help us in the search of an appropriate model.

Filtering

Filtering denotes the computation of the distribution of the current state z_t using the observation $\mathbf{x}_{1:t}$: $p(z_t|\mathbf{x}_{1:t})$. Using the Markovian assumptions, one can derive

$$p(z_{t+1}|\mathbf{x}_{1:t}; \boldsymbol{\theta}) = \int p(z_{t+1}|z_t, x_t; \boldsymbol{\theta}) p(z_t|\mathbf{x}_{1:t}; \boldsymbol{\theta}) dz_t. \tag{3.2}$$

One can then update the conditional distribution with the current observation using Bayes rule

$$\begin{aligned} p(z_{t+1}|\mathbf{x}_{1:t+1}; \boldsymbol{\theta}) &= \frac{p(\mathbf{x}_{1:t+1}|z_{t+1}; \boldsymbol{\theta}) p(z_{t+1}; \boldsymbol{\theta})}{p(\mathbf{x}_{1:t+1}|\boldsymbol{\theta})} \\ &= \frac{p(x_{t+1}|z_{t+1}; \boldsymbol{\theta}) p(\mathbf{x}_{1:t}|z_{t+1}; \boldsymbol{\theta}) p(z_{t+1}; \boldsymbol{\theta})}{p(\mathbf{x}_{1:t+1}|\boldsymbol{\theta})} \\ &= \frac{p(x_{t+1}|z_{t+1}; \boldsymbol{\theta}) p(\mathbf{x}_{1:t}, z_{t+1}|\boldsymbol{\theta})}{p(\mathbf{x}_{1:t}|\boldsymbol{\theta}) p(x_{t+1}|\mathbf{x}_{1:t}; \boldsymbol{\theta})} \\ &= \frac{p(x_{t+1}|z_{t+1}; \boldsymbol{\theta}) p(z_{t+1}|\mathbf{x}_{1:t}; \boldsymbol{\theta})}{p(x_{t+1}|\mathbf{x}_{1:t}; \boldsymbol{\theta})} \\ &\propto p(z_{t+1}|\mathbf{x}_{1:t}; \boldsymbol{\theta}) p(x_{t+1}|z_{t+1}; \boldsymbol{\theta}). \end{aligned} \tag{3.3}$$

These equations allows us to naturally derive the predictive distribution of both state and the observation after the current step T, as

$$p(z_{T+1}|\mathbf{x}_{1:T};\boldsymbol{\theta}) = \int p(z_{T+1}|z_T, x_T;\boldsymbol{\theta})p(z_T|\mathbf{x}_{1:T};\boldsymbol{\theta})dz_T, \tag{3.4a}$$

$$p(x_{T+1}|\mathbf{x}_{1:T};\boldsymbol{\theta}) = \int p(x_{T+1}|z_{T+1};\boldsymbol{\theta})p(z_{T+1}|\mathbf{x}_{1:T};\boldsymbol{\theta})dz_{T+1}, \tag{3.4b}$$

where we simply reused (3.2).

We can then iteratively compute the predictive distribution for multiple steps ahead. For $j \geq 1$, we have

$$p(z_{T+j+1}|\mathbf{x}_{1:T};\boldsymbol{\theta}) = \int p(z_{T+j+1}|z_{T+j};\boldsymbol{\theta})p(z_{T+j}|\mathbf{x}_{1:T};\boldsymbol{\theta})dz_{T+j}, \tag{3.5a}$$

$$p(x_{T+j+1}|\mathbf{x}_{1:T};\boldsymbol{\theta}) = \int p(x_{T+j+1}|z_{T+j+1};\boldsymbol{\theta})p(z_{T+j+1}|\mathbf{x}_{1:T};\boldsymbol{\theta})dz_T. \tag{3.5b}$$

Likelihood

The likelihood describes the probability of observing $\mathbf{x} = (x_1, .., x_T)$ given parameter $\boldsymbol{\theta}$.

$$p(\mathbf{x}|\boldsymbol{\theta}) = p(x_1|\boldsymbol{\theta}) \prod_{t=1}^{T-1} p(x_{t+1}|\mathbf{x}_{1:t};\boldsymbol{\theta}) \tag{3.6a}$$

where we can inject equation 3.4b. The likelihood will therefore play the role of an objective function for learning the parameters of the model.

3.1.2 Linear State Space Model

In the linear version of a state space model restricts one restricts the state transition and observation equations to be linear. Such a prior is very interesting, as it allows to easily derive the filtering equations with the so called Kalman filtering algorithm. One way of formulating it is the following

$$z_{t+1} = a_t + A_t z_t + u_t, \tag{3.7a}$$

$$x_t = b_t + B_t z_t + v_t, \tag{3.7b}$$

Where we have a_t, b_t being vectors and A_t, B_t being matrices. $\boldsymbol{u} = (u_1, u_2, ..)$ and $\boldsymbol{v} = (v_1, v_2, ..)$ are pairwise independent white noise processes, i.e

$$\begin{aligned}\mathbb{E}[u_t] &= 0,\\ \mathbb{E}[u_t u_t^T] &= U_t,\\ \mathbb{E}[u_t u_s^T] &= 0, \text{ for } s \neq t.\end{aligned}$$

The same holds for $\boldsymbol{v}$, v_t having as correlation matrix V_t. $\boldsymbol{u}$ and $\boldsymbol{v}$ are denominated the **state transition noise** and the **observation noise** respectively. Furthermore, we suppose that the initial state, z_0, satisfies $\mathbb{E}[z_0] = \mu_0$ and $\mathrm{Var}[z_0] = S_0$.

Let us fix the dimensions of all variables : $x_t \in \mathbb{R}^d$, $z_t \in \mathbb{R}^r$, $u_t \in \mathbb{R}^m$, $v_t \in \mathbb{R}^d$. Then we have for the parameters : $b_t \in \mathbb{R}^d$, $a_t \in \mathbb{R}^r$, $\mu_0 \in \mathbb{R}^r$, $B_t \in \mathbb{R}^{n\times d}$, $A_t \in \mathbb{R}^{n\times n}$, $V_t \in \mathbb{R}^{d\times d}$, $U_t \in \mathbb{R}^{m\times m}$, $S_0 \in \mathbb{R}^{r\times r}$. All these parameters together constitute the overall model parameter $\boldsymbol{\theta}$.

We can index the model parameters by t, because the joint process $(x_t, z_t; t \geq 1)$ is Markovian. In the case of the linear SMM, we would have

$$\begin{aligned}\theta_0 &= (\mu_0, S_0),\\ \theta_t &= (A_t, B_t, U_t, V_t, a_t, b_t), \ \forall t \geq 1.\end{aligned} \tag{3.8}$$

A very suitable property arises when we assume that the initial distribution of the state $p(z_1|\boldsymbol{\theta})$, and the noises $p(u_t|\boldsymbol{\theta})$, $p(v_t|\boldsymbol{\theta})$ are Gaussian, i.e $z_1 \sim \mathcal{N}(\mu_0, S_0)$, $u_t \sim \mathcal{N}(0, U_t)$, $v_t \sim \mathcal{N}(0, V_t)$. Then, we can deduce that all of the previously derived distribution are Gaussian as well because of the linearity in the observation and state transition equations. This is very useful to us as we can now easily compute the filtering distribution, thus the prediction distribution in closed form.

Kalman filtering

We define the sequence of **innovations** and denoted $\boldsymbol{\delta} = (\delta_1, .., \delta_T, ..)$ as an equivalent to the sequence of observations $\mathbf{x} = (x_1, .., x_T, ..)$ with

$$\delta_1 = x_1 - \mathbb{E}[x_1], \tag{3.9a}$$
$$\delta_{t+1} = x_{t+1} - \mathbb{E}[x_{t+1}|\mathbf{x}_{1:T}]. \tag{3.9b}$$

Let $R_t = \mathrm{Var}(x_t|\mathbf{x}_{1:t-1}) = \mathrm{Var}(\delta_t)$ denote the innovation variance and $z_{t|s} = \mathbb{E}[z_t|\mathbf{x}_{1:s}]$, $S_{t|s} = \mathrm{Var}[z_t|\mathbf{x}_{1:s}]$ the filtering expectation and variance.

Proposition 3.1 (Kalman Filtering equations) *We have the following equalities*

$$\delta_t = x_t - B_t z_{t|t-1} - b_t, \tag{3.10a}$$
$$R_t = B_t S_{t|t-1} B_t^T + V_t, \tag{3.10b}$$
$$z_{t|t} = z_{t|t-1} + S_{t|t-1} B_t^T R_t^{-1} \delta_t, \tag{3.10c}$$
$$S_{t|t} = S_{t|t-1} - S_{t|t-1} B_t^T R_t^{-1} B_t S_{t|t-1}, \tag{3.10d}$$
$$z_{t+1|t} = a_t + A_t z_{t|t}, \tag{3.10e}$$
$$S_{t+1|t} = A_t S_{t|t} A_t^T + U_t. \tag{3.10f}$$

The details about the derivation of the previous equalities can be found in Giordani et al. (2011), and an in depth theoretical study of the Kalman filtering algorithm is available at Masnadi-Shirazi et al. (2019).

We call $z_{t|t-1}$ and $S_{t|t-1}$ the **a priori state estimate** and **a priori estimate covariance**. Similarly, we call $z_{t|t}$ and $S_{t|t}$ the **a posteriori state estimate** and **a posteriori estimate covariance**.

Thus we can compute $z_{t|s} = \mathbb{E}[z_t|\mathbf{x}_{1:s}]$, $S_{t|s} = Var[z_t|\mathbf{x}_{1:s}]$ as

$$z_{t+j+1|t} = a_{t+j} + A_{t+j} z_{t+j|t}, \tag{3.11a}$$
$$S_{t+j+1|t} = A_{t+j} S_{t+j|t} A_{t+j}^T + U_{t+j}, \tag{3.11b}$$
$$x_{t+j+1|t} = b_{t+j+1} + B_{t+j+1} z_{t+j+1|t}, \tag{3.11c}$$
$$\mathrm{Var}[x_{t+j+1}|\mathbf{x}_{1:t}] = B_{t+j+1} S_{t+j+1|t} B_{t+j+1}^T + V_{t+j+1}. \tag{3.11d}$$

Likelihood

Using the innovation terms introduced previously, we can easily factorise the likelihood of the observations, as

$$\begin{aligned} p(\mathbf{x}_{1:T}|\boldsymbol{\theta}) &= p(\delta_{1:T}|\boldsymbol{\theta}) \\ &= \prod_{t=1}^{T} p(\delta_t|\boldsymbol{\theta}) \\ &= \prod_{t=1}^{T} (\det(2\pi R_t))^{-1/2} \exp(-\frac{1}{2}\delta_t^T R_t^{-1} \delta_t) \end{aligned} \tag{3.12}$$

We used the fact that the innovations are simply a transformation of the observations, the independence of the innovations, and the the expression of the Gaussian density.

3.1.3 Model Learning

In the previous section, we based ourselves on a model parametrized by an unknown $\boldsymbol{\theta}$. It comprises all the parameters from the state transition and observation equation, and the initial state distribution.

We aim at learning appropriate parameters that fit the data generation process. As mentioned earlier, one natural way of formulating this is using the likelihood. Formally, we seek for the parameter that maximises the likelihood of the observations, thus the maximum likelihood estimator (MLE). Suppose we have input time series of length T, $\{\mathbf{x}^{(i)}\}_{i=1..N}$. The objective can be formulated as

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmax}} \prod_{i=1}^{N} p(\mathbf{x}^{(i)}|\boldsymbol{\theta}), \tag{3.13}$$

where Θ is the paramter space.

Keeping the current formulation does not seem to be a good idea, as the model is highly over-parameterized compared to the dimensionality of the input (the observations). For a single time series of length T, we have an input of size dT compared to a size of $T(d + r + rd + r^2 + dm + rm + m^2) = \mathcal{O}(T)$ for $\boldsymbol{\theta}$. This over-parameterization comes from the fact that we have parameters for each time index t. Furthermore, in this naive approach, we do not account for any similarities in the time indexed parameters θ_t, neither any smoothness of θ_t with respect to t. More specifically, This aspect is also problematic as we can only deal with time series having a fixed time horizon T.

A first approach to overcome this over-parameterization is to make the model time invariant. Specifically, it consists in replacing each θ_t by a time invariant parameter θ. This formulation has the advantage to highly reduce the complexity of the problem, while enforcing both similarity and smoothness in the θ_t. However, it might not be able to capture any time dependent behaviours such as periodic events. Note as illustration that under certain assumptions on the initial state distribution, the resulting state space model is stationary (Jong and Chu-Chun-Lin, 1994).

It might also be the case that the state transition not only depends on the time, but also other external factors that influence the dynamics of the process. Some authors (Rangapuram et al., 2018; Du et al., 2022) have tried to tackle those issues by parameterizing the parameters θ_t themselves, and by building a function mapping additional time dependent covariates to θ_t. More specifically, assume that we are given $\mathbf{y} = (y_1, y_2, ..)$ gathering information about the the dynamics of the process. The idea is to build a mapping Ψ parameterized by φ such that

$$\theta_t = \Psi(\mathbf{y}_{1:t}; \varphi). \tag{3.14}$$

For instance in order to encode the time dependence, y_t can be an encoding of the time t. If one assumes there is a periodicity in the dynamics, with a period length of $\mathcal{T}$, then a natural choice of the features of y_t could be $f(t)$, where f is a $\mathcal{T}$-periodic function, such as $t \mapsto \cos(2\pi t/\mathcal{T})$.

Limitations

We identified several limitations in learning and inferring the predictive distribution with the state space model approach. First of all, we made it clear that the joint process (the concatenation of state and the observation process) of a SSM is Markovian. This can become an issue if there are important path dependencies in the observed process we aim to model. This issue can be alleviated when the observed process is not perfectly Markovian but has some lagged dependencies. For instance, it has been shown in Hamilton (1994) that one can model an autoregressive moving average (ARMA) process with a linear SSM. Secondly, linear SSMs do have quite restrictive assumptions, and may not be able to capture the true dynamics. Involving non-linear SSMs is significantly more challenging and involves simulation or approximation methods. Finally, SSMs do not allow to model the dynamics of the time series continuously throughout time. This is not necessarily an issue in our true data, but could be limiting in other applications.

3.2 Path-dependent Neural Jump ODE Forecasting

PD-NJODE is a framework described in Krach et al. (2022) that allows to forecast general stochastic processes, such as non-markovian and discontinuous ones. It is able to learn from a dataset consisting of irregularly and partially observed time series with random observations times and observed coordinates. Under certain assumptions, PD-NJODE is theoretically guaranteed to converge to the optimal predictions, where optimal is meant in the sense of the L^2 norm. This framework is designed for online forecasting, that is, it forecasts continuously in time, with predictions depending solely on past observations.

3.2.1 Motivations

Here, we give some motivations for using PD-NJODE in our anomaly detection algorithm.

First of all, this framework is based on a neural ODE-RNN which allows to model the dynamics of the data generation process continuously in time. This is a major difference to more traditional RNNs that are only able to model the dynamics for regular time intervals. Thus, one advantage is that one is able to forecast for any time in the future and not solely on regular

time grids. Furthermore, such a model offers a more natural way to model metrics that are believed to evolve continuously in time, although they are measured at regular time intervals as it is the case in our setup.

Secondly, PD-NJODE offers theoretical convergence guarantees under reasonable assumptions. We will discuss in Subsection 3.2.2 the applicability of those assumptions on both synthetic data and real data.

Thirdly, the framework is able to deal with irregularly observed data. It is not obvious why this feature might be of interest as we have regular observations in our true dataset. However, learning with irregular data by deleting some observations can be useful in two ways. On the one hand, it uses the data more efficiently, i.e can ingest more time series samples, and process samples more quickly. On the other hand, it makes the model robust. Indeed, the model is forced to learn multiple time steps ahead, rather than simply one step ahead. Thus the model is more likely to learn the true dynamics of the data generating process because of a varying forecasting horizon.

Lastly, it is probably an interesting framework when the quality of the dataset is not perfect, as in our case. While observing the data, we saw that there is often a significant portion of the KPI values that are zero. (Indeed, when plotting the distribution of KPI values, we often see an isolated mode at zero values). This is abnormal as most KPIs are expected to take strictly positive values (response time, transfer size). It could either be caused by true anomalies, or most commonly, an absence of measurement. In the second case, the ability to have irregular observations or masked coordinates (masking certain KPIs) might be useful. However, we did not exploit the coordinates masking feature and leave this to future work.

Setting

Let $X = (X_t)_{t\in[0,T]}$ be a d_X-dimensional adapted and càdlàg (right continuous, with left limit) stochastic process on a filtered probability space $(\Omega, \mathcal{F}, \mathbb{F} = (\mathcal{F}_t)_{0\leq t\leq T}, \mathbb{P})$.

Furthermore, one considers random observation times of the process, which are modelled as stopping times of another filtered probability space $(\tilde{\Omega}, \tilde{\mathcal{F}}, \tilde{\mathbb{F}} = (\tilde{\mathcal{F}}_t)_{0\leq t\leq T}, \tilde{\mathbb{P}})$,

- $n : \tilde{\Omega} \to \mathbb{N}_{\geq 0}$ is the random number of observations, which is integrable : $\mathbb{E}_{\mathbb{P}}[n]$,
- $t_i : \tilde{\Omega} \to [0, T]$, $0 \leq i \leq n$ are sorted stopping times,
- $\tau : [0, T] \times \tilde{\Omega} \to [0, T]$ maps t to the last observation time before t, i.e $(t, \tilde{\omega}) \to \max\{t_i(\tilde{\omega}) | t_i(\tilde{\omega}) \leq t\}$.

The information available up to time $t \in [0, T]$ is modelled as a σ-algebra

$$\mathcal{A}_t = \sigma(X_{t_i}, t_i | t_i \leq t),$$

which defines a filtration $\mathbb{A} = (\mathcal{A}_t)_{0 \leq t \leq T}$. We also have $\mathcal{A}_t = \mathcal{A}_{\tau(t)}$.

The goal of the framework is to estimate the conditional expectation process $\hat{X} = (\hat{X}_t)_{t \in [0,T]}$ defined as $\hat{X}_t = \mathbb{E}[X_t | \mathcal{A}_{\tau(t)}]$. Indeed, this process is the optimal approximation of X in $L^2(\Omega \times \tilde{\Omega}, \mathbb{A}, \mathbb{P} \times \tilde{\mathbb{P}})$.

The process $\tilde{X}^{\leq t}$ is introduced, being the continuous version of the rectilinear interpolation of X until t. This process carries all information of $\mathcal{A}_{\tau(t)}$ (up to potential observations that are the exact interpolations of the two neighbouring observations. In that case, one uses an extended version of $\tilde{X}^{\leq t}$). Please refer to Krach et al. (2022) for more details on $\tilde{X}^{\leq t}$. Using this statement, one can deduce by Doob-Dynkin Lemma that there exists a measurable function $F : [0, T] \times [0, T] \times BV^c([0, T]) \to \mathbb{R}^{d_X}$ such that $\hat{X}_t = F(t, \tau(t), \tilde{X}^{\leq \tau(t)})$, where $BV^c([0, T])$ denotes the set of continuous processes of bounded variation on $[0, T]$. The general idea of the framework is to model the function F.

One significant challenge arises here. It comes from the fact that F takes as input a path, the realisation of $\tilde{X}^{\leq \tau(t)}$, which is an element of an infinite dimensional space. One can observe that since $\tilde{X}^{\leq \tau(t)}$ is a rectilinear interpolation of observations until $\tau(t)$, it can actually be encoded in a finite dimensional vector. However, the dimension of this vector depends on the number of observations before $\tau(t)$. Thus, we cannot have a unified neural network to model F by taking $\tilde{X}^{\leq \tau(t)}$ directly as an input. In order to overcome this issue, the authors of Krach et al. (2022) propose to replace $\tilde{X}^{\leq \tau(t)}$ by its truncated signature. We denote by $\pi_m(\tilde{X}^{\leq \tau(t)})$ the truncated signature of order m of $\tilde{X}^{\leq \tau(t)}$.

Remark 3.2 *The authors actually propose a more general formulation of the framework, where observations of X are allowed to be partial. This means that at observation time t_i, only a subset of coordinates $J_i \subset \{1, .., d_X\}$ are actually observed.*

3.2.2 Framework

Here, we describe the general architecture of the framework.

Let $\mathcal{X} \subset \mathbb{R}^{d_X}$, $\mathcal{H} \subset \mathbb{R}^{d_H}$, $\mathcal{Y} \subset \mathbb{R}^{d_Y}$ be the observation, latent, and target space respectively, with $d_X, d_H, d_Y \in \mathbb{N}$ (usually $d_X = d_Y$). d_m denotes the dimension of the truncated signature of order m. We have the following feed-forward Neural networks

- $f_{\theta_1} : \mathbb{R}^{d_H} \times \mathbb{R}^{d_m} \times \mathbb{R}^{d_X} \times [0, T] \times [0, T] \to \mathbb{R}^{d_H}$ modelling the ODE dynamics,
- $\rho_{\theta_2} : \mathbb{R}^{d_H} \times \mathbb{R}^{d_m} \times \mathbb{R}^{d_X} \to \mathbb{R}^{d_H}$ modelling jumps at new observations,

- $g_{\theta_3} : \mathbb{R}^{d_H} \to \mathbb{R}^{d_Y}$ modelling the readout map.

The Path-Dependent Neural Jump ODE model is defined by

$$\begin{aligned} H_0 &= \rho_{\theta_2}(0, X_0, 0), \\ dH_t &= f_{\theta_1}(H_{t-}, \pi_m(\tilde{X}^{\leq \tau(t)} - X_0), X_{\tau(t)}, \tau(t), t)dt, \\ &\quad + \left(\rho_{\theta_2}(H_{t-}, \pi_m(\tilde{X}^{\leq \tau(t)} - X_0), X_{\tau(t)}) - H_{t-}\right) du_t \\ Y_t &= g_{\theta_3}(H_t), \end{aligned} \tag{3.15}$$

where $H = (H_t)_{t\in[0,T]}$ and $Y = (Y_t)_{t\in[0,T]}$ are the latent and output processes respectively, i.e. solutions of the above SDE system. The set parameters is denoted by $\theta := (\theta_1, \theta_2, \theta_3)$. The jump process u is simply defined by

$$\begin{aligned} u : \tilde{\Omega} \times [0, T] &\to \mathbb{R}, \\ (\tilde{\omega}, t) &\mapsto \sum_{i=1}^{n(\tilde{\omega})} \mathbf{1}_{[t_i(\tilde{\omega}),T]}(t). \end{aligned}$$

Here, we expose the theoretical objective function. Let $\mathbb{D}$ be the set of d_X dimensional adapted processes on $(\Omega \times \tilde{\Omega}, \mathbb{A}, \mathbb{P} \times \tilde{\mathbb{P}})$, then

$$\Psi : \mathbb{D} \to \mathbb{R}_{\geq 0}, \tag{3.16}$$

$$Z \mapsto \mathbb{E}_{\mathbb{P}\times\tilde{\mathbb{P}}}\left[\frac{1}{n}\sum_{k=1}^{n}(|X_{t_k} - Z_{t_k}|_2 + |X_{t_k} - Z_{t_k-}|_2)^2\right]. \tag{3.17}$$

The objective function is derived from Ψ as

$$\Phi : \Theta \to \mathbb{R}_{\geq 0}, \tag{3.18}$$

$$\theta \mapsto \Psi(Y^\theta(X)), \tag{3.19}$$

where Θ is the parameter space, and Y^θ is the frameworks output for parameter θ.

In practice, the Monte Carlo approximation is used. For N independent realisations $X^i \sim X$, and $(n^i, t_1^{(i)}, .., t_{n^{(i)}}^{(i)}) \sim (n, t_1, .., t_n)$, $i \leq N$, it takes the following form

$$\hat{\Phi}_N : \Theta \to \mathbb{R}_{\geq 0}, \tag{3.20}$$

$$\theta \mapsto \frac{1}{N}\sum_{i=1}^{N}\frac{1}{n^{(i)}}\sum_{k=1}^{n^{(i)}}\left(|X^i_{t_k^{(i)}} - Y^{\theta,i}_{t_k^{(i)}}|_2 + |X^i_{t_k^{(i)}} - Y^{\theta,i}_{t_k^{(i)}-}|_2\right)^2, \tag{3.21}$$

where $Y^{\theta,i} = Y^\theta(X^i)$.

Assumptions

We briefly state the assumptions on the stochastic process X and the observation times for the PD-NJODE framework to converge to the true conditional expectation. As we don't consider the possibility for certain coordinates to be masked at observation times, this aspect is not elaborated here.

First of all, the framework imposes that any pair of observation time can not be arbitrarily close with non-zero probability. This condition is always fulfilled in our data as the difference between two observation times is lower bounded by the sampling period of 5 minutes. Furthermore, it is assumed that X is observed at a jump (i.e point of discontinuity of X) with probability zero. We consider this to be always fulfilled in our case as the underlying data generating process is believed to be continuous.

Thus, we are left with the two assumptions

1. the jth coordinate of F, F_j is continuous and differentiable in t. By denoting f_j its derivative with respect to t, f_j and F_j are polynomially bounded in $X^* = (\sup_{s\in[0,t]} |X_s|_1)_{t\in[0,T]}$, i.e there exist $B, p > 0$ such that

$$|F_j(t, \tau(t), \tilde{X}^{\leq\tau(t)})| + |f_j(t, \tau(t), \tilde{X}^{\leq\tau(t)})| \leq B(X_t^* + 1)^p,$$

2. X^* is L^{2p}-intergrable, i.e $\mathbb{E}[(X_T^*)^{2p}] < \infty$.

We show in B.1.3 that those condition are fulfilled with our synthetic design (see 2.1). Lets briefly discuss the real data case. As mentioned earlier, each KPI univariate time series has values living in the $[0, 1]$ interval. Therefore, the integrability assumption always holds in our case. Thus, we are left with the first assumption. However, it is rather difficult to check whether F is continuous or differentiable in practice, so we can only assume that this holds.

3.2.3 Guarantees

The PD-NJODE framework is interesting as it comes with theoretical guaranties which we briefly discuss here. The reader can refer to Krach et al. (2022) for details.

First, this framework exploits signature theory, a powerful operator on bounded variation continuous paths, that yields a universal approximation property. The signature of a path maps paths to an infinite dimensional space. Therefore in practice, one uses the truncated signature. Theorem 3.7 (Krach et al., 2022) gives an approximation property for the truncated signature. Under certain conditions, one can approximate uniformly well

any function on a compact set of bounded variation continuous paths that are not tree-like equivalent (see paper), using a linear mapping of the truncated signature and provided that the degree of the truncated signature is sufficiently high. This result is beneficial as it allows to extract all (up to approximation) information present in $\tilde{X}^{\leq\tau(t)}$, into a fixed size vector, and regardless of the number of observations occurring before t. This aspect is particularly useful if we try to learn on non-Markovian processes. All possible $\tilde{X}^{\leq\tau(t)}$ lie in a compact subset because $\tilde{X}^{\leq\tau(t)}$ is piece-wise linear, and one can make sure that the paths are not tree-like equivalent by appending the time. Regarding the convergence of the loss function, it it first shown (Theorem 4.1) that the optimal parameter for a certain level of signature truncation and NN size converges with the truncation level to the overall optimum. This means that if one can learn the optimal parameters of the theoretical loss function for a sufficiently high level of signature truncation and NN size, one should get nearly optimal predictions. Afterwards, it is shown (Theorem 4.4) that for a fixed signature level and and NN size, the Monte Carlo approximation of the loss function converges almost surely to the theoretical loss function uniformly on the parameter space, with the number of samples tending to infinity. The final result shows that with sufficiently high signature level and NN size and sufficiently many samples, one can learn the true conditional expectation process using the Monte Carlo approximation of the loss function.

3.3 Predictive Distribution Estimation

In this section, we discuss how we can recover an approximation of the conditional distribution $X_t|\mathcal{A}_{\tau(t)}$, using the PD-NJODE framework, which is nothing more than the predictive distribution of X_t, given previous observations. This aspect was briefly elaborated in Krach et al. (2022), Section 5. For reasons that are explained later, we investigate how we can get an approximations of the conditional distribution given a finite number of conditional moments.

Leads from PD-NJODE paper

Firstly, the framework can be extended to estimate conditional moments of the process, $\mathbb{E}[X_t^m|\mathcal{A}_{\tau(t)}]$ where $m \in \mathbb{N}$. In order to theoretically guarantee the convergence of the predictions to the true conditional moments, the same assumptions on X have to apply to X^m. One can then ingest the joint process $Z := (X^m)_{m\in\mathcal{M}}$ into the framework, where $\mathcal{M} \subset \mathbb{N}$ corresponds to the finite set of conditional moments we want to estimate. We observe that the dimension of X_t^m grows exponentially with m, as we have to account for each combination of coordinates. For instance if $m = 2$, $\mathbb{E}[X_t^2|\mathcal{A}_{\tau(t)}]$ comprises $\mathbb{E}[X_{t,i}X_{t,j}|\mathcal{A}_{\tau(t)}]$ for $i, j = 1..d_X$, where $X_{t,i}$ denotes the ith coordinate of X_t.

According to this, the dimension of X_t^m should be $\binom{d_X+m-1}{m}$. However, if we only consider the marginal conditional moments, that is simply $\mathbb{E}[X_{t,i}^m|\mathcal{A}_{\tau(t)}]$ for $i = 1, .., d_X$, this dimension drops down to only d_X. In a nutshell, this first approach consists in using conditional moments to predict the conditional distribution, or simply the per-coordinate marginal conditional distribution.

A second approach would be to approximate the conditional moment generating function (CMGF) $u \mapsto \mathbb{E}[\exp(uX_t)|\mathcal{A}_t]$ on an open set $(-\delta, \delta)$. Indeed, the conditional distribution $X_t|\mathcal{A}_t$ is uniquely characterised by the previous function on an open interval including zero. Then one can estimate the conditional distribution by applying an inverse Laplace transformation on the CMGF. However, this process has some bottlenecks. First, applying PD-NJODE to estimate $\mathbb{E}[\exp(uX_t)|\mathcal{A}_t]$ for $u \in \mathbb{R}$ assumes that the hypothesis of the framework (on X) hold for $\exp(uX)$. Secondly, one can only estimate $\mathbb{E}[\exp(uX_t)|\mathcal{A}_t]$ for finitely many $u \in \mathbb{R}$ which will thus simply give us an approximation of the CMGF. The quality of the approximation then surely depends on the number of input / outputs we have from the CMGF, and this would be a crucial but difficult analysis. Finally, the inverse Laplace transformation takes the form of an integral, and is therefore difficult to compute.

Approach

Given the two leads mentioned above, we choose the first approach for two principal reasons. First, because approximating both the CMDF, and then the conditional distribution is complex, as explained previously. Secondly, because the conditional moments alone provide enlightening and interpretable information on the conditional distribution. Furthermore, extending the PD-NJODE framework to approximate conditional moments is straightforward in theory.

Related Moment Problems

We mention related moment problems that have been well studied by researchers.

In the literature, the problem of reconstructing a function given the moments is denoted as the **classical moment problem** (Bandyopadhyay et al., 2005). Let μ be a measure on $I \subset \mathbb{R}$. Given a set of values $\{m_i|i \in \mathbb{N}\}$, one would like to find necessary or sufficient conditions under which there exists one (or more) μ-measurable function f such that the moments of f match these values, i.e $m_i = \int_I x^i f(x)dx$ for all $i \in \mathbb{N}$. This problem is denoted **Hausdorff problem**, and **Stieltjes problem** (Tagliani, 1999) respectively for $I = [0, 1]$ and $I = \mathbb{R}_+$. It is known that in case a solution f exists, it is unique for the Hausdorff problem, but there can be infinitely many solutions for the

Stieltjes problem. When a solution f exists but is not unique, some authors propose to formulate the problem as an optimisation problem, where the goal is to compute the function that has maximal entropy (Bandyopadhyay et al., 2005; Frontini and Tagliani, 2007).

As a first remark, when linking this problem to probabilistic distributions, we restrict ourselves to distributions that admit a density function. In our case, the values of the KPIs are normalised in the $[0, 1]$ interval, which is encouraging considering the last remark on solutions uniqueness. However, the same conclusion can not hold when we only consider a finite set of moments. In this case, the problem is denominated as the **finite-moment problem**.

The so-called method of moments (MOM) (John et al., 2007) is one of the most common and simplest methods addressing this issue, when given a finite number of moments. Basically, it assumes that the distribution comes from a parameterised family of distributions $\mathcal{D} = \{D_\theta | \theta \in \Theta\}$. One example is the family of normal distributions $\mathcal{D} = \{\mathcal{N}(\mu, \sigma^2) | \theta = (\mu, \sigma) \in \mathbb{R} \times \mathbb{R}_+\}$. Using this assumption, and assuming that we know some of the moments m_i, the goal is to estimate the parameter $\theta \in \Theta$ such that the moments of D_θ match all the given m_i.

As we can immediately see, there is not necessarily a unique $D_\theta \in \mathcal{D}$. On the one side it is possible that their is more than one solution, while on the other side, there can be no solution at all. Lets consider the family of normal distributions described previously. If we simply know that the first moment has value μ, we cannot find a unique θ. However, if we look for θ such that the first three moments are respectively $0, 1, 1$ then we won't find any solution since for a centred Gaussian distribution (first moment 0), we necessarily have that the third moment is also 0. However, we note that for some distribution families, the distributions are uniquely characterised by some of its moments. For instance Gaussian distributions are uniquely characterised by their first two moments.

As we shall see later, this simple approach is very convenient when we are dealing with a small amount of moments. Furthermore, it does not require expensive computations.

Chapter 4

Proof of Concept : Anomaly Detection Pipeline

In this chapter, we expose the structure of our proposed anomaly detection pipeline. In 4.1, we explain how the PD-NJODE forecasting framework is integrated into our pipeline, as well as the different modifications that were tested and build in. In practice, we use this framework to solely predict the two first moments of the conditional distribution (expectation and second moment). This is already challenging as we have to make both predictions consistent with one another, i.e. so that the conditional variance is positive. However, both the conditional expectation and variance should give sufficient information to infer the range of admissible observations (i.e. that are not outliers), provided that the conditional distribution is unimodal and not too complex. The conditional expectation gives a reference level, while the conditional variance provides a measure of variability. In 4.2, we explain how we proceed to estimate the conditional distribution resulting from the output conditional moments given by the forecasting module. Then, we describe in 4.3 how to score the different observations based on the estimated conditional distribution, as well as how those scores are aggregated with one another. The final scores are in turn used to predict a label, based on a fixed threshold.

We briefly explain the setting for our pipeline. We suppose that we are given two datasets; one "clean" dataset $\mathcal{D}_c$ to train the forecasting model, and one dataset $\mathcal{D}_a$ which contains anomalies, on which we run inference through the pipeline.

For clarity, we illustrate what these datasets correspond to in practice, both for synthetic data, which we use as a benchmark, and for the true data. In the synthetic case, both datasets are generated using the same data generating process. However, for $\mathcal{D}_a$, we ingest the previously described anomalies. Usually, we only include one anomaly type into $\mathcal{D}_a$. The purpose of this is to

specifically design scoring modules that target specific anomaly types, and for which we can evaluate their effectiveness. Indeed, as we will see later, the scoring module has learnable aggregating weights and some weighting schemes are more suited for certain anomaly types than for others. Since the aggregation weights take a simple and interpretable form, we can reuse them directly in a real setting, or use them to handcraft new aggregation weights. Since the forecasting module requires significantly more computations than the other modules, it is perfectly reasonable to run the pipeline with different aggregation weights, to detect anomalies of different types. In a real setting, it is difficult to get access to a clean dataset, as the data is usually not labelled. Thus, $\mathcal{D}_a$ and $\mathcal{D}_c$ would have the same underlying distribution. If we are able to learn the forecasting model without over-fitting the data, we expect it to still be effective to detect anomalies as the model will usually better fit normal events compared to rare events.

The assumptions on the dataset can be stated as follows.

- The datasets originate from a d_X-dimensional data generating process $X = (X_t)_{t\in[0,T]}$. It consists of time series which are regularly observed; we denote by Δt the smallest observation period. That is, it consists of samples $X^{(j)}$ which are observed at times $(t_0, .., t_n)$ where $t_k = i\Delta t$ and $n = T/\Delta t$. Because of the regularity of observations, we will refer to an elementary time difference Δt between observation as one time step. Each time difference between observation can thus be written as $s\Delta t$, $s \in \mathbb{N}$, i.e s time steps.
- $\mathcal{D}_c$ includes irregularly sub-sampled time series. We will give some motivations for this aspect later.

$$\begin{aligned}\mathcal{D}_c &= \{x^{(i)}\}_{i=1..N}\\ x^{(i)} &= (X^{(i)}_{t'_0}, .., X^{(i)}_{t'_{n^{(i)}}}) \ \ \forall i = 1..N\\ X^{(i)} &\sim X\end{aligned}$$

  $(t'_0, .., t'_{n^{(j)}})$ is a random subset of $(t_0, .., t_n)$, where each observation time t_k is independently selected with probability p.
- For $\mathcal{D}_a$, the data generating process is $\tilde{X}$, a modification of X which also ingests anomalies. However, it includes the complete regularly observed time series.

$$\begin{aligned}\mathcal{D}_a &= \{x^{(i)}\}_{i=1..\tilde{N}}\\ x^{(i)} &= (X^{(i)}_{t_0}, .., X^{(i)}_{t_n}) \ \ \forall i = 1..\tilde{N}\\ X^{(i)} &\sim \tilde{X}\end{aligned}$$

- Furthermore, in the case of synthetic data, $\mathcal{D}_a$ includes anomaly labels as well. For each $x^{(i)}$, there is a binary time series $l^{(i)}$ of the same size where each entry $l^{(i)}_{t_k,j} \in \{0,1\}$ for time t_k and dimension j.

4.1 Forecasting Module

The forecasting module exploits the PD-NJODE framework to learn and infer the conditional moments of the data generating process. Here, we explain the setting for both training and inference. We also discuss some key modifications made to the framework, especially for enforcing the consistency of predicted moments.

As stated before, we are interested in estimating the marginal conditional moments. That is, for each dimension $j \in \{1..d_X\}$, and for moments $m \in \{1..M\}$, we aim for $\mathbb{E}[X^m_{t,j}|\mathcal{A}_{\tau(t)}]$. Therefore, the input consists in the concatenation of the time series at power $m \in \{1..M\}$ among the dimension, which results in a md_X-dimensional input.

Training

During training, we use irregularly observed time series. The main motivation is to force the model to learn the conditional moments for arbitrary forecasting horizons. Thus, the observation time selection probability p is a sensitive parameter which should depend on how many steps ahead we want to learn to forecast. It also makes the training more efficient, as each sample has less observations and is quicker to process. As first improvement, we propose to do the random selection directly in the training loop. This allows to train on the same data sample several times with different observation times, and artificially grows the dataset. We denominate this improvement as **on-the-fly random observation selection**. As we will show later, this improvement is especially important when the dataset size is small. Indeed, in the real data, we are often confronted with smaller datasets, especially when we solely train a model for a single system.

Inference

During inference, there are few advantages to the use of irregularly observed time series. The main difference is that we do multiple horizon forecasts. Let $t = k\Delta t$ be the time for which we would like to compute the conditional moments. Then, one can compute $\mathbb{E}[X^m_{t,j}|\mathcal{A}_{\tau(t)}]$ for $\tau(t) = (k-s)\Delta t$, $s \in \mathbb{N}$. We select a set $S \subset \mathbb{N}$ of forecasting horizons.

Forecasting with multiple forecasting horizons is a key aspect of our method and requires an explanation. The intuition is that depending on the type of

anomaly, and the position of observation in the anomaly sequence (beginning, end), different forecasting horizons can bring an unequal amount of information to classify the observation as anomalous, or not.

Even if the method is probably generalisable to multiple moments, we simply experimented with two. This is already challenging since the moments have to be consistent with each other.

Key modifications

We made certain key modifications to the PD-NJODE framework for several reasons. Firstly, because of some assumptions or beliefs that we have on the data, we can incorporate some inductive bias into the model. Secondly, the task of predicting conditional moments that are consistent with each other is more complicated in practice than just predicting them independently. This last reason explains why in our experiment, we restricted ourselves to the prediction of the conditional moments 1 and 2.

4.1.1 Training Loss function

Firstly, we observed that a slight modification of the objective function could make the convergence of the model faster. In 3.20, we replace the objective

$$\left(|X_{t_k} - Y_{t_k}|_2 + |Y_{t_k} - Y_{t_k-}|_2\right)^2$$

with

$$|X_{t_k} - Y_{t_k}|_2^2 + |X_{t_k} - Y_{t_k-}|_2^2 .$$

One part of the modification is to swap Y_{t_k} with X_{t_k} in the second term, and was already mentioned in Krach et al. (2022). The other one is to square both norm terms directly before adding them up. We intuitively believe that this could improve the training, since both terms are unbalanced. Since Y_{t_k} is computed using observation X_{t_k} (that is after a jump), it is an easier task for neural networks to learn how to make $|X_{t_k} - Y_{t_k}|$ small rather than to make $|X_{t_k} - Y_{t_k-}|$ small. However, in the initial objective, the value of the bigger term $|X_{t_k} - Y_{t_k-}|$ will have a relatively high influence on the gradient of the parameters with respect to the smaller term $|X_{t_k} - Y_{t_k}|$. This can result in some learning instabilities. Furthermore, the new formulation does not involve the computation of a square root, which results in numerical advantages as the gradient of the square explodes at 0.

Making the predictions of the conditional moments consistent with each other is not an easy task. For instance, for moments 1 and 2, we always have in theory that the conditional variance $\mathrm{Var}[X_t|\mathcal{A}_{\tau(t)}] = \mathbb{E}[X_t^2|\mathcal{A}_{\tau(t)}] - (\mathbb{E}[X_t|\mathcal{A}_{\tau(t)}])^2 \geq 0$, but it is not guaranteed for the predictions. Furthermore,

this expression often takes values that are near zero. Indeed, we always have $\mathrm{Var}[X_t^2|\mathcal{A}_t] = 0$, that is the conditional variance falls to zero at each observation.

In order to tackle this issue, we decided to change the loss function by introducing some consistency terms. For notational simplicity, let us denote the predictions of the framework as

$$\begin{aligned} E_t &:= \hat{\mathbb{E}}[X_t|\mathcal{A}_{\tau(t)}], \\ S_t &:= \hat{\mathbb{E}}[X_t^2|\mathcal{A}_{\tau(t)}], \end{aligned}$$

where the square is applied element-wise.

A first approach is to calibrate the prediction of the conditional variance $S_{t_k-} - E_{t_k-}^2$ with the empirical conditional variance, by adding the following loss term.

$$\left|(S_{t_k-} - E_{t_k-}^2) - (X_{t_k} - E_{t_k-})^2\right|_2^2 \tag{4.1}$$

Another possibility is to predict an additional variable for the conditional variance, which we will denote by V_t. This has the advantage to decouple to some extend the predicted conditional variance from the predicted conditional moments. We therefore included two loss terms that force the model to fit this prediction with the prediction of the two conditional moments, and the empirical conditional variance. We also added a term to enforce this prediction to be zero at observations.

$$\begin{gathered} \left|V_{t_k-} - (S_{t_k-} - E_{t_k-}^2)\right|_2^2, \\ \left|V_{t_k-} - (X_{t_k} - E_{t_k-})^2\right|_2^2, \\ |V_{t_k}|_2^2. \end{gathered} \tag{4.2}$$

Other solutions exist in order to explicitly enforce the positiveness of the conditional variance such as the following terms

$$\begin{gathered} \mathrm{ReLU}(-V_{t_k-})^2, \\ \mathrm{ReLU}(E_{t_k-}^2 - S_{t_k-})^2. \end{gathered} \tag{4.3}$$

In case we explicitly predict V_{t_k}, one can use a readout activation function that forces the predictions to be positive such as the exponential, or ReLU. However, we noticed that these have a tendency to bias the prediction towards larger values than expected. Furthermore, even though the conditional variance should be zero at observations, the exponential activation would never result in zero predictions. On the other hand, the ReLU activation could suffer from multiple issues such as dying neurons, and non-differentiability at 0.

Finally, we observed that making the weights of different loss terms vary over time leads to better results. Intuitively, the quality of the prediction of the conditional variance highly depends on the prediction of the conditional expectation. Therefore, we put more weights on terms involving the prediction of the conditional expectation at the beginning of the training, and then increase the other terms over training epochs. However, one could also consider to make separate models, one for each prediction term (i.e conditional expectation / variance) and training them sequentially.

4.1.2 Validation Loss Function

We propose a lead to build the validation loss, which is used to select the best model over the training. This aspect is particularly important in the real data case, where we cannot compare the predictions of the model with the true optimal predictions. There are several reasons why in our case, we might have to choose a validation loss which is different than the training loss.

First of all, not all the loss terms mentioned above do reflect the performance of the model. Instead, we are interested in a loss that quantifies in a simple fashion the quality of the predictions for the conditional expectation and variance, before and after observations. Secondly, it would not be consistent to take a loss function that varies over the training, as it is the case when we vary the weights of the different loss terms in the training loss.

In response to the first remark, we propose a loss function with the terms

$$\begin{gathered} |X_{t_k} - E_{t_k}|_2^2, \\ |X_{t_k} - E_{t_{k-}}|_2^2, \\ |V_{t_{k-}} - (X_{t_k} - E_{t_{k-}})^2|_2^2, \\ |V_{t_k}|_2^2. \end{gathered} \tag{4.4}$$

4.1.3 Other improvements

- **NN input scaling :** Scaling the inputs is always an important aspect when training neural networks, and often result in a more stable learning process Heiss et al. (2021). This concerns several variables in our setting. First, we make sure that the range of the process is approximately in the $[0, 1]$ interval. Note that there is no particular operation to perform on the real dataset, as it is already the case. We also rescaled the observation times with a factor of $1/T$ so that each observation time belongs to the $[0, 1]$ interval. Lastly, we scaled the time difference $t - \tau(t)$ in (4.5) with a constant factor, such that it has an average value of 1. This factor depends on the selection probability p.

- **Time periodic feature ODE-NN input :** As there is some periodicity in our data (as argued before), the idea of this improvement is to add a feature to our model that accounts for the position of time in the period, while making the model less dependent on the absolute time. Therefore, we replaced the input observation time t in the ODE network, by features encoding the position of t in the period. For a period $\mathcal{T}$, we used the features $\cos(2\pi t/\mathcal{T})$, and $\sin(2\pi t/\mathcal{T})$. In real world data $\mathcal{T}$ could correspond to 1 day or 1 week depending on the periodicity we want to incorporate. These features have the advantage of being smooth as opposed to a more naive solution like the modulus $t \bmod \mathcal{T}$.
- **Training with Artificial points :** The model is supposed to learn to jump at new observation times, through both the jump and the readout NN (ρ_{θ_2} and g_{θ_3} as in 3.2.2). Since in our case we do not deal with incomplete observations, we know in advance what the composition of those networks should give as an output; we want the output for the conditional moment of order m to be exactly the observations to the power m. Formally, suppose that we have an observation at time t ($\tau(t) = t$), then we have for $m = 1$

  $$g_{\theta_3} \circ \rho_{\theta_2}(H_{t-}, \pi_n(\tilde{X}^{\leq t} - X_0), X_t) = X_t.$$

  Following the same reasoning, the conditional variance should jump to 0 at observations (this would also hold for all central moments of higher order). In case the dataset is small, the model might actually struggle to learn this. Therefore, we propose an improvement that consists in training the jump and readout networks on artificially generated points, alongside from the regular training. Concretely, we generate random data for each input term H_{t-}, $\pi_n(\tilde{X}^{\leq t} - X_0)$ and X_t. The loss function is then simply the squared difference between $g_{\theta_3} \circ \rho_{\theta_2}(H, \pi, X)$ and X.

The final version of the framework is the following :

$$\begin{aligned} H_0 &= \rho_{\theta_2}(0, X_0, 0) \\ dH_t &= f_{\theta_1}(H_{t-}, \pi_n(\tilde{X}^{\leq \tau(t)} - X_0), X_{\tau(t)}, \tau(t), s(t-\tau(t)), \\ &\quad \cos(2\pi t/\mathcal{T}), \sin(2\pi t/\mathcal{T}))dt \\ &\quad + \left(\rho_{\theta_2}(H_{t-}, \pi_n(\tilde{X}^{\leq \tau(t)} - X_0), X_{\tau(t)}) - H_{t-}\right) du_t \\ Y_t &= g_{\theta_3}(H_t) \end{aligned} \tag{4.5}$$

4.2 Distribution Inference Module

In the previous section, we predicted the marginal conditional moments of the marginal conditional distribution $X_{t,j}|\mathcal{A}_s$, for each coordinate j. From

those, we aim at inferring an estimate $\hat{Q}^j_{s,t}$ of $X_{t,j}|\mathcal{A}_s$.

Inspired from MoM John et al. (2007), we are given moments μ_1, .., μ_k and we aim to infer a distribution Q that matches those moments. Assume that the distribution belongs to a distribution class $f_Q(x;\theta)$ parameterised by k parameters θ_1,..,θ_k. Assume furthermore that the moments can be expressed as functions of those parameters,

$$\mu_i = \mathbb{E}[W^i] = g_i(\theta_1, .., \theta_k), \tag{4.6}$$

where W is a random variable with distribution Q.

The method consists in estimating the parameters $\theta_1, .., \theta_k$ such that 4.6 is verified. We briefly illustrate the method with the following example.

Gaussian Distribution Inference

Suppose we choose the class of Gaussian distributions $\mathcal{N}(\mu, \sigma^2)$, which is parameterised by $\theta_1 = \mu$, $\theta_2 = \sigma^2$. Then we directly have

$$\mu_1 = \mu \tag{4.7}$$
$$\mu_2 = \sigma^2 + \mu^2 \tag{4.8}$$

We chose to implement this approach in our pipeline. Following John et al. (2007), there are multiple distribution classes that can be reconstructed using the two first moments, such as the half-normal, log-normal, beta, or gamma distribution.

4.3 Scoring Module

Here we discuss how given the conditional distribution, one can identify an observation as an anomaly.

By combining the framework of PD-NJODE with the method of moments, we are able to formulate a belief on the conditional distribution $Q_{t,s}$ of X_t based on all the information up to time $s \leq t$, $\mathcal{A}_s$. We assume that :

$$X_t|\mathcal{A}_s \sim Q_{t,s} \tag{4.9}$$

We call this our belief under to the normality assumption, i.e. if X is not anomalous at time t.

Now having an observation x_t of X_t, we want to infer a value quantifying the "likelihood" of x_t with respect to $Q_{t,s}$.

We want to infer scores that are small for normal observations and big for anomalous observations. Moreover, we would like these scores to be always positive. The following is a scoring method proposition.

P-value based Scoring

In case where $Q_{t,s}$ is a simple distribution, i.e. having a single mode, one can assume that the farther x_t lands in the tails of $Q_{t,s}$, the less likely it is (and the more probable it is to have an anomaly).

In order to quantify this, one can compute the (two-sided) p-value of x_t conditioned on the hypothesis $\mathcal{H}$ that $X_t|\mathcal{A}_s \sim Q_{t,s}$, given by

$$p_{s,t} = 2 * \min(\mathbb{P}[X_t \leq x_t|\mathcal{H}], \mathbb{P}[X_t \geq x_t|\mathcal{H}]). \tag{4.10}$$

Note that if x_t diverges towards $\pm\infty$, it will solely have its p-value converging towards 0, whereas it might become "arbitrarily abnormal". Thus, it is probably preferable to apply some transformation to $p_{s,t}$ in order to compute a score that is better able to differentiate normal from anomalous observations. The logarithmic function is a good candidate. Thus, we set $v_{s,t} = -\log(p_{s,t})$.

4.4 Score Aggregation Module

The idea of our score aggregation module is to quantify the "abnormality" of x_t not solely based on $\mathcal{A}_s$ for a single s, but rather for several $s < t$ values. Also, since most anomalies are not punctual but rather prolonged, it is of interest to consider scores of neighbouring observations in time. This could also be beneficial in order to recover anomaly windows, as it would smooth the scores.

Method

Suppose we have a time series sample $\{x_{t_j}, j = 0..n\}$ for an increasing sequence of times $t_0 < .. < t_n$, and let $i \in \{0, .., n\}$. We compute v_{t_j,t_i} for each $j < i$.

We would like to have a unified score for the "abnormality" of x_{t_i}. To address this, we simply build this score as a weighted sum of the v_{t_j,t_i} values

$$\text{score}'_{t_i} = \sum_{j<i} a_{t_i-t_j} v_{t_j,t_i}. \tag{4.11}$$

In order to aggregate scores from neighbouring observation in time, one can again opt for a weighted sum

$$\text{score}_{t_i} = \sum_j b_{|t_i-t_j|} \text{score}'_{t_j}. \tag{4.12}$$

Since we are given regularly sampled time series with smallest time interval Δt, we can simply represent the weights $a_{k\Delta t}$ for $k \in \mathbb{N}$. Also, we restrict ourselves to a finite horizon forecasting, where we only compute v_{t_j,t_i} for i, j such that $t_i - t_j \leq K\Delta t$ with $K \in \mathbb{N}$ the maximal forecasting horizon. The same reasoning applies to the b weights. Therefore, we can store finite many $a_{k\Delta t}$ and $b_{k\Delta t}$. For ease of notation, we can rewrite $a_{k\Delta t}$ and $b_{k\Delta t}$ as a_k and b_k respectively.

Thus, the complete aggregated scoring formula would be

$$\text{score}_{t_i} = \sum_{l=-L}^{L} b_l \sum_{k=1}^{K} a_k v_{t_{i+l-k},t_{i+l}} \tag{4.13}$$

One advantage of this procedure is that it is interpretable. If we are given some weights $w_{k\Delta t}$ that perform well for differentiating between observations of anomalous and normal regions, then one can identify the best forecasting horizons leading to this differentiation.

One can naturally generalise this procedure by combining the weights a_k and b_k.

$$\text{score}_{t_i} = \sum_{l=-L}^{L} \sum_{k=1}^{K} w_{lk} v_{t_{i+l-k},t_{i+l}}. \tag{4.14}$$

Design

The design of the weights in the score aggregation module requires some further explanation.

First, different weights might be more or less effective to detect certain types of anomalies. Some anomalies might be easier to detect if we account for the scores obtained from longer forecasting horizons. Indeed, if an anomaly results in a sudden spike in the time series values, then a small forecasting horizon might be sufficient. However, if the anomaly progressively changes the trajectory of the time series values, one might be more inclined to rely on scores from longer forecasting horizons. In the latter case, the weights a_k would need to be significant for large k's. Furthermore, aggregating the scores from neighbouring observations might be helpful to detect anomalies on a given observation. Take for instance the noise anomaly described in section 2.3, i.e. the observations are more noisy for a certain period. Then an observation might (by luck) be less noisy than neighbouring observations. However, it can still be considered to be included in the anomalous regions. Therefore, the neighbouring observation scores would be important to classify this observation as anomalous. In this case, one would make the weights

a_l significant for $l \neq 0$. On the contrary, for spike anomalies, one would be inclined to set $a_l = 0$ for $l \neq 0$.

One way of designing those weights is to formulate the problem as a classification problem in the synthetic data setting, by targeting the true anomaly labels. Concretely, the idea is to pre-define the types of anomalies we want to detect in the real data. Then, one uses the pipeline to learn the corresponding weights in an annotated synthetic dataset, containing the targeted anomaly. Finally, one reuses the learned weights for the real world case. Otherwise, one could simply handcraft the weights, according to simple reasoning such as the above. This explains why we made the score aggregation module so simple; the purpose is to make it interpretable, and easily designable with respect to the targeted anomaly type.

In the following, we detail how to formulate the weighted aggregation module as a machine learning classification problem in the synthetic data setting. First of all, we aim to infer anomaly labels $l_{t_i} \in \{0, 1\}$ for each observation, where 1 corresponds to an anomaly. Therefore, we aim to output final scores that fall in the interval $[0, 1]$. A convenient way is to apply a Sigmoid readout function on the aggregated scores. In order to enforce the positiveness of weights, we propose to pre-parameterise the weights. Indeed, we don't want that the $v_{t_j,t_i} > 0$ influence negatively the final scores. Therefore, we can set

$$a_l = \sigma(a'_l) > 0, \tag{4.15}$$

$$b_l = \sigma(b'_l) > 0, \tag{4.16}$$

where σ is a positive activation function. With this, one can directly optimise the a'_l and b'_l.

4.5 Thresholding

Finally, we use the aggregated scores score_{t_i} to output labels $l_{t_i} \in \{0, 1\}$. This is through a pre-defined threshold. We claim that in our method, the threshold does not need to be dynamic since the conditional variance already accounts for varying discrepancy of observations.

Chapter 5

Experiments

The implementation of the synthetic data generation, and the anomaly detection pipeline including the forecasting pipeline can be found at : https://github.com/MarkusChardonnet/Probabilistic_forecasting_for_Anomaly_Detection. It is mainly based on the implementation of Krach et al. (2022), the code of which is available at : https://github.com/FlorianKrach/PD-NJODE. The code uses the programming language Python, together with the Pytorch 1.12 package. It is designed to run on CPUs or GPUs with CUDA API.

5.1 Synthetic Data Generation

5.1.1 Ornstein-Uhlenbeck based generator

Here, we explain how to sample time series from the data generation model introduced in 2.2.1.

First, we set the starting point of the process x_0 to be equal to $m(0)$. We then defined a small time difference Δt, and iteratively sample $X_{i\Delta t}$ for $i = 1..T/\Delta t$ such as :

$$X_{t+\Delta t} = X_t - \theta(X_t - m(t))\Delta t + \Sigma dW_t$$

where $dW_t \sim \mathcal{N}(0, \Delta t I)$. This results in a time series of length $T/\Delta t$. As we will see later, especially for the training phase of the forecasting model, we can produce downsampled and irregular versions of this time series by dropping some observations.

In order to generate m such that it is continuous and periodic, we randomly generated a neural network. This NN takes as input $\cos(2\pi t/\mathcal{T})$ and $\sin(2\pi t/\mathcal{T})$ in order to enforce the $\mathcal{T}$-periodicity. All the coordinates of m were generated without enforcing particular dependencies. One could enforce similarities in the different coordinates of the multivariate time series

by using d_X perturbed copies of the same neural network for each coordinate m_j, $j \in \{1..d_X\}$.

An example of a generated data sample can be found in Figure 5.1.

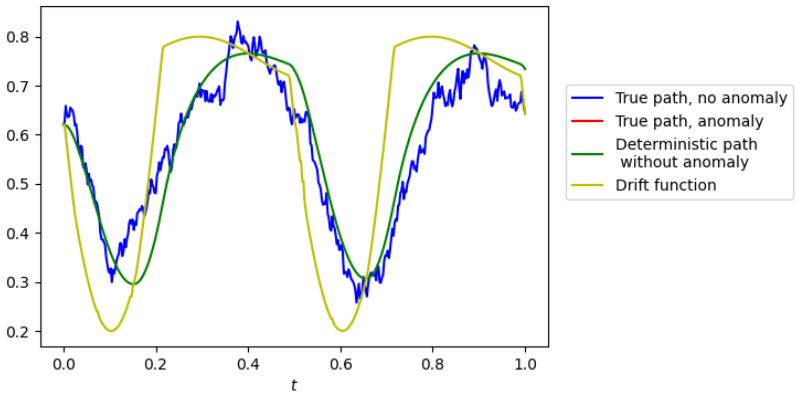
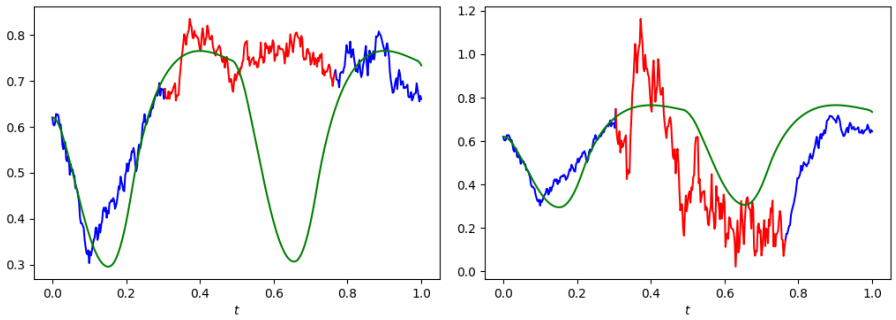


Figure 5.1: Sample of the Orstein-Uhlenbeck based process, with $S = 400$ steps and $P = 2$ periods. The final data path is in blue, the deterministic path (i.e without the diffusion term) in green, and the periodic m function in yellow

5.1.2 Anomaly Ingestion

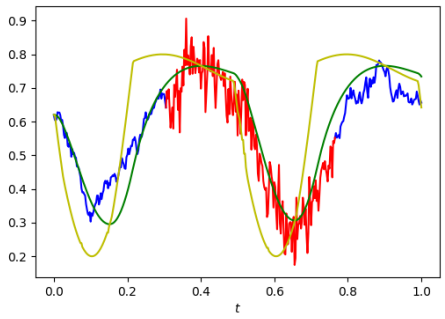
In order to inject the anomalies introduced in Section 2.3, we proceeded in a similar way. For the time range of anomalies, one simply changes the parameters of the drift or the diffusion, except for spike and noise anomalies. For noise anomalies, the noise is added on top of the original path during the anomaly time range, while for spikes it is the same except that the anomalies are punctual. Each timestamp during an anomaly, which is the time when some modification is done in the generation process, is labelled as an anomaly ($l = 1$). The rest of the timestamps are labelled as normal ($l = 0$). Note that for certain anomalies we might have a certain inertia in the anomalous behaviour. For instance, if we generate a deformation anomaly, it might be the case that just after the anomaly, the values of the path are far from the periodic m function, and could still "look" anomalous. Therefore, the labelling of anomalies might not appear perfect in those cases. The reader can refer to Figure 5.2 for examples of paths with anomalies.

The generation of the anomaly time windows' start and duration, is random and follows a uniform law. Furthermore, in a multivariate setting, the implementation allows to have either anomaly windows that coincide, or that are independent. However, we injected at most one anomaly window per

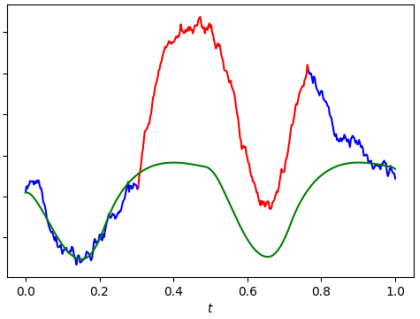


(a) Cutoff

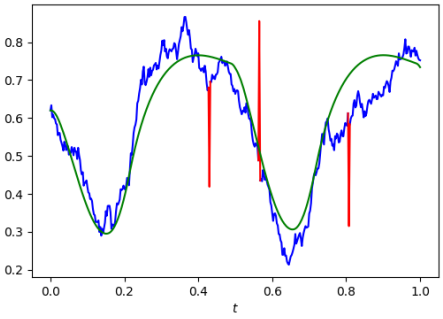
(b) Diffusion



(c) Noise



(d) Scale



(e) Spike

Figure 5.2: Sample of the Orstein-Uhlenbeck based process, with $S = 400$ steps and $P = 2$ periods. The final data path is in blue (no anomaly) and red (anomaly). The deterministic path without anomaly is in green.

path. For spikes this is different as the duration of anomalies is 1 timestamp. Therefore, each timestamp gets spiky with a certain probability and independent from the others.

5.2 Forecasting

In this section, we detail the different experiments performed for training the forecasting module. We first started with an ablation study on the different improvements added on top of the original PD-NJODE framework, detailed in Section 4.1. All these experiments where performed using the synthetic data. Then, we briefly show some further experiments on the synthetic data with different setups. Finally, we explain some results on the true KPI data.

5.2.1 Ablation study

We conducted an ablation study for the different modifications and improvements of the forecasting module explained in the last chapter. The ablation study was performed on the following base setup.

Data details

We generated a dataset of uni-dimensional data (without anomalies) generated for $P = 2$ periods and $S = 400$ timestamps (except for one case specified later), with parameters $\theta = 15$, $\sigma = 0.3$ (refer to Equation (2.1)). That the number of periods P intuitively corresponds to the duration in days (or weeks depending on the periodicity we consider) of the each path. We used $N = 80K$ paths for training in most cases, but also restricted ourselves to smaller training sizes in some cases of the ablation study ($N = 1000$ and $N = 200$). The periodic function m was generated by a randomly initialised NN with two layers of size 16 and ReLU activation. The function takes as input $\cos(2\pi t/\mathcal{T})$ and $\sin(2\pi t/\mathcal{T})$ and outputs a value in $\mathbb{R}$. The data was rescaled linearly to approximately fit the desired range (between 0 and 1).

Training Details

The models were trained over 50 epochs when the training size N was 80K. When we used a smaller dataset size, the number of epochs was increased in order to have the same amount of forward passes; 4K epochs for $N = 1000$ paths and 20K epochs for $N = 1000$ paths. We used a learning rate of 0.001 and a batch size of 200. Regarding the NN architectures of PD-NJODE, we had 2 layers with bias for each of the three networks (jump, ODE and readout), with the configuration as in Table 5.1.

During training, we use a dropout rate of 0.1 for each NN and layer.

Evaluation Metrics

For evaluating the predictions of the forecasting model, we compared them with the ground truth of the conditional moments of the described process, using the results stated in Subsection 2.2.1. Concretely, we computed the

	layer 1 (neurons, activation)	layer 2 (neurons, activation)
Jump NN	200,tanh	200,tanh
ODE NN	300,tanh	300,ReLU
Readout NN	200,tanh	200,tanh

Table 5.1: NN configuration

mean square error (MSE) between the predictions and the truth at each timestamp before and after jumps. We simply used the Euler method whenever the computation of the ground truth involved an integral. For the second moment, the MSE was computed using the conditional standard deviation, by applying the square root to the conditional variance V_t. In case the prediction of the conditional variance was negative, it was simply replaced by zero. The model was evaluated during training after each epoch. In case the dataset was smaller, the evaluation was done each 80 and 400 epochs for $N = 1000$ and $N = 200$ paths respectively. In each case, we performed the evaluation on the same $N_{test} = 200$ paths. We then reported the best evaluation (in MSE) over the whole training, together with the corresponding standard deviation (of the MSE) with the notation : $mse \pm stdmse/\sqrt{N_{test}}$. The evaluation is reported for both the conditional expectation and variance.

Reference Model

First, we introduce the reference model that includes the most relevant improvements that we made for the forecasting framework.

We start by detailing the training loss function for our base model. Let X^i, $i \in \{1..B\}$ be the sample paths in a batch observed at times $(t_1^{(i)}, .., t_{n^{(i)}}^{(i)})$. Note that in this version, we used the additional prediction of V_t.

The loss function is :

$$\frac{1}{N} \sum_{i=1}^{B} \frac{1}{n^{(i)}} \sum_{k=1}^{n^{(i)}} L_k^{(i)} \tag{5.1}$$

$$
\begin{aligned}
L_k^i = \lambda_1 \left| X^{(i)}_{t_k^{(i)}} - E^{(i)}_{t_k^{(i)}} \right|_2^2 + \lambda_2 \left| X^{(i)}_{t_k^{(i)}} - E^{(i)}_{t_k^{(i)}-} \right|_2^2 \\
+ \lambda_3 \left| (X^{(i)}_{t_k^{(i)}})^2 - S^{(i)}_{t_k^{(i)}} \right|_2^2 + \lambda_4 \left| (X^{(i)}_{t_k^{(i)}})^2 - S^{(i)}_{t_k^{(i)}-} \right|_2^2 \\
+ \lambda_5 \left| V^{(i)}_{t_k^{(i)}-} - (S^{(i)}_{t_k^{(i)}-} - (E^{(i)}_{t_k^{(i)}-})^2) \right|_2^2 \\
+ \lambda_6 \left| V^{(i)}_{t_k^{(i)}-} - (X^{(i)}_{t_k^{(i)}} - E^{(i)}_{t_k^{(i)}-})^2 \right|_2^2 + \lambda_7 \left| V^{(i)}_{t_k^{(i)}} \right|_2^2
\end{aligned}
\tag{5.2}
$$

where we used the same notation as in Section 4.1 for E, S, V. We empirically tested several hyperparamter combinations and decided for the following combination : $\lambda_1 = \lambda_2 = \lambda_7 = 1$, $\lambda_3 = \lambda_4 = 0.1$. For λ_5 and λ_6, we put as an initial value 0, and grow them linearly throughout epochs to 0.1 and 0.05 respectively. In the two last lines of the loss function, E_{t_k-} is detached for the computation of the gradients.

Each time step is selected as an observation independently and with probability $p = 0.1$. This means that in average, each path will be processed using $n^{(i)} \approx Sp = 40$ irregular observation times. The times of each path are scaled between 0 and 1, i.e. $t_i = i/S$ for $i = 0..S$, before being processed by the model.

Furthermore, we include some modifications from the last chapter, namely 1) on-the-fly random observation selection 2) NN input scaling and 3) periodic feature input. For each of the following ablation studies we will always compare this model with this exact configuration with a similar model that has only one modification. Each modification will be mentioned explicitly.

Qualitative results of the base model are shown in Figure 5.3. The model is optimal whenever the orange line matches the green dashed line (for the conditional expectation), and the orange and green zones overlap (conditional standard deviation). The performance of the jump NN (and the readout NN) is reflected at jumps : the conditional expectation should jump to the the new observations and the conditional standard deviation should jump to zero. What happens in between observations reflects the performance of the ODE NN.

1. **Adding the prediction of the Variance**

   Here, we stress the advantage of directly predicting the variance (V_t).

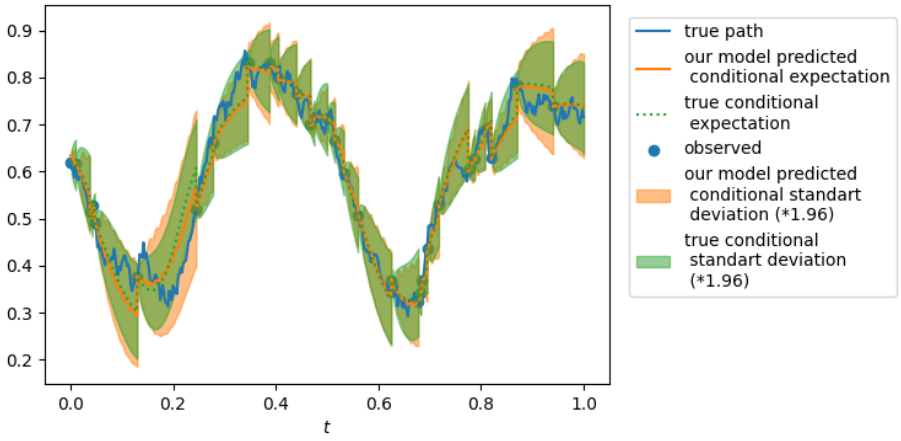


Figure 5.3: Example of predictions for the conditional expectation and standard deviation on the base model. The true path (blue) is only observed at certain points (blue dots). The predictions and the ground truth are displayed in orange and green respectively.

Alternatively, we use the loss function

$$\begin{aligned} L_k = \lambda_1 \left|X_{t_k} - E_{t_k}\right|_2^2 + \lambda_2 \left|X_{t_k} - E_{t_{k-}}\right|_2^2 \\ + \lambda_3 \left|(X_{t_k})^2 - S_{t_k}\right|_2^2 + \lambda_4 \left|(X_{t_k})^2 - E_{t_{k-}}\right|_2^2 \\ + \lambda_5' \left|(X_{t_k} - E_{t_{k-}})^2 - (S_{t_{k-}} - (E_{t_{k-}})^2)\right|_2^2 \end{aligned}$$

where λ_5' grows linearly throughout epochs from 0 to 0.1. In this setting, the predicted conditional variance is obtained by computing $S_{t_k} - E_{t_k}^2$. Similarly as before, $E_{t_{k-}}$ is detached in the last line.

model	cond. exp. MSE ($\times 10^{-4}$)	cond. std MSE ($\times 10^{-4}$)
ref. model	$\mathbf{1.058 \pm 0.2679}$	$\mathbf{0.4902 \pm 0.09143}$
without V	1.205 ± 0.3920	11.90 ± 0.5448

Table 5.2: Results comparing the reference model and the reference model without the prediction of V. See 5.2.1 for details about the reported metrics.

The results displayed in Table 5.2 show that there is a clear improvement in the predicted conditional standard deviation when using V. The reader can compare Figure 5.4 with Figure 5.3 to get an idea of the qualitative improvement.

2. **Enforcing the positiveness of the Variance**

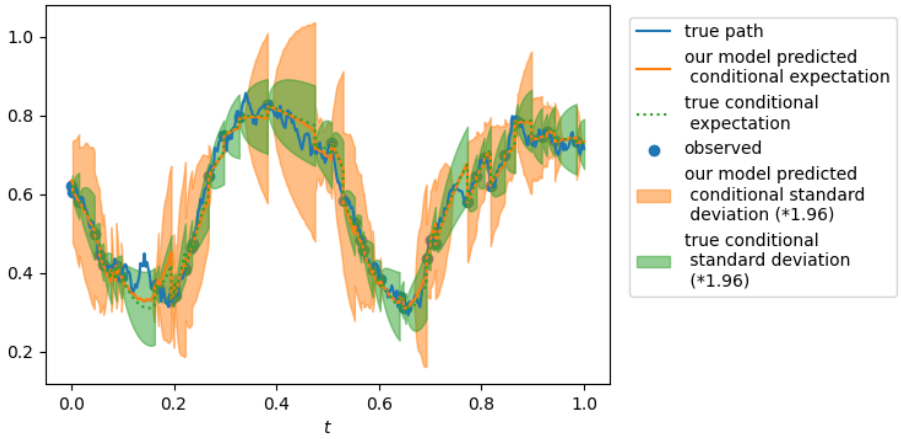


Figure 5.4: Example of forecasting predictions on the alternative model, without the direct prediction of the variance.

Here, we compare the performance of the base loss, with a similar loss which consists in adding the following term :

$$\lambda_8 \text{ReLU}(-V_{t_k-})^2$$

with $\lambda_8 = 1$. This term aims to enforce that the predicted variance is positive, by penalising negative predictions.

model	cond. exp. MSE ($\times 10^{-4}$)	cond. std MSE ($\times 10^{-4}$)
ref. model	$\mathbf{1.058 \pm 0.2679}$	0.4902 ± 0.09143
with	1.092 ± 0.2859	$\mathbf{0.3943 \pm 0.08535}$

Table 5.3: Results comparing the reference model and the reference model with the additional term enforcing the positiveness of V. See 5.2.1 for details about the reported metrics.

The evaluation results, displayed in Table 5.3, suggest that there is a small improvement of the variance prediction, which is not significant. We also observed that adding these terms has the tendency to bias the variance prediction at early training stages, as shown in Figure 5.5. Therefore, we did not include it in the reference model. One could consider adding this loss term to the base model in future work, with a smaller weight $\lambda_8 < 1$.

3. **Varying the weights of the loss function**

   We stress the advantage of varying the weights of the loss function (λ_5 and λ_6). Therefore, we compared the resulting evaluation of the

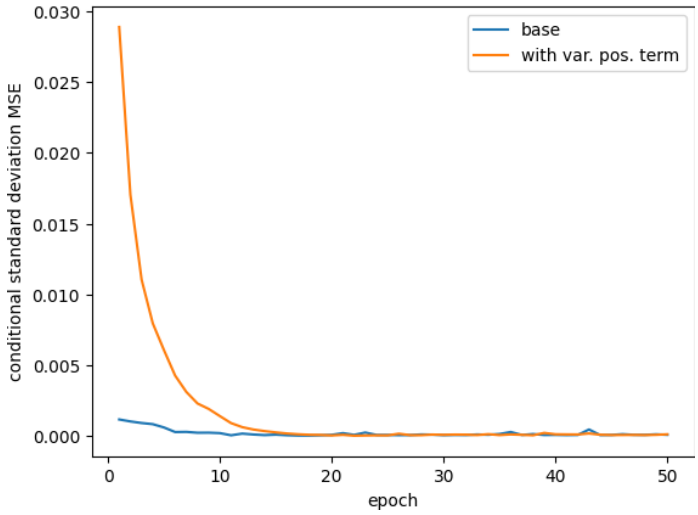


Figure 5.5: Evaluation of the conditional standard deviation for the two versions of the loss function, namely with and without the term enforcing variance's positiveness.

model by having fixed values for λ_5 and λ_6, which we set to 0.1 and 0.05 respectively. The underlying idea is to calibrate the prediction of V_t with E_t and S_t later in the training, when the model has had time to learn E_t and S_t accurately. We believe that if we put too much weight on the terms 5 and 6 at the beginning of the training, the model might learn sub-optimal predictions of V_{t_k} based on sub-optimal E_{t_k} and S_{t_k} .

We report the results in Table 5.4. We observe that this improvement leads to better predictions for both the conditional expectation and standard deviation. Furthermore, it reduces the std of the MSE, especially for the conditional expectation, which suggest a better stability of the model. As an explanation, one could argue the following. Since the prediction of the conditional variance highly depends on the prediction of the conditional expectation, learning the former in early stages and then the latter in later stages is arguably a good lead. If one learns both prediction jointly during the whole training, the model may have the tendency to find local optimas that are sub-optimal, since it will try to fit the conditional variance without a calibrated conditional expectation.

4. **Varying dataset sizes**
   As stated before, we compare the performance of models trained on different dataset sizes N, and show that we can still obtain comparable results by reducing the number of training paths by an order of 10^2.

   The results are reported in Table 5.5. They show that the model man-

model	cond. exp. MSE ($\times 10^{-4}$)	cond. std MSE ($\times 10^{-4}$)
ref. model	$\mathbf{1.058 \pm 0.2679}$	$\mathbf{0.4902 \pm 0.09143}$
fixed weights	1.325 ± 0.5160	0.7484 ± 0.1016

Table 5.4: Results comparing the reference model and the reference model with fixed weights for all terms of the loss function. See 5.2.1 for details about the reported metrics.

ages to learn on much less data with comparable accuracy. We will see in the next experiment that this is mainly due to our data augmentation trick. The results are encouraging because of the limitation of data in many real world settings, especially in our case when we train a specific model on a single system. Note that N is the number of training paths before the data augmentation trick. However, each of those paths is ingested at each iteration with different randomly selected observation times (thanks to the on-the-fly random observation selection explained before), so that the resulting training dataset is intuitively N times the number of epochs.

dataset size N	cond. exp. MSE ($\times 10^{-4}$)	cond. std MSE ($\times 10^{-4}$)
80K	1.058 ± 0.2679	$\mathbf{0.4902 \pm 0.09143}$
1000	$\mathbf{1.044 \pm 0.2714}$	0.6123 ± 0.09461
200	1.220 ± 0.3002	0.6075 ± 0.09242

Table 5.5: Results comparing the reference model trained on different dataset sizes. See 5.2.1 for details about the reported metrics.

5. **On-the-fly random observation selection**

   Here, we aim at providing a justification for selecting random observations during training. The idea is to exploit the entire data paths, while still training on irregularly observed paths for previously mentioned reasons. Therefore, we compare the performance of a model trained on this setting, with a model that is trained on paths for which the observations are randomly sampled before the training starts, once and for all. We chose to compare the results when training on small dataset sizes i.e $N = 200$ paths. Indeed, the advantage of this "data augmentation technique" should be most visible for limited training data.

   The evaluation reported in Table 5.6 clearly shows the benefits of this technique when the dataset is small, as paths are observed at different observation times.

6. **Training with artificial points**

selection method	cond. exp. MSE ($\times 10^{-4}$)	cond. std MSE ($\times 10^{-4}$)
ref. model	$\mathbf{1.220 \pm 0.3002}$	$\mathbf{0.6075 \pm 0.09242}$
fixed obs. times	2.429 ± 0.5053	2.7484 ± 0.3038

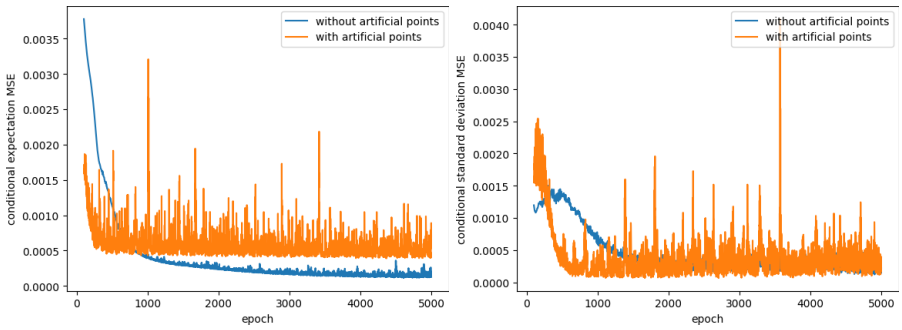
Table 5.6: Results comparing the reference model and the reference model trained on paths with fixed observation times. See 5.2.1 for details about the reported metrics.

In this experiment, we aim at understanding whether additionally training the jump and readout NN with artificial points could help to calibrate the model at observations. We expected this improvement to be especially useful when the training dataset is small. The model should more easily learn to jump to the correct values since it gets fewer examples.

Therefore, we compare the training with and without those additional points on a training set of 200 paths. We performed additional epochs for the artificial points. At each of those epochs, we generate a batch of random triplets (H, π, X), with the same dimensions as the hidden state, the truncated signature and the input respectively. X is transformed as the input, by including X^2 for the prediction of the second moment and the variance. Then we compute the mean square loss between $g_{\theta_3} \circ \rho_{\theta_2}(H, \pi, (X, X^2)) = (E, S, V)$ and $(X, X^2, 0)$. We perform 10K batch iterations with those artificial points, before starting the actual training. Then, we perform one batch iteration in between each batch iteration with the training paths.

The results reported in Table 5.7 show that the model in fact performs poorer when this technique is used, both for the expectation and the standard deviation. By observing Figure 5.6, we can see that while the MSE is lower at early stages of the training, it quickly stops to improve when using this technique, while the metrics get better without it. Furthermore, the evaluation metric oscillates much more, which suggest that the learning process is less stable. In summary, with this technique, the model initially performed well to jump at the correct values, but was worse for prediction in between jumps in later stages of the training.

Although we did not achieve good results with this technique, it might be possible to improve it by further tuning some of its hyperparameters. Pre-training with artificial points might harm the training as model weights could get stuck in a local minima. In addition, the learning rate of the artificial training stage might be reduced in order to alleviate the oscillating behaviour. We leave these leads to future work.



(a) MSE, conditional expectation (b) MSE, conditional std

Figure 5.6: Evolution of the evaluation MSE throughout epochs on models trained with and without the artificial points.

model	cond. exp. MSE ($\times 10^{-4}$)	cond. std MSE ($\times 10^{-4}$)
ref. model	$\mathbf{1.044 \pm 0.2714}$	$\mathbf{0.6123 \pm 0.09461}$
with artificial points	3.902 ± 1.235	0.9347 ± 0.1064

Table 5.7: Results comparing the reference model and the reference model trained additionally with artificial points. See 5.2.1 for details about the reported metrics.

7. **Periodic time features**

Here, we evaluate the benefits of feeding the ODE-NN with periodic time features, when the time series data has some periodicity. Specifically, we compared the performance of two models, where the ODE NN takes the following inputs.

Without the time periodic features:

$$f_{\theta_1}(H_{t-}, \pi_n(\tilde{X}^{\leq \tau(t)} - X_0), X_{\tau(t)}, \tau(t), s(t-\tau(t)), t). \tag{5.3}$$

With the time periodic features:

$$\begin{aligned} f_{\theta_1}(H_{t-}, \pi_n(\tilde{X}^{\leq \tau(t)} - X_0), X_{\tau(t)}, \tau(t), s(t-\tau(t)), \\ \cos(2\pi t/\mathcal{T}), \sin(2\pi t/\mathcal{T})). \end{aligned} \tag{5.4}$$

Here, the experiments were performed on time series with more steps ($S = 1000$ instead of $S = 400$) and more periods ($P = 5$ instead of $P = 2$). We trained our model on $N = 40K$ paths and evaluated it on 200 paths, as before.

The results are reported in Table 5.8. There is a significant improvement in the prediction of the conditional expectation, while the conditional variance is slightly worse. We believe that this feature could play

the role of an inductive bias in the data. This means that the model would solely rely on where the current observation lies with respect to the period, instead of where in the whole time interval.

model	cond. exp. MSE ($\times 10^{-4}$)	cond. std MSE ($\times 10^{-4}$)
ref. model	$\mathbf{0.5067 \pm 0.1807}$	0.5941 ± 0.07092
without periodic feat.	0.9766 ± 0.3016	$\mathbf{0.5289 \pm 0.09963}$

Table 5.8: Results comparing the reference model and the reference model without the periodic time features. See 5.2.1 for details about the reported metrics.

8. **Modification of the loss function**

Finally, we analyse the modification of the loss function, that consists in separating the two base terms of the loss function described in Krach et al. (2022). Specifically, we compared the old version

$$\left(\left| (X_{t_k}, X_{t_k}^2) - (E_{t_k}, S_{t_k}) \right|_2 + \left| (E_{t_k}, S_{t_k}) - (E_{t_k-}, S_{t_k-}) \right|_2 \right)^2,$$

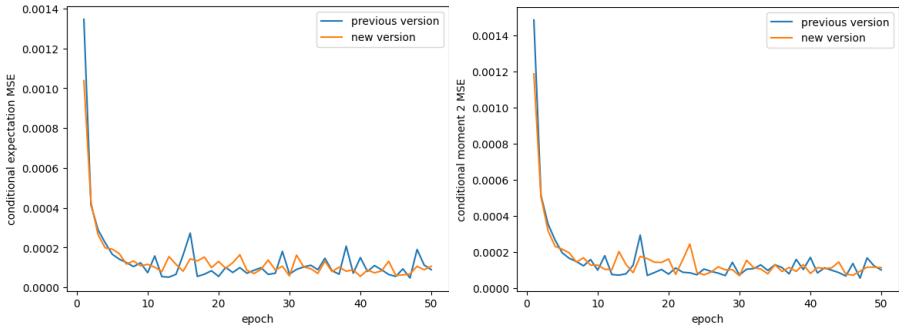
with the new one

$$\left| (X_{t_k}, X_{t_k}^2) - (E_{t_k}, S_{t_k}) \right|_2^2 + \left| (E_{t_k}, S_{t_k}) - (E_{t_k-}, S_{t_k-}) \right|_2^2.$$

We evaluated the predictions of the conditional expectation and the conditional 2nd moment (instead of the conditional standard deviation), as we did not include any of the terms to calibrate the conditional variance. The results in Table 5.9 show a slight performance alteration in the new version. However, we did not identify a significant gap, as shown in Figure 5.7.

model	cond. exp. MSE ($\times 10^{-4}$)	cond. mom. 2 MSE ($\times 10^{-4}$)
previous	$\mathbf{0.4739 \pm 0.1454}$	$\mathbf{0.5457 \pm 0.1420}$
new	0.5553 ± 0.1457	0.6721 ± 0.1673

Table 5.9: Results comparing the previous and the new version of the loss function, as detailed above. We evaluate the predictions of the conditional expectation and 2nd moment (instead of the variance). See 5.2.1 for details about the reported metrics.



(a) MSE, conditional expectation (b) MSE, conditional moment 2

Figure 5.7: Evolution of the evaluation MSE throughout epochs for models with two version of the loss : square outside (previous version), and square inside (new version).

5.2.2 Additional experiments

Forecasting 2-dimensional process

We tested our reference model (detailed in Subsection 5.2.1) on a two dimensional synthetic process. The purpose is to verify that the model also performs properly when the coordinates of the process have dependencies. The data was generated for 2 periods and $S = 400$ steps, with the following parameters (refer to Section (2.1))

$$\theta = \begin{pmatrix} 10 & 5 \\ 0 & 15 \end{pmatrix}, \qquad \sigma = \begin{pmatrix} 0.3 & 0.15 \\ 0.15 & 0.3 \end{pmatrix}. \tag{5.5}$$

The non-diagonal term of the drift speed θ introduces dependency between the coordinates, while the non-diagonal diffusion factor σ injects shared noise. The model is trained using the same training setup as in the reference model, on $N = 8K$ paths for 100 epochs.

A qualitative example is given in Figure 5.8. The plot shows the ability of the model to learn the conditional expectation. The predictions of the conditional variance are not bad, but more unstable for the first coordinate. This is probably due to the hyper-parameters of the model, which are the same as in the one-dimensional case. Training for more epochs with bigger networks might resolve the issue.

5.2.3 Experiment on Cloud KPI data

We conduct some qualitative experiments on the KPI cloud data in order to check whether the forecasting method is able to learn the conditional distribution.

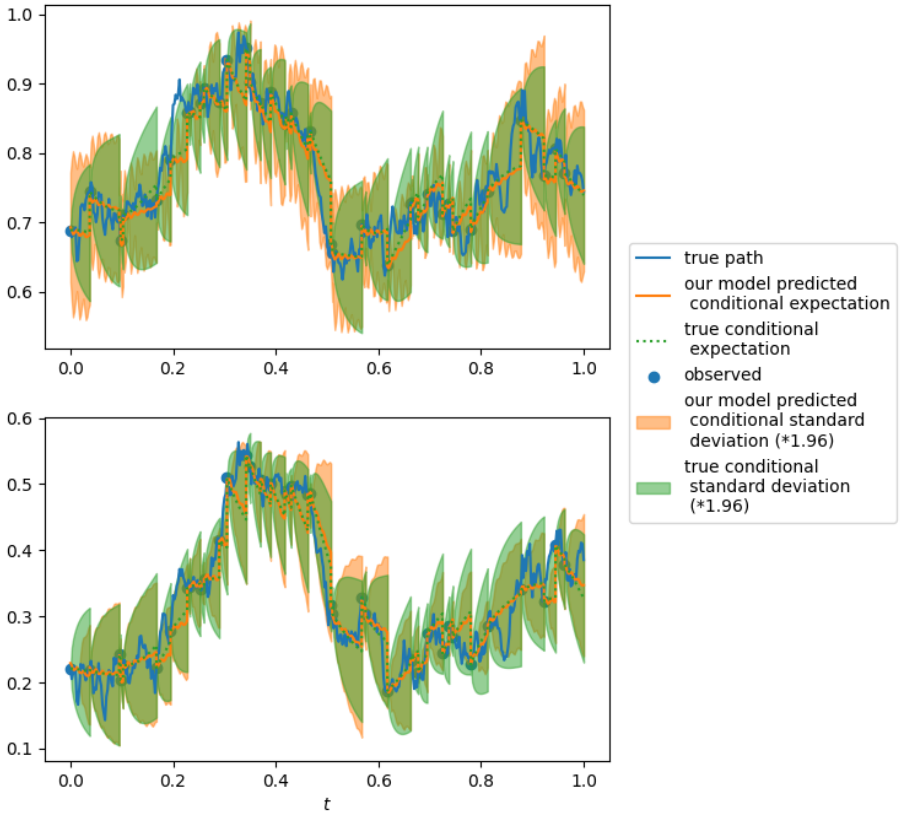


Figure 5.8: Example of predictions on the 2-dimensional synthetic process.

Data details

The data was gathered from a single system and on a period of 401 days. The multivariate time series consists of the concatenation of three KPIs : read I/O rate, read response time and read transfer size. We select those KPIs as they probably have some cross-dependencies, which could be captured in the conditioning term $\mathcal{A}_t$ as explained previously. Each path consists of 1 day of data, i.e $S = 288$ steps. Consequently, we have one period per path $P = 1$. We use 320 paths for training and the remaining 81 for evaluation and visual inspection. We do not provide with any visualisation because of data privacy.

Model

We use the same reference model as described in the previous section with the same hyper-parameters.

We train two separate models by using differently pre-processed data. In the former model, the data is linearly scaled while in the second case, we apply the transformation

$$\frac{2\left(\log(x+0.01)-\log(0.01)/2\right)}{-\log(0.01)},$$

where x are the KPI values. The motivation for this latter transformation is the following. The distribution of KPI values is often right skewed. Therefore the true conditional distribution often is so as well. However, when applying the Gaussian distribution inference, we get a symmetric distribution, which often has a significant area under the curve for negative values. This is unrealistic as KPI values lie in $[0,1]$. As a result, we assign low anomaly scores to unexpectedly small KPI observations, which therefore are less likely to be classified as anomalies. The purpose of this transformation is therefore to get close to a symmetric distribution, where the Gaussian distribution inference with the two-sided p-value based scoring is better justified. Furthermore, note that the resulting values are still bounded, as they lie in $[-1,1]$.

Models are trained for 10K epochs with a batch size of 160.

Qualitative analysis

As before, we plot the forecasting results with random observation times in order to show the ability of the model to learn the predictions for multiple forecasting horizons. First, we notice that both the predictions for the conditional variance and the conditional expectation jump to the appropriate values at new observations. Then, we further observe that the conditional expectation follows properly the trend of the time series with both models. The standard deviation also has a plausible amplitude, and increases consistently after observations.

The two models show differences in the confidence bound displayed by the conditional standard deviation (i.e. $\pm 1.96 \times std$). For the untransformed model, the lower confidence bound is much lower than it should (i.e. < 0), as expected, while the upper confidence bound often lies below observations. For the transformed data, this phenomena is alleviated, and the confidence bounds seem to better fit the range of observations. Of course, the anomaly detection method does not necessarily rely on these confidence bounds, but rather on the inferred distribution, and the scoring method. However, they

give a good idea of the scores if we infer a Gaussian, and use the two-sided p-value scoring.

Note that one might still choose not to transform the data, but rather use a different distribution class for inference, such as the log-normal distribution which has a support on $(0, +\infty)$. In this case, the two-sided p-value based scoring would assign high anomaly scores to values close to zero.

5.3 Anomaly Detection on Synthetic Data

In this section, we show some anomaly detection experiments on synthetic data.

Data details

We generate four datasets of one-dimensional time series, each with one type of anomaly: cutoff, noise, diffusion, spike (see Section 2.3). Each dataset has 1000 paths. We essentially use the same setting as in the previous section, i.e. time series are generated by the synthetic process (see Equation (2.1)) for $P = 2$ periods with $S = 400$ steps and with parameters $\theta = 15$, $\sigma = 0.3$. We also generate one dataset of clean data with the same parameters, on which we train the forecasting model. Note that we always use the same periodic driving function. For cutoff, noise and diffusion anomalies, we select a random range $r \sim \mathcal{U}([0.1, 0.6])$, which is the proportion of the time series which is anomalous. The starting time of the anomaly is set uniformly at random over the entire time interval. For the spike dataset, we simply sample each timestamp independently with probability 0.005 that it is an anomaly. The anomaly labels take values in $\{0, 1\}$, where 1 corresponds to an anomaly. The concrete anomalous behaviour is the following.

- **Cutoff:** For the cutoff anomalies, we set the periodic driving function m to a constant for the duration of the anomaly. This constant is simply the value of the original m at the starting time of the anomaly.
- **Noise:** For the noise anomalies, we inject Gaussian white noise with standard deviation 0.05, for the duration of the anomaly.
- **Diffusion**: The diffusion factor σ is scaled by a factor of 5 for the duration of the anomaly.
- **Spike:** For each random spike, we add a randomly sampled value $s \sim \mathcal{U}([-0.5, -0.2] \cup [0.2, 0.5])$ to the original path (without anomaly).

Pipeline parameters

Here we explain the setting of the pipeline that we use in this experiment. It relies on a forecasting model which was trained on data without anomalies.

The specifics of the forecasting model are the same as in the reference model (Subsection 5.2.1). We infer a Gaussian distribution from the predicted expectation and variance for the anomaly detection module, and chose a two sided p-value based scoring. Finally, the design of the score aggregation module is

$$\text{score}_{t_i} = \sum_{l=-L}^{L} b_l \sum_{k \in \mathcal{K}} a_k v_{t_{i+l-k}, t_{i+l}}, \tag{5.6}$$

where $L = 5$ steps (that is we smooth the scores with the 10 neighbouring scores), and forecasting horizons $k \in \mathcal{K} = \{1, 3, 5, 7, 10, 12, 15, 18, 20\}$ (in steps). We apply the sigmoid function to get positive weights

$$a_l = \sigma(a'_l), \tag{5.7}$$
$$b_l = \sigma(b'_l). \tag{5.8}$$

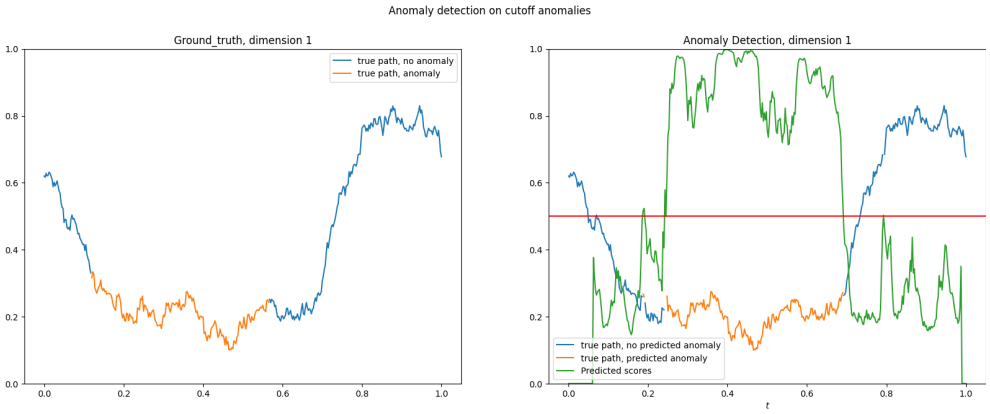
Finally we re-apply the sigmoid function to the final aggregated scores to have values in $[0, 1]$.

The weights of the score aggregation module are learned on 80% of the data, using Adam optimiser with learning rate 0.01. In practice, we optimise the weights a'_l and b'_l, before applying the activation. We use the Cross Entropy loss between the final predicted scores and the anomaly labels, together with L2 weight regularisation.

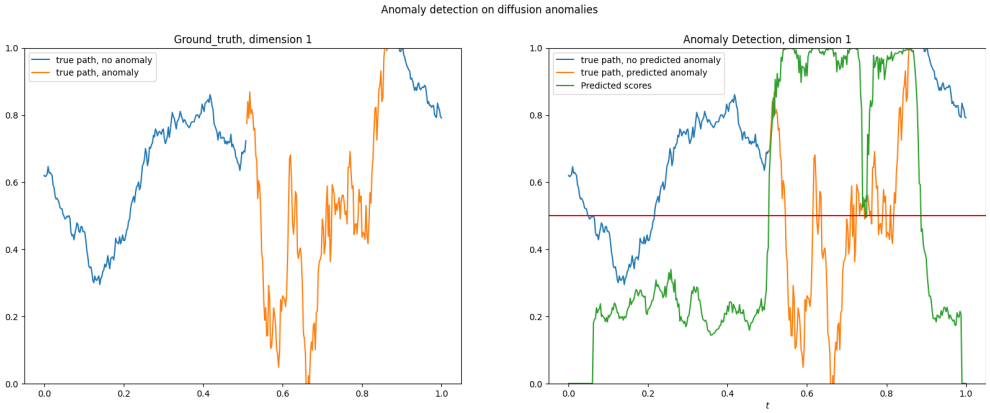
During inference, we set a threshold of 0.5 on the resulting scores. We tested the pipeline on the remaining 20% of the data.

Results

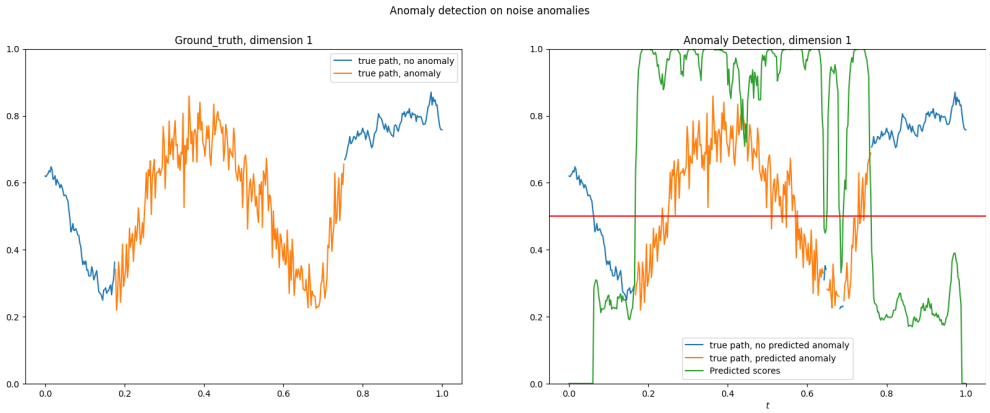
We provide qualitative examples (Figure 5.9) of anomaly detection, by showing side-by-side the synthetic ground truth and the predictions with the corresponding scores. We also display the different score aggregation weights as heatmaps (see Figure 5.10). The idea is to show examples of aggregation schemes for different types of anomalies. As we can see in Figure 5.10, different weighting schemes are suited for different anomaly types. When the targeted anomaly is a spike, one relies more on the current observations, rather than neighbouring (past or future) observations. For noise and diffusion anomalies, looking at neighbouring observations seem to be useful, but one would solely need a forecasting horizon of one step. For cutoff anomalies, one should choose a bit longer forecasting horizon.



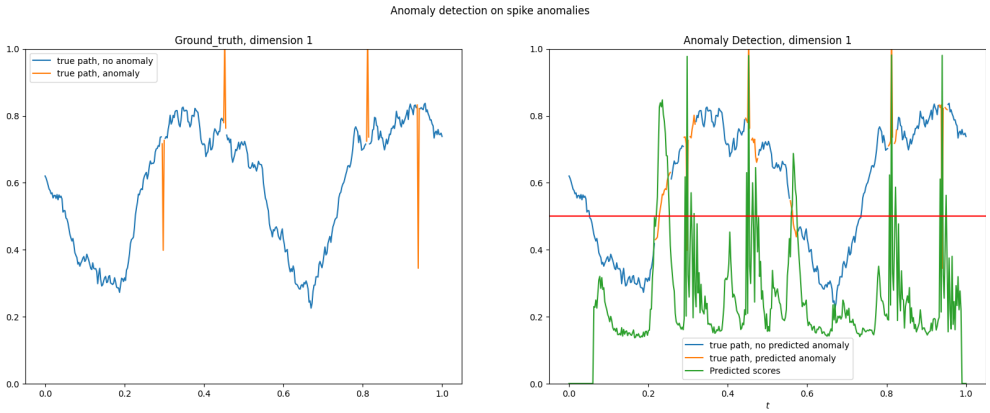
(a) Cutoff



(b) Diffusion

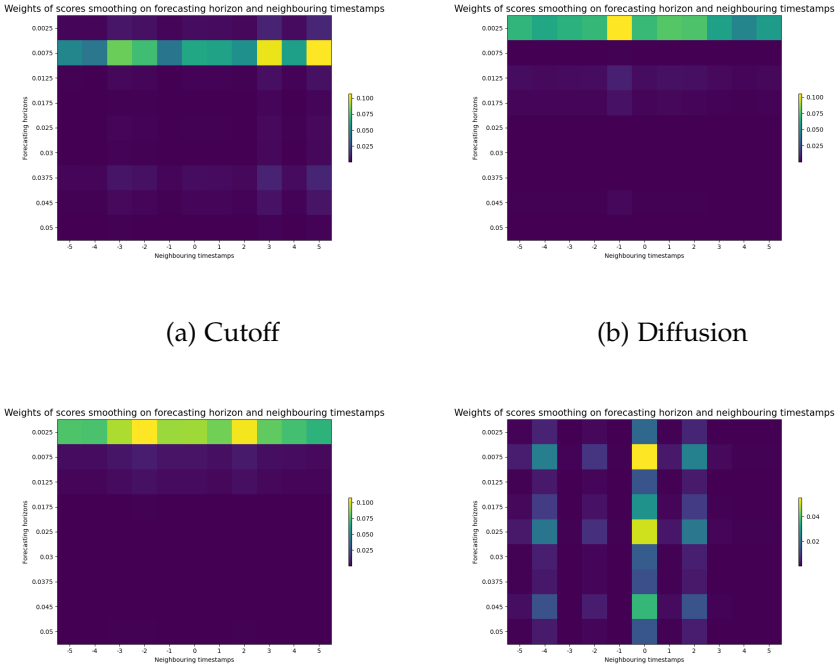


(c) Noise



(d) Spike

Figure 5.9: Example plots of anomaly detection. The ground truth is on the left, the prediction on the right. The non-anomalous parts are in blue, the anomalous ones in orange. For the prediction, scores are in green, and the red line shows the threshold level of scores.



(a) Cutoff

(b) Diffusion

(c) Noise

(d) Spike

Figure 5.10: Learned score aggregation weights for different anomaly types. The horizontal axis shows the influence of neighbouring scores (past observations on the left and future on the right), while the vertical axis shows the influence of different forecasting horizons (low forecasting horizons on top, and high horizons on the bottom). Note that the resulting numbers on the vertical axis correspond to $k\Delta t$ where Δt is the smallest time difference.

Chapter 6

Conclusion

This work tackles the problem of time series anomaly detection under the spectrum of probabilistic forecasting. We elaborated a new multivariate time series anomaly detection methodology which aims to extend and improve the forecasting based approach. After introducing the context of cloud systems and the state of the art on time series anomaly detection and probabilistic forecasting, approaches were discussed to model the predictive distribution of future time series observations. The recent PD-NJODE forecasting framework was incorporated in the anomaly detection pipeline. Then, the functioning of probabilistic forecasting and anomaly detection was demonstrated on synthetic data. We also showed how different configurations of the designed method can be applied to detect different types of anomalies.

Since the approach is novel there are multiple directions that could be explored. First of all, as anomaly detection is an unsupervised task, it is important to pay attention to the expected outcomes of the method at hand. First of all, one might put some effort into identifying the characteristics of anomalies that are targeted. We observed that, depending on the anomaly type, one should make different choices in the design of the pipeline. Synthetic data can be a useful tool for this purpose. Although we tried to incorporate elements of the true data in the synthetic data generation process, one could further tune it to make it even more similar to the true data. Furthermore, one could conduct synthetic experiments on more complex stochastic processes to test the performance of the model when we have some path dependencies, or time / space dependent volatility.

We invested time into making the prediction of the conditional moments consistent and observed that this was already challenging when solely dealing with the first two moments. We saw that predicting the centred second moment (i.e the variance), was a fruitful lead towards this purpose. We believe that training different models for the different moments could be promising. Specifically, one could first learn the conditional expectation

with a first model before learning the conditional variance with another model, that relies on the first one. One could further investigate how to generalise the method to further moments, while keeping the consistency of moments. Finally, we restricted ourselves to predicting the per-coordinate marginal moments, conditioned on previous observations from all coordinates. One could extend the method towards learning the joint conditional moments, in order to analyse the anomalousness of all coordinates jointly. Then, one could search for proper distribution classes that would fit the data at hand. This could be tackled by looking at the empirical distribution of observations, subtracted by the predicted conditional expectation.

Regarding the rest of the pipeline, further work could be conducted into finding proper score aggregation weights, depending on the type of anomaly. Since, the most expensive computation resides in the forecasting module, it is perfectly feasible to use the pipeline with different weights (as demonstrated). Furthermore, this could be used to infer the type of anomaly by looking at the highest observed scores for different filters. We claim that in our method the threshold is much less critical to set than in forecasting methods that often have to rely on dynamic threshold. It can stay fixed in our case as we already account for the expected discrepancy of observations from the expectation by computing the conditional variance. However, one could still fix it with more care by tuning false positive / negative rates.

We believe that the mentioned leads are worth exploring in order to design a customised, intepretable and theoretically justified multivariate time series anomaly detection method in future work.

Appendix A

KPI Causal Relations Learning

In this chapter, we introduce the topic of Causal Discovery in Time Series, in an attempt to identify causal relations between the KPIs monitoring cloud systems. We will first briefly introduce the topic and mention the fundamental ideas behind causal discovery. Then, we will emphasise on two causal discovery methods with which we attempted to identify causal relations between KPIs.

A.1 Causality background

A.2 Causality in Time Series

Defining causality in time series is a challenge in itself. Most of recent studies of this topic rely on the widespread probabilistic concept of Granger Causality, while other focus on other concepts such as Intervention Causality, or Structural Causality. Before formulating these concepts, we will outline the underlying hypothesis of Time series Causality.

There are two common properties Eichler (2013) for defining cause-effect relationships in time-series :

- Temporal precedence : a cause precedes its effect in time
- Physical Influence : manipulations of the cause change the effects

While the second property better aligns with the common understanding of causal relations, interventions are usually not possible and we have to deal with a fixed dataset of time series. Therefore, we will restrict ourselves to the hypothesis of temporal precedence.

A.2.1 Granger Causality

Granger Causality Granger (1969) is a probabilistic concept, formulating two principles :

- the past may cause the future, but the future cannot cause the past (temporal precedence)
- a cause time series entails some unique information determining the effect time series

Let t denote the present time and $X = (X_t)_{t\in\mathbb{Z}}, Y = (Y_t)_{t\in\mathbb{Z}}, Z = (Z_t)_{t\in\mathbb{Z}}$ be three time-indexed discrete stochastic processes on a filtered probability space $(\Omega, \mathcal{F}, \mathbb{F} = (\mathcal{F}_t)_{t\in\mathbb{Z}}, \mathbb{P})$. We also use the notation $X^{\leq t} = \{X_s, s \leq t\}$.

In this setting, X and Y are the variables of interest for which we aim study the causal relationship, while Z denotes all the other variables present in the universe.

Definition A.1 (Granger Causality) *X does not Granger causes Y with respect to (X, Y, Z) if for all $t \in \mathbb{Z}$,*

$$Y_{t+1} \perp\!\!\!\perp_{\mathbb{P}} X^{\leq t} | Z^{\leq t}, Y^{\leq t}$$

This means that Y_{t+1} and $X^{\leq t}$ are conditionally to $Z^{\leq t}$ and $Y^{\leq t}$ independent. Otherwise, X Granger causes Y with respect to (X, Y, Z).

where $\perp\!\!\!\perp_{\mathbb{P}}$ denotes the probabilistic independence.

In practice, we usually are interested in discovering the causal relations between N variables that evolve through time. Therefore, we instead have N stochastic processes $\{X^j\}_{j=1..N}$, one for each variable. Referring to the above definition, one takes each pair of stochastic processes to play the role of X and Y, while Z would be the concatenation of all the others.

Granger Causality is a reference definition, from which most causal discovery methods take inspiration. One of its most common application form involves autoregressive linear models. Basically, one assumes that their exists coeeficients $A_{jk,u}$, $j, k = 1..N$, $u \in \mathbb{N}$ such that :

$$X_{j,t} = \sum_{u=1}^{\infty} \sum_{k=1}^{N} A_{kj,u} X_{k,t-u} + \varepsilon_t^j$$

The principle is that if we can decide, using statistcal procedures such as hypothesis testing, that $A_{kj,u} = 0$ for each time lag $u \in \mathbb{N}$, then one deduces that variable k does not Granger cause variable j.

This use case example links to another formulation of Causality.

A.2.2 Structural Causality

Structural Causal models (SCM) (Pearl, 2009) is another popular formalisation of causality, where the effect is directly modelled as functions of its causes, and noise. In time series causality, the temporal precedence principle stated in the previous definition also holds. Using the previously introduced notation, we equip ourselves with a set of time-evolving variables $\{X_j\}_{j=1..N}$. In SCMs in general one also considers the time lags in the causal relations. More specifically, consider $X_{j,t}$ (that is X_j at time t) and assume it has a set of parents $\mathcal{P}(X_{j,t}) \subset \{X_{k,t'}\}_{k=1..N,t'<t}$ such that :

$$X_{j,t} = f_{j,t}(\mathcal{P}(X_{j,t}), N_{j,t})$$

where $N_{j,t}$ is a noise term.

For simplicity, SCMs often take the form of additive models, which are time invariant :

$$X_{j,t} = \sum_{X_{k,t'} \in \mathcal{P}(X_{j,t})} f_{kj,t-t'}(X_{k,t'}) + N_{j,t}$$

A.2.3 Causal Graphs

The objective behind causal discovery is usually to extract a causal graph, that is a directed graph which nodes, and edges correspond to variables and cause-effect relations respectively.

Some works focus on enforcing that the resulting causal graph is Directed Acyclic Graph (DAG) , that is a graph without cycles. This is because we commonly assume that a variable cannot be the cause of itself. This aspect however gets more complicated in Time Series as because of the temporal dimension. Indeed, it can be that the past of a variable causes its own future.

There are two notions of causal graphs for time series. In the Full Causal Graph, nodes correspond to variables at each time, i.e causal relations such as $X_{k,t'} \to X_{j,t}$. The Summary Graph is a compressed version of it where we look for $X_k \to X_j$ whenever there are times $t' < t$ such that $X_{k,t'} \to X_{j,t}$. In the following, we discuss methods that seek for summary graphs.

A.3 Time Series Causal Discovery Methods

A.3.1 Minimum Predictive Information Regularisation (MPIR)

MPIR Wu et al. (2020) is a method aiming at discovering causal directional relations from multivariate time series. The methods supports the use of deep learning models to discover nonlinear relations. It introduces a regularisation by feeding input with learnable corruption noise amplitude. The

objective of the regularisation is to learn how much a specific variable can be predicted in order to still yield comparable prediction accuracy. The idea is that as a mean to predict a specific variable, the other variables that bring some unique and relevant information will yield worse predictions if they are corrupted, while for irrelevant variables would give comparable results.

Description

This method consists in building one forecasting model per variable, which gets as an input the past of all the other variables. The learnable parameters comprise the ones of the forecasting model and the noise amplitude. The objective loss has two terms; the mean square prediction error of the corrupted input, and the mutual information term quantifying the similarity between the corrupted and the original input. The first term will tend to lower the noise amplitude, at least for relevant variables, while the second will optimise the mutual information of the corrupted and original input, and therefore have the tendency to increase the noise amplitude.

Let $\mathbf{x}$ be a time series data sample, and consider the current time t. $\mathbf{x}^{>t}$,$\mathbf{x}^{\leq t}$ denote the past and the future, i.e the input and the output of the forecasting model. In addition, let $\mathbf{x}_i$ be the $i$the coordinate corresponding to a variable of the system, where $i = 1..N$. The corrupted input will be noted $\tilde{\mathbf{x}}^{(\boldsymbol{\eta})}$ where $\boldsymbol{\eta} \in \mathbb{R}^N$ is the noise amplitude, and η_j its per variable coordinates. The objective loss is :

$$\mathbb{E}\left[\left(x_i^{>t} - f_\theta(\tilde{\mathbf{x}}^{\leq t,(\boldsymbol{\eta})})\right)^2\right] + \lambda \sum_{j=1}^{N} I(\tilde{\mathbf{x}}_j^{\leq t,(\eta_j)}; \mathbf{x}_j^{\leq t}) \tag{A.1}$$

where is I is the mutual information, f_θ the prediction model and $\lambda > 0$ a hyperparameter.

Let $\boldsymbol{\eta}^*$ be the learned noise amplitude. The mutual information term is used as a result to quantify the causal strength, i.e

$$W_{ji} = I(\tilde{\mathbf{x}}_j^{\leq t,(\eta_j^*)}; \mathbf{x}_j^{\leq t})$$

where W_{ji} is the causal strength between cause j and effect i.

In case one wants to establish a boundary between causal and non-causal relations, the author propose to use fake variables. Basically, one adds fake variables that are generated randomly and without causal relations. Then, one can statistically compare the resulting causal strength on fake and original variables to decide for the most significant causal relations. The mutual

information is defined as the Kullback Leibler divergence between the joint distribution and the product of marginal distributions (Cover and Thomas, 2001).

Discussion

This method is interesting from a theoretical point of view, and in the meantime allows to learn non linear relations using deep learning architecture. The authors use an MLP, but one could easily imagine to replace it with a CNN or RNN to better capture the temporal dependencies while reducing the number of parameters. Having one model for each target variable, which depends on all the other variables (which are causes candidates) has two advantages. First it allows to identify the variables that bring some unique information, i.e that is not present in other variables, which goes in the direction of the Granger causal definition. Secondly, it reduces the complexity to only N as opposed to the naive approach of having one model per cause-effect variable pair (i.e N^2 models). This aspect becomes important when dealing with an increasing number of KPIs. Furthermore, regularisation often has a natural benefit of stabilisation. Indeed adding the random corruption noise to the input can prevent the model from being stuck in a local minima.

A.3.2 Amortized Causal Discovery

Amortized Causal Discovery Löwe et al. (2020) is a causal discovery framework that is designed to work on time series samples having different underlying causal relations, but that share dynamics. In opposition to the last method, the idea is no longer to learn intangible causal relations, but that can depend on the context. It relies on an encoder-decoder like architecture, where the encoder infers the causal relations.

Method

This method relies on the key assumption that there is a fixed function describing the dynamics of any sample time series $\mathbf{x}$, given past observations and an underlying causal graph. This means that the distribution of future observations of a given variable will only depend on the past observations of neighbouring variables in the causal graph (and the variable itself). This is illustrated in the following equation :

$$\mathbf{x}^{>t} = g(\mathbf{x}^{\leq t}, \mathcal{G}_{\mathbf{x}}) + \boldsymbol{\varepsilon}^{>t}$$

where $\mathcal{G}_{\mathbf{x}}$ is the underlying causal graph of $\mathbf{x}$, and g is the function describing the dynamics. The latter can be assimilated to a forecasting model mentioned earlier.

The proposed model has two differentiable networks : one encoder, and one decoder. One one hand, the encoder network f_φ infers a Causal (directed) graph for the given sample. One the other hand, the decoder network f_θ predicts future observations and aims at approximating g. Both functions are learnt jointly during training time.

$$x_{t+1} \approx f_\theta(\mathbf{x}_{\leq t}, f_\varphi(\mathbf{x}_{\leq t})) \tag{A.2}$$

At inference time, one is only interested in the output of the encoder, i.e the predicted causal graph $f_\varphi(\mathbf{x})$. Note that f_θ depends on $\mathbf{x}_{\leq t}$ just by restricting itself to information propagated through the inferred causal graph $f_\varphi(\mathbf{x}_{\leq t})$.

The training objective is formulated the following way.

$$\varphi^*, \theta^* = \text{argmin}_{\varphi,\theta} \mathcal{L}(\mathbf{X}_{train}, f_\varphi, f_\theta) \tag{A.3}$$

$$\mathcal{L}(\mathbf{X}_{train}, f_\varphi, f_\theta) = \sum_{\mathbf{x} \in \mathbf{X}_{train}} \sum_{t=1}^{T-1} l(x_{t+1}, f_\theta(\mathbf{x}_{\leq t}, f_\varphi(\mathbf{x}))) + r(f_\varphi(\mathbf{x})) \tag{A.4}$$

where φ^* and θ^* are the targeted parameters. r is a regularisation term on the resulting graph for enforcing sparsity, and T is the number of timestamps in a particular sample.

The problem is formalised as a variational inference problem. The encoder and decoder are modelled as conditional distributions $q_\varphi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x}|\mathbf{z})$ respectively, where $\mathbf{z}$ is a binary encoding of the causal graph. We briefly explain the construction of the encoder and decoder networks. The encoder infers aims at inferring a discrete representation of the graph, where each edge of the complete graph gets a category (the edge is absent or not). This is done by first inferring a value for each category, and then taking the argmax. In order to make the encoder differentiable, they use the Gumbel softmax trick, in order to approximate the argmax. The decoder predicts future values of the time series by propagating information through the graph.

Discussion

There are different scenarios where inferring varying causal relations for different time series samples can be useful for Cloud KPI data. First of all, because the context is important when studying the dependencies between KPIs. This context can be related to the state of the system, i.e its current activity or usage. For instance, it has been observed that the Response Time and Transfer Size are usually independent. However, if the Transfer Size gets saturated, then the Response Time suddenly explodes, introducing a strong

correlation. The context can also be dependent of the particular system that is being monitored, or the considered period of time. Furthermore, this framework can find a direct application towards Anomaly Detection. Indeed, investigating the occurrences of the causal relations through the different time series samples can help to discover which are the most likely and at which period of time. Therefore, it can inform us whether some detected causal relations are unusual, or conversely if the causal relations do not appear where we expect them. With this, one can potentially identify outlier time series samples.

Appendix B

Derivation Details

B.1 Orstein-Uhlenbeck based Synthetic Data Generating Process

B.1.1 Theoretical preliminaries

In this section, we provide useful definitions and results for designing the synthetic data generating process. These elements are taken from the lecture *Brownian Motion and Stochastic Calculus* by Prof. Martin Schweizer (Schweizer, 2022).

Îto Diffusion : Let $(\Omega, \mathcal{F}, \mathbb{F} = (\mathcal{F}_t)_{0\leq t\leq T}, \mathbb{P})$ be a filtered probability process, and $(W_t)_{0\leq t\leq T}$ an adapted d_W-dimensional Brownian motion, then an Îto diffusion is a d_X-dimensional process $X = (X_t)_{0\leq t\leq T}$ given by the solution of the stochastic differential equation

$$\begin{aligned} dX_t &= \mu(t, X_t)dt + \sigma(t, X_t)dW_t, \\ X_0 &= Y, \end{aligned} \tag{B.1}$$

where $\mu : [0, T] \times \mathbb{R}^{d_X} \to \mathbb{R}^{d_W}$ and $\sigma : [0, T] \times \mathbb{R}^{d_X} \to \mathbb{R}^{d_X \times d_W}$ are measurable functions called **drift** and **diffusion** respectively.

A solution of the last system is formalised by the following definition.

Definition B.1 *A **strong solution** of B.1 is a continuous $\mathbb{F}$-adapted process $X = (X_t)_{0\leq t\leq T}$ satisfying B.1 and such that*

$$\int_0^t (|\mu(s, X_s))| + |\sigma(s, X_s)|^2)ds < \infty.$$

Theorem B.2 *Under the assumptions*

1. $\mathbb{E}[Y^2] < \infty$,

2. *μ and σ are Lipschitz-continuous in x, uniformly in t, meaning that*

$$|\mu(t,x) - \mu(t,y)| \leq K|x-y| \quad \text{for all } t \in [0,T], x, y \in \mathbb{R}^{d_X},$$

3. *μ and σ have linear growth in x, uniformly in t, meaning that*

$$|\mu(t,x)| \leq K(1+|x|) \quad \text{for all } t \in [0,T], x \in \mathbb{R}^{d_X},$$

B.1 has a unique strong solution, up to indistinguishability, satisfying $||X_T^*||_{L^2(P)} < 0$.

Theorem B.3 (Îto formula for continuous semimartingales) *Let X be an $\mathbb{R}^{d_X}$-dimensional continuous semimartingale and $f : \mathbb{R}^{d_X} \to \mathbb{R}$ be a function of class $C^2(\mathbb{R}^{d_X}, \mathbb{R})$. Then $f(X)$ is also a continuous semimartingale and we have $\mathbb{P} - a.s$, for all $t \geq 0$*

$$f(X_t) = f(X_0) + \sum_{i=1}^{d_X} \int_0^t \frac{\partial f}{\partial x^i}(X_s, s) dX_s^i + \frac{1}{2} \sum_{i,l=1}^{d_X} \int_0^t \frac{\partial f}{\partial x^i \partial x^l}(X_s, s) d\langle X_s^i, X_s^l \rangle. \tag{B.2}$$

Remark B.4 *A strong solution to B.1 is a semimartingale. Thus, we can apply the Îto formula in this context.*

Theorem B.5 (Generalised Îto isometry) *Let $X = (X_t)_{t\in[0,T]}$, $Y = (Y_t)_{t\in[0,T]}$ be local martingales and H, G be predictable processes that are integrable with respect to X and Y respectively. Then we have*

$$\mathbb{E}\left[\int_0^T H_s dX_s \int_0^T G_s dY_s\right] = \mathbb{E}\left[\int_0^T H_s G_s d[X,Y]_s\right].$$

Theorem B.6 (Approximation of stochastic integral) *Let X by a semimartingale, H be an adapted RCLL process, and $(\Pi_n)_{n\in\mathbb{N}}$ a partition of $[0,T]$ with $\lim_{n\to\infty} |\Pi_n| = 0$. Then we have $\forall t \in [0,T]$,*

$$\int_0^t H_{s-} dX_s = \lim_{n\to\infty} \sum_{t_i \in \Pi_n} H_{t_i}(X_{t_{i+1}\wedge t} - X_{t_i \wedge t}) \quad \text{in probability.}$$

B.1.2 Process Characterisation

Theorem B.7 *The SDE defined in (2.1) admits a unique strong solution.*

Proof B.8 *Lets define $\mu(x,t) = -\theta(x - m(t))$ and $\sigma(x,t) = \Sigma$. We immediately see that μ and σ are Lipschitz-continuous in x uniformly in t. Also, since m is continuous and defined on a close interval, then it is also bounded by a constant $M \geq 1$. Thus, $|\mu(t,x)| \leq |\theta|(|M| + |x|) \leq |\theta M|(1 + |x|)$ and μ has linear growth in x uniformly in t (and σ as well). By Theorem B.2, (2.1) admits a unique strong solution.*

Theorem B.9 *The conditional distribution of the solution of (2.1) is Gaussian, given by*

$$X_t|\mathcal{A}_{\tau(t)} \sim \mathcal{N}\left(e^{-\theta(t-\tau(t))}X_{\tau(t)} + \int_{\tau(t)}^{t} e^{-\theta(t-s)}\theta m(s)ds, \right. \tag{B.3}$$

$$\left. \int_{\tau(t)}^{t} e^{-\theta(t-s)}\sigma\sigma^T e^{-\theta^T(t-s)}ds\right) \tag{B.4}$$

Proof B.10 *In the following, $X_{t,j}$ denotes the jth coordinate of X_t. For a matrix $A \in \mathbb{R}^{n\times m}$, $A_j \in \mathbb{R}^{1\times m}$, denotes its jth line, and A_{ji} the element of A line j column i. We define the continuous function $f : \mathbb{R}^{d_X} \times [0,T] \to \mathbb{R}^{d_X}$ such $f(x,t) = e^{\theta t}x$. We apply the Îto formula B.3 to f_j, the jth coordinate of f in order to derive the expression of X as*

$$\begin{aligned}
f_j(X_t,t) &= f_j(X_0,0) + \sum_{i=1}^{d_X}\int_0^t \frac{\partial f_j}{\partial x_i}(X_s,s)dX_{s,i} + \int_0^t \frac{\partial f_j}{\partial t}(X_s,s)ds \\
&+ \sum_{i,l=1}^{d_X}\int_0^t \frac{\partial f_j}{\partial x_i \partial x_l}(X_s,s)d\langle X_{s,i}, X_{s,l}\rangle \\
&= f_j(X_0,0) + \sum_{i=1}^{d_X}\int_0^t \frac{\partial f_j}{\partial x_i}(X_s,s)dX_{s,i} + \int_0^t \frac{\partial f_j}{\partial t}(X_s,s)ds \\
&= X_{0,j} - \sum_{i=1}^{d_X}\int_0^t (e^{\theta s})_{ji}\left(\theta_i(X_s - m(s))ds - \sigma_i dW_s\right) + \int_0^t (e^{\theta s}\theta)_j X_s ds \\
&= X_{0,j} - \sum_{i=1}^{d_X}\int_0^t (e^{\theta s})_{ji}\sum_{k=1}^{d_X}\theta_{ik}(X_{s,k} - m_k(s))ds + \sum_{i=1}^{d_X}\int_0^t (e^{\theta s}\theta)_{ji}X_{s,i}ds \\
&+ \sum_{i=1}^{d_X}\int_0^t (e^{\theta s})_{ji}\sigma_i dW_s \\
&= X_{0,j} + \int_0^t (e^{\theta s})_j \theta m(s)ds + \int_0^t (e^{\theta s})_j \sigma dW_s
\end{aligned}$$

In the first line, the terms with $d\langle X_{s,i}, s\rangle$, $d\langle s, X_{s,l}\rangle$, $d\langle s,s\rangle$ are ruled out, because the process defined by $Z_t = t$ has finite variation. We also note that the second partial derivatives of f_j with respect to x_i and x_l, and the partial derivatives of f_j with respect to x_i with $i \neq j$ are zero, which gives us the 3rd line.

In line 4, we develop the last expression by noting that $\frac{\partial f_j}{\partial x_i}(X_s, s) = (e^{\theta s})_{ji}$ *and* $\frac{\partial f_j}{\partial t}(X_s, s) = (e^{\theta s}\theta)_j X_s$. *Finally, by developing, the terms in line 5, we see that the terms* $(e^{\theta s})_j, \sum_{k=1}^{d_X} \theta_{ik} X^k = (e^{\theta s}\theta)_{ji} X_{s,i}$ *cancel themselves.*

Therefore,

$$X_t = e^{-\theta t} X_0 + \int_0^t e^{-\theta(t-s)} \theta m(s) ds + \int_0^t e^{-\theta(t-s)} \sigma dW_s, \tag{B.5}$$

or equivalently,

$$X_t = e^{-\theta(t-\tau(t))} X_{\tau(t)} + \int_{\tau(t)}^t e^{-\theta(t-s)} \theta m(s) ds + \int_{\tau(t)}^t e^{-\theta(t-s)} \sigma dW_s. \tag{B.6}$$

The last expression implies in particular that X *is a* ***(non-stationary) Markov process****.*

We can further decompose the j*th coordinate of the last term as*

$$\int_0^t (e^{\theta s}\sigma)_j dW_s = \sum_{k=1}^{d_W} \int_0^t (e^{\theta s}\sigma)_{jk} dW_{s,k}.$$

Let's denote Z *the deterministic process such that* $Z_t := (e^{\theta t}\sigma)_{jk}$. *It is continuous and thus predictable.* Z *is integrable with respect to* W_k *since*

$$||Z||^2_{L^2(W_k)} = \mathbb{E}\left[\int_0^T (Z_s^{jk})^2 d[W_k]_s\right] = \mathbb{E}\left[\int_0^T (e^{\theta t}\sigma)^2_{jk} ds\right] < \infty,$$

which justifies that the last term is valid.

W *is a martingale centred in zero and therefore, the stochastic integral* $Z \bullet W$ *is still a martingale centred in zero. One can immediately compute the conditional expectation*

$$\mathbb{E}[X_t | \mathcal{A}_{\tau(t)}] = e^{-\theta(t-\tau(t))} X_{\tau(t)} + \int_{\tau(t)}^t e^{-\theta(t-s)} \theta m(s) ds. \tag{B.7}$$

Now, we compute the conditional covariance, i.e. $Cov(X_t | \mathcal{A}_{\tau(t)})$. *The* i, j *entry of this matrix is*

$$
\begin{aligned}
&\mathbb{E}\left[\left(X_{t,i} - \mathbb{E}[X_{t,i}|\mathcal{A}_{\tau(t)}]\right)\left(X_{t,j} - \mathbb{E}[X_{t,j}|\mathcal{A}_{\tau(t)}]\right)|\mathcal{A}_{\tau(t)}\right] \\
&= \mathbb{E}\left[\left(\int_{\tau(t)}^{t} e^{-\theta(t-s)}\sigma dW_s\right)_i \left(\int_{\tau(t)}^{t} e^{-\theta(t-s)}\sigma dW_s\right)_j |\mathcal{A}_{\tau(t)}\right] \\
&= \sum_{k=1}^{d_W}\sum_{l=1}^{d_W}\mathbb{E}\left[\left(\int_{\tau(t)}^{t}(e^{-\theta(t-s)}\sigma)_{ik}dW_{s,k}\right)\left(\int_{\tau(t)}^{t}(e^{-\theta(t-s)}\sigma)_{jl}dW_{s,l}\right)\right] \\
&= \sum_{k=1}^{d_W}\sum_{l=1}^{d_W}\mathbb{E}\left[\int_{\tau(t)}^{t}(e^{-\theta(t-s)}\sigma)_{ik}(e^{-\theta(t-s)}\sigma)_{jl}d[W_k, W_l]_s\right] \\
&= \sum_{k=1}^{d_W}\mathbb{E}\left[\int_{\tau(t)}^{t}(e^{-\theta(t-s)}\sigma)_{ik}(\sigma^T e^{-\theta^T(t-s)})_{lj}ds\right] \\
&= \int_{\tau(t)}^{t}(e^{-\theta(t-s)}\sigma\sigma^T e^{-\theta^T(t-s)})_{ij}ds.
\end{aligned}
$$

In the second line, we observed that $\int_{\tau(t)}^{t} e^{-\theta(t-s)}\sigma dW_s$ *is independent of* $\mathcal{A}_{\tau(t)}$*, and decomposed the scalar products. For line three, we applied Theorem B.5. The fourth line is a consequence of* $d[W_k, W_l] = 0$ *for two independent Brownian Motions* W_k *and* W_l*.*

Therefore, we have

$$
Cov(X_t|\mathcal{A}_{\tau(t)}) = \int_{\tau(t)}^{t} e^{-\theta(t-s)}\sigma\sigma^T e^{-\theta^T(t-s)}ds. \tag{B.8}
$$

Furthermore, we can prove that $X_t|\mathcal{A}_{\tau(t)}$ *is normally distributed. By looking at the stochastic integral*

$$
\int_{\tau(t)}^{t} e^{\theta s}\sigma dW_s
$$

we see that the integrand is deterministic (thus adapted) and continuous. We can invoke theorem B.6 to get for a partition $(\Pi_n)_{n\in\mathbb{N}}$ *of* $[0, T]$ *with* $\lim_{n\to\infty}|\Pi_n| = 0$

$$
\int_{\tau(t)}^{t} e^{\theta s}\sigma dW_s = \lim_{n\to\infty}\sum_{t_i\in\Pi_n} e^{\theta t_i}\sigma(W_{(t_{i+1}\wedge t)\vee\tau(t)} - W_{(t_i\wedge t)\vee\tau(t)}) \quad \textit{in probability.} \tag{B.9}
$$

Because of the properties of the Brownian Motion, i.e independent and normally distributed increments, we get

$$\sum_{t_i \in \Pi_n} e^{\theta t_i} \sigma (W_{t_{i+1} \wedge t} - W_{t_i \wedge t}) \sim \mathcal{N}\left(0, \sum_{t_i \in \Pi_n} (t_{i+1} \wedge t - t_i \wedge t)(e^{\theta t_i})^T \sigma \sigma^T e^{\theta t_i}\right)$$

Therefore, the limit is also normally distributed. Finally, we have

$$X_t | \mathcal{A}_{\tau(t)} \sim \mathcal{N}\left(e^{-\theta(t-\tau(t))} X_{\tau(t)} + \int_{\tau(t)}^t e^{-\theta(t-s)} \theta m(s) ds, \right. \tag{B.10}$$

$$\left. \int_{\tau(t)}^t e^{-\theta(t-s)} \sigma \sigma^T e^{-\theta^T (t-s)} ds \right) \tag{B.11}$$

One can easily derive a closed form expression for the covariance term in case θ is diagonal. Lets denote its diagonal elements θ_j for $j \in \{1..d_X\}$, and $\Sigma := \sigma\sigma^T$. Note that $\theta_j > 0$ since θ is positive definite.

$$\int_{\tau(t)}^t e^{-\theta(t-s)} \sigma \sigma^T e^{-\theta^T (t-s)} ds = e^{-\theta(t-\tau(t))} \left(\int_0^{t-\tau(t)} e^{\theta s} \nu e^{-\theta^T s} ds \right) e^{-\theta^T (t-\tau(t))}$$

Let $(\Pi_n)_{n \in \mathbb{N}}$ be such that for $0 \leq i \leq n$ and $t_i \in \Pi_n$, $t_i = i\frac{t-\tau(t)}{n}$. By reusing the j,k entry of the covariance term, as in (B.9), we get for all $t - \tau(t) > 0$, $n \geq 1$

$$\begin{aligned}
\frac{t}{n} \sum_{i=0}^{n-1} (e^{\theta_j i (t-\tau(t))/n}) \Sigma_{jk} (e^{\theta_k i (t-\tau(t))/n})^T &= \Sigma_{jk} \frac{t}{n} \sum_{i=0}^{n-1} \left(e^{(\theta_j + \theta_k)(t-\tau(t))/n}\right)^i \\
&= \Sigma_{jk} \frac{t}{n} \left(1 - e^{(\theta_j+\theta_k)(t-\tau(t))/n}\right)^{-1} \left(1 - e^{(\theta_j+\theta_k)(t-\tau(t))}\right) \\
&\xrightarrow[n \to \infty]{} \Sigma_{jk} (\theta_j + \theta_k)^{-1} (e^{(\theta_j+\theta_k)(t-\tau(t))} - 1),
\end{aligned}$$

and therefore,

$$\left(\int_{\tau(t)}^t e^{-\theta(t-s)} \sigma \sigma^T e^{-\theta^T (t-s)} ds \right)_{jk} = \Sigma_{jk} (\theta_j + \theta_k)^{-1} (1 - e^{-(\theta_j+\theta_k)(t-\tau(t))}). \tag{B.12}$$

B.1.3 PD-NJODE Assumptions verifications

We verify that the modified version of the Orstein-Uhlenbeck process satisfies the assumptions of the PD-NJODE framework. Since we aim at estimating the conditional moments of the process, our study not only concerns X, but also X^m for $m \in \mathbb{N}$. For simplification, we only consider the case where the dimension $d_X = 1$.

Theorem B.11 *The process X defined with 2.1 satisfies the assumptions 3.2.2 of the PD-NJODE framework.*

Proof B.12 *First of all, it is a continuous process so we don't need to worry about jumps.*

Let's denote $\hat{X}^m$ the mth conditional moment with $\hat{X}_t^m = \mathbb{E}[X_t^m|\mathcal{A}_{\tau(t)}]$. Furthermore, let $\tilde{X}^{m,\leq s}$ be the rectilinear interpolation up to time s and F^m the function such that $F^m(t,\tau(t),\tilde{X}^{m,\leq\tau(t)}) = \hat{X}_t^m$ (we essentially took the same notation as in Krach et al. (2022)).

For $m=1$, we have for F^1 that

$$F^1(t,\tau(t),\tilde{X}^{m,\leq\tau(t)}) = (e^{-\theta(t-\tau(t))})X_{\tau(t)} + \int_{\tau(t)}^t (e^{-\theta(t-s)}\theta)m(s)ds. \tag{B.13}$$

It is continuous and differentiable in t and its derivative with respect to t can be computed with Leibniz's integral rule and is

$$\begin{aligned} f^1(t,\tau(t),\tilde{X}^{m,\leq\tau(t)}) = &-e^{-\theta(t-\tau(t))}\theta X_{\tau(t)} \\ &+ \left(-\int_{\tau(t)}^t e^{-\theta(t-s)}\theta^2 m(s)ds + \theta m(t)\right). \end{aligned} \tag{B.14}$$

The second term of both expression (B.13),(B.14) is bounded by a constant that depends on T, θ, and $\sup_{s\in[0,T]}|m(s)|_1$, while the first term is linearly bounded by $|X_{\tau(t)}|_1] \leq \sup_{s\in[0,t]}|X_s|_1 = X_t^*$. Therefore, Assumption 1 of the framework is fulfilled for $m=1$, i.e.

$$\left|F^1(t,\tau(t),\tilde{X}^{m,\leq\tau(t)})\right| + \left|f^1(t,\tau(t),\tilde{X}^{m,\leq\tau(t)})\right| \leq B_1(X_t^*+1), \tag{B.15}$$

for $B_1 > 0$ a constant depending on T, θ, and $\sup_{s\in[0,T]}|m(s)|_1$.

For $m=2$, the conditional moment is $\mathbb{E}[X_t^2|\mathcal{A}_{\tau(t)}] = (\mathbb{E}[X_t|\mathcal{A}_{\tau(t)}])^2$ $+ Var[X_t^m|\mathcal{A}_{\tau(t)}]$. Therefore,

$$\begin{aligned} F^2(t,\tau(t),\tilde{X}^{m,\leq\tau(t)}) = &\left(e^{-\theta(t-\tau(t))}X_{\tau(t)} + \int_{\tau(t)}^t e^{-\theta(t-s)}\theta m(s)ds\right)^2 \\ &+ \int_{\tau(t)}^t e^{-\theta(t-s)}\sigma^2 e^{-\theta^T(t-s)}ds, \end{aligned}$$

which is also continuous and differentiable in t. Since the variance term is also bounded (by T, θ, and σ), we get by a similar argument that

$$\left|F^2(t,\tau(t),\tilde{X}^{m,\leq\tau(t)})\right| + \left|f^2(t,\tau(t),\tilde{X}^{m,\leq\tau(t)})\right| \leq B_2(X_t^*+1)^2,$$

for some $B_2 > 0$.

For $m > 2$, *one can proceed similarly by observing that* $\mathbb{E}[X_t^m|\mathcal{A}_{\tau(t)}]$ *depends polynomially on* $\mathbb{E}[X_t|\mathcal{A}_{\tau(t)}]$ *and* $Var[X_t^m|\mathcal{A}_{\tau(t)}]$, *in order to verify Assumption 1.*

For Assumption 2, one can use Lemma 16.1.4 from Cohen and Elliott (2015) since both drift and diffusion terms satisfy the linear growth condition. Consequently, for all $p \in \mathbb{N}$,

$$\mathbb{E}[(X_T^*)^p] < \infty.$$

Therefore, this holds also for all moments of order $m \in \mathbb{N}$

$$\mathbb{E}[(X_T^{m,*})^p] < \infty. \tag{B.16}$$

If the process has dimension $d_X > 1$, *the arguments are similar, except that one has to account for the cross-coordinates covariance terms. For instance for* $d_X = 2$ *and* $m = 2$, *the assumptions have to be verified for* $\mathbb{E}[X_{t,1}X_{t,2}|\mathcal{A}_{\tau(t)}]$ *as well.*

Bibliography

K. Bandyopadhyay, A. K. Bhattacharya, Parthapratim Biswas, and D. A. Drabold. Maximum entropy and the problem of moments: A stable algorithm. *Physical Review E*, 71(5), may 2005. doi: 10.1103/physreve.71.057701. URL https://doi.org/10.1103%2Fphysreve.71.057701.

Seif-Eddine Benkabou, Khalid Benabdeslem, and Bruno Canitia. Unsupervised outlier detection for time series by entropy and dynamic time warping. *Knowl. Inf. Syst.*, 54(2):463–486, feb 2018. ISSN 0219-1377. doi: 10.1007/s10115-017-1067-8. URL https://doi.org/10.1007/s10115-017-1067-8.

Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A. Lozano. A review on outlier/anomaly detection in time series data, 2020.

Diego Carrera, Beatrice Rossi, Daniele Zambon, Pasqualina Fragneto, and Giacomo Boracchi. Ecg monitoring in wearable devices by sparse models. In Bettina Berendt, Björn Bringmann, Élisa Fromont, Gemma Garriga, Pauli Miettinen, Nikolaj Tatti, and Volker Tresp, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 145–160, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46131-1.

Zekai Chen, Dingshuo Chen, Zixuan Yuan, Xiuzhen Cheng, and Xiao Zhang. Learning graph structures with transformer for multivariate time series anomaly detection in iot. *CoRR*, abs/2104.03466, 2021. URL https://arxiv.org/abs/2104.03466.

Kukjin Choi, Jihun Yi, Changhwa Park, and Sungroh Yoon. Deep learning for anomaly detection in time-series data: Review, analysis, and guidelines. *IEEE Access*, 9:120043–120065, 2021. doi: 10.1109/ACCESS.2021.3107975.

Samuel N. Cohen and Robert J. Elliott. *Lipschitz Stochastic Differential Equations*, pages 397–426. Springer New York, New York, NY, 2015. ISBN 978-1-

4939-2867-5. doi: 10.1007/978-1-4939-2867-5_16. URL https://doi.org/10.1007/978-1-4939-2867-5_16.

T.M. Cover and J.A. Thomas. *Entropy, Relative Entropy and Mutual Information*, chapter 2, pages 12–49. John Wiley & Sons, Ltd, 2001. ISBN 9780471200611. doi: https://doi.org/10.1002/0471200611.ch2. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/0471200611.ch2.

Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. *CoRR*, abs/2106.06947, 2021. URL https://arxiv.org/abs/2106.06947.

Liwei Deng, Xuanhao Chen, Yan Zhao, and Kai Zheng. Hifi: Anomaly detection for multivariate time series with high-order feature interactions, 2021.

Nan Ding, HaoXuan Ma, Huanbo Gao, YanHua Ma, and GuoZhen Tan. Real-time anomaly detection based on long short-term memory and gaussian mixture model. *Computers & Electrical Engineering*, 79:106458, 2019. ISSN 0045-7906. doi: https://doi.org/10.1016/j.compeleceng.2019.106458. URL https://www.sciencedirect.com/science/article/pii/S0045790618334372.

Heming Du, Shouguo Du, and Wen Li. Probabilistic time series forecasting with deep non-linear state space models. *CAAI Transactions on Intelligence Technology*, n/a(n/a), 2022. doi: https://doi.org/10.1049/cit2.12085. URL https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cit2.12085.

Michael Eichler. Causal inference with multiple time series: principles and problems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1997):20110613, 2013. doi: 10.1098/rsta.2011.0613. URL https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2011.0613.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery and Data Mining*, 1996.

Marco Frontini and Aldo Tagliani. The moment problem and maximum entropy: numerical investigation. 2007.

Paolo Giordani, Michael Pitt, and Robert Kohn. 6061 Bayesian Inference for Time Series State Space Models. In *The Oxford Handbook of Bayesian Econometrics*. Oxford University Press, 09 2011. ISBN 9780199559084. doi: 10.1093/oxfordhb/9780199559084.013.0004. URL https://doi.org/10.1093/oxfordhb/9780199559084.013.0004.

Mononito Goswami, Cristian Challu, Laurent Callot, Lenon Minorics, and Andrey Kan. Unsupervised model selection for time-series anomaly detection, 2023.

C W J Granger. Investigating Causal Relations by Econometric Models and Cross-Spectral Methods. *Econometrica*, 37(3):424–438, July 1969. URL https://ideas.repec.org/a/ecm/emetrp/v37y1969i3p424-38.html.

James Douglas Hamilton. *Time Series Analysis*. Princeton University Press, Princeton, 1994. ISBN 9780691218632. doi: doi:10.1515/9780691218632. URL https://doi.org/10.1515/9780691218632.

Jakob Heiss, Josef Teichmann, and Hanna Wutte. How implicit regularization of relu neural networks characterizes the learned function – part i: the 1-d case of two layers with random first layer, 2021.

Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &amp Data Mining*, pages 387–395, 07 2018. doi: 10.1145/3219819.3219845.

V. John, I. Angelov, A.A. Öncül, and D. Thévenin. Techniques for the reconstruction of a distribution from a finite number of its moments. *Chemical Engineering Science*, 62(11):2890–2904, 2007. ISSN 0009-2509. doi: https://doi.org/10.1016/j.ces.2007.02.041. URL https://www.sciencedirect.com/science/article/pii/S0009250907002072.

Pidt de Jong and Singfat Chu-Chun-Lin. Stationary and non-stationary state space models. *Journal of Time Series Analysis*, 15(2):151–166, 1994. doi: https://doi.org/10.1111/j.1467-9892.1994.tb00182.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9892.1994.tb00182.x.

E. Keogh, J. Lin, and A. Fu. Hot sax: efficiently finding the most unusual time series subsequence. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8 pp.–, 2005. doi: 10.1109/ICDM.2005.79.

Eamonn Keogh, Stefano Lonardi, and Bill 'Yuan-chi' Chiu. Finding surprising patterns in a time series database in linear time and space. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 550–556, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 158113567X. doi: 10.1145/775047.775128. URL https://doi.org/10.1145/775047.775128.

Tung Kieu, Bin Yang, and Christian S. Jensen. Outlier detection for multi-dimensional time series using deep neural networks. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)*, pages 125–134, 2018. doi: 10.1109/MDM.2018.00029.

Florian Krach, Marc Nübel, and Josef Teichmann. Optimal estimation of generic dynamics by path-dependent neural jump odes, 2022. URL https://arxiv.org/abs/2206.14284.

Sindy Löwe, David Madras, Richard Zemel, and Max Welling. Amortized causal discovery: Learning to infer causal graphs from time-series data, 2020. URL https://arxiv.org/abs/2006.10833.

Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Lstm-based encoder-decoder for multi-sensor anomaly detection, 2016.

Larry Manevitz and Malik Yousef. One-class svms for document classification. *Journal of Machine Learning Research*, 2:139–154, 01 2001. doi: 10.1162/15324430260185574.

Hamed Masnadi-Shirazi, Alireza Masnadi-Shirazi, and Mohammad-Amir Dastgheib. A step by step mathematical derivation and tutorial on kalman filters, 2019.

Judea Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, 3 (none):96 – 146, 2009. doi: 10.1214/09-SS057. URL https://doi.org/10.1214/09-SS057.

Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/5cf68969fb67aa6082363a6d4e6468e2-Paper.pdf.

Faraz Rasheed and Reda Alhajj. A framework for periodic outlier pattern detection in time-series sequences. *IEEE Transactions on Cybernetics*, 44(5): 569–582, 2014. doi: 10.1109/TSMCC.2013.2261984.

Huorong Ren, Mingming Liu, Zhiwu Li, and Witold Pedrycz. A piecewise aggregate pattern representation approach for anomaly detection in time series. *Knowledge-Based Systems*, 135:29–39, 2017. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2017.07.021. URL https://www.sciencedirect.com/science/article/pii/S0950705117303465.

Martin Schweizer. Brownian motion and stochastic calculus, lecture notes, 2022.

Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19,

page 2828–2837, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016. doi: 10.1145/3292500.3330672. URL https://doi.org/10.1145/3292500.3330672.

Aldo Tagliani. Hausdorff moment problem and maximum entropy: A unified approach. *Appl. Math. Comput.*, 105:291–305, 1999.

P. Vatiwutipong and N. Phewchean. Alternative way to derive the distribution of the multivariate ornstein–uhlenbeck process, 2019.

Chengyu Wang, Kui Wu, Tongqing Zhou, Guang Yu, and Zhiping Cai. Tsagen: Synthetic time series generation for kpi anomaly detection. *IEEE Transactions on Network and Service Management*, 19(1):130–145, 2022. doi: 10.1109/TNSM.2021.3098784.

Tailin Wu, Thomas Breuel, Michael Skuhersky, and Jan Kautz. Discovering nonlinear relations with minimum predictive information regularization, 2020. URL https://arxiv.org/abs/2001.01885.

Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. Multivariate time-series anomaly detection via graph attention network, 2020. URL https://arxiv.org/abs/2009.02040.

Yan Zhao, Liwei Deng, Xuanhao Chen, Chenjuan Guo, Bin Yang, Tung Kieu, Feiteng Huang, Torben Bach Pedersen, Kai Zheng, and Christian S. Jensen. A comparative study on unsupervised anomaly detection for time series: Experiments and analysis, 2022.