Spring 5-15-2021

Anomaly detection in smart grids using multi-task Gaussian processes

Aman Karra
University of New Mexico

Aman Karra

Candidate

Electrical and Computer Engineering

Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Manel Martinez-Ramon, Chairperson

Tameem Albash

Ramiro Jordan

Anomaly detection in smart grids using multi-task Gaussian processes

by

Aman Karra

B.Tech, Shiv Nadar University, 2017

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Computer Engineering

The University of New Mexico

Albuquerque, New Mexico

May, 2021

Dedication

To my parents, Supriya and Shastry, for their love and support.
And my brother, Tarun, for some much needed humor.

Acknowledgments

I would like to thank my advisor, Professor Manel Martinez Ramon, for his support, patience, and guidance as I struggled to understand things that initially seemed beyond me. A very special thanks to Miguel for his generous help and for answering questions that I was hesitant to ask elsewhere. I also want to thank Aswathy for being such a great collaborator.
I am grateful to Sravani and Rajesh for inviting me to their home and providing a much needed break from the isolation brought about by the pandemic.
There are a few others who helped in ways unexpected. Vikas, with his delicious cooking that restores the spirit; Sabya, who got me started with LaTex; Ganesh and Aman Geetey, by being good critics of my ideas and my writing; Keerthana, with her timely encouragement. I thank them all.

Anomaly detection in smart grids using multi-task Gaussian processes

by

Aman Karra

B.Tech, Shiv Nadar University, 2017

M.S., Computer Engineering, University of New Mexico, 2021

Abstract

The availability of data and computing infrastructure in smart grids creates new challenges that invite solutions based on algorithmic techniques. A particular problem of interest in these systems is fault detection. It is difficult to characterize faults because of the number of different ways in which these faults can occur. Moreover, simulating a faulty mode of the system can be expensive. This work addresses fault detection by framing it as the task of detecting anomalies in the sensor data of smart grids. The assumption made is that faults are events of low probability. But no reliance is made on any possible availability of information classifying the sensor data as faulty or normal. Thus, the problem is solved purely based on the structure of the data, by building an anomaly detection system consisting of a multi-task Gaussian process (MTGP) model coupled with a One-Class Support Vector Machine (OC-SVM).

Contents

List of Figures

List of Tables

Glossary

$\mathbf{A^T}$	Transpose of a matrix $\mathbf{A}$.
Tr($\mathbf{A}$)	Trace of a matrix $\mathbf{A}$.

Chapter 1

Introduction

1.1 Introduction

Modern electrical grids have only been increasing in complexity, with the power generation distributed across multiple sources like hydro, solar photo-voltaic, wind power plants etc [18]. Moreover, the relationship of consumers with the grid has also changed with a percentage of the consumers installing small power generation systems at homes. This requires the grid to employ power electronics to control frequency and voltage levels which makes the grid less robust to faults [2]. Thus, the detection and diagnosis of faults has become an important problem.

To address such challenges, smart grids are being embedded with greater measurement and communication capabilities. These systems produce large amounts of data about the grid which means there is potential to apply algorithms, especially from machine learning, to solve problems of detecting and diagnosing faults. Faults can mean two different events: there is either a physical fault in the grid or there is some abnormal behavior in the communication systems. Research related to fault detection in smart grids can also be partitioned along these lines. Typically, in these

works, the availability of information about faulty states is assumed. And sometimes, the ability to simulate a faulty state and use the knowledge that the state being simulated is a faulty one is assumed.

This thesis deals with detecting faults based on the sensor readings corresponding to the physical state of the smart grid. The fault detection is done by assuming that faults are events of low probability. When viewed from the perspective of system data, this means the problem is that of anomaly detection. But we do not assume the availability of any information on which part of the system data represents faults. Thus, in machine learning jargon, we are trying to do unsupervised anomaly detection. The anomaly detection idea used here is based on the following known technique from the literature: build a regression model for the data and use the prediction error or residual to infer if a given sample is an anomaly [6]. It is important to ensure that the regression model is robust against outliers to ensure good performance.

Research on smart grids requires reliable and realistic simulations of the grids. There are multiple standard simulated systems that are called test grids. Commonly used ones are provided by the IEEE, classified according to the number of buses or node: 4, 13, 34, 37, 123, 324, and 8500 [18]. The data for this work was generated from the IEEE 123 test grid. Note that although faulty modes of the system were simulated, this information is not used when detecting the faults.

The thesis is organized as follows:

1. Chapter 2: Briefly introduces the most commonly used techniques on unsupervised anomaly detection.

2. Chapter 3: Introduces Gaussian processes and their multi-task extension.

3. Chapter 4: Explains the data, the experimental methodology, results, and their implications.

4. Chapter 5: Summarizes the main takeaways and talks of future work.

Chapter 2

Anomaly Detection

2.1 Introduction

Anomaly detection is the task of identifying samples in data that either do not belong to the distribution from which the data has been sampled, or are outliers within the same distribution. Techniques for anomaly detection can rely on the presence of labels identifying a subset of the data as normal or anomaly. This is the supervised and semi-surpervised anomaly detection case. On the other hand, when no labels are present it is called unsupervised anomaly detection. This work deals with the unsupervised case.

An important assumption for the unsupervised case is that the occurrence of anomalies in the data is infrequent. If this condition is not met, then it can be expected to see a high false alarm rate. In the unsupervised setting, anomaly detection techniques come in a few flavors: distance based, probabilistic model based, and model-free. Techniques of the first type rely on a distance metric applicable for the problem; in the second type, the underlying distribution of the data is modeled; and the last class of techniques rely simply on the structure of the data. A brief

introduction to the algorithms of each type follows, starting with the model-free ones.

2.2 Model-Free Techniques

The two most important algorithms in the model-free framework are One-Class Support Vector Machine (OC-SVM) and Support Vector Data Descriptor (SVDD).

2.2.1 One-Class SVM

The main idea is to estimate the support of the probability distribution of the given data. This is done by constructing a hyperplane that maximizes the separation of the data from the origin. Any data point that falls beyond the hyperplane is classified as an anomaly [16].

Consider that we have a training dataset $X = \{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}\}$, for some $n \in \mathbb{N}$, the set of natural numbers. Each element of the set is a vector in $\mathbb{R}^D$. SUppose that $\mathbb{H}$ is a Hilbert space with inner product $\langle, \rangle$. Assume that a mapping $\phi : \mathbb{R}^D \to \mathbb{H}$, where $\mathbb{H}$ is a Hilbert space is applied to the elements of X. The inner product in $\mathbb{H}$ can be represented in terms of the $\mathbf{x_i}$ themselves by a kernel function $k(\mathbf{x_i}, \mathbf{x_j}) = \langle \phi(\mathbf{x_i}), \phi(\mathbf{x_j}) \rangle$. The details of the existence of such kernel functions will be explained in the next chapter.

We want a function $f : \mathbb{H} \to \{-1, +1\}$ defined as $f(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}\phi(\mathbf{x}) \rangle + b)$. This f takes the value $+1$ only in the region where our data can possibly live. Everywhere else it is -1. This is achieved by finding a hyperplane that maximally separates the data from the origin. Specifically, this means that the hyperplane should allow as large a margin as possible between the origin and the data points, i.e, if a hyperplane

is defined by $(\mathbf{w}, b)$ then we want $\mathbf{w}$ and b that satisfy:

$$\begin{aligned} &\underset{\mathbf{w},b}{\text{argmax}} \ \min |\langle \mathbf{w}, \mathbf{x_i} \rangle + b|, \text{ such that,} \\ &\qquad\qquad \forall i, f(\mathbf{x_i}) > 0, \text{ and} \\ &\qquad\qquad |\mathbf{w}|^2 = 1. \end{aligned} \tag{2.1}$$

The above problem can be simplified as:

$$\begin{aligned} &\underset{\mathbf{w},b}{\text{argmin}} \, |\mathbf{w}|^2, \text{ s.t.,} \\ &\quad \forall i, f(\mathbf{x_i}) \geq 1. \end{aligned} \tag{2.2}$$

An additional requirement is that we want to avoid finding a hyperplane that is biased by the presence of outliers. This means that misclassifications need to be allowed, but with a penalty. Adding this requirement the final optimization problem becomes [16]:

$$\begin{aligned} &\min_{\mathbf{w}} \frac{1}{2}||\mathbf{w}||^2 + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i - b, \text{ subject to} \\ &\qquad\qquad \langle \mathbf{w}, \phi(\mathbf{x_i}) \rangle \geq b - \xi_i, \\ &\qquad\qquad \xi_i \geq 0, \forall i \in \{1, 2, \cdots, n\}, \\ &\qquad\qquad b \in \mathbb{R}, \text{ the set of all real numbers.} \end{aligned} \tag{2.3}$$

The ξ_i are known as slack variables. Adding them gives the property that for some solution $\mathbf{w}^*, b^*$ to (2.3), f will be $+1$ for most of the x_i in the training set, for a small $|\mathbf{w}^*|$. This tradeoff is controlled by the parameter ν. This ν is an upper bound on the number of outliers. It is also a lower bound on the number of support vectors the solution returns. The support vectors are the data points that lie on the margin.

2.2.2 Support Vector Data Description

The SVDD was introduced by Tax and Duin in [19]. This technique constructs a hypersphere using the given data. Any data point that lies outside this hypersphere is classified as an anomaly.

The objective is defined as a search for a hypersphere with the smallest volume, that completely encompasses the data. Assuming we have the same training dataset as described in the previous section, this can be expressed as:

$$\begin{aligned} &\min\ \mathbf{R}^2, \text{ such that:} \\ &|\mathbf{x_i} - \mu|^2 \leq \mathbf{R}^2, \forall i, \end{aligned} \tag{2.4}$$

where μ is the center of the hypersphere and $\mathbf{R}$ is the radius. This optimization problem can only be used when we are sure that the training data does not contain any anomalies. To accommodate for the possibility of the presence of anomalies, we need to allow some of the training data points to be left outside the hypersphere, potentially misclassifying them. This requirement can be baked into (2.4) using a similar idea from the previous section: allow points to lie outside the boundary but penalize the objective function for every point that it leaves outside. The objective function now becomes:

$$\begin{aligned} \min\ \mathbf{R}^2 + C\sum_{i=1}^{n} \xi_i \text{ such that} \\ ||\mathbf{x_i} - \mu||^2 \leq \mathbf{R}^2 + \xi_i, \forall i, \text{ and,} \\ \xi_i > 0, \forall i. \end{aligned} \tag{2.5}$$

Upon solving this and finding a center and radius $(\mu^*, \mathbf{R}^*)$, a new test point $\mathbf{x}_*$ is classified an anomaly if $|\mathbf{x}_* - \mu^*|^2 > (\mathbf{R}^*)^2$. The parameter C is similar to ν from the one-class SVM.

2.3 Probabilistic Techniques

In these techniques a probabilistic model is constructed based on the training data and new data is checked for a good fit with respect to the model. Based on the fit an anomaly score is assigned. Constructing a probabilistic model can be parametric or non-parametric.

2.3.1 Parametric Approach

Consider that we have i.i.d. data $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}\}$. Let θ be a vector of parameters that completely characterize the assumed underlying distribution from which this data has been generated. The log likelihood of observing this data given θ is given as [17]:

$$L(\mathbf{X}; \theta) = \log\left(\prod_{i=1}^{n} P_\theta(\mathbf{x_i})\right) = \sum_{i=1}^{n} \log(P_\theta(\mathbf{x_i})), \tag{2.6}$$

where P_θ is the parametric probability distribution of $\mathbf{X}$. This log likelihood needs to be maximized with respect to the parameters; the value of θ that does it is called the maximum likelihood estimate (MLE).

A new test point $\mathbf{x}$ is classified as an anomaly if $P_\theta(\mathbf{x}) < \delta$ is true, where δ is a threshold. Setting the threshold can be done by heuristics, but there are more principled approaches based on Extreme Value Theory (EVT). The Fisher–Tippett theorem [7] states that extreme values of any distribution follow the Weibull, Gumbet, or a Feichel distribution. A general definition of an extreme value for a probability density function f is (see [7]):

$$\underset{\mathbf{x}}{\operatorname{argmin}}\, f(\mathbf{x}), \forall \mathbf{x} \in \mathbf{X} \tag{2.7}$$

To set the threshold δ, apply a transformation on f that maps to a unimodal EVD, like a Gumbet if the initial distribution is Gaussian, and then compute the MLE of the resulting distribution to learn a good threshold based on the learned parameters. Details of closed form analytical and numerical solutions for a variety of distributions can be found in [7].

When modeling the underlying distribution, a commonly used representation involves latent variables. If Y represents the latent random variable that takes values from the finite set $\{y_1, y_2, \cdots, y_m\}$, the log likelihood of one data sample $\mathbf{x_k}$ is written

as:

$$\log P_\theta(\mathbf{x_k}) = \log\left(\sum_{i=1}^{m} P_\theta(\mathbf{x_k}, y_i)\right). \tag{2.8}$$

To find the MLE for all the data we will need to maximize a sum of logarithms:

$$\operatorname{argmax} \sum_{k=1}^{n} \left(\log\left(\sum_{i=1}^{m} P_\theta(\mathbf{x_k}, y_i)\right)\right). \tag{2.9}$$

Solving (2.9) is hard because of the presence of the latent variables. Instead, let's define a new function F [17]:

$$F(Q, \theta) = \sum_{i=1}^{n}\sum_{j=1}^{m} Q_{y_j} \log(P_\theta[\mathbf{x_i}, y_j]). \tag{2.10}$$

The expectation maximization algorithm can be used to solve (2.11). The EM algorithm has two steps:

1. E-step:

$$Q_{y_j}^t = P_{\theta^{t-1}}(y_j|\mathbf{x_k}) \tag{2.11}$$

2. M-step:

$$\theta^t = \operatorname*{argmax}_\theta F(Q^t, \theta). \tag{2.12}$$

A common technique is to model the data as a mixture of Gaussians. Using the EM algorithm we can assign the samples to one of K clusters. GMMs have allowed people to use a variety of heuristics to decide on what an anomaly looks like in their problem. An example would be to allow all the clusters to represent modes of normal behavior of a system [14].

2.3.2 Non-Parametric Approaches

The most straightforward non-parametric technique is constructing histograms of the data. If the data happens to be multivariate, histograms are built for individual

variables. If one is confident that the data mostly represents normal behavior, any new test data that does not fit any bin in the appropriate histogram can be classified as an anomaly. Such approaches have been used for detecting network intrusions, and fraud in financial institutions [6]. An important trade-off is between the probability of false alarm, the probability of anomaly detection, and bin size. Too small a bin size will lead to a greater probability of false alarm while too large a bin size will lead to a lower probability of detection.

Another non-parametric technique is the kernel density estimator. This technique is also called the Parzen-Rosenblatt window. For a given $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, \cdots, \mathbf{x_n}\}$ that are independent and identically distributed, with a probability density function f, the estimated density is given by [11]:

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right), \tag{2.13}$$

where K is a kernel (same as those seen in One-Class SVM) and $h > 0$ is used for smoothing the estimated density.

2.4 Distance Based

These techniques require a definition of distance that can be applied to the data. For continuous data, this is usually the Euclidean distance. The k-nearest neighbor is the most commonly used one. In this, for every sample, the algorithm computes the k nearest neighbors. This information is used to assign an anomaly score for the current sample in different ways, depending on the problem. For example, the anomaly score could simply be the distance between the current sample and its k^{th} nearest neighbour, or the sum of all the k distances. A different method involves computing the global density of any sample by considering the number of points contained within a hypersphere of radius r centered at that point. The number of points within the hypersphere divided by the volume of the hypersphere is considered

the density. The anomaly score is the inverse of this density because the assumption is that the more densely populated a region, the less likely it is to be an region of anomaly. For a hypersphere in $\mathbb{R}^n$ the anomaly score becomes (see [3]):

$$\begin{aligned} &\frac{k\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}+1\right)}r^n, \text{ with} \\ &\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt. \end{aligned} \tag{2.14}$$

An improved technique to handle regions of varying density is Local Outlier Factor (LOF) [6], [4]. This algorithm introduces local density for any data point. The local density is computed as the ratio of k nearest neighbors and the volume of the smallest hypersphere that contains those k neighbors. Finally, to compute the LOF score take the ratio of the average local density of the closest k neighbors and the local density of the current point. A high LOF score will indicate an anomaly.

Such distance based clustering techniques can be very useful for quick analysis of a reasonable sample size, provided a distance metric can be easily constructed. Moreover, their modeling of the distribution of the data is entirely dependent on the observed data and there is no need for prior assumptions about the underlying distribution, except for the fact that anomalies occur infrequently, and far away from regions of "normality". This required assumption can end up being useful or harmful. An important concern is the runtime of these algorithms. For N points of dimensions d, efficient implementations can run in $\mathcal{O}(Nd)$.

All the techniques discussed in this chapter have a wide variety of applications. They have been used in network intrusion detection, fraud detection, mechanical structure failure prediction, medical anomaly detection and many more. A good list of references on the individual applications of these techniques in a specific domain can be found in [6] and [12]. Advances in deep learning have also led to new techniques and applications in anomaly detection. A survey of techniques can be found in [5]. A unified survey of classical and deep learning techniques for anomaly detection can be found in [14].

Chapter 3

Gaussian Processes

3.1 A Gentle Introduction

Let $[\mathbf{m}] = \{1, 2, \cdots, m\}$ and suppose that $\mathbf{X}$ is a subset of $\mathbb{R}^n$. Then consider points $\mathbf{x_i} \in \mathbf{X}$ for i in $[\mathbf{m}]$. We are given the result of applying a function $f : \mathbb{R}^n \to \mathbb{R}$, denoted by $\mathbf{y}$, where $\mathbf{y} = \{y_1, \cdots y_m\}$. Our task is to learn the function f reasonably well, so that, given a new point $\mathbf{x}_* \in \mathbb{R}^n$,we can make an estimate of its corresponding $\mathbf{y}_*$. Note that these $\mathbf{x_i}$ are assumed to be drawn independently from a fixed underlying distribution that is not known.

For now let us simplify the problem as much as possible. Assume that we are given an extra piece of information: the relationship between the $\mathbf{x}$ and y can be approximated by a linear model. What this means is that $f(\mathbf{x}) = \mathbf{x}^T\mathbf{w}$, where $\mathbf{w}$ too is a vector in $\mathbb{R}^n$. Let us also assume that our observations y contain noise, and we write out the model as,

$$y = f(\mathbf{x}) + \epsilon \tag{3.1}$$

where ϵ is the noise. This noise could be of any kind but for our case we assume that its values are samples from a Gaussian distribution with zero mean and variance σ_n^2

that are drawn independently, i.e.,

$$\epsilon \sim \mathcal{N}(0, \sigma_n^2). \tag{3.2}$$

Let us put all the $\{\mathbf{x_i}, y_i\}$ pairs in a set and call it the training set S. It appears that if we can find some good $\mathbf{w}$, we can solve our problem. So let's try to find the distribution of $\mathbf{w}$ given the y_i and the $\mathbf{x_i}$, denoted as

$$P(\mathbf{w}|\mathbf{X}, \mathbf{y}). \tag{3.3}$$

This looks like a something that we could use Baye's rule on. If we assume the above quantity to be the posterior distribution, we need a prior, a likelihood, and a marginal distribution.

For the likelihood we can write the probability of observing any y_i given the corresponding $\mathbf{x_i}$ and some $\mathbf{w}$ as:

$$P(y_i|x_i, w) = (\frac{1}{\sigma_n\sqrt{2\pi}}) \exp\left(\frac{-(y_i - \mathbf{x_i}^T\mathbf{w})^2}{2\sigma_n^2}\right), \tag{3.4}$$

which, for all the given samples can be written as a product because of the independence assumption:

$$\begin{aligned} &P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{n}[P(y_i|\mathbf{x_i}, \mathbf{w})] \\ =&\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(\frac{-(y_i - \mathbf{x_i}^T\mathbf{w})^2}{2\sigma_n^2}\right) \\ =&\mathcal{N}(\mathbf{X}^T\mathbf{w}, \sigma_n^2 I). \end{aligned} \tag{3.5}$$

For the prior, we can assume that $\mathbf{w}$ is drawn from a multivariate Gaussian with a zero mean vector and a covariance matrix $\mathbf{\Sigma}$,

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}). \tag{3.6}$$

The marginal likelihood will become $P(\mathbf{y}|\mathbf{X})$. Putting it all together we get

$$P(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{P(\mathbf{y}|\mathbf{X},\mathbf{w})P(\mathbf{w})}{P(\mathbf{y}|\mathbf{X})} \tag{3.7}$$

Since the marginal likelihood does not feature $\mathbf{w}$ in its formulation we can think of it as a normalizing constant for now, and drop it from the equation to get [13]:

$$P(\mathbf{w}|\mathbf{x},\mathbf{y}) \sim \mathcal{N}(\overline{w} = \frac{1}{\sigma_n^2}\mathbf{A^{-1}}\mathbf{X}\mathbf{y}, \mathbf{A^{-1}}), \tag{3.8}$$

where $\mathbf{A} = \sigma_n^{-2}\mathbf{X}\mathbf{X}^T + \mathbf{\Sigma_p^{-1}}$. When given a test input $\mathbf{x}_*$, we make our prediction f_* by averaging over all the possible $\mathbf{w}$, or all the linear models, using the posterior predictive distribution [13]:

$$\begin{aligned} P(f_*|\mathbf{x}_*,\mathbf{X},\mathbf{y}) &= \int P(f_*|\mathbf{x}_*,\mathbf{w})P(\mathbf{w}|\mathbf{X},\mathbf{y})d\mathbf{w} \\ &= \mathcal{N}(\frac{1}{\sigma_n^2}{\mathbf{x}_*}^T\mathbf{A}^{-1}\mathbf{X}\mathbf{y}, {\mathbf{x}_*}^T\mathbf{A}^{-1}\mathbf{x}_*). \end{aligned} \tag{3.9}$$

What do we do when the relationship between $\mathbf{x}$ and y cannot be expressed as a linear model? We need to make our model more expressive. For that, we can project the vectors $\mathbf{x}$ into a higher dimensional vector space, where a linear model in that space might be a good fit for the data.

Consider a function $\phi : \mathbb{R}^d \to \mathbb{R}^m$, where $m > d$. We can use this function to map our inputs $\mathbf{x}$ to $\mathbb{R}^m$ and create a new set of inputs for a new model. This means that our weight vector $\mathbf{w}$ is also in $\mathbb{R}^m$ now. Since the target y stays the same, we can simply write our new model as:

$$\begin{aligned} f(\mathbf{x}) &= \phi(\mathbf{x})^T\mathbf{w} \\ \implies y &= \phi(\mathbf{x})^T\mathbf{w} + \epsilon. \end{aligned} \tag{3.10}$$

This model is still linear in $\mathbf{w}$. As the equation probably suggests, we simply need to replace the set $\mathbf{X}$ with $\Phi(\mathbf{X})$ in the previous linear model. The new posterior becomes [13]:

$$f_*|\mathbf{x}_*,\mathbf{X},\mathbf{y} \sim \mathcal{N}(\frac{1}{\sigma_n^2}\phi(\mathbf{x}_*)^T\mathbf{A}^{-1}\Phi\mathbf{y}, \phi(\mathbf{x}_*)^T\mathbf{A}^{-1}\phi(\mathbf{x}_*)) \tag{3.11}$$

where $\mathbf{A} = \sigma_n^{-2}\Phi\Phi^T + \Sigma_p^{-1}$. To make predictions on a new sample we will need to invert the new $N \times N$ matrix $\mathbf{A}$. Since this can be problematic for large N, we rely on a result called the matrix inversion lemma [10].

3.1.1 Matrix Inversion Lemma

Consider a general partitioned matrix $\mathbf{M} = \begin{pmatrix} \mathbf{E} & \mathbf{F} \\ \mathbf{G} & \mathbf{H} \end{pmatrix}$, where we assume $\mathbf{E}$ and $\mathbf{H}$ are invertible. We have,

$$\begin{aligned} (\mathbf{E} - \mathbf{F}\mathbf{H}^{-1}\mathbf{G})^{-1} &= \mathbf{E}^{-1} + \mathbf{E}^{-1}\mathbf{F}(\mathbf{H} - \mathbf{G}\mathbf{E}^{-\mathbf{1}}\mathbf{F})^{-1}\mathbf{G}\mathbf{E}^{-1} \\ (\mathbf{E} - \mathbf{F}\mathbf{H}^{-1}\mathbf{G})^{-1}\mathbf{F}\mathbf{H}^{-1} &= \mathbf{E}^{-1}\mathbf{F}(\mathbf{H} - \mathbf{G}\mathbf{E}^{-\mathbf{1}}\mathbf{F})^{-1} \end{aligned} \tag{3.12}$$

Using this lemma, we can rewrite the predictive posterior as [13]:

$$\begin{aligned} &f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y} \sim \mathcal{N}(\phi(\mathbf{x}_*)^T\mathbf{\Sigma}_p\Phi(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{y}, \\ &\phi(\mathbf{x}_*)^T\mathbf{\Sigma}_p\phi(\mathbf{x}_*) - \phi(\mathbf{x}_*)\mathbf{\Sigma}_p\Phi((\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\Phi^T\Sigma_p\phi_*), \end{aligned} \tag{3.13}$$

where $K = \Phi^T\Sigma_p\Phi$.

3.2 Kernels

What we still do not know here is the kind of mapping $\phi(.)$ has to be or what the properties of the high dimensional space that we are mapping to are. Moreover, there is no guarantee that $\phi(.)$ does not map to infinite dimensions. To address these questions, we will need two results: Mercer's theorem[1] and the Representer Theorem[15].

3.2.1 Mercer's Theorem

Given a function $K(\mathbf{x}, \mathbf{x}') : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ such that

$$\iint_{\mathbb{R}^D \times \mathbb{R}^D} g(\mathbf{x}) K(\mathbf{x}, \mathbf{x}') g(\mathbf{x}') \mathrm{d}\mathbf{x} \mathrm{d}\mathbf{x}' \geq 0, \quad \forall g(\mathbf{x}) : \int g^2(\mathbf{x}) d\mathbf{x} \leq \infty \tag{3.14}$$

then, a function $\phi(.)$ exists that maps all $\mathbf{x}$ into a Hilbert space H, i.e., $\mathbf{x} : \mathbb{R}^D \to \phi(\mathbf{x}) : H$, whose inner product is defined as:

$$\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle = K(\mathbf{x}, \mathbf{x}') \tag{3.15}$$

3.2.2 Representer Theorem

Assume that we are given a K that has a corresponding $\phi(.)$ according to Mercer's theorem. Then, if the $\mathbf{w}$ that we need to find in the Hilbert space H is solved for by optimizing a function of the form:

$$\min_{\mathbf{w}} E[|\mathbf{y} - \mathbf{e}|] + \Omega(|\mathbf{w}|), \tag{3.16}$$

where $\Omega(.)$ is a non-decreasing function, the solution will be of the form:

$$\mathbf{w}^* = \sum_i \alpha_i \phi(\mathbf{x}_i) \tag{3.17}$$

Now, in Gaussian process our objective is to maximize the posterior distribution,i.e.,

$$\max P(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \max P(\mathbf{y}|\mathbf{X}, \mathbf{w}) P(\mathbf{w}) \tag{3.18}$$

Taking the log of the above expression we get:

$$\max \log P(\mathbf{y}|\mathbf{X}, \mathbf{w}) + \log P(\mathbf{w}). \tag{3.19}$$

Note that both the terms in the summation are logarithms of Gaussian distributions. Solving this further,

$$\begin{aligned} &\max \log P(\mathbf{y}|\mathbf{X}, \mathbf{w}) + \log P(\mathbf{w}) \\ =&\max -|\mathbf{y} - \mathbf{w^T X}|^2 - \mathbf{w^T \Sigma^{-1} w} \\ =&\min |\mathbf{y} - \mathbf{w^T X}|^2 + \mathbf{w^T \Sigma^{-1} w} \end{aligned} \tag{3.20}$$

Using the Representer theorem we can say that the solution will be of the form

$$\mathbf{w} = \sum_i \alpha_i \phi(\mathbf{x_i}) \tag{3.21}$$

Substituting this back into Equation 3.10, we get

$$\begin{aligned} &\mathbf{y} = \mathbf{w^T} \phi(\mathbf{X}) + \epsilon \\ \implies &\mathbf{y} = \sum_i \alpha_i \phi^T(\mathbf{x_i}) \phi(\mathbf{X}) + \epsilon \\ \implies &\mathbf{y} = \sum_i K(\mathbf{x_i}, \mathbf{X}) + \epsilon \end{aligned} \tag{3.22}$$

The last expression is a result of applying Mercer's theorem. The function K is called a kernel. Using kernels makes the use of non-linear models computationally tractable.

Having familiarized ourselves with Gaussian processes, let us shift our perspective and approach them from a different point of view.

3.3 Gaussian Processes – Functional Space Perspective

The formal definition of a Gaussian process is:

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

Thus, Gaussian processes generalize multivariate Gaussians, with each random variable being Gaussian distributed. Multivariate Gaussians are characterized by their mean and covariance function. For some real process $f(\mathbf{x})$, these functions are defined as [13]:

$$m(\mathbf{x}) = E[f(\mathbf{x})], \text{ and } k(\mathbf{x}, \mathbf{x}') = E[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \tag{3.23}$$

Thus, $f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$.
For some finite number of variables N, the multivariate Gaussian has a mean vector belonging to $\mathbb{R}^N$ and a covariance function that corresponds to a matrix of size $N \times N$. But consider that instead of taking a finite set of points $\mathbf{X}$, we now take all the points in that vector space. If we take the target set $\mathbf{Y}$ corresponding to this new set $\mathbf{X}$, we find that $\mathbf{X}$ and $\mathbf{Y}$ contain all the possible values that could ever be given to us as data.Then the Gaussian will be infinite dimensional. Drawing a curve through all the data points will give us a function that is the true function which explains our data. So essentially, we are placing a distribution over the functions that could all possibly explain the data.

But we can never work with infinite data or dimensions in real life. Fortunately, Gaussian distributions have two properties that allow us to circumvent this problem [9] :

1. Marginalizing a multivariate Gaussian gives a Gaussian.

2. Conditioning a Gaussian results in a Gaussian.

The first property tells us that if we marginalize a larger set of random variables, we can study the distribution of a smaller number of remaining random variables because their distribution will not change. Thus, if we split all our random variables into two mutually exclusive subsets A, B, we can write their joint Gaussian distribution as [9]:

$$P_{A,B} = [A, B] \sim \mathcal{N}(\mu, \mathbf{\Sigma}) = N\left([\mu_{\mathbf{A}}\mu_{\mathbf{B}}], \begin{bmatrix} \mathbf{\Sigma_{AA}} & \mathbf{\Sigma_{AB}} \\ \mathbf{\Sigma_{BA}} & \mathbf{\Sigma_{BB}} \end{bmatrix}\right) \tag{3.24}$$

Using the marginalization property, we can get rid of subset B completely when we only want to examine A, and rewrite the distribution for A as $A \sim N(\mu_A, \Sigma_{AA})$, and similarly for B. This means that all those infinite variables that appeared before can be safely dropped by marginalization.

We already saw the second property in action when discussing the linear model. To put it in terms of subsets of random variables A and B we say [9]:

$$A|B \sim \mathcal{N}(\mu_{\mathbf{A}} + \mathbf{\Sigma_{AB}\Sigma_{BB}^{-1}}(B - \mu_{\mathbf{B}}), \mathbf{\Sigma_{AA}} - \mathbf{\Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}}) \tag{3.25}$$

This means that if B were the training data and A the test data, we will force our candidate functions to pass through all the points in B. Also, when we condition, it is only the mean that depends on the variables that we conditioned on. The covariance matrix is independent.

Now we can easily retrieve our linear model from this. Recall that the weight vector $\mathbf{w}$ had a prior distribution $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$. Assuming that the inputs are mapped to a higher dimensional feature space using a function $\mathbf{\Phi}(.)$, we can write the mean and covariance for our model $f(\mathbf{x}) = \mathbf{\Phi}(\mathbf{x})^{\mathbf{T}}\mathbf{w}$ as [13]:

$$\begin{aligned} &E[f(\mathbf{x})] = \mathbf{\Phi}(\mathbf{x})^{\mathbf{T}}E[\mathbf{w}] = \mathbf{0}, \text{because} E[\mathbf{w}] \text{is assumed to be } \mathbf{0}; \\ &E[f(\mathbf{x})f(\mathbf{x}')] = \mathbf{\Phi}(\mathbf{x})^{T}E[\mathbf{w}\mathbf{w}^{\mathbf{T}}]\mathbf{\Phi}(\mathbf{x}') = \mathbf{\Phi}(\mathbf{x})^{T}\mathbf{\Sigma}\mathbf{\Phi}(\mathbf{x}') \end{aligned} \tag{3.26}$$

Looking at the covariance function carefully we can see that it is the dot product of two vectors $\mathbf{\Phi}(\mathbf{x})$ and $\mathbf{\Phi}(\mathbf{x}')$, one of which is also acted upon by a linear operator $\mathbf{\Sigma}$. This is again some kernel function K that we use as our covariance function.

Dot products can be thought of as a measure of similarity between vectors. Thus, if we decide to take a kernel that computes some similarity measure between two

given inputs, we have a covariance function that immediately decides the Gaussian distribution we start with. This choice of the kernel can play an important role in the success of our model, and people rely on their domain expertise to decide which kernel to use based on the source of the data.

Now, let's include back the noise that we originally had assumed was present in our observations $\mathbf{Y}$. Since the noise was also assumed to be an independent, and identically distributed Gaussian, and was being added to $f(\mathbf{x})$, we can write:

$$\text{cov}(y) = K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}. \tag{3.27}$$

The prior joint distribution of the given $\mathbf{Y}$ and the possible $f_*(.)$ for all given $\mathbf{x}$ can be written as [13]:

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(\mathbf{X}, \mathbf{X} + \sigma_n^2 \mathbf{I} & K(\mathbf{X}, \mathbf{X}_*) \\ K(\mathbf{X}_*, \mathbf{X}) & K(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}\right) \tag{3.28}$$

Conditioning on the given $\mathbf{X}, y,$ and $\mathbf{X}_*$ we get:

$$\mathbf{f}_* | \mathbf{X}, y, \mathbf{X}_* \sim \mathcal{N}(\overline{\mathbf{f}_*}, \text{cov}(\mathbf{f}_*)) \tag{3.29}$$

with the definitions being,

$$\overline{\mathbf{f}_*} = E[\mathbf{f}_* | \mathbf{X}, \mathbf{y}, \mathbf{X}_*] = K(\mathbf{X}_*, \mathbf{X})[K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}]^{-1} K(\mathbf{X}, \mathbf{X}_*) \tag{3.30}$$

$$\text{cov}(\mathbf{f}_*) = K(\mathbf{X}_*, \mathbf{X}_*) - K(\mathbf{X}_*, \mathbf{X})[K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}]^{-1} K(\mathbf{X}, \mathbf{X}_*) \tag{3.31}$$

These describe the posterior distribution that we need for making predictions.

3.4 Multitask Gaussian Process

Consider that for a given input set of $\mathbf{x_i} \in \mathbb{R}^n$, for $i = \{1, 2, \cdots, m\}$ the targets values are $\mathbf{y_i} \in \mathbb{R}^t$, for $i = \{1, 2, \cdots, m\}$. This means that using an input sample $\mathbf{x}$

we are predicting the values of t functions $f_j,\ j = \{1, 2, \cdots, t\}$, when each of those functions is given the $\mathbf{x}$ as an input. The linear model defined previously for some mapping $\mathbf{\Phi}(.)$ of the inputs becomes:

$$\mathbf{y_i} = \mathbf{W^T \Phi}(\mathbf{x_i}) + \epsilon_\mathbf{i} \tag{3.32}$$

where the $\mathbf{W}$ is a matrix of weight vectors, with one weight vector for each of the function values we are trying to predict, and $\epsilon_\mathbf{i} \in \mathbb{R}^t$ is the noise. To use Gaussian process regression in such a setting, it is assumed that correlations exists between the tasks, and the final covariance function is modeled as a Kronecker product of the covariance between tasks and the covariance of the samples. For a pair of inputs $\mathbf{x}, \mathbf{x}'$, the covariance between tasks i, j is defined as [20]:

$$\langle f_i(\mathbf{x}) f_j(\mathbf{x}') \rangle = K^f_{ij} k(\mathbf{x}, \mathbf{x}'), \tag{3.33}$$

where K^f is a positive semi-definite matrix representing inter-task correlations and k is the regular kernel function defining covariance between the inputs. Also, the distribution over any task i for the input $\mathbf{x_j}$ can be written as [20]:

$$y_{ji} \sim \mathcal{N}(f_i(\mathbf{x_j}), \sigma_i^2), \tag{3.34}$$

where σ_i^2 is the noise variance for the i^{th} task.
This joint Gaussian distribution over $\mathbf{y}$ has the property that observations of one task can affect the predictions on another task. Putting it all together, we can write out the inference equations for a test point $\mathbf{x}_*$ as [20]:

$$\begin{aligned} \overline{f_i}(\mathbf{x}_*) &= (\mathbf{k_i^f} \otimes \mathbf{k_*^{x^T}}) \mathbf{\Sigma}^{-1} \mathbf{y} \\ \mathbf{\Sigma} &= \mathbf{K^f} \otimes \mathbf{K^x} + \mathbf{D} \otimes \mathbf{I} \end{aligned} \tag{3.35}$$

where $\overline{f_i}$ is the mean of the prediction for the test input, $\mathbf{K^x}$ a matrix representing covariances of all training data, D is a diagonal matrix with the (i,i) element $= \sigma_i^2$.

3.5 Training a multi-task Gaussian process

The task of training a Gaussian process of any type boils down to finding optimal values for the parameters of the kernel, also known as the hyperparameters. To understand how this is done, consider again the equation for the posterior distribution:

$$P(\mathbf{w}|\mathbf{y}, \mathbf{X}, \theta_\mathbf{k}) = \frac{P(\mathbf{y}|\mathbf{X}, \mathbf{w}, \theta_\mathbf{k})P(\mathbf{w}|\theta_\mathbf{k})}{P(\mathbf{y}|\mathbf{X}, \theta_\mathbf{k})}. \tag{3.36}$$

Here the θ_k is a vector containing all the hyperparameters of the kernel. If we assume that the hyperparameters also have some distribution, their posterior w.r.t the data can be written as:

$$P(\theta_\mathbf{k}|\mathbf{y}, \mathbf{X}) = \frac{P(\mathbf{y}|\mathbf{X}, \theta_\mathbf{k})P(\theta_\mathbf{k})}{P(\mathbf{y}|\mathbf{X})}. \tag{3.37}$$

Notice that the marginal likelihood from (3.36) became the likelihood in (3.37). It can be hard to optimize posterior distribution of the hyperparameters directly. A commonly adopted strategy is to maximize the marginal likelihood of (3.36). [13].

For the multi-task Gaussian process, maximizing the marginal likelihood can be done using gradient based optimization. Computing the gradients directly can get expensive. An alternative is to use an algorithm based on expectation maximization(EM). A practical EM algorithm is described in [20] and is reproduced here.

A note on notation: $\mathbf{F}, \mathbf{Y}$ represent the matrix of values corresponding to $\mathbf{f}, \mathbf{y}$ respectively. Also, $\mathbf{y_{.j}}$ represents the j^{th} column of $\mathbf{Y}$ for all the inputs m. Similarly for $\mathbf{f}$. So $\mathbf{Y}, \mathbf{F}$ are both m x t matrices. $\theta_\mathbf{x}$ is the vector of parameters for $\mathbf{k_x}$. The complete-data log likelihood is:

$$\begin{aligned} L_{\text{comp}} = &- \frac{m}{2}\log|\mathbf{K^f}| - \frac{t}{2}\log|\mathbf{K^x}| - \frac{1}{2}\operatorname{Tr}\left[(\mathbf{K^f})^{-1}\mathbf{F^T}(\mathbf{K^x})^{-1}\mathbf{F}\right] \\ &- \frac{m}{2}\sum_{i=1}^{t}\log\sigma_i^2 - \frac{1}{2}\operatorname{Tr}\left[(\mathbf{Y}-\mathbf{F})\mathbf{D^{-1}}(\mathbf{Y}-\mathbf{F})^\mathbf{T}\right] - \frac{mt}{2}\log 2\pi \end{aligned} \tag{3.38}$$

From this, the update steps are given by($\wedge$ represents the updated value):

$$\begin{aligned}
&\hat{\theta}_x = \underset{\boldsymbol{\theta}_\mathbf{x}}{\operatorname{argmin}}(m \log \left|\left\langle \mathbf{F^T}(\mathbf{K^x}(\theta_\mathbf{x}))^{-\mathbf{1}}\mathbf{F}\right\rangle\right| + t \log |\mathbf{K^x}(\theta_\mathbf{x})|), \\
&\hat{\mathbf{K}}^\mathbf{f} = \frac{1}{m}\left\langle \mathbf{F^T}(\mathbf{K^x}(\hat{\theta}_\mathbf{x}))^{-\mathbf{1}}\right\rangle \qquad \hat{\sigma}_l^2 = \frac{1}{m}\left\langle (\mathbf{y_{.j}} - \mathbf{f_{.j}})^T(\mathbf{y_{.j}} - \mathbf{f_{.j}})\right\rangle .
\end{aligned} \tag{3.39}$$

Chapter 4

Methodology

4.1 Methodology

The full procedure that will be used to finally detect anomalies is this:

1. Train a multi-task Gaussian Process model on the voltage data to predict the corresponding current values.

2. Test the model on the training set and the test set.

3. Compute the prediction error. If the true current sample is $\mathbf{i_t}$ and the predicted current sample value is $\mathbf{i_p}$, the error is given as $\mathbf{e} = \mathbf{i_t} - \mathbf{i_p}$.

4. Whiten each error vector $\mathbf{e}$ using the predictive covariance matrix $\mathbf{V}$ associated with the input sample. Whitening is done as ($\mathbf{e}$ is assumed to be a column vector here):

$$\begin{aligned} &\mathbf{e_w} = \mathbf{We}, \text{ where} \\ &\mathbf{WW^T} = \mathbf{V^{-1}}. \end{aligned} \tag{4.1}$$

Whitening has the effect of de-correlating the values across each dimension. What this means is that after whitening, the prediction errors will form a multivariate Gaussian whose covariance matrix is the identity matrix.

5. Train the one-class support vector machine (OC-SVM) on the whitened prediction errors generated by testing the MTGP on the training set. Make the OC-SVM classify the whitened test set prediction errors and this becomes the decision for the corresponding voltage sample too.

A note on implementation: the multi-task Gaussian process is implement using the library GPyTorch with a linear kernel [8]. The one-class SVM is from scikit-learn, and uses a radial basis function kernel.

4.2 Data

A smart grid with bus type IEEE 123 was simulated in MATLAB and a time series data was generated. This grid operates at a voltage of 4.16kV. The data contained 5.4 million samples with 11 features: time , V_1, V_2, V_3, I_1, I_2, I_3, breaker status, fault impedance, fault location, fault type. Each sample was generated with a gap of 5 seconds.

The Vs and Is represent line voltages and currents respectively. There are three types of faults and the last feature takes values {0,1,2,3}, with 0 being the no-fault case. Of these, the most commonly occurring fault in this particular simulation is of type-1 (80%), followed by type-2 (15%) and type-3 (5%). Every time a fault occurs, the fault impedance value varies between 0.1 Ω and 5 Ω randomly.

The grid is divided into 10 relays and each relay has 4 configurations. There is a different dataset for each combination of relay and configuration. In total, there are 40 datasets.

4.2.1 Data Preprocessing

The objective is to use the 3 voltage values to predict the corresponding 3 current values and then use the error from the prediction to classify the voltage samples as normal or anomalous. To achieve this, we first need to construct a training and test dataset, while making sure that these sets do not contain any common data points. It was decided to use the first 2.7 million samples to construct the training set and the next 2.7 million samples for the test set.

Constructing the training set

The training set was constructed in the following way:

1. Discard samples that contain NaNs.
2. Take the voltage and current values only. A single voltage sample has 3 values (features) and its corresponding current value has 3 values (features).
3. From the first 2.7 million samples pick 7000 normal samples at random.

Constructing the test set

The test set is simpler to construct:

1. Discard all samples with NaNs in them.
2. Split the samples into two sets containing just the voltage and current values respectively.

Thus the test set will have almost all of the 2.7 million samples, except for those discarded due to NaNs.

4.3 Results

The performance of the anomaly detector is measured by checking if for a high probability of detecting an anomaly, there is a small probability of false alarm. Before looking at how PD and PFA are calculated, some nomenclature:

1. True Positive (TP) = Count of anomaly samples correctly classified as anomaly.
2. False Positives (FP) = Count of normal samples incorrectly classified as an anomaly.
3. False Negative (FN) = Count of anomaly samples incorrectly classified as normal.
4. True Negative (TN) = Count of normal samples correctly classified as normal.

The two quantities are calculated as:

$$\begin{aligned} PD &= TP/(TP + FN) \\ PFA &= FP/(FP + TN) \end{aligned} \tag{4.2}$$

4.3.1 PD vs PFA plots

Plots of probability of detection (PD) vs probability of false alarm (PFA) when detecting anomalies in a window of 30,000 samples, for all 4 configurations of one relay are given next. Here, the ν value was fixed to 0.001 and the decision threshold varied.

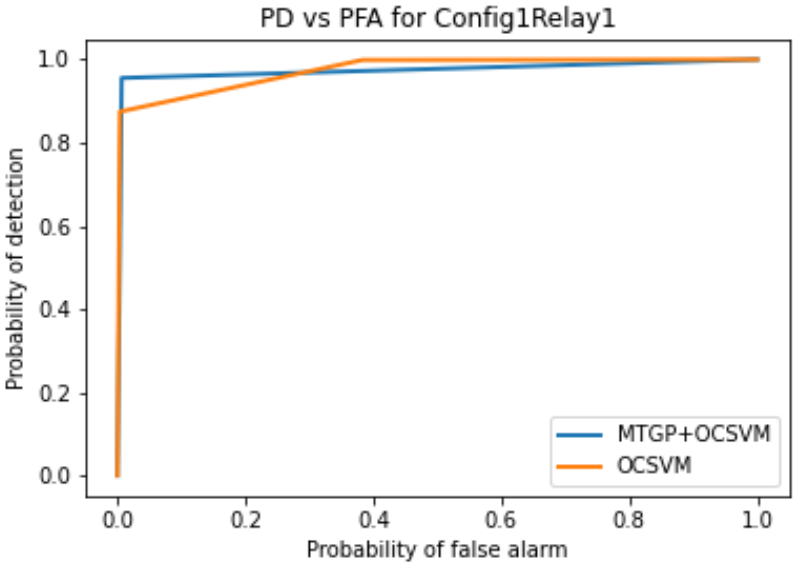

Figure 4.1: Relay1Config1-PD vs PFA

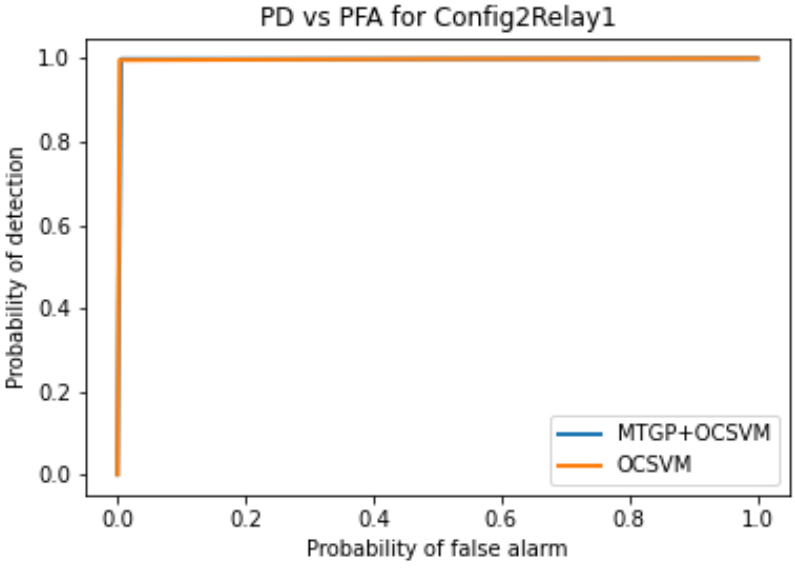

Figure 4.2: Relay1Config2-PD vs PFA

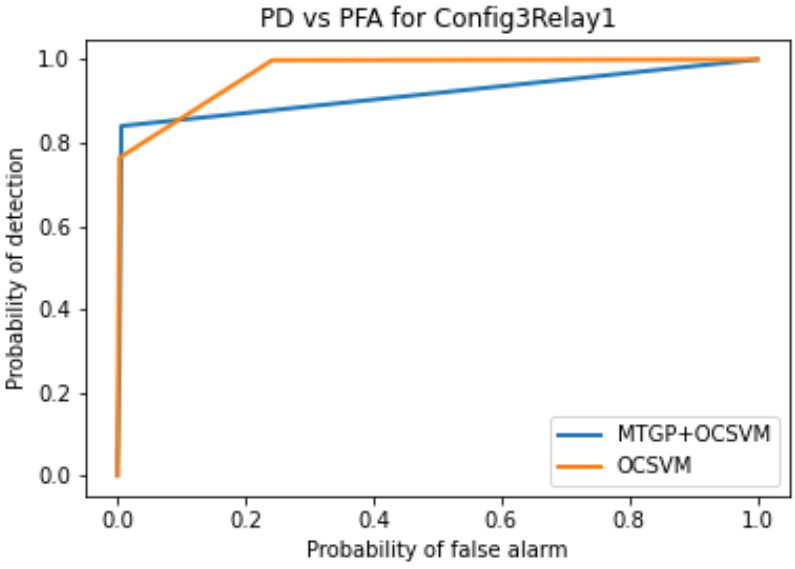

Figure 4.3: Relay1Config3-PD vs PFA

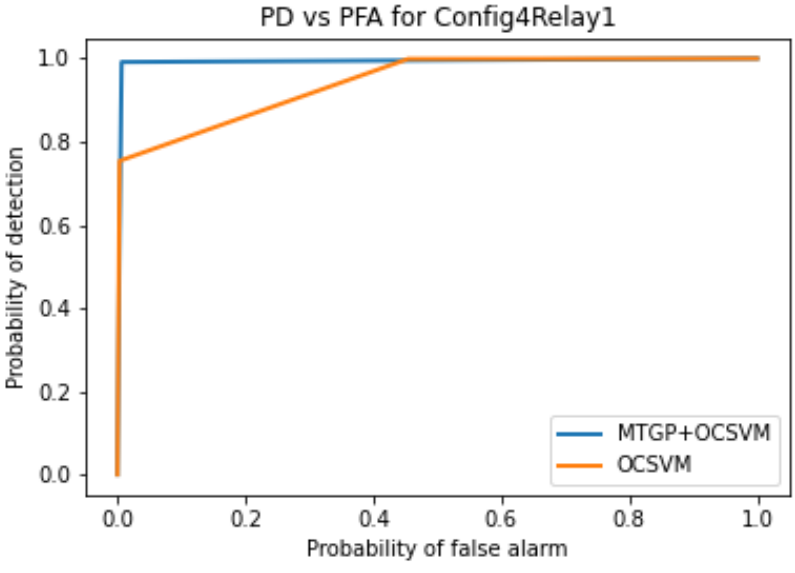

Figure 4.4: Relay1Config4-PD vs PFA

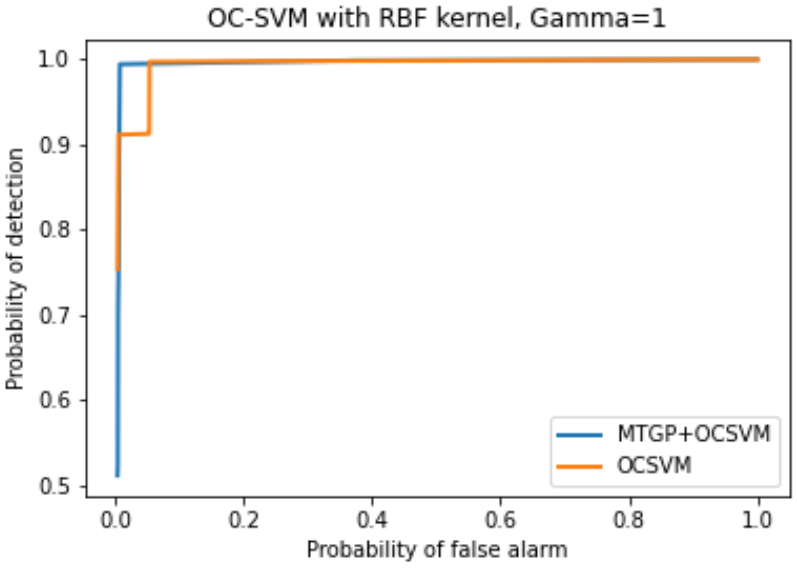

Figure 4.5: Relay1Config4-PD vs PFA - Zoomed in

4.3.2 Algorithm performance comparison

The MTGP+OC-SVM model is compared against the simple OC-SVM, where the OC-SVM is given the test data directly (after normalizing with respect to the training data). The performance is given in the table below:

	$\nu = 10^{-3}, \gamma = 1$				Cross Validated ν and γ	
Configuration	OC-SVM		MTGP+OC-SVM		OC-SVM	
	1-PD	PFA	1-PD	PFA	1-PD	PFA
1	1×10^{-3}	5.7×10^{-1}	1.5×10^{-2}	6×10^{-3}	6×10^{-3}	4×10^{-3}
2	1×10^{-3}	3.5×10^{-1}	7×10^{-3}	5×10^{-3}	8×10^{-3}	3×10^{-3}
3	2×10^{-3}	2.4×10^{-1}	1.3×10^{-1}	6×10^{-3}	5.6×10^{-2}	3×10^{-3}
4	2×10^{-3}	4.5×10^{-1}	4.4×10^{-2}	6×10^{-3}	7×10^{-3}	3×10^{-3}

Table 4.1: Performance Comparison on Relay1

Upon cross-validation, $\nu = 0.006, \gamma = 0.006$ give the best performance for OC-SVM.

Chapter 5

Conclusion

This thesis introduces the idea of using multi-task Gaussian process for unsupervised anomaly detection in smart-grids. The MTGP model is used to make predictions about the sensor data from smart-grids and the prediction errors are computed and transformed into a spherical Gaussian, through a process called whitening. These whitened prediction errors are then given to a one-class SVM for anomaly detection.

An important requirement is that the rate of false alarms be low because it is expensive to repsond to a false alarm in these systems. When using a simple OC-SVM model, low false alarm rates can be achieved using cross-validation of the ν and γ parameters. But in an unsupervised setting, cross-validation is not possible. This is where the advantage of using a multi-task Gaussian process comes in. It allows us to whiten the prediction errors which means that they are transformed to resemble a spherical Gaussian with variance 1. Since we are using a Gaussian kernel with the one-class SVM, the boundary is formed by combining univariate Gaussians. Setting the γ to a similar scale as the variance of the data allows for a smooth boundary. Too low a value and the boundary becomes noisy while too high a value leads to an excessively smooth boundary. Which is why it is reasonable to set $\gamma = 1$ and expect close to optimal performance. Since we made the assumption that the anomalies are

events of low probability, we can set $\nu = 0.001$ for both cases - MTGP+OC-SVM and simple OC-SVM.

Table 4.1 shows that without the help of cross-validation, the MTGP+OC-SVM model performs better than the simple OC-SVM. When cross-validtion is sued to find the best possible performance of the OC-SVM, it becomes just as good as the MTGP+OC-SVM, with better detection rates. But since the detection rates and the false alarm rates are comparable for both models, it proves that in a truly unsupervised setting the multi-task Gaussian process can be used.

The next steps will be to try different kernels for the multi-task Gaussian process and compare against the performance of the one-class SVM. Along with this, the entire dataset will also be tested upon for completeness.

References

[1] M. A. Aizerman, E. A. Braverman, and L. Rozonoer, *Theoretical foundations of the potential function method in pattern recognition learning.*, Automation and Remote Control,, Automation and Remote Control,, no. 25, 1964, pp. 821–837.

[2] Christian Andre Andresen, Bendik Nybakk Torsæter, Hallvar Haugdal, and Kjetil Uhlen, *Fault detection and prediction in smart grids*, 2018 IEEE 9th International Workshop on Applied Measurements for Power Systems (AMPS), IEEE, 2018, pp. 1–6.

[3] Avrim Blum, John Hopcroft, and Ravindran Kannan, *Foundations of data science*, Cambridge University Press, 2020.

[4] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander, *Lof: identifying density-based local outliers*, Proceedings of the 2000 ACM SIGMOD international conference on Management of data, 2000, pp. 93–104.

[5] Raghavendra Chalapathy and Sanjay Chawla, *Deep learning for anomaly detection: A survey*, arXiv preprint arXiv:1901.03407 (2019).

[6] Varun Chandola, Arindam Banerjee, and Vipin Kumar, *Anomaly detection: A survey*, ACM computing surveys (CSUR) **41** (2009), no. 3, 1–58.

[7] David Andrew Clifton, Samuel Hugueny, and Lionel Tarassenko, *Novelty detection with multivariate extreme value statistics*, Journal of signal processing systems **65** (2011), no. 3, 371–389.

[8] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson, *Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration*, Advances in Neural Information Processing Systems, 2018.

[9] Jochen Görtler, Rebecca Kehlbeck, and Oliver Deussen, *A visual exploration of gaussian processes*, Distill (2019), https://distill.pub/2019/visual-exploration-gaussian-processes.

[10] Kevin P Murphy, *Machine learning: a probabilistic perspective*, MIT press, 2012.

[11] Emanuel Parzen, *On estimation of a probability density function and mode*, The annals of mathematical statistics **33** (1962), no. 3, 1065–1076.

[12] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko, *A review of novelty detection*, Signal Processing **99** (2014), 215–249.

[13] Carl Edward Rasmussen and Christopher K. I. Williams, *Gaussian processes for machine learning (adaptive computation and machine learning)*, The MIT Press, 2005.

[14] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller, *A unifying review of deep and shallow anomaly detection*, Proceedings of the IEEE (2021).

[15] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola, *A generalized representer theorem*, International conference on computational learning theory, Springer, 2001, pp. 416–426.

[16] Bernhard Schölkopf, Robert C Williamson, Alexander J Smola, John Shawe-Taylor, John C Platt, et al., *Support vector method for novelty detection.*, NIPS, vol. 12, Citeseer, 1999, pp. 582–588.

[17] Shai Shalev-Shwartz and Shai Ben-David, *Understanding machine learning: From theory to algorithms*, Cambridge university press, 2014.

[18] Aleksandar M Stanisavljević, Vladimir A Katić, Boris P Dumnić, and Bane P Popadić, *A brief overview of the distribution test grids with a distributed generation inclusion case study*, Serbian Journal of Electrical Engineering **15** (2018), no. 1, 115–129.

[19] David MJ Tax and Robert PW Duin, *Support vector data description*, Machine learning **54** (2004), no. 1, 45–66.

[20] Chris Williams, Edwin V Bonilla, and Kian M Chai, *Multi-task gaussian process prediction*, Advances in neural information processing systems (2007), 153–160.