# ANOMALY DETECTION IN TIME SERIES

by

Onur Poyraz

B.S., Electrical and Electronics Engineering, Boğaziçi University, 2016

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computational Science and Engineering
Boğaziçi University

2019

# ACKNOWLEDGEMENTS

I would like to thank my supervisor Prof. Ali Taylan Cemgil for his support and mentorship. I feel genuinely privileged to study under his supervision. It was an honor for me to work with him. I also received a lot of support from my instructors in my graduate years. I especially want to thank Dr. Emre Uğur, and Prof. Cem Say. I would also like to thank Dr. Suzan Üsküdarlı for her cheerful approach.

I have spent incredible years in my graduate study. For this, first of all, I should thank my friends in our lab. I feel lucky to be able to work with the members of PILAB Sigma, and I am grateful for this. Especially I thank my old friend and my fate partner Semih Akbayrak, Burak Kurutmaz who was once my business partner and will be my 'brö' forever, my great friend Serkan Buğur, my lovely friend Merve Ünlü, and finally my knight brother Gökhan Çapan, for their excellent support on me. I also want to thank Hakan Kalaycı, Caner Türkmen, Çağlar Hızlı, Özge Bozal, Melih Barsbey, Serhan Daniş, Çağrı Sofuoğlu, İlker Gündoğdu and Burak Suyunu for their friendship, great academic discussions, and joyful coffee breaks. Finally, I thank Mine Öğretir for everything she has brought me and for her great support. They all made me love the department.

On the other side, I want to thank my family. I would like to express my deepest gratitude for the opportunity they provided and their endless moral support. I wouldn't finish my graduate studies without them.

Finally, I want to thank Borusan Arge and Bankalararası Kart Merkezi (BKM) for their data support. I participated in their projects, and it was a pleasure to work with them.

# ABSTRACT

## ANOMALY DETECTION IN TIME SERIES

Anomaly detection (AD) is the discovery of the observations which does not conform with the rest of the observations. The types of anomalies and their occurrences that exist in the data set are tried to be determined. On the other hand, time series structures have dynamic structures, which are evolving over time, and in such structures, observations will be affected by previous observations. This thesis focuses on the anomaly detection process under time series structures. This problem is not always straightforward because the definition of anomaly could change with the context of the dynamic structure and anomaly detection process in the system could interfere with the intense noises at the observations.

In this thesis, we try to identify anomalies in the sub-sequences of the streaming data. When doing so, we also want to discriminate the anomalies in the system with the faulty observations. Therefore we investigate collective anomalies in the data. We propose both statistical inference methods and deep learning approaches for such type of anomaly detection in time series (ADTS) problem. We use a Gaussian mixture model (GMM) and a customized hidden Markov model (HMM) as statistical approaches, while we use Recurrent Neural Networks (RNN) and Long Short-Term Memories (LSTM) as deep learning approaches. Except for GMMs, we take into account the sequential structures of data sets in the models proposed above. We apply our methodologies to the Borusan wind turbines data and we compare the model results with the experiments we performed on this dataset.

# ÖZET

## ZAMAN SERİLERİNDE OLAĞANDIŞILIK SEZİMİ

Olağandışılık Sezimi (AD), mevcut veri kümesinin diğer gözlemleriyle örtüşmeyen gözlemlerin sezimlenmesidir. Olağandışılık türleri ile bunların veri kümesi içindeki oluşumları belirlenmeye çalışılır. Öte yandan, zaman serisi yapıları zaman içinde gelişen devingen yapılara sahiptir ve bu tür yapılarda gözlemler önceki gözlemlere bağlıdır. Bu tez, zaman serisi yapıları altındaki olağandışılık sezimi sürecine odaklanmaktadır. Bu problem her zaman kolay değildir, çünkü olağandışılığın tanımı, sürecin devingen yapısı bağlamında değişebilir ve sistem içerisindeki olağandışılık sezimi işlemi gözlemlerdeki yüksek gürültülerle karışabilir.

Bu tezde, akış verilerinin alt kümelerinde meydana gelene olağandışılıkları belirlemeye çalışıyoruz. Bunu yaparken, sistemdeki olağandışılıkları hatalı gözlemlerden ayırt etmek istiyoruz. Bu nedenle verilerdeki toplu olağandışılıkları araştırmaktayız. Bu özelliklere sahip olan zaman serisinde olağandışılık sezimi (ADTS) için hem istatistiksel çıkarım yöntemleri hem de derin öğrenme yaklaşımları öneriyoruz. Derin öğrenme yaklaşımları olarak Tekrarlayan Sinir Ağları (RNN) ve Uzun Kısa Süreli Bellek'i (LSTM) kullanırken, Gaussian karışım modelini (GMM) ve özelleştirilmiş bir gizli Markov modelini (HMM) istatistiksel yaklaşımlar olarak kullanıyoruz. GMM'ler hariç, yukarıda önerilen modellerde veri kümelerinin sıralı yapılarını dikkate alıyoruz. Yöntemlerimizi, Borusan rüzgar türbinleri verilerine uyguluyoruz ve model sonuçlarını bu veri kümesi üzerinde yaptığımız deneylerle karşılaştırıyoruz.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF SYMBOLS

| | |
|---|---|
| $\mathcal{BE}(.)$ | Bernoulli distribution |
| $\boldsymbol{C}$ | Cell state |
| $\mathcal{D}(.\|\|.)$ | Divargence metric |
| $\mathcal{D}_{KL}(.\|\|.)$ | Kullback-Leibler divergence |
| $\mathbb{E}_{p(.)}[.]$ | Expectation with respect to function $p(.)$ |
| $\mathcal{E}_t$ | Anomaly prediction for time $t$ |
| $\boldsymbol{\mathcal{L}}$ | Loss |
| $\mathcal{N}(.)$ | Gaussian distribution |
| $p(.)$ | Probability distribution function |
| $p$ | Neural network input parameters at time $t$ |
| $\mathcal{P}$ | Input dimension of NN |
| $q(.)$ | Variational distribution function |
| $\mathcal{Q}(.,.)$ | Energy function between two variables |
| $r$ | Neural network output parameters at time $t$ |
| $\hat{r}$ | Neural network predictions at time $t$ |
| $\mathcal{R}$ | Output dimension of NN |
| $s_t$ | Hidden variable at time $t$ |
| $s_{1:\mathcal{T}}$, $\boldsymbol{s}$ | Set of hidden variables |
| $S$ | Hidden state dimension |
| $\mathcal{T}$ | Number of time slices |
| $\boldsymbol{\mathcal{W}}$ | Weight matrix of the neural network |
| $x_t$ | Observed variable at time $t$ |
| $x_{1:\mathcal{T}}$ , $\boldsymbol{x}$ | Set of observations |
| $y_t$ | Result at time $t$ |
| $y_{1:\mathcal{T}}$, $\boldsymbol{y}$ | Set of results |
| $\hat{y}_t$ | Predicted variable at time $t$ |
| $\hat{y}_{1:\mathcal{T}}$, $\hat{\boldsymbol{y}}$ | Set of predictions |

| | |
|---|---|
| $\alpha(s_t)$ | Forward recursion |
| $\beta(s_t)$ | Backward recursion |
| $\gamma(s_t)$ | Correction smoother |
| $\eta$ | Learning rate |
| $\Theta$ | Parameter set |
| $\mu$ | Mean |
| $\boldsymbol{\pi}$ | Initial latent state parameter of hidden Markov Model |
| $\sigma(.)$ | Sigmoid Function |
| $\sigma$ | Variance |
| $\Sigma$ | High variance |
| $\phi(s_t)$ | Most likely hidden state sequence |
| $\tau$ | Time index |
| $\boldsymbol{\Psi}$ | State transition matrix of hidden Markov Model |
| $\Psi_{\hat{s},\hat{s}'}$ | Probability of state transition from state $\hat{s}'$ to $\hat{s}$ |
| $\boldsymbol{\Omega}$ | Emission probability of hidden Markov Model |
| $\Omega_{\hat{x},\hat{s}}$ | Emission probability of observation $\hat{x}$ from given state $\hat{s}$ |

# LIST OF ACRONYMS/ABBREVIATIONS

| | |
|---|---|
| 2D | Two Dimensional |
| 3D | Three Dimensional |
| AD | Anomaly Detection |
| ADTS | Anomaly Detection in Time Series |
| ANN | Artificial Neural Networks |
| AR | Auto-Regressive |
| BPTT | Backpropagation Throuh Time |
| EM | Expectation-Maximization |
| GMM | Gaussian Mixture Model |
| HMM | Hidden Markov Model |
| KL | Kullback-Leibler |
| LDS | Linear Dynamical Systems |
| LSTM | Long-Short Term Memory |
| MAP | Maximum a Posteriori |
| MD | Multidimensional |
| MLE | Maximum Likelihood Estimation |
| MAD | Median Absolute Deviation |
| MAE | Mean Absolute Error |
| MAPE | Mean Absolute Percentage Error |
| MSE | Mean Square Error |
| NN | Neural Network |
| RMSE | Root Mean Square Error |
| RNN | Recurrent Neural Network |
| SMC | Sequential Monte Carlo |
| SSM | State-Space Models |

# 1. INTRODUCTION

Anomalies are patterns or items in observations that do not conform to a well-defined notion of rational behavior of the system [1]. In the literature, anomalies are also known as outliers, novelties, noise, deviations, and exceptions [2]. Identification of these anomalies in the defined context is known as *'anomaly detection'* (AD) [3]. The importance of anomaly detection is mainly due to the fact that anomalies usually contain critical and valuable information. For example, fraudulent credit card transactions [4], tumors in the brain MRIs [5] and heavy traffic on the network [6] are examples of anomalies and detection of them is vitally important. On the other hand, it may be equally important to describe the normal working behavior or pattern of any system, and so we need to identify the anomalies within the observations to extract such a pattern.

Anomaly detection is an omnipresent problem that has been researched in many different application areas. It is similar to *noise removal* [7] or *noise accommodation* [8] or *novelty detection* [9, 10] problems in the literature. All of these methods deal with unfamiliar observations in the data that do not conform to the usual pattern or distribution. In the *noise removal* and *noise accommodation* problem, these observations are undesirable and ineffective. On the other hand, in the *novelty detection* and *anomaly detection*, most of the valuable information and 'interestingness' of the data lies in the unfamiliar observations. *Novelty detection* methods have the purpose of incorporating the novel pattern into the regular model. In contrast to these, the primary goals of the *anomaly detection* are to find or detect the problematic observations which may indicate some malfunctions or defects and try to distinguish abnormal observations and patterns by learning standard patterns.

There are, basically, two types of anomaly detection. The first type of anomaly detection methods focuses on static structures, which do not change over time and are capable of representing only a single snapshot of observations [11]. However, real-world scenarios have dynamic structures, which are evolving over time. So, both normal be-

havior and definition of the anomaly may vary. Therefore, the second type of anomaly detection methods takes into account the dynamic structures of the observations [1]. Such methods evaluate observations under some 'context'. The structure of the system, that generates the observations, determines the notion of the context, which is a part of the problem. In literature, one of the essential contexts is the time series and called as *anomaly detection in time series* (ADTS). In the context of the time series, observations will be affected by previous observation(s) (i.e., dynamical structures). In ADTS, observations are evaluated under a context (i.e., time) and do not necessarily have to be an anomaly to be considered as anomalies. Such structures are named as *contextual anomaly* or *conditional anomaly* [12]. In such structures, one observation could be considered as an anomaly according to the position in the sequence, i.e., a high number of transactions on the weekend is usual while fewer transactions are expected in the week. In particular, in the context of the anomaly detection in time series the interesting objects are often not rare objects, but 'unexpected bursts' in the activity. Therefore, to detect abnormal movements in such a context, one should consider the time-series effect of the system. In this work, we are interested in such problems.

### 1.1. Related Work

The literature on anomaly detection is quite extensive. Excellent overviews of the anomaly detection methods are presented by [1, 2], which covers almost all anomaly detection literature. In other overviews, [13,14] investigates anomaly detection in time series problems. This two survey mention most of the current work on this subject. In another work, [15] studies the evaluation of the problem of anomaly detection. This work is essential since the evaluation of the AD is the omnipresent problem.

On the other side of this work, we have to review the time-series literature, which is also an extensive literature. In the surveys [16, 17], authors investigate the nature of time series data in detail. In the work [18], the authors mentioned the most popular time series forecasting techniques and analysis. In [19, 20], they give Bayesian treatment for the solution of time-series problem, and they give detail about the most well-known time series models, HMM [21] and AR model. More recently, in [22],

authors examine the deep learning approaches for unsupervised feature learning in the time series problems.

So far, we talked about more general perspectives. However, there are many applications related to ADTS. One of the most recent and well-known studies is [23]. In this study, a deep learning sequence model for ADTS is developed, and this work is applied to the *electro cardiogram* (ECG), space shuttle, power demand, and multi-sensor engine dataset. Similarly, ADTS models are used for credit card fraud detection [4, 24], industrial damage detection [25, 26] and attack detection in networks [27].

### 1.2. Outlooks of the Problem

In this section, we will cover the outlooks of anomaly detection frequently encountered in applications. Basically, it is possible to distinguish these points of view as anomaly definition, problem definition, data types, and availability of the labels.

Proper identification of the anomalies is crucial, and it is the first step in anomaly detection. There are mainly two definitions of an anomaly which vary according to the problem, outlined in the literature which are;

(i) *Point Anomalies.* Point anomaly refers to cases where abnormal observations occur individually [28]. The single observation could be considered as an anomaly concerning the rest of the concerning the rest of the observations or unusual according to the data stream. In this type of anomaly, one should define the boundary of typical region and observations which are different from the usual observations are called anomalies. Since the problem includes time as the context, the normal region should be determined according to this context.

(ii) *Collective Anomalies.* Collective anomaly refers to the sequential collection of abnormal data samples according to the complete data stream [1]. In this type, observations can be considered as anomalies together even though they may not be anomalous individually. In this context, both anomalies in observation intervals and changes in the generative model in the long term are examined.

(iii) *Contextual Anomalies.* Contextual anomalies refer to the evaluation of observation according to a specific context. If an observation instance is anomalous in that particular context (but not otherwise), then it is termed as a contextual anomaly (also referred to as conditional anomaly). In that approach, it is assumed that each observation has two sets of attributes; *Contextual attributes* and *Behavioral attributes*. These attributes define the characteristics of the observations.

Even though the definition of the anomalies is the primary objective, the description of the problems requires different approaches and solutions. In the literature, most of the research focus on three main problem definitions;

(i) *Sequence-based AD.* The primary purpose is to detect the abnormal sequences concerning data set, which consist of similar time series sequences. Since different sequences are compared, each sequence could be considered as *multi-dimensional* (MD) data that accepts each timestep as a dimension. Therefore classical anomaly detection methods can be valid in this type of problem.

(ii) *Subsequence-based AD.* The primary aim is to detect the unusual sub-sequence concerning the rest of the time series data set. The sub-sequences that does not fit the rest of the long sequence is searched. Therefore, most of the sequence is assumed to be healthy, and sub-sequences most different than to the rest of the sequence are considered as an anomaly. In this work, we usually deal with sub-sequence based AD problems.

(iii) *Pattern frequency based AD.* This approach is slightly different from previous methods. It takes into account the occurrence rates of the observations, and if an observation has not occurred in the correct time than it is considered as an anomaly.

It is essential to determine the nature of the input regardless of the problem because all systems operate according to specific inputs types and the nature of data determines both the model and methodology. Both inputs and observations could be

*binary*, *categorical* or *continuous* as a data type and could be *univariate* or *multivariate* according to data dimension. In addition to these, observations could be count data, or there may be other constraints on them. Furthermore, there are periodicity and synchronization features of time series data. Both the nature of the data and the constraints affect the models to be applied. Finally, one who will deal with ADTS problem should evaluate the relations between the observations [29]. In this work, we consider observations as a time series and divide them into three parts according to data types regarding time series effect;

(i) *Continuous streams.* Observations in which the flow occurs continuously. Observation indexes are defined within a finite or infinite interval.

(ii) *Discrete streams.* Observations are formed by event logs, so it is discrete sequences. Our work is based on discrete streams.

(iii) *Multi-dimensional streams.* Each observation is fed from multiple sources and includes more than one instance. According to data sources, it could be both *continuous stream* or *discrete stream* or the mixture of them.

One of another main outlook of the anomaly detection problem is the availability of the labeled data. In the literature, there are typically three classes of anomaly detection according to the availability of the data labels [1].

(i) *Supervised Anomaly Detection.* It requires a labeled dataset for each observation. Therefore the main problem becomes a classification problem. The only difference is that abnormal observations do not act as a class because of the unbalanced nature of anomalies.

(ii) *Semi-supervised Anomaly Detection.* It requires the knowledge of the ordinary behavior at the training set. In this technique, the usual working behavior of the system will be learned from the training data labeled as healthy. At the test phase, abnormal observations will be detected.

(iii) *Unsupervised Anomaly Detection.* The primary assumption of this technique is that one should suppose that there is a small amount of the abnormal observations

in both train and the test dataset. The main goal is to detect abnormal pattern rather than finding abnormal observations.

Labeling all observations in sequential data is costly. Therefore, in most cases, getting a labeled set of data is almost impossible. Moreover, if one of the system or anomalies have a dynamical structure, then a new class of patterns or anomalies can arise. It means that there could be no labeled data for a while.

### 1.3. Challenges

Anomaly detection problems are straightforward at the conceptual level. One should define the representation of the ordinary behavior or region of the observations and then declare any observation which does not comply with this definition as an anomaly. However, there are some challenges in the anomaly detection problem, which made the problem hard to solve.

(i) *Definition of normal behavior*: This problem is a crucial point for almost any machine learning problem. Especially in the unsupervised methods, observations include both normal and abnormal data, so one can not directly define the regular pattern of the system according to data. So one needs to detect anomalies to find the normal region and then determine the normal region to declare anomalies. Obviously, this problem is recursive and more complicated than it seems.

(ii) *Evolution of normal behavior*: In many domains, the definition of healthy behavior is evolving. In such cases, the previous definition of healthy behavior may not be sufficient to explain the new one. So, one has to take this phenomenon into account in anomaly detection application to accurately determine the anomaly.

(iii) *Definition of the anomaly*: Applications usually have a wide variety of nature and include a different kind of data. For example, an ECG, the periodicity of the observations is vital to detect anomalies while fraudulent transactions appear on single observations. On the other hand, an observation within a series could be continuous or discrete, and corresponding anomalies change form accordingly.

Additionally, one could investigate single point in series or some sub-sequence or whole sequence, and the desired anomaly affects the entire model. In addition to all these, sometimes evolution in the streaming observations could be an anomaly.

(iv) *Adversarial anomalies*: In most problems, the anomalies are not clearly visible. Furthermore, in some problems, adversarial anomalies are trying to hide within normal data (i.e., fraudulent transactions in the transaction records).

(v) *Noise*: In real-world applications, observations usually collected in a noisy environment. On the other hand, anomalies are also some kind of noise. Therefore, to develop successful anomaly detection method, one should separate the noise and anomalies.

(vi) *Availability of the labeled data*: Available labels are vital for almost any machine learning problem but especially in anomaly detection. One needs to define the typical behavior pattern to declare some observations as an anomaly. To evaluate the correctness of the model, one directly needs labeled data. Since the time series data is usually too large, the labels needed are often unreachable.

(vii) *Computational Complexity*: Time series data is usually broad and continues to expand. Also, because the observations are dependent on each other, the relationship between observations should be learned. Therefore, the ADTS problem requires a lot of computational power.

(viii) *Evaluation*: Almost all time series data is not adequately labeled. There may not even be a label in some of them. In some cases, there may not be a direct label, in which the given label could be only foreseen. Furthermore, sometimes anomalies in the system interfere with the intense noise. So, the evaluation of ADTS could change according to the problem and the application.

In ADTS problems, there is no general solution. Solutions depend on the outlooks of the problem that are mentioned, such as type of data, the type of anomaly. The proposed solutions take shape according to the nature of the problems and try to cope with some of the above difficulties.

## 1.4. Scope of Our Work

So far, we defined the anomaly detection problem in time series data along with the motivation of this thesis and the previous works. We want to detect collective or point anomalies in discrete and multivariate streams using sub-sequence based anomaly detection approaches with semi-supervised anomaly detection methods. We also want to discriminate the anomalies created by the system and the anomalies caused by the errors in the observations. To deal with such problems, we propose both statistical methods and deep learning algorithms. To successfully model the sequential observations we use the 'Hidden Markov Model' (HMM) as a statistical inference algorithm, while we use 'Recurrent Neural Network' (RNN) and 'Long Short Term Memory' (LSTM) as deep learning approaches. Then, we compare the observations with the model predictions in order to create an anomaly score for each step.

Our work focuses on two main applications, which are anomaly detection in wind turbines and anomaly detection in merchant transactions. In wind turbines, we define a set of observations as an anomaly while in merchant transactions, we interested in single point anomalies. Both applications rely on semi-supervised methods.

The rest of the thesis is organized as follows: The theoretical background required for the continuation of this thesis is given in Chapter 2. Chapter 3 contains the methods and algorithms that we propose to detect anomalies in time series. The data descriptions, experimental settings, and experimental results are given in Chapter 4. Chapter 5 includes the final comments about the works and further researches.

# 2. THEORETICAL BACKGROUND

In the previous chapter, we discussed the problem of anomaly detection in time series. This chapter is devoted to providing some concepts and methodologies which are addressed in our work. We will cover different statistical time series models in general terms and introduce our notation. Subsequently, we will address learning and inference methods in such models. Afterward, we will cover the deep learning sequence models.

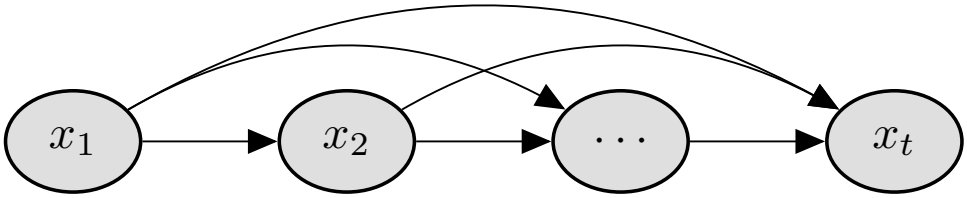
The entropy maximizing model for the set of observation $x_{1:T} = \{x_1, x_2, ..., x_{\mathcal{T}}\}$ is to assume the instances are independent and identically distributed (i.i.d.) random variables (RV) [30]. However, in the context of the time series, it is natural to consider models consistent with the causal nature of time [20]. So, we can get the following causal form of joint distribution $p(x_{1:\mathcal{T}})$ by applying Bayes' rule recursively:

$$p\left(x_{1:\mathcal{T}}\right) = p\left(x_{\mathcal{T}} \mid x_{1:\mathcal{T}-1}\right) p\left(x_{1:\mathcal{T}-1}\right) \tag{2.1}$$

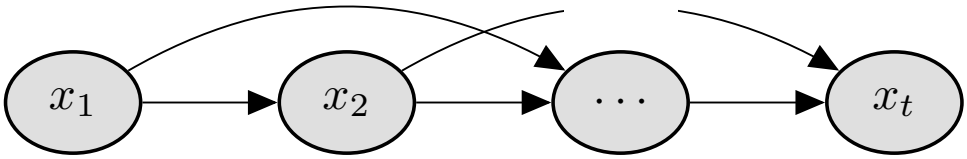
$$= \prod_{t=1}^{\mathcal{T}} p\left(x_t \mid x_{1:t-1}\right) \tag{2.2}$$

This representation of time series has natural causal interpretation in which observations depend on all the past information. This phenomenon corresponds to the cascade graph, shown in Figure 2.1(a), which is the most general form of belief network. Obviously, it is an intractable and expensive problem for a large scale model.

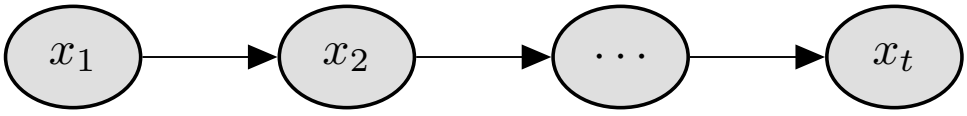
The fundamental solution to this problem is the assumption of conditional independence. Intuitively, it corresponds to the removal of the edges in the cascade graph. One of the most well-known models for conditional independence assumption on time-series model is the Markov model. In which, $L$th-order Markov model assumes that given observation only depends on previous $L$ observations. Mathematically, assumption of the $L$th-order Markov model is as the following:




(a) Cascade graph for time-series




(b) Second-order Markov model




(c) First-order Markov model

Figure 2.1. Graphical models for time-series models

$$p\left(x_t \mid x_{1:t-1}\right) \;=\; p\left(x_t \mid x_{t-L:t-1}\right) \tag{2.3}$$

Although $L$th-order Markov model is a simplified representation of the cascade graph, it still has relatively high complexity in which posterior distribution of the latent space is still untractable. On the other hand, the most straightforward representation of the Markov model is the 1st-order Markov model, which assumes each observation in the sequence only depend on previous observation. Although it looks easy, it has a tractable posterior distribution, and therefore, it is an influential and powerful model which allows stronger modeling. Mathematical interpretation of the first-order Markov model is as follows:

$$p\left(x_t \mid x_{1:t-1}\right) \;=\; p\left(x_t \mid x_{t-1}\right) \tag{2.4}$$

Graphical models for cascaded graph, second($L$th)-order Markov model and first-order Markov model are shown in Figure 2.1(b) and Figure 2.1(c), respectively. The complexity increases with the order of the network.

## 2.1. Hidden Markov Model

Up to the present, we examined the models which are directly built on observations. However, such models are suffering from low representation power on data. The reason for this is that observations are not always accurate and can deceive the model. Therefore, in the literature, a more general framework of time series models exists which uses latent, unobservable variable $s_t$, which generates observations.

From now on, instead of building models where observations depend on previous observations, we build models in which the observations depend on the hidden variable. One of the most well-known models with this structure is the 'state-space models'

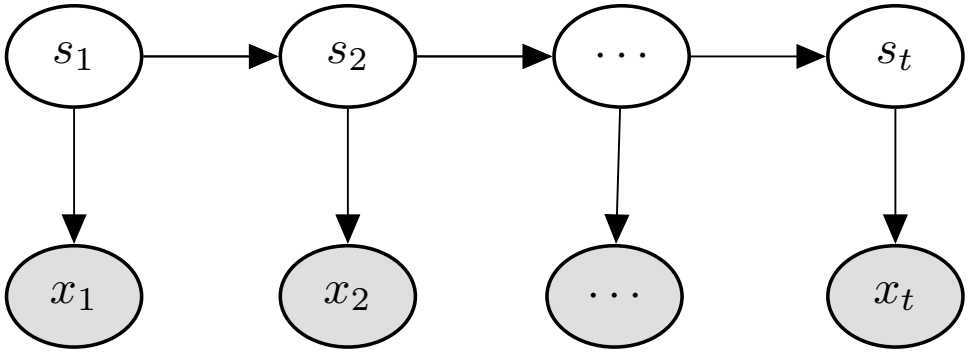



Figure 2.2. Graphical model for hidden Markov Model

(SSM). In this model, each observation is assumed to be generated from a latent (or hidden) variable. Therefore, Markov structure is formed on latent variables, $s_{1:\mathcal{T}}$, not on the observations, $x_{1:\mathcal{T}}$. The observed variables are dependent on the hidden variables through an emission, $p(y_t \mid x_t)$ [31]. Hidden Markov Model is the particular case of SSM in which hidden variables are only dependent on the previous hidden variable [21]. Hence, state-transition dynamics are shown as $p(s_t \mid s_{t-1})$. In other words, Hidden Markov models are first order state-space models. Graphical model for HMMs is given in Figure 2.2. The joint probability distribution for this model is given below. Initial state $s_1$ can be considered as dependent on initial hidden state $s_0$ and $p(s_1) = p(s_1 \mid s_0)$ so that one could get a simpler expression for joint probability.

$$p(s_{1:\mathcal{T}}, x_{1:\mathcal{T}}) = \left[\prod_{t=1}^{\mathcal{T}} p(x_t \mid s_t)\right]\left[\prod_{t=2}^{\mathcal{T}} p(s_t \mid s_{t-1})\right] p(s_1) \tag{2.5}$$

$$= \prod_{t=1}^{\mathcal{T}} p(x_t \mid s_t)\, p(s_t \mid s_{t-1}) \tag{2.6}$$

There are different naming for hidden Markov models because both discrete and continuous models share the same graphical model structure. For the convention, we use the term 'state-space models'(SSM) as a generic name for latent state Markov models, 'hidden Markov Model'(HMM) for the discrete latent state Markov models and 'linear dynamical systems'(LDS) for continuous latent state Markov models.

The hidden variables in HMM are always discrete, while observations could be both discrete or continuous. Therefore for a HMM, in the case of $S$ different states, the state transition distribution $p(s_t \mid s_{t-1})$ can be defined by an $S \times S$ transition matrix $\mathbf{\Psi}$ and, $\Psi_{\hat{s},\hat{s}'} = p\left(s_t = \hat{s} \mid s_{t-1} = \hat{s}'\right)$ denotes the probability of going from state $\hat{s}'$ to state $\hat{s}$ at the time $t$. It is important to note that the transition matrix $\mathbf{\Psi}$ is build from non-negative entries $\Psi_{\hat{s},\hat{s}'}$, and sum of entries in columns of transaction matrix is equal to 1, $\sum_{\hat{s}} \Psi_{\hat{s},\hat{s}'} = 1$ Similarly, in the case of $X$ discrete observations, emission distribution $p(x_t|s_t)$ can be defined by $X \times S$ emission matrix $\mathbf{\Omega}$ and $\Omega_{\hat{x},\hat{s}} = p\left(x_t = \hat{x} \mid s_t = \hat{s}\right)$. If the output is continuous then $s_t$ selects one of the potential $S$ output distributions $p(x_t \mid s_t)$.

## 2.2. Inference in Hidden Markov Model

Hidden Markov models have widespread applications with different purposes in many different domains, such as speech recognition, bioinformatics, and time series forecasting. Therefore, different outputs or statistics may be requested from the model. One may try to infer the current latent state $s_t$ useing all the observations $x_{1:t}$ from the past to the present. It can be shown as $p\left(s_t \mid x_{1:t}\right)$. This approach is named as *filtering* in the literature. On the other hand one can try to infer the past $p\left(s_{t-L} \mid x_{1:t}\right)$ or try to infer the future $p\left(s_{t+L} \mid x_{1:t}\right)$ with positive $L$, which are known as *smoothing* and *prediction* respectively. Finally one can try to identify the most likely hidden path $\arg\max_{s_{1:\mathcal{T}}} p\left(s_{1:\mathcal{T}} \mid x_{1:\mathcal{T}}\right)$ which is known as *Viterbi path.* and can be calculated with *Viterbi algorithm.* In the following parts, we will discover these methods.

### 2.2.1. Filtering

Filtering is the estimation of the current hidden state by using all observations so far, $p\left(s_t \mid x_{1:t}\right)$. To compute that, one can first find joint marginal $p\left(s_t, x_{1:t}\right)$ which is proportional to conditional marginal $p\left(s_t \mid x_{1:t}\right)$ and conditional marginal can be reached by the normalization.

$$
\begin{aligned}
p\left(s_t, x_{1:t}\right) &= \sum_{s_{t-1}} p\left(s_t, s_{t-1}, x_{1:t-1}, x_t\right) && (2.7)\\
&= \sum_{s_{t-1}} p\left(x_t \mid x_{1:t-1}, s_t, s_{t-1}\right) p\left(s_t \mid x_{1:t-1}, s_{t-1}\right) p\left(x_{1:t-1}, s_{t-1}\right) && (2.8)\\
&= \sum_{s_{t-1}} p\left(x_t \mid s_t\right) p\left(s_t \mid s_{t-1}\right) p\left(s_{t-1}, x_{1:t-1}\right) && (2.9)
\end{aligned}
$$

If we define $\alpha\left(s_t\right) = p\left(s_t, x_{1:t}\right)$ and put that equation into the Equation 2.9 we could get the following recursive equation which known as *$\alpha$-recursion* or *forward recursion*;

$$
\alpha\left(s_t\right) = \underbrace{p\left(x_t \mid s_t\right)}_{\text{corrector}} \underbrace{\sum_{s_{t-1}} p\left(s_t \mid s_{t-1}\right) \alpha\left(s_{t-1}\right)}_{\text{predictor}} \qquad (2.10)
$$

where $\alpha\left(s_1\right) = p\left(x_1, s_1\right) = p\left(x_1 \mid s_1\right) p\left(s_1\right)$. This recursive formula shows that filtered distribution $\alpha(.)$ is propagated through forward in each time-step and it acts like a 'prior' distribution for the following time-step. In other words, at each time step the calculated posterior becomes the new prior the for following time-step [31].

### 2.2.2. Smoothing

Smoothing is basically the estimation of the past hidden states from the given data. Mathematically it can express as $p(s_t \mid x_{1:\mathcal{T}})$ where the $\mathcal{T}$ is the length of the sequence and $t < \mathcal{T}$. Similar to what have we done in the previous section, we calculate the joint marginal $p(s_t, x_{1:\mathcal{T}})$ instead of conditional marginal $p(s_t \mid x_{1:\mathcal{T}})$. There are conceptually two main approaches to calculate smoothing: *Parallel Smoother* and *Sequential Smoother*.

2.2.2.1. Parallel Smoother. In this approach, the posterior distribution is rewritten in the form with contributions from the past and the future, taking advantage of *d-separation* in the Equation 2.13. It is the best-known smoothing method in the literature [21] and algebraically it can be shown as follows.

$$
\begin{aligned}
p\left(s_t, x_{1:\mathcal{T}}\right) &= p\left(s_t, x_{1:t}, x_{t+1:\mathcal{T}}\right) && (2.11)\\
&= p\left(s_t, x_{1:t}\right) p\left(x_{t+1:\mathcal{T}} \mid s_t, x_{1:t}\right) && (2.12)\\
&= \underbrace{p\left(s_t, x_{1:t}\right)}_{\text{past}} \underbrace{p\left(x_{t+1:\mathcal{T}} \mid s_t\right)}_{\text{future}} && (2.13)\\
&= \alpha\left(s_t\right) \beta\left(s_t\right) && (2.14)
\end{aligned}
$$

where $\beta(s_t)$ is called as *$\beta$-recursion* or *backward recursion*. Since $\alpha(.)$ and $\beta(.)$ recursions are independent of each other, and they may be run in parallel. From now on, the $\beta$-recursion needs to be calculated. The derivation of the $\beta$-recursion is as follows:

$$
\begin{aligned}
p\left(x_{t+1:\mathcal{T}} \mid s_t\right) &= \sum_{s_{t+1}} p\left(x_{t+1}, x_{t+2:\mathcal{T}}, s_{t+1} \mid s_t\right) && (2.15)\\
&= \sum_{s_{t+1}} p\left(x_{t+1} \mid x_{t+2:\mathcal{T}}, s_{t+1}, s_t\right) p\left(x_{t+2:\mathcal{T}}, s_{t+1} \mid s_t\right) && (2.16)\\
&= \sum_{s_{t+1}} p\left(x_{t+1} \mid s_{t+1}\right) p\left(s_{t+1} \mid s_t\right) p\left(x_{t+2:\mathcal{T}} \mid s_{t+1}, s_t\right) && (2.17)\\
&= \sum_{s_{t+1}} p\left(x_{t+1} \mid s_{t+1}\right) p\left(s_{t+1} \mid s_t\right) p\left(x_{t+2:\mathcal{T}} \mid s_{t+1}\right) && (2.18)\\
\beta(s_t) &= \sum_{s_{t+1}} p\left(x_{t+1} \mid s_{t+1}\right) p\left(s_{t+1} \mid s_t\right) \beta\left(s_{t+1}\right) && (2.19)
\end{aligned}
$$

This $\alpha - \beta$ recursion is known as *Forward-Backward* algorithm. Smoothed posterior can be obtained by the normalization of the result of the *Forward-Backward* algorithm.

2.2.2.2. Sequential Smoother. In this approach, the recursion is directly formed for smoothed posterior. In the literature, it is also known as *correction smoother*. This method, again, uses the advantages of the *d-separation* this time in a different way. It makes future observations unnecessary by the conditioning on the latent present state [32].

$$
\begin{aligned}
p\left(s_t \mid x_{1:\mathcal{T}}\right) &= \sum_{s_{t+1}} p\left(s_t, s_{t+1} \mid x_{1:\mathcal{T}}\right) && (2.20)\\
&= \sum_{s_{t+1}} p\left(s_t \mid s_{t+1}, x_{1:t}, x_{t+1:\mathcal{T}}\right) p\left(s_{t+1} \mid x_{1:\mathcal{T}}\right) && (2.21)\\
&= \sum_{s_{t+1}} p\left(s_t \mid s_{t+1}, x_{1:t}\right) p\left(s_{t+1} \mid x_{1:\mathcal{T}}\right) && (2.22)
\end{aligned}
$$

If we define $\gamma(s_t) = p\left(s_t \mid x_{1:\mathcal{T}}\right)$ and put that equation into the Equation 2.22, then we could get the following recursive equation which known as $\gamma$*-recursion*;

$$
\gamma(s_t) = \sum_{s_{t+1}} p\left(s_t \mid s_{t+1}, x_{1:t}\right) \gamma(s_{t+1}) \tag{2.23}
$$

In this smoother, the probability $p\left(s_t \mid s_{t+1}, x_{1:t}\right) \propto p\left(s_{t+1} \mid s_t\right) p\left(s_t \mid x_{1:t}\right)$. So, it may directly be calculated from the filtered results. It is called *dynamic reversal* because it equals to change the directions in the latent space. This methods also named as *correction smoother* because it changed the filtered result. One could realize that sequential smoother is proportional to parallel smoother. Mathematically it can be shown as;

$$
\gamma\left(s_t\right) \propto \alpha\left(s_t\right) \beta\left(s_t\right) \tag{2.24}
$$

### 2.2.3. Prediction

Prediction of the succeeding states and observations is another important topic in the latent space models. One can find the following sequence of length $L$ as follows:

$$p\left(x_{t+1:t+L} \mid x_{1:t}\right) = \sum_{s_{t:t+L}} \left( \prod_{\tau=t+1}^{t+L} p\left(x_\tau \mid s_\tau\right) p\left(s_\tau \mid s_{\tau-1}\right) \right) p\left(s_t \mid x_{1:t}\right) \quad (2.25)$$

If the $L$-step ahead predictive distribution is more important than the predictive distribution of the 1-step ahead, the sum is taken over future observations in the Equation 2.25. So the new density is as follows:

$$p\left(x_{t+1:t+L} \mid x_{1:t}\right) = \sum_{s_{t:t+L}} p\left(x_{t+L} \mid s_{t+L}\right) \left( \prod_{\tau=t+1}^{t+L} p\left(s_\tau \mid s_{\tau-1}\right) \right) p\left(s_t \mid x_{1:t}\right) \quad (2.26)$$

### 2.2.4. Viterbi Algortihm

Finding the most likely sequence $s_{1:\mathcal{T}}$ of hidden states is significant problem [33, 34]. The most likely sequence $s_{1:\mathcal{T}}$ for posterior $p\left(s_{1:\mathcal{T}} \mid x_{1:\mathcal{T}}\right)$ is the same with the most likely sequence for joint probability $p\left(s_{1:\mathcal{T}}, x_{1:\mathcal{T}}\right)$. This joint probability is equivalent to Equation 2.6.

$$\underbrace{\max_{s_{1:t}} p\left(s_{1:t}, x_{1:t}\right)}_{\phi(s_t)} = \left( \max_{s_t} p\left(x_t \mid s_t\right) p\left(s_t \mid s_{t-1}\right) \right) \underbrace{\max_{s_{1:t-1}} p\left(s_{1:t-1}, x_{1:t-1}\right)}_{\phi(s_{t-1})} \quad (2.27)$$

In this recursion, the message is sent from the beginning to the end of the chain. From the set of observations $x_{1:\mathcal{T}}$, the Viterbi algorithm starts to find the most likely states $s_t$ beginning from the last.

## 2.3. Learning in Probabilistic Models

Learning is the estimation of the model parameters $\theta$ from the given data $\boldsymbol{x} \equiv x_{1:\mathcal{T}}$. Parameter estimation methods such as *maximum likelihood estimation* (MLE) and *maximum a posteriori* (MAP) are based on the idea of defining a probability distribution on data $\boldsymbol{x}$ and tuning the parameters $\theta$ of the distribution such that likelihood of observed data is maximized under this particular probability distribution [35]. Maximum a Posteriori requires an additional prior distribution on parameters which is equivalent to regularization. Additionally, probability distributions are functions distributed between 0 and 1 and can take very small values. Therefore, for numerical stability, they are examined on a logarithmic scale, which is a monotonic function. So the general solution form is as follows;

- *Maximum Likelihood Estimation*: $\arg\max_\theta \log p\left(\boldsymbol{x} \mid \theta\right)$
- *Maximum a Posteriori*: $\arg\max_\theta \log p\left(\theta \mid \boldsymbol{x}\right) \equiv \arg\max_\theta \left(\log p\left(\boldsymbol{x} \mid \theta\right) + \log p\left(\theta\right)\right)$

In latent space models, the hidden variables $\boldsymbol{s} \equiv s_{1:\mathcal{T}}$ should be marginalized out in order to calculate the likelihood of the data. In time series problem, since the hidden dimensions grows with time, it is usually impossible to track marginalization $\sum_{\boldsymbol{s}} p(\boldsymbol{x}, \boldsymbol{s} \mid \theta)$. Therefore it is impossible to calculate the likelihood of the data directly. Intractable likelihood of the HMM is as follows:

$$p\left(x_{1:\mathcal{T}}\right) \;=\; \sum_{s_\mathcal{T}} p\left(s_\mathcal{T}, x_{1:\mathcal{T}}\right) = \sum_{s_\mathcal{T}} \alpha\left(s_\mathcal{T}\right) \tag{2.28}$$

where $\alpha\left(s_\mathcal{T}\right)$ is shown in Equation 2.10. So, the goal is the learning of the maximum likelihood parameters [19] which can be carried out by the following algorithms. It is important to note that sum is exponential in the sequence length. Therefore, although the above formula appears straightforward, it is often impossible to calculate it directly.

### 2.3.1. Expectation-Maximization Algorithm

Maximizing the likelihood is an essential problem under missing data or latent variables [36]. We mentioned that this is usually an intractable problem in time series models. There is an iterative and convenient method in order to solve such problems, which is called *Expectation-Maximization* (EM) Algorithm. The primary goal of the EM algorithm is to find a $\theta$ that maximizes the marginal likelihood $p(\boldsymbol{x} \mid \theta)$ or log marginal likelihood $\log p(\boldsymbol{x} \mid \theta)$. Since, the logarithm is a monotonic function, maximizing $\theta$ for these two likelihood is the same. Usually, log marginal likelihood is preferred because it has numerical stability in calculations. The main idea of the EM algorithm is to form an alternative objective function for which individual parameter updates can be achieved. By doing so, the marginal likelihood will be replaced by a lower bound. So, one can iteratively maximize this lower bound.

To derive a lower bound on log marginal likelihood $\log p(\boldsymbol{x} \mid \theta)$, *Kullback-Leibler* (KL) divergence which is always non-negative and measures the distance between probability distributions is considered [37]. In EM algorithm, one define a 'variational' distribution $q(\boldsymbol{s} \mid \boldsymbol{x})$. Then, the distance between 'variational' distribution $q(\boldsymbol{s} \mid \boldsymbol{x})$ and the parametric model $p(\boldsymbol{s} \mid \boldsymbol{x}, \theta)$ can be measured by KL divergence as follows:

$$\begin{aligned} \mathcal{D}_{\mathrm{KL}}\left(q\left(\boldsymbol{s} \mid \boldsymbol{x}\right) \| p\left(\boldsymbol{s} \mid \boldsymbol{x}, \theta\right)\right) &= \mathbb{E}_{q(\boldsymbol{s}|\boldsymbol{x})}\left[\log q\left(\boldsymbol{s} \mid \boldsymbol{x}\right) - \log p\left(\boldsymbol{s} \mid \boldsymbol{x}, \theta\right)\right] \quad (2.29) \\ &= \mathbb{E}_{q(\boldsymbol{s}|\boldsymbol{x})}\left[\log q\left(\boldsymbol{s} \mid \boldsymbol{x}\right) - \log p\left(\boldsymbol{s}, \boldsymbol{x} \mid \theta\right)\right] + \log p\left(\boldsymbol{x} \mid \theta\right) \\ &\geq 0 \quad (2.30) \end{aligned}$$

Equation 2.29 is obtained from Bayes' rule $p\left(\boldsymbol{s} \mid \boldsymbol{x}, \theta\right) = p\left(\boldsymbol{s}, \boldsymbol{x} \mid \theta\right) / p\left(\boldsymbol{x} \mid \theta\right)$ where $p\left(\boldsymbol{x} \mid \theta\right)$ does not depend on $\boldsymbol{s}$. So, lower bound on log marginal likelihood could be obtain by rearranging the inequality between Equation 2.29 and Equation 2.30:

$$\log p\left(\boldsymbol{x} \mid \theta\right) \geq \underbrace{-\mathbb{E}_{q(\boldsymbol{s}|\boldsymbol{x})}\left[\log q\left(\boldsymbol{s} \mid \boldsymbol{x}\right)\right]}_{\text{Entropy}} + \underbrace{\mathbb{E}_{q(\boldsymbol{s}|\boldsymbol{x})}\left[\log p\left(\boldsymbol{s}, \boldsymbol{x} \mid \theta\right)\right]}_{\text{Energy}} \quad (2.31)$$

The right-hand side of the Equation 2.31 is known as a lower bound for log-likelihood and represented as $\mathcal{L}(q, \theta)$ which depends on the choice of distribution $q$ and the model parameters $\theta$. Energy term is, on the other hand, known as 'expected complete log-likelihood' [19]. Log-likelihood of the data is shown as follows:

$$\log p\left(\boldsymbol{x} \mid \theta\right) \;=\; \mathcal{L}\left(q, \theta\right) + \mathcal{D}_{\mathrm{KL}}\left(q\left(\boldsymbol{s} \mid \boldsymbol{x}\right) \| p\left(\boldsymbol{s} \mid \boldsymbol{x}, \theta\right)\right) \tag{2.32}$$

Therefore, by minimizing divergence in Equation 2.32, one can achieve to maximize the lower bound on the log-likelihood of the observations. The lower bound $\mathcal{L}$ depends both on model parameters $\theta$ and 'variational distributions' $q$. The EM algorithm tries to find this lower bound iteratively by optimizing it w.r.t. $\theta$ and $q$, respectively. This algorithm is built upon two steps which are;

- *Expectation (E-step)*: Finds the variational distribution $q\left(\boldsymbol{s} \mid \boldsymbol{x}\right)$ for fixed $\theta$
- *Maximization (M-step)*: Finds the model parameters $\theta$ for fixed $q\left(\boldsymbol{s} \mid \boldsymbol{x}\right)$

The algorithm starts with initial model parameters $\theta^{old}$. In the E-step, the variational distribution $q\left(\boldsymbol{s} \mid \boldsymbol{x}\right)$ is set to posterior distribution $p\left(\boldsymbol{s} \mid \boldsymbol{x}, \theta^{old}\right)$ and log-likelihood is calculated under $\theta^{old}$. In the M-step, $\theta^{new}$ will be found which maximizes the log-likelihood. Since only the 'energy' term of the lower bound in Equation 2.31 depends on $\theta^{new}$, M-step corresponds to maximization of the energy. By denoting 'energy' as $Q\left(\theta, \theta^{old}\right)$, E and M steps could be shown as follows:

$$\text{E-step} : Q\left(\theta, \theta^{old}\right) \;=\; \mathbb{E}_{p\left(\boldsymbol{s} \mid \boldsymbol{x}, \theta^{old}\right)}\left[\log p\left(\boldsymbol{s}, \boldsymbol{x} \mid \theta\right)\right] \tag{2.33}$$

$$\text{M-step} : \theta^{new} \;=\; \arg\max_{\theta} Q\left(\theta, \theta^{old}\right) \tag{2.34}$$

These steps are repeated until the convergence. E-step and M-step could also be considered as an inference step and learning step. In the E-step, posterior of the latent variables are inferred, and in the M-step, the new model parameters $\theta^{new}$ are learned.

### 2.3.2. Baum-Welch Algorithm

Baum-Welch algorithm is special case of the EM algorithm which is for learning model parameters of a hidden Markov model (HMM) [38]. In the HMM, state transition matrix $\boldsymbol{\Psi}$, where $\Psi_{\hat{s},\hat{s}'} = p\left(s_t = \hat{s} \mid s_{t-1} = \hat{s}'\right)$, emission matrix $\boldsymbol{\Omega}$, where $\Omega_{\hat{x},\hat{s}} = p\left(x_t = \hat{x} \mid s_t = \hat{s}\right)$, and initial state parameter $\boldsymbol{\pi}$, where $\pi_{\hat{s}} = p\left(s_1 = \hat{s}\right)$, could be learned from the given set of data $\boldsymbol{x}$ under the assumption that the number of hidden states S is known. Therefore model parameters $\theta$ are parameterized by state transition $\boldsymbol{\Psi}$, emission probability $\boldsymbol{\Omega}$ and initial state $\boldsymbol{\pi}$, in such $\theta = (\boldsymbol{\pi}, \boldsymbol{\Psi}, \boldsymbol{\Omega})$ [39].

The energy term of the HMM is obtained from the logarithm of the joint probability density of HMM in Equation 2.6 under the assumption of the *independent and identically distributed* (i.i.d.) which is as follows:

$$
\begin{aligned}
Q\left(\theta, \theta^{old}\right) &= \sum_{n=1}^{N} \mathbb{E}_{p\left(\boldsymbol{s}^n \mid \boldsymbol{x}^n, \theta^{old}\right)} \left[\log p\left(\boldsymbol{s}^n, \boldsymbol{x}^n \mid \theta\right)\right] && (2.35) \\
&= \sum_{n=1}^{N} \mathbb{E}_{p\left(s_1^n \mid \boldsymbol{x}^n, \boldsymbol{\pi}^{old}\right)} \left[\log p\left(s_1^n\right)\right] \\
&\quad + \sum_{n=1}^{N} \sum_{t=2}^{\mathcal{T}_n} \mathbb{E}_{p\left(s_t^n, s_{t-1}^n \mid \boldsymbol{x}^n, \boldsymbol{\Psi}^{old}\right)} \left[\log p\left(s_t^n \mid s_{t-1}^n\right)\right] \\
&\quad + \sum_{n=1}^{N} \sum_{t=1}^{\mathcal{T}_n} \mathbb{E}_{p\left(s_t^n \mid \boldsymbol{x}^n, \boldsymbol{\Omega}^{old}\right)} \left[\log p\left(x_t^n \mid s_t^n\right)\right] && (2.36)
\end{aligned}
$$

One need to maximize energy term specified in Equation 2.36. We need to maximize this equation according to our three different parameters. This procedure corresponds to the M-step of the EM algorithm.

Optimizing Equation 2.36 with respect to $p\left(s_1\right)$ and forcing $p\left(s_1\right)$ to be a distribution one can get M-step of the initial parameters $\boldsymbol{\pi}$ as follows:

$$
\pi_{\hat{s}}^{new} \equiv p\left(s_1 = \hat{s} \mid \theta^{old}\right) = \frac{1}{N} \sum_{n=1}^{N} p\left(s_1^n = \hat{s} \mid \boldsymbol{x}^n, \theta^{old}\right) \qquad (2.37)
$$

This is the average number of time where initial state $s_1$ is in state $\hat{s}$ w.r.t. $\boldsymbol{\pi}^{old}$. Similarly one should optimize Equation 2.36 w.r.t. $p\left(s_t \mid s_{t-1}\right)$ to get M-step of the transition paramaters $\boldsymbol{\Psi}$ which is as follows:

$$\begin{aligned} \Psi_{\hat{s},\hat{s}'}^{new} &\equiv p\left(s_t = \hat{s} \mid s_{t-1} = \hat{s}', \theta^{old}\right) && (2.38) \\ &\propto \sum_{n=1}^{N}\sum_{t=2}^{\mathcal{T}_n} p\left(s_t^n = \hat{s}, s_{t-1}^n = \hat{s}' \mid \boldsymbol{x}^n, \theta^{old}\right) && (2.39) \end{aligned}$$

Which is equivalent to the average number of times that transition from latent state $\hat{s}'$ to $\hat{s}$. One should force the rows of the $\boldsymbol{\Psi}$ to be a distribution; therefore, these rows should be normalized. The following equation could achieve it:

$$\Psi_{\hat{s},\hat{s}'}^{new} = \frac{\sum_{n=1}^{N}\sum_{t=2}^{\mathcal{T}_n} p\left(s_t^n = \hat{s}, s_{t-1}^n = \hat{s}' \mid \boldsymbol{x}^n, \theta^{old}\right)}{\sum_{\hat{s}}\sum_{n=1}^{N}\sum_{t=2}^{\mathcal{T}_n} p\left(s_t^n = \hat{s}, s_{t-1}^n = \hat{s}' \mid \boldsymbol{x}^n, \theta^{old}\right)} \quad (2.40)$$

By doing so, one force to $\boldsymbol{\Psi}$ to be probability of transition from latent state $\hat{s}'$ to $\hat{s}$. Similar to the previous update equations, M-step for the emission parameters are obtained by the optimization of Equation 2.36 w.r.t $p\left(x_t \mid s_t\right)$. Update equation is as follows:

$$\begin{aligned} \Omega_{\hat{x},\hat{s}}^{new} &\equiv p\left(x_t = \hat{x} \mid s_t = \hat{s}\right) && (2.41) \\ &\propto \sum_{n=1}^{N}\sum_{t=1}^{\mathcal{T}_n} p\left(s_t^n = \hat{s} \mid \boldsymbol{x}^n, \theta^{old}\right) \mathbb{I}\left[x_t^n = \hat{x}\right] && (2.42) \end{aligned}$$

This is equivalent to the average probability of emission form latent state $\hat{s}$ to observation state $\hat{x}$. Similar to transition probability, to get stationary probability distribution on emission matrix $\boldsymbol{\Omega}$ we need to normalize this equation for emission probability as follows:

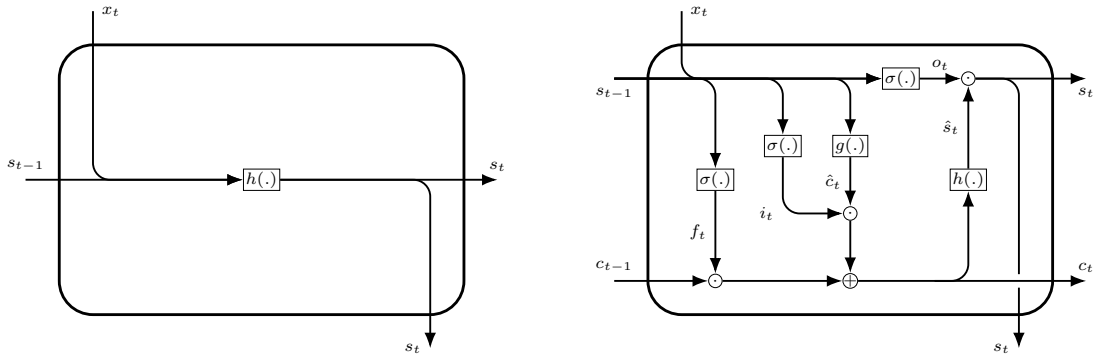
$$\Omega_{\hat{x},\hat{s}}^{new} = \frac{\sum_{n=1}^{N}\sum_{t=1}^{\mathcal{T}_n} p\left(s_t^n = \hat{s} \mid \boldsymbol{x}^n, \theta^{old}\right) \mathbb{I}\left[x_t^n = \hat{x}\right]}{\sum_{n=1}^{N}\sum_{t=1}^{\mathcal{T}_n} p\left(s_t^n = \hat{s} \mid \boldsymbol{x}^n, \theta^{old}\right)} \quad (2.43)$$

In the E-step, three quantities $p\left(s_1^n = \hat{s} \mid \boldsymbol{x}^n, \theta^{old}\right)$, $p\left(s_t^n = \hat{s}, s_{t-1}^n = \hat{s}' \mid \boldsymbol{x}^n, \theta^{old}\right)$ and $p\left(s_t^n = \hat{s} \mid \boldsymbol{x}^n, \theta^{old}\right)$ which are used in M-step update equations, should be calculated. The derivations of this quantities are shown in the section 2.2.

## 2.4. Deep Learning Sequence Models

Up to the present, we examined statistical models and algorithms. However, deep learning techniques for making inferences in time series are also very successful. In this section, we will continue to more in-depth dive into such sequence models. It should be noted that we will be only interested in networks that produce an output at each time step.

A neural network (NN), in the case of artificial neurons, is called an artificial neural network (ANN). It resembles an interconnected group of neurons which mathematically correspond to non-linear activation functions. That is, neural networks are non-linear statistical data modeling or decision making tools which can be used to model complex relationships between inputs and outputs or to find patterns in data. Additionally, these structures are transformed into deep models with an increased number of hidden layers. We will use the name 'traditional, fully connected feedforward network' for all systems that share such similar ideas. Such networks would have separate parameters for each input feature so they would need to learn all parameter space separately at each position in the sequence. By doing so, these models technically assume that all inputs and outputs are independent of each other. However, observations are not independent of each other, and modeling the relationship between them allows us to achieve more successful results. To overcome this problem, deep learning sequence models use the parameter sharing approach. Parameter sharing makes it possible to share features across different sequence positions of the network. By this modification, NN approaches gains the ability to work in a sequence prediction since it takes into account the previous inputs to predict the next output. 'Recurrent Neural Networks' (RNN) and 'Long Short-Term Memory' are the two instance of such models.




(a) RNN Cell (b) LSTM Cell

Figure 2.3. Cell structures of the RNN and LSTM

### 2.4.1. Recurrent Neural Networks

Recurrent Neural Networks (RNN) are the neural networks for processing sequence data, which contains cycles in the structure [40]. In the literature, any NN with the cycles can be considered as an RNN. Such cycles allow detecting recurrences of patterns and networks share the same weights across several time-steps [41]. This chain structure makes them powerful to analyze sequences and time series problems. The folded and unfolded computational graphs of the RNNs and LSTMS are shown in Figure 2.4. These graphs represents the mapping between an input sequence $\boldsymbol{x}$ and corresponding output sequence $\hat{\boldsymbol{y}}$. $\boldsymbol{\mathcal{L}}$ refers to loss and shown as $\mathcal{D}(\boldsymbol{y}\|\hat{\boldsymbol{y}})$ while $\boldsymbol{\mathcal{W}}$'s correspond to the weight matrix of the network. $\boldsymbol{C}$ represents the neural network in the loop and called as 'cell'. These computational graphs inspire these networks with the loops where each loop in the network represents the influence of the previous value of each variable on the next value of the same variable.

The forward propagation of the RNN assumes that there is a non-linear activation function at the hidden units. Hidden state of the network is the result of this activation function. Outputs are the linear transformations of the hidden states. Graphically recurrent neural networks can be shown as Figure 2.4 with the cell in Figure 2.3(a). Forward propagation begins with the $s_0$. Then for each time step following update equations will be applied:

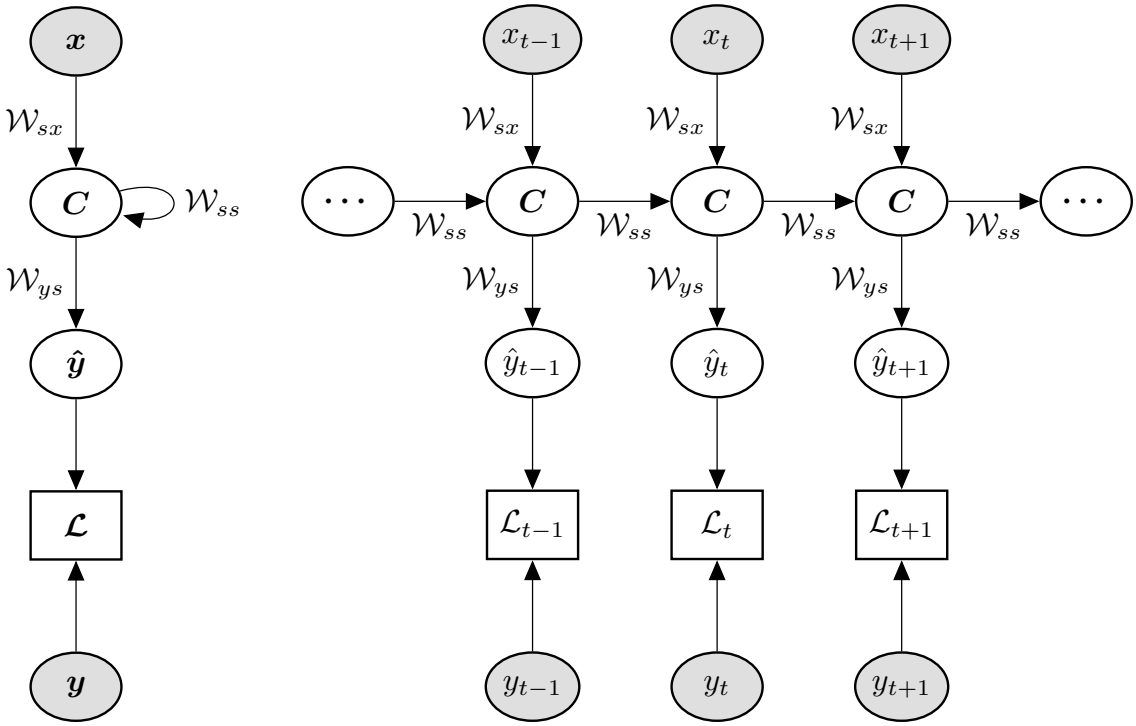



Figure 2.4. Folded and unfolded representation of the computational graphs

$$s'_t = \mathcal{W}_{ss} s_{t-1} + \mathcal{W}_{sx} x_t + b_s \tag{2.44}$$

$$s_t = h\left(s'_t\right) \tag{2.45}$$

$$\hat{y}_t = \mathcal{W}_{ys} s_t + c_y \tag{2.46}$$

By the help of the chain structure and recurrence, these neural networks can learn sequences and patterns. However, there are some disadvantages to RNNs. Since the gradients are transferred over time and pass through the non-linear activation function at each time step, the gradients tend to be decreasing exponentially and disappears after a few time steps. Therefore networks tend to forget the previous inputs after a few time steps and accordingly cannot handle the long-term dependencies. This problem is usually called as the 'vanishing gradient' in the literature. [42].

### 2.4.2. Long Short-Term Memory

Learning long-term dependencies in dynamical systems is one of the main challenges in deep learning researches. RNNs achieve to learn the short-term dependencies but they usually fail on long-term dependencies. Long Short-Term Memory (LSTM) structure is designed to capture long-term dependencies, as a special kind of RNN. The idea behind that structure is to remember the information for an extended period by default. Therefore LSTMs have emerged as effective and scalable models for several learning problems related to sequential data [43]. They are a special kind of RNN that are good at learning long-term dependencies. Similar to all kind of RNNs, LSTMs have the form of a chain of repeating modules of neural networks [44].

LSTM is a kind of gated RNN. The idea behind the gated RNNs is creating paths through time that have derivatives that neither vanish nor explode [40]. Gated RNNs accomplish this idea by the gates that change connection weights at each time step. The central idea behind the LSTM architecture is a memory cell which can maintain its state over time, and non-linear gating units which regulate the information flow into and out of the cell [45]. The idea of introducing internal recurrence, in addition to the outer recurrence of the RNN, is to produce paths where the gradient can flow for long time-steps. In standard RNN, repeating structure (cell) has a simple structure, an activation function. On the other hand, LSTM cell has four layers which interact with each other. The combination of the Figure 2.4 and Figure 2.3(b) could form graphical representation of the LSTMs. Forward propagation and interactions in the single LSTM cell are derived step by step as following:

(i) It should be decided which information should be thrown away from the previous cell state. Decision is made by 'forget gate' $f_t$. It is derived as follows:

$$f_t = \sigma\left(\mathcal{W}^f_{ss} s_{t-1} + \mathcal{W}^f_{sx} x_t + b^f\right) \tag{2.47}$$

(ii) It should be decided which new information coming from input $x_t$ will be stored in new cell state. In order to do this, first a proposal $\hat{c}_t$ should be made for the

new cell state and then it should be decided how much of this proposal will be accepted. The decision maker for this process is the 'input gate' $i_t$. These are derived as follows:

$$\hat{c}_t = g\left(\mathcal{W}_{ss}^{\hat{c}} s_{t-1} + \mathcal{W}_{sx}^{\hat{c}} x_t + b^{\hat{c}}\right) \tag{2.48}$$

$$i_t = \sigma\left(\mathcal{W}_{ss}^{i} s_{t-1} + \mathcal{W}_{sx}^{i} x_t + b^{i}\right) \tag{2.49}$$

(iii) The new step is the determination of the new cell state. For this purpose, the outputs obtained in the previous steps should be used. Derivation of the updated cell state is as follows:

$$c_t = f_t \odot c_{t-1} + i_t \odot \hat{c}_t \tag{2.50}$$

(iv) The new hidden state $s_t$ is calculated after the calculation of the new cell status $c_t$. But this transformation is done in 3 steps. First, the cell state $c_t$ is passed through the activation function $h(.)$ as in RNN and unfiltered hidden state $\hat{s}_t$ is obtained. Then the obtained result is filtered by the 'output gate' $o_t$ and the hidden state $s_t$ is obtained . The derivations as follows:

$$\hat{s}_t = h\left(c_t\right) \tag{2.51}$$

$$o_t = \sigma\left(\mathcal{W}_{ss}^{o} s_{t-1} + \mathcal{W}_{sx}^{o} x_t + b^{o}\right) \tag{2.52}$$

$$s_t = o_t \odot \hat{s}_t \tag{2.53}$$

(v) The predictions $\hat{y}_t$ of the network are obtained as follows. It is the linear transformation of the hidden state $s_t$ at time $t$, with the output weights of the network $\mathcal{W}_{ys}$, as in RNN.

$$\hat{y}_t = \mathcal{W}_{ys} s_t + c_y \tag{2.54}$$

LSTMs are the improved version of the RNNs. It has a better capacity of holding past information to current state and known as state of the art sequence model [45].

### 2.4.3. Learning in RNN and LSTM

After forward propagation, the loss $\mathcal{L}$ between $y_t$ and $\hat{y}_t$ and gradients will be calculated. Computation of the gradients through an RNN and an LSTM is a straightforward problem [40]. A particular version of the backpropagation algorithm, which is known as *Backpropagation Through Time* (BPTT), should be applied [46]. BPTT operates on an unfolded computational graph of RNN in time. The unfolded network contains $t$ inputs and outputs, but the network shares the same parameters over all networks. Then the backpropagation algorithm is used to find the gradient of the loss function concerning all the network parameters.

# 3. MODELS AND ALGORITHMS FOR ANOMALY DETECTION

In Chapter 1, we briefly mentioned the anomaly detection in time series problem and showed that several studies in the literature address different approaches for the problem. In Chapter 2, we mentioned the models and inference algorithms that are necessary for the development of anomaly detection in time series applications. In this chapter, the development of anomaly detection algorithms will be examined. Anomaly detection models should be able to distinguish the anomalies arising from the outliers in the system and the anomalies resulting from the errors in the observations. Besides, these models should be able to cope with missing observations to achieve more fruitful results, because such missing observations may disrupt the resulting anomaly score. Another issue is the importance of the models that can detect anomalies online because the anomalies are often wanted to be detected as soon as possible, even before they happen. The rest of this section contains details about the works we have done by paying attention to the above details.

## 3.1. Gaussian Mixture Model for Anomaly Detection

The problem of anomaly detection is more common in real data than in virtual data. Therefore, while modeling the system, it should be ensured that erroneous observations do not disturb the model. Therefore, we should be able to model these erroneous observations while developing our model. In other words, we can say that systems feed on multiple sources where one of them produce erroneous observations. If we can develop a model in this way, we can better detect system behavior and detect anomalies. We can develop such a model with Gaussian mixtures. Let assume; we have a one-dimensional observation. We assume that these observations come from 2 different distributions. So, we define two distributions. The first one is for non-outlier observations, and it has Gaussian distribution with a mean and ideal amount of variance that is calculated during the expectation maximization process. The second

distribution is for outlier observations and has 0 mean and $\infty$ variance. Then, let us assume, there is another Bernoulli distributed parameter $r$ which decides either given observation is an outlier or not. Then, we evaluate the model for parameters of the first distribution during the process. So our generative model is as follows:

$$r_t \sim \mathcal{BE}(\pi) \tag{3.1}$$

$$p\left(y \mid x, r, \mu, \sigma\right) \sim \mathcal{N}\left(\mu, \sigma^2\right)^{r=0} \mathcal{N}\left(0, \Sigma\right)^{r=1} \tag{3.2}$$

Here, $r = 1$ represents the outlier observations and $r = 0$ represents the non-outlier ones. Under the assumption of prior outlier probability $r$, this model is intended to find the possibility of $\mu^{(n+1)}$ and $\sigma^{(n+1)}$ which are maximizing the expectation of $\log p\left(y, r \mid x, \mu, \sigma\right)$ under the probability distribution $p\left(r \mid x, y, \mu^{(n)}, \sigma^{(n)}\right)$. Mathematically it could be shown as follows:

$$\mu^{(n+1)}, \sigma^{(n+1)} = \arg\max_{\mu,\sigma} \mathbb{E}_{p\left(r|x,y,\mu^{(n)},\sigma^{(n)}\right)} \left[\log p\left(y, r \mid x, \mu, \sigma\right)\right] \tag{3.3}$$

To calculate Equation 3.3, one should derive the probability distribution inside the expectation in Equation 3.3, which is $\log p\left(y, r \mid x, \mu, \sigma\right)$, and $p\left(r \mid x, y, \mu^{(n)}, \sigma^{(n)}\right)$. Following equations shows the derivation of the $p\left(y, r \mid x, \mu, \sigma\right)$, which is essential to calculate the previous two distributions:

$$p\left(y, r \mid x, \mu, \sigma\right) = \prod_t p\left(y_t, r_t \mid x_t, \mu, \sigma\right) \tag{3.4}$$

$$= \prod_t p\left(y_t \mid x_t, r_t, \mu, \sigma\right) p\left(r_t\right) \tag{3.5}$$

$$= \prod_t \left(\frac{1-\pi_0}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(\mu_t - y_t)^2}{2\sigma_t^2}\right)\right)^{(1-r_t)} \times \left(\frac{\pi_0}{\sqrt{2\pi\Sigma}} \exp\left(\frac{-y_t^2}{2\Sigma}\right)\right)^{(r_t)} \tag{3.6}$$

After the calculation of $p\left(y, r \mid x, \mu, \sigma\right)$, one can directly calculate the outlier probability $r$ given all other parameters. To calculate this probability, there is a need for the sum of the probability in Equation 3.6 over $r$. It can be calculated with the following form:

$$\begin{aligned} p\left(r \mid x, y, \mu, \sigma\right) &= \frac{p\left(r, y \mid x, \mu, \sigma\right)}{p\left(y \mid x, \mu, \sigma\right)} && (3.7) \\ &= \frac{p\left(r, y \mid x, \mu, \sigma\right)}{\Sigma_r p\left(r, y \mid x, \mu, \sigma\right)} && (3.8) \end{aligned}$$

Up to the present, we derived the conditional distribution of outlier probability $p\left(r \mid x, y, \mu, \sigma\right)$ and we derived the calculations of the probability $p\left(y, r \mid x, \mu, \sigma\right)$. Now, let $\pi_t^{(n)}$ be the probability of $r_t = 1$ given all other parameters. Then $\pi_t^{(n)}$ will be equal to Equation 3.10;

$$\begin{aligned} \pi_t^{(n)} &= p\left(r_t = 1 \mid x_t, y_t, \mu^{(n)}, \sigma^{(n)}\right) && (3.9) \\ \pi_t^{(n)} &= \frac{\frac{\pi_0}{\Sigma} \exp\left(\frac{-y_t^2}{2\Sigma}\right)}{\frac{\pi_0}{\Sigma} \exp\left(\frac{-y_t^2}{2\Sigma}\right) + \frac{1-\pi_0}{\sigma} \exp\left(\frac{-(\mu_t - y_t)^2}{2\sigma^2}\right)} && (3.10) \end{aligned}$$

From now on, we will calculate the log likelihood of the probability in the Equation 3.3. The objective is the maximization of the log-likelihood. The derivations for the calculation of the log-likelihood is as follows:

$$\begin{aligned} \mathbb{E}_{p\left(r|x,y,\mu^{(n)},\sigma^{(n)}\right)}\left[\log p\left(y, r \mid x, \mu, \sigma\right)\right] &= \sum_t \mathbb{E}_{p\left(r|x,y,\mu^{(n)},\sigma^{(n)}\right)}\left[\log p\left(y_t, r_t \mid x, \mu, \sigma\right)\right] \\ &= \sum_t \left\langle r_t \left(\frac{-y_t^2}{2\Sigma^2}\right) \right\rangle \\ &+ \sum_t \left\langle (1 - r_t) \left( \left(\frac{-(\mu_t - y_t)^2}{2\sigma_t^2}\right) - log\sigma_t \right) \right\rangle \\ Q(\mu, \sigma) &\propto \sum_t (1 - \pi_t^{(n)}) \left( \frac{(\mu_t - y_t)^2}{2\sigma_t^2} + log\sigma_t \right) \qquad (3.11) \end{aligned}$$

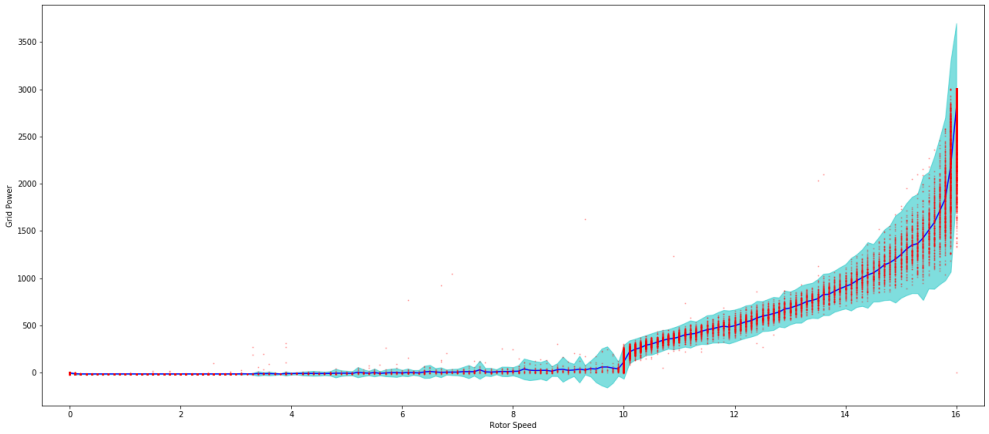



Figure 3.1. One dimensional Gaussian mixtures for the each value on the x-axis

Now, we can iteratively calculate the $\mu^{(n+1)}$ and $\sigma^{(n+1)}$ values that maximize $Q(\mu, \sigma)$ such that;

$$\mu^{(n+1)} = \arg\max_{\mu} Q(\mu, \sigma^{(n)}) \tag{3.12}$$

$$\sigma^{(n+1)} = \arg\max_{\sigma} Q(\mu^{(n)}, \sigma) \tag{3.13}$$

In this model, the system does not consider the time series properties. Figure 3.1 shows an intuitive but straightforward example of the Gaussian mixtures. The model decides the outlier observations and then learns the system behaviors. $r$ parameters are learned during the training, and they will become the anomaly score of the system.

## 3.2. Hidden Markov Model for Anomaly Detection

The purpose of this section is to create a model that will learn the operation of a system as a time series and generate a warning against distortions and deterioration in this system. The model is also expected to catch anomalies in an unsupervised manner because the anomaly detection needs to be done online, and it is often not possible to find the data marked as an anomaly. Also, usually, there is not enough data available to model or validate the deterioration phase. For this reason, the behavior of the system in normal working conditions has been modeled. The model will generate a warning if the system works outside of the normal working conditions. [47]

In this approach, the behavior of the system is modeled as a *Hidden Markov Model*. Also, it is assumed that the system always selects a state from a *discrete state space*, and according to this state, the system creates multi-dimensional observations. The state that the system chooses at any time depends only on the previous state. This state space corresponds intuitively with the phases that the system enters during operation (such as being closed or working on full power). However, one of the main difficulties of the problem is that the data set does not have an attribute, such as the phase of the system. Therefore, the state space should be found with the unsupervised learning algorithm. There are also some *missing observations* and *outliers* in the data set due to the sensors. These are other factors that must be accounted for in the Hidden Markov Model.

In the model, hidden variables of the system for the states at the $t$th time step is shown as $s_t$ and observations of the system at the $t$th time step is shown as $x_t$. Observations are usually multi-dimensional and expressed as a vector with a length of $\mathcal{P} + \mathcal{R}$, such that $x_t = \begin{bmatrix} p_t^1 & \cdots & p_t^{\mathcal{P}} & r_t^1 & \cdots & r_t^{\mathcal{R}} \end{bmatrix}^T$. There are usually two kinds of the subgroup of observations which are observations that drives the system and observations that are driven by the systems and shown as $p^{1:\mathcal{P}}$ and $r^{1:\mathcal{R}}$, respectively. We will show $p^{1:\mathcal{P}}$ as $p$ and $r^{1:\mathcal{R}}$ as $r$ for the simplicity of the representation, unless otherwise required.

### 3.2.1. Generative Model and Learning

The discrete state space of the system consists of $S$ different states which indicate the typical operating phases, and there is 1 additional state which indicates contradictory observations. In fact, with the slight modification of the observation model, instead of defining a separate state for the modeling of contradictory observations, it can be assumed that each observation is observed to be very noisy with a small likelihood. However, as far as we can tell from our analysis, outlier values are more likely to be seen in succession. For this reason, the state space of the model is defined in a way where there is an additional latent feature.

In our model the probability of transitions between states is shown in parameter $\mathbf{\Psi} \in \mathbb{R}^{(S+1)\times(S+1)}$. In other words, the transition probability from state $\hat{s}'$ to state $\hat{s}$ is $\Psi_{\hat{s},\hat{s}'}$. Each situation has its own observation distribution. The observation distribution where the outliers are observed ($\hat{s} = 0$) is a uniform distribution defined in the space of the observation. The observational distributions of the other states ($\hat{s} > 0$) are multivariate Gaussian distributions with the variables $\mu_{\hat{s}}$ and $\Sigma_{\hat{s}}$:

$$p(s_1) = \mathcal{U}\{0, S\} \tag{3.14}$$

$$p(s_t = \hat{s} \mid s_{t-1} = \hat{s}') = \Psi_{\hat{s}\hat{s}'} \tag{3.15}$$

$$p(x_t \mid s_t = \hat{s}) = \begin{cases} c, & \hat{s} = 0 \\ \mathcal{N}(x_t; \mu_{\hat{s}}, \Sigma_{\hat{s}}), & \hat{s} > 0 \end{cases} \tag{3.16}$$

The next step is the calculation of the variables $\Psi_{\hat{s}\hat{s}'}$, $\mu_{\hat{s}}$ and $\Sigma_{\hat{s}}$ from the data. Since this operation is expensive to perform over the whole dataset, random sequences with the length of $\tau$ can be selected from the dataset. Let $x_{1:\tau}^{(i)}$ be the $i$th selected sequence and suppose that corresponding $s_{1:\tau}^{(i)}$ are known for this sequence. Then sufficient statistics for the variables can be calculated as follows:

$$\left\langle x_t x_t^T \right\rangle_{p(x_t|s_t=\hat{s})} \approx \mathcal{V}_{\hat{s}}^{(i)} = \frac{\sum_{t=1}^{\tau} [s_t^{(i)} = \hat{s}] x_t x_t^T}{\sum_{t=1}^{\tau} [s_t^{(i)} = \hat{s}]} \tag{3.17}$$

$$\left\langle x_t \right\rangle_{p(x_t|s_t=\hat{s})} \approx m_{\hat{s}}^{(i)} = \frac{\sum_{t=1}^{\tau} [s_t^{(i)} = \hat{s}] x_t}{\sum_{t=1}^{\tau} [s_t^{(i)} = \hat{s}]} \tag{3.18}$$

$$C_{\hat{s}\hat{s}'}^{(i)} = \sum_{t=2}^{\tau} [s_{t-1}^{(i)} = \hat{s}'][s_t^{(i)} = \hat{s}] \tag{3.19}$$

The average of sufficient statistics ($\mathcal{V}_{\hat{s}}^{(i)}$, $m_{\hat{s}}^{(i)}$, $C_{\hat{s}\hat{s}'}^{(i)}$) of selected sequences can be used to estimate the final values of $\mathcal{V}_{\hat{s}}$, $m_{\hat{s}}$ and $C_{\hat{s}\hat{s}'}$. *Moving averages* can be computed using a $\eta_i$ variable in the range $[0, 1]$ to make this process less costly. Estimation of variables of distributions from the sufficient statistics are as follows:

$$\Psi_{\hat{s}\hat{s}'} \approx C_{\hat{s}\hat{s}'} / \sum_{k} C_{k\hat{s}'} \tag{3.20}$$

$$\mu_{\hat{s}} \approx m_{\hat{s}} \tag{3.21}$$

$$\Sigma_{\hat{s}} \approx \mathcal{V}_{\hat{s}} - m_{\hat{s}} m_{\hat{s}'}^{T} \tag{3.22}$$

The states $s_{1:\tau}^{(i)}$ must be known for the calculations of Equation 3.17, Equation 3.18 and Equation 3.19. If the variables $\Psi_{\hat{s}\hat{s}'}$, $\mu_{\hat{s}}$ and $\Sigma_{\hat{s}}$ are known, $s_{1:\tau}^{(i)}$ can be estimated using the following recursive equation with the *Viterbi algorithm*:

$$\underbrace{\max_{s_{1:t}^{(i)}} p\left(s_{1:t}^{(i)}, x_{1:t}^{(i)}\right)}_{\phi\left(s_t^{(i)}\right)} = \left(\max_{s_t^{(i)}} p\left(x_t^{(i)} \mid s_t^{(i)}\right) p\left(s_t^{(i)} \mid s_{t-1}^{(i)}\right)\right) \underbrace{\max_{s_{1:t-1}^{(i)}} p\left(s_{1:t-1}^{(i)}, x_{1:t-1}^{(i)}\right)}_{\phi\left(s_{t-1}^{(i)}\right)} \tag{3.23}$$

Therefore, the desired variables can be deduced by a recursive algorithm which first assumes the states constant and makes *maximization* over the variables and then assumes the variables constant and makes *maximization* over the states.

If there are *missing data*, the only thing that will change is the distribution of observations. Fortunately, one of the advantageous properties of the Gaussian distribution and the uniform distribution is that the *marginal probability* can be easily calculated. For example, in the case where observation $p_*$ is not observed, the distribution with Gaussian joint distribution will be transformed from the $\mu$ and $\Sigma$ into $(\mathcal{P} + \mathcal{R} - 1)$ dimensional Gaussian distributions obtained by subtracting all the rows and orders for the observation $p_*$.

### 3.2.2. Calculation of Predictive Distribution

In the previous section, the process of learning the normal working pattern of the system is performed. After that, the observed behavior should be evaluated with the

learned model. In our opinion, calculating the *predictive distribution* to do this task is one of the most efficient ways. One of the important points is that instead of the predictive distribution $p(x_t \mid x_{1:t-1})$ of $x_t$, predictive distribution of the system driven parameters conditioned on the parameter which drives the system $p(r_t \mid p_t, x_{1:t-1})$ is more meaningful. Because the behavior of the system is how the system produces its output under given external conditions. The predictive distributions of the system driven observations, which are conditional on system driver parameters, are in fact the ratio of the predictive distribution of all observations to the predictive distribution of the system driver observations:

$$p\left(r_t \mid p_t, x_{1:t-1}\right) = \frac{p\left(x_t \mid x_{1:t-1}\right)}{p\left(p_t \mid x_{1:t-1}\right)} \tag{3.24}$$

Additionally, the predictive distributions of all observations and the predictive distributions of the system driver observations could be calculated by following recursive equations:

$$\begin{aligned}
p\left(x_t \mid x_{1:t-1}\right) &= \sum_{s_t} p\left(x_t, s_t \mid x_{1:t-1}\right) && (3.25)\\
&= \sum_{s_t} p\left(x_t \mid s_t\right) p\left(s_t \mid x_{1:t-1}\right) && (3.26)\\
&= \sum_{s_t} p\left(x_t \mid s_t\right) \sum_{s_{t-1}} p\left(s_t, s_{t-1} \mid x_{1:t-1}\right) && (3.27)\\
&= \sum_{s_t} p\left(x_t \mid s_t\right) \sum_{s_{t-1}} p\left(s_t \mid s_{t-1}\right) p\left(s_{t-1} \mid x_{1:t-1}\right) && (3.28)\\
p\left(p_t \mid x_{1:t-1}\right) &= \sum_{s_t} p\left(p_t \mid s_t\right) \sum_{s_{t-1}} p\left(s_t \mid s_{t-1}\right) p\left(s_{t-1} \mid x_{1:t-1}\right) && (3.29)
\end{aligned}$$

In order to find posterior probabilities $p\left(s_{t-1} \mid x_{1:t-1}\right)$ in these equations, it is sufficient to find and normalize *forward probabilities*. The forward probabilities can also be found in the following recursive equation:

$$\underbrace{p\left(s_t, x_{1:t}\right)}_{\alpha(s_t)} = p\left(x_t \mid s_t\right) \sum_{s_{t-1}} p\left(s_t \mid s_{t-1}\right) \underbrace{p\left(s_{t-1}, x_{1:t-1}\right)}_{\alpha(s_{t-1})} \tag{3.30}$$

In order to generate anomaly prediction $\mathcal{E}_t$ from predictive distribution, the performance of the predictive distribution calculated by HMM is compared with the performance of the uniform distribution. If the uniform distribution is better than HMM, $\mathcal{E}_t$ will be higher. We foresee these cases as anomalies:

$$\mathcal{E}_t = \frac{c}{c + p\left(r_t \mid p_t, x_{1:t-1}\right)} \tag{3.31}$$

### 3.3. Deep Learning Sequence Models for Anomaly Detection

In this section, deep learning models are developed for anomaly detection in time series. The developed models in this section are also expected to handle missing observations, and they are developed to find deterioration and anomalies in the system. Since usually there is no deterioration or anomaly label, these model should be an unsupervised model. Therefore, the model uses the system outputs $r$ as a label and trained under system inputs $p$ and system outputs $r$. In other words, the model learns the system. The developed model will perform analysis on time series; so, RNNs and LSTMs, which are deep learning sequence models, are used. We will decide on the anomalies according to the conformity of the observations to the outputs of the model.

In this approach, we designed deep models that learn the relationship between the system driver parameters (input) and the system driven parameters (output), and the model also learns patterns of the system from the given input [48, 49]. Both inputs and outputs are multidimensional and expressed as a vector with the lenghts of $\mathcal{P}$ and $\mathcal{R}$ as $p_t = \begin{bmatrix} p_t^1 & \cdots & p_t^{\mathcal{P}} \end{bmatrix}^T$ and $r_t = \begin{bmatrix} r_t^1 & \cdots & r_t^{\mathcal{R}} \end{bmatrix}^T$, respectively. Therefore the developed models should generate system driven observations from the system driver

observations, so it will act as a kind of generative model and learn to generate outputs regarding the system behavior [23]. Because deep learning sequence models can learn system dynamics in more detail, we prefer to look at the error of the outputs instead of defining an outlier state as in HMM.

### 3.3.1. Forward Propagation in RNN

The inputs and outputs of the system are defined. Now, in this step, the model is expected to learn the pattern between inputs and outputs. Since these networks are expensive to perform over whole dataset, random sequence with length of $\tau$ can be selected at each training epoch. Let $p_{1:\tau}^{(i)}$ be the $i$th selected input sequence and $r_{1:\tau}^{(i)}$ is the corresponding output sequence. The model will calculate the predicted output sequence $\hat{r}_{1:\tau}^{(i)}$. Forward propagation of RNN for this model is as follows:

$$s_t' = \mathcal{W}_{ss} s_{t-1} + \mathcal{W}_{sp} p_t^{(i)} + b_s \tag{3.32}$$

$$s_t = h\left(s_t'\right) \tag{3.33}$$

$$\hat{r}_t^{(i)} = \mathcal{W}_{rs} s_t + c_y \tag{3.34}$$

Where $b_s$ corresponds to bias terms in the hidden layer of the NN and $c_y$ corresponds to bias term in the output layer. $\mathcal{W}$s are the model weights that are learned through training. This forward propagation is calculated from $1:\tau$ for the $i$th subsequence at each epoch. There will be $\mathcal{T}/\tau$ subsequence at each training step. $\mathcal{T}$ corresponds to the number of total data instance.

### 3.3.2. Forward Propagation in LSTM

In a similar way with the RNN, random sequence with length of $\tau$ can be selected at each training epoch. Let $p_{1:\tau}^{(i)}$ be the $i$th selected input sequence and $r_{1:\tau}^{(i)}$ is the corresponding output sequence. The model will calculate the predicted output sequence $\hat{r}_{1:\tau}^{(i)}$, as in RNN. The forward propagation of LSTM for such a model is as follows:

$$
\begin{aligned}
f_t &= \sigma\left(\mathcal{W}^f_{ss} s_{t-1} + \mathcal{W}^f_{sp} p^{(i)}_t + b^f\right) && (3.35)\\
\hat{c}_t &= g\left(\mathcal{W}^c_{ss} s_{t-1} + \mathcal{W}^c_{sp} p^{(i)}_t + b^c\right) && (3.36)\\
i_t &= \sigma\left(\mathcal{W}^i_{ss} s_{t-1} + \mathcal{W}^i_{sp} p^{(i)}_t + b^i\right) && (3.37)\\
c_t &= f_t \odot c_{t-1} + i_t \odot \hat{c}_t && (3.38)\\
\hat{s}_t &= h\left(c_t\right) && (3.39)\\
o_t &= \sigma\left(\mathcal{W}^o_{ss} s_{t-1} + \mathcal{W}^o_{sp} p^{(i)}_t + b^o\right) && (3.40)\\
s_t &= o_t \odot \hat{s}_t && (3.41)\\
\hat{r}^{(i)}_t &= \mathcal{W}_{rs} s_t + c_r && (3.42)
\end{aligned}
$$

Where $b$s correspond to the bias term in the gates and hidden layers, and $c_y$ is the bias at the output layer. $f_t, i_t$ and $o_t$ corresponds to forget gate, input gate, and output gate, respectively. $\hat{c}$ is the proposed cell state, while $c_t$ is the cell state of the network. In a similar manner, $\hat{s}_t$ is the proposed hidden state for the network, while $s_t$ is the filtered and resulted in the hidden state of the network.

### 3.3.3. Learning

After the calculation of forward propagation and generation of the predictions $\hat{r}^{(i)}_{1:\tau}$, the loss should be calculated. We could adjust the model by changing the model parameters $\boldsymbol{\mathcal{W}}$. So, the objective is the minimization of the loss concerning $\boldsymbol{\mathcal{W}}$ which is as follows:

$$
\arg\min_{\boldsymbol{\mathcal{W}}} \mathcal{L}\left(r^{(i)}_{1:\tau} \| \hat{r}^{(i)}_{1:\tau}\right) \tag{3.43}
$$

The loss will be calculated and then the weights, $\boldsymbol{\mathcal{W}}$, are updated with the time series specific gradient descent algorithm which is backpropagation through time (BPTT) at each time-step [46]. The forward propagation and the weight update procedure with

BPTT algorithm will continue with the new sub-sequences $i'$ at each epoch until the convergence. Thus, the model parameters, $\boldsymbol{\mathcal{W}}$, are learned. As a result of this procedure, the model is able to reproduce system outputs.

### 3.3.4. Prediction and Anomaly Score

We developed models which learn the system to be analyzed. The next step is to find anomalies. After the training phase, let us assume that there is a sequence, with a length $\mathcal{T}$, and we represent it as $p_{1:\mathcal{T}}$. Then the model will calculate the predicted outputs $\hat{r}_{1:\mathcal{T}}$ with the learned weights and bias terms as in the equations 3.42 and 3.34. Then, to detect the anomalies, the loss $\mathcal{L}$ between the observed data $r_{1:\mathcal{T}}$ and model predictions $\hat{r}_{1:\mathcal{T}}$ should be calculated separately for each time step.

$$\mathcal{E}'_t \;=\; \mathcal{L}\left(r_t \| \hat{r}_t\right) \tag{3.44}$$

We have linear units at the output layer; therefore, our predictions are the result of that linear units. So, we need to select appropriate loss function. *Root mean square error* (RMSE), *mean square error* (MSE) and *mean absolute error* (MAE) are the most appropriate loss functions. However, since system outputs are multi-dimensional and some of the sensors return numerically large values and dominate the error, such loss functions, especially MSE and RMSE, have drawbacks. One way to handle this drawback is to use another distance metric, which measures the percentage error. This metric is called as a *mean absolute percentage error* (MAPE). However, it brings out other problems such as high error rate at the points where observations are too small. We have one additional problem during training, which are the anomalies in the training set. The model we developed performs unsupervised learning. Therefore, the model should not be affected by anomalous observations during training. This could be achieved by applying robust optimization techniques at the loss function [50]. So, we need a loss function that will not be influenced by the anomalies; which is *robust loss function* [51]. The solution of this problem is in the robust statistics literature [52], and we use the *Tukey's biweight* loss function as a robust optimization method for a

deep regression [51]. We compare different loss functions and detailed analysis of the loss functions could be found in Appendix A.

We normalize all the inputs and outputs and model calculates the results and errors on the normalized data. When doing reconstruction, we unnormalize it. When it comes to producing an anomaly score, we are applying the following steps:

(i) Calculate the error of each observation and create an unnormalized anomaly score from the error for each observation.
(ii) Calculate the average loss of the fake observations which are uniformly selected within the range of the space of the feature space.
(iii) Compare unnormalized error with the uniformly selected samples error and create normalized anomaly score.

Let $c$ represents the average loss of the uniformly selected random observations over observations space, and let $\mathcal{E}'_t$ shows the unnormalized error for the $t$th observation. Then we can obtain normalized anomaly score with the following equation:

$$\mathcal{E}_t = \frac{\mathcal{E}'_t}{c + \mathcal{E}'_t} \tag{3.45}$$

If there is a higher probability of error for the specified time step, then corresponding observation will be marked as an anomaly. For the detection of collective anomaly [27], moving averages of the resulting errors can be computed using a $\eta$ variable in the range $[0, 1]$, which is as follows:

$$\hat{\mathcal{E}}_t = \hat{\mathcal{E}}_{t-1} \times \eta + \mathcal{E}_t \times (1 - \eta) \tag{3.46}$$

In this section, we developed two similar models which use RNN and LSTM, respectively. If there are long-term dependencies of the system, LSTM is expected to give more accurate results, if not, RNN is expected to give similar results.

# 4. EXPERIMENTS AND RESULTS

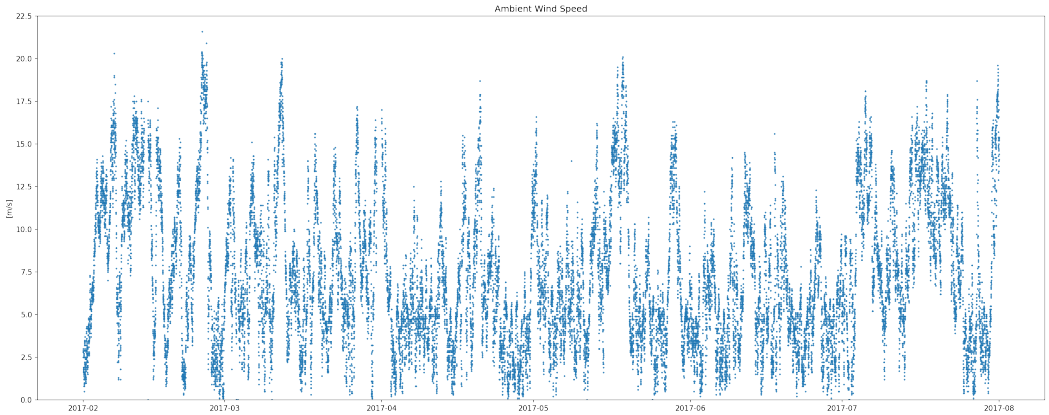
In this chapter, we present applications of the anomaly detection models described in Chapter 3. In the beginning, we apply GMM, which does not take into account the sequential feature of the data, and then we implement the HMM, which is a probabilistic sequence model. Then, we expand our work with the application of deep learning sequence models. We perform experiments and compare the results with models constructed on RNN and LSTM, respectively.

In this study and the experiments presented in this chapter, data collected from the wind turbines of Borusan in Bandırma were used.

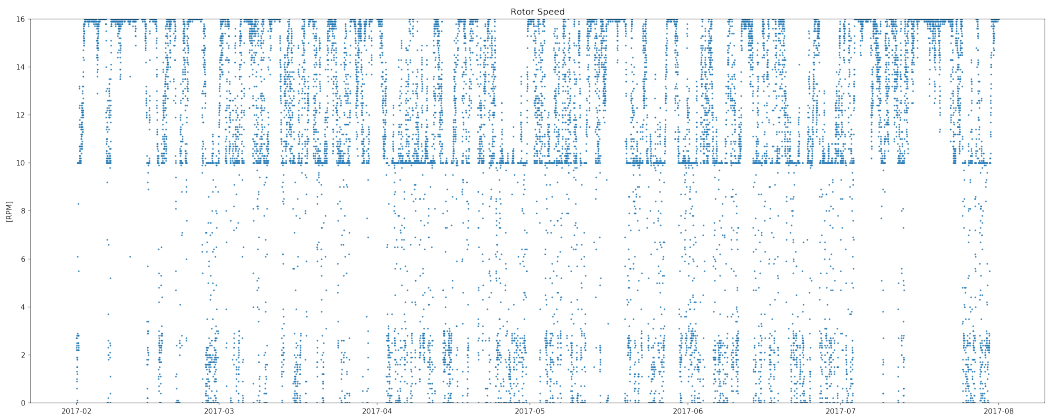
## 4.1. Wind Turbine Dataset

Borusan Vestas V90-3MW wind turbines dataset was created from 20 different wind turbines which are identical. For each wind turbine, there are observations collected at 25737 different consecutive time-steps. Each observation is the collection of wind speed, rotor speed, and generated power values. This data set was created with data collected at 10-minute intervals from February to July. There are only two reported anomalies in the data set. However, it is known that there are unreported and undetected anomalies. Additionally, this dataset contains some Null and incorrect sensory information because of the faulty sensor measurements. These errors occur randomly and do not continue. Therefore, we know that there is no accurate data in the data set and that there are errors and missing data in the observations. The visualization of the wind, rotor, and power data could be found in Figure 4.1.

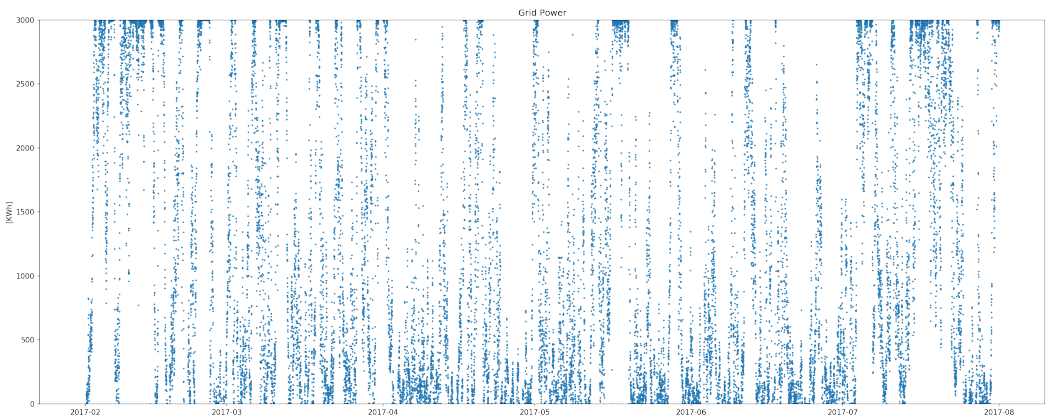
In this dataset, wind speed can be seen as an input to the system while the rotor speed and the generated power are the outputs of the system. However, it can be seen from Figure 4.1 that there is not a direct relationship between these features. The reason behind this phenomena is that previous observations and working condition of the turbine affect the current time observations because of the physical relationships




(a) Wind speed observations as a time-series




(b) Rotor speed observations as a time-series




(c) Generated grid power observations as a time series

Figure 4.1. Individual sequential observations for the sensory information

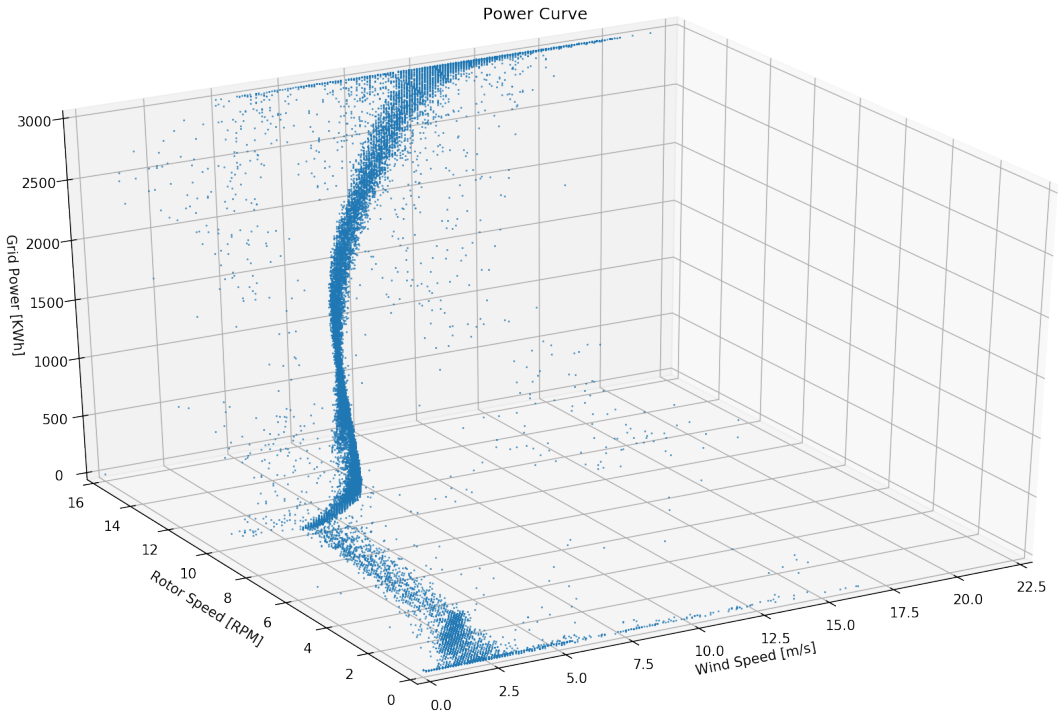



Figure 4.2. The relationship between the wind speed, rotor speed and grid power

in the mechanical system of the wind turbines. Therefore, the outputs cannot be generated directly according to input data, and it requires the system state, which indicates how the system reacts to the given input. The interaction graph which is obtained from a limited time interval and does not consider the time series feature can be seen in Figure 4.2.

### 4.1.1. Modeling The Power Curve of Wind Turbine

The power curve of wind turbines defines the relationship of a wind speed or rotor speed, or both, to the amount of power generation. Since, the physical machinery has a sophisticated control mechanism, as well as environmental variations that are not directly measurable, makes this relationship more complicated. Therefore, we have to create a model that works under different circumstances according to basic working principals of the wind turbines. For this purpose, we use Bayesian approaches to generate the power curve of the wind turbine efficiently.

There are different approaches to the modeling of the power curve. Some of the works use wind speed and grid power to evaluate the power curve [25] and some other uses rotor speed and grid power to evaluate power curve [26]. On the other hand, we use both rotor speed and wind speed because only wind cannot give too much information because it has too much uncertainty and there is a lot of sudden changes in the wind speed. On the other hand, rotor speed itself again cannot give desired results because there is break points in the rotor speed - power curve graph. In addition to that, such a discontinuity is not desired in such a model. Therefore we use both wind speed and rotor speed to analyze power curve model. By this selection, we see that the discontinuity in the rotor speed - grid power model is gone, and the high variance in the wind speed - power curve is also reduced. Therefore increasing the dimensionality of the power curve significantly strengthens our hands in terms of better analysis.

The power curve shows us the critical features of the wind turbine machinery system. However, although the wind speed, rotor speed, and grid power are included, the turbine still contains some discontinuity in the power curve as a result of some states of the turbine. Since there is a gearbox in the turbine, we expect that kind of relationship, and we can see the location of this change points from the power curve.

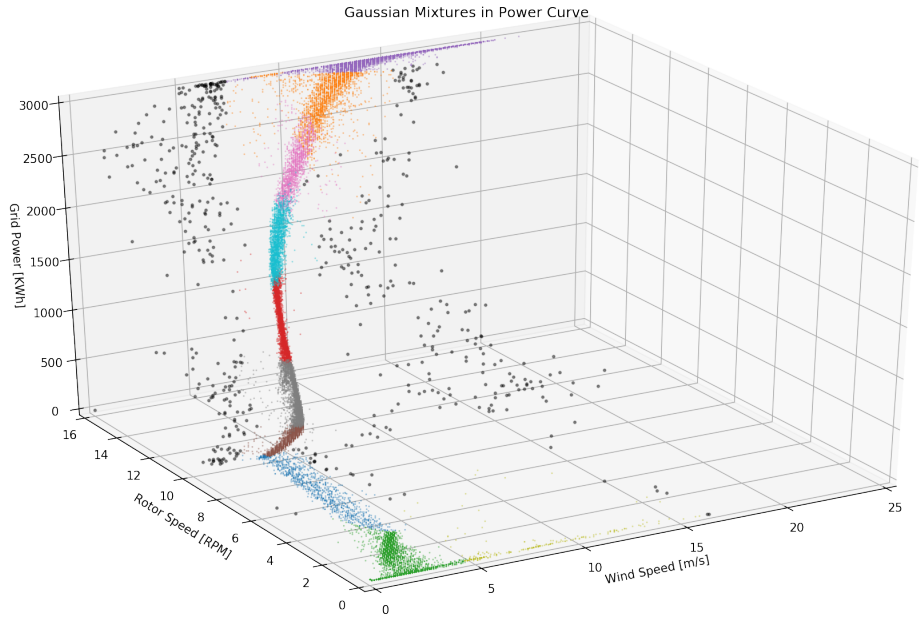
In addition to the previous details, we can also add the time factor to the account. When we do this, we create the power curve using the incoming data sequence. On this page, we can analyze not only the power curve but also the trends of change on the power curve.

We aim to find the optimum power curve for each turbine individually because the physical machines are complex systems, and even if they work with precisely the same mechanism, they can show different production values. The power curve includes the main line that the turbine is likely to generate and the variance that occurs. It is important to note that since wind turbines are rotating systems, the variance of the production will increase with the rotor speed. Therefore our model should take into account that property.

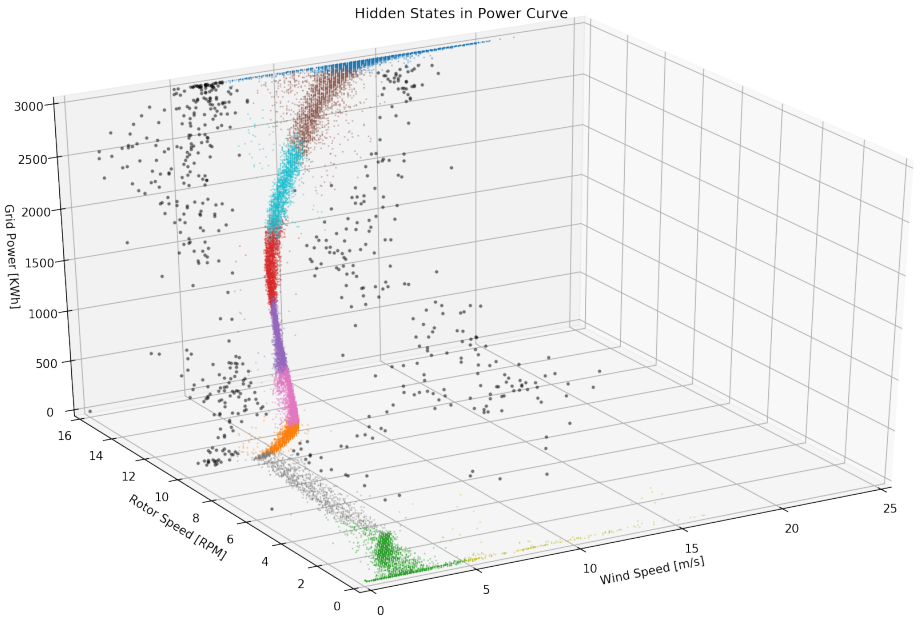
### 4.1.2. Experiments with the Probabilistic Models

We started this study by reconstruction of the power curves of the wind turbines using *Gaussian Mixture Model*(GMM). This model learns the power curve of the wind turbine as a mixture of Gaussian distribution over the available data. At this point, we assume that our observations include both turbine states and anomalies. Therefore we assume that, in the model, there are 10 different Gaussian distributions for turbine operation states which should be learned and there is 1 Gaussian distribution with the infinite variance, uniform over output space, for sensory information errors or anomalies. These Gaussian distributions are then learned using the EM algorithm, except the distribution corresponds to anomalies. Once the distributions are learned, we can determine which distribution the incoming data belongs to, and what observations are anomalies. Therefore, according to this model, the incoming data has a predictive error rate. This model, which is more straightforward than HMM, does not take into account the time effect and how the consecutive observations should behave is not taken into account. Therefore, the Gaussian distributions in the model are learned without this knowledge. Since the incoming data is not analyzed as a time series, the model is more susceptible to faulty data coming from the sensors and can not catch the faulty transitions between states. The mixtures are shown in Figure 4.3(a). Moreover, the model may be insufficient to detect unexpected fluctuations in the power output of the turbine.

In the expanse of this study, the power curves of the wind turbines are constructed as time series using HMM. This model also learns the power curve of the wind turbine as a mixture of Gaussian distribution over the available data. Therefore we assume that, as in GMM, there are 10 different Gaussian distributions for turbine operation states which should be learned and there is 1 Gaussian distribution with the infinite variance, uniform over output space, for sensory information errors or anomalies. We design our model in such a way as to determine which distribution will come from which distribution. Therefore, the previous distribution will have an impact on the distribution of the observations. Therefore the learned mixtures of the Gaussian distributions of the states will be slightly different from that obtained in the GMM. The




(a) The mixture of Gaussians




(b) Distributions in the latent space

Figure 4.3. Comparison of the mixture of Gaussians in the GMM with the mixture of Gaussians in the HMM which defined on the power curve

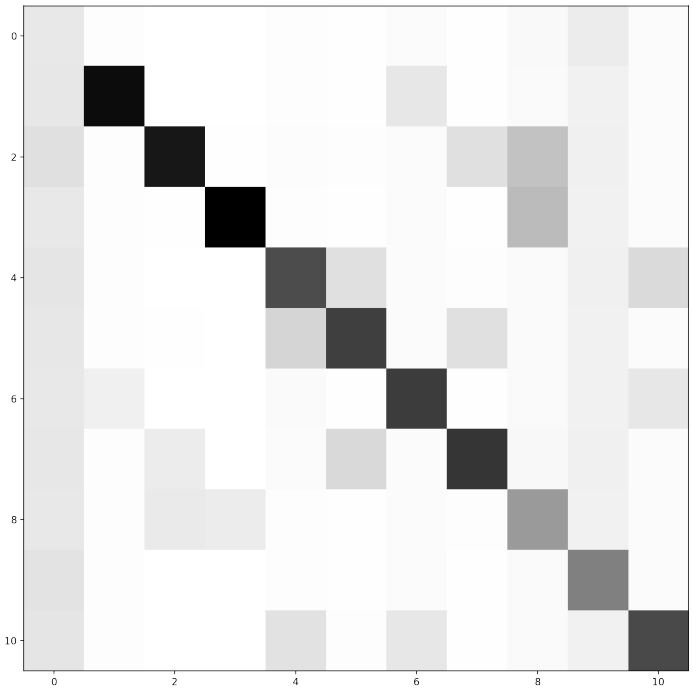
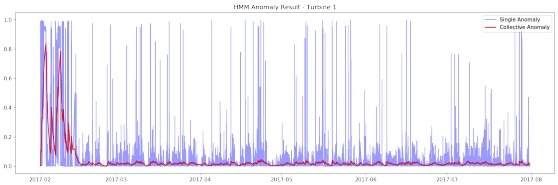
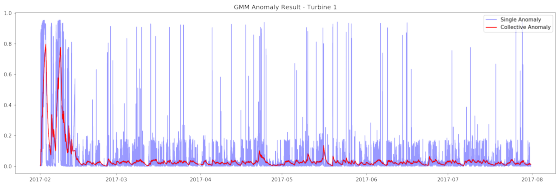



Figure 4.4. State transition diagram of the HMM

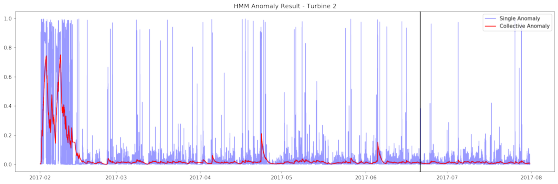
learned Gaussian mixtures by HMM are shown in Figure 4.3(b). The better modeling of the system, as can be seen in Figure 4.3 has resulted in more obvious mixtures. This model was trained by processing the sample mini-series of 144 measurements, which corresponds to one-day observations. During this training, the learning speed was chosen as $\eta_t = 1/(t+1)$. On the other hand, the state transition diagram of the HMM is shown in Figure 4.4. As can be seen from the transition diagram, it is possible to switch from any state to anomaly. Therefore, the HMM model is more tolerant to incorrect measurements from the detectors and can detect abnormal changes in the power curve without being affected by false observations as it learns the operation of the turbine. As a result of these studies, it can be seen that HMM can make error prediction more clear than GMM, according to Figure 4.5(c) and Figure 4.5(d). However, the most critical point of HMM is that it minimizes the false positive error prediction. In most cases, the GMM tends to produce false positive predictions, as shown in Figure 4.5(f), but HMM has achieved much better results in this regard. A detailed comparison between GMM and HMM model is given in Figure 4.5. Malfunctions are shown as black vertical lines in graphs. The collective anomaly result is cumulatively calculated from the anomaly forecast for each observation and converted into a warning signal.



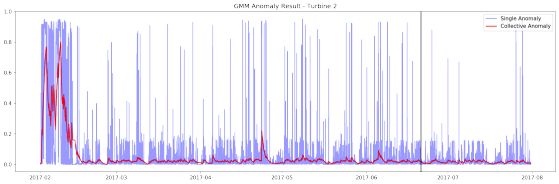

(a) The error prediction of HMM on the turbine which works under normal conditions



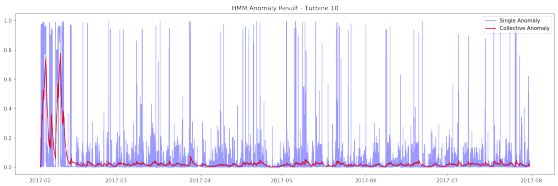

(b) The error prediction of GMM on the turbine which work under normal conditions



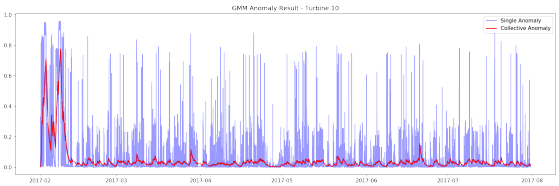

(c) HMM generates two clear warning before malfunction on the June




(d) Warnings of GMM are not clear as warning generated by HMM




(e) HMM does not predict error for the turbine which works under normal conditions




(f) GMMs tend to produce false positive warnings on the normally working turbine

Figure 4.5. Comparison of the performance of HMM with GMM

### 4.1.3. Experiments with the Deep Learning Sequence Models

In this part of the study, we are interested in deep learning models to detect anomalies. Therefore we apply such models to the power curve of the wind turbines to learn about the system dynamics. The main idea behind this study is to reconstruct the power curve of the wind turbines with developed models. Then, in the test phase, we will first reconstruct the power curve, and compare this reconstruction with the new observations and measure how wrong they are. This measurement will also determine our anomaly score. In the experimental setup, we basically compare the RNN network with LSTM network. While making this comparison, we will do our experiments by changing two additional parameters. The first parameter to examine the effect on the model is the depth of the network. For both RNN and LSTM, we compare the results with 1-layer network with 2-layer network (stacked) [23]. The other parameter to examine the effect on the model is loss function. We examine the effect of the L1 loss and MSE loss on the developed model. We focus on these two losses because both models have the rectified linear output layer, and these loss functions are more appropriate for such a model. In the rest of the experimental setup, we set all hyperparameters to the same for the healthy comparison. We set learning rate $\eta = 0.01$ for all of the network setups, and we train each network for 1000 number of the epoch. All the experimental setups use the same batch size, 144 observation time-step for one batch. In the experiments, both RNN and LSTM network has the number of hidden layer size 128.

In these experiments, we implement 12 different experiment setup. These setups include RNN or LSTM, 1-layer or 2-layer, and L1 loss or MSE loss or Tukey's biweight loss. Our experiments show that both RNN and LSTM setups have almost similar performance on the system. This is because wind turbine dataset does not contain long-term dependencies and to obtain good estimation we do not need to know information from the future. On the other side, experiments show us that stacked networks perform better on the power curves. This situation may be due to the complexity of the power curve. Finally, we see in the experiments that L1 loss is a better choice to model such data because data includes anomalous observations and MSE loss not robust for

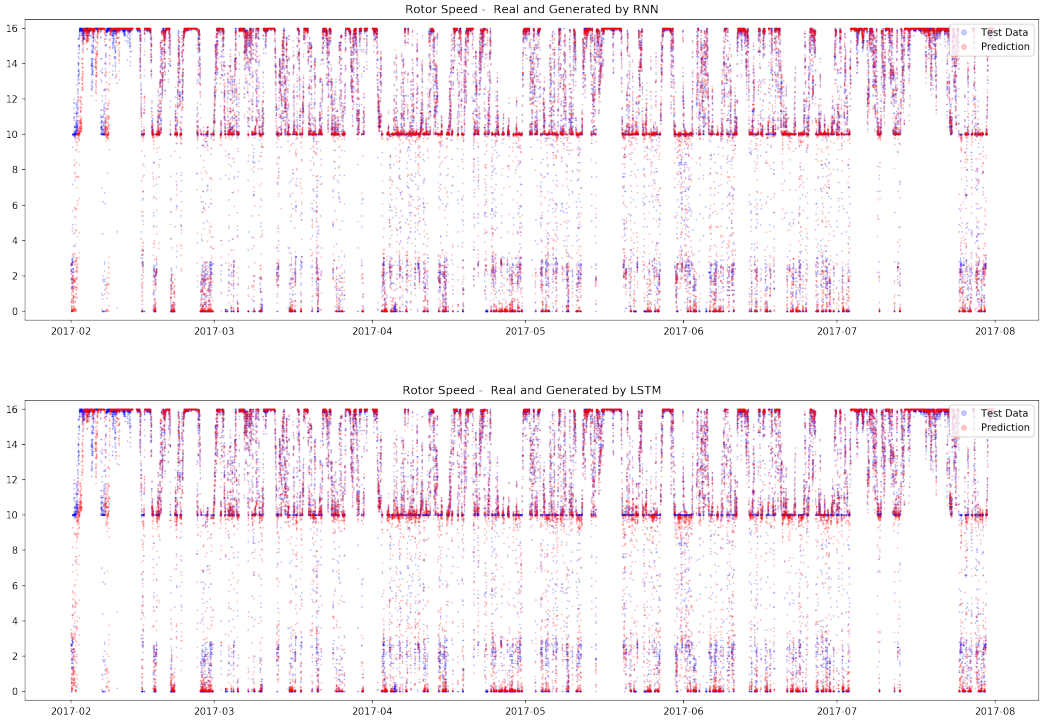



Figure 4.6. Predicted rotor speed observations with RNN and LSTM models which use Tukey's biweight loss

such observations while L1 does. Tukey's biweight loss, on the other hand, allow us to perform more robust optimizations and perform better on our task. These networks almost totally correctly estimate the generated power and rotor speed values from the given wind speed, as in Figure 4.6, Figure 4.7, Figure 4.8 and Figure 4.9. The first two figures are generated from the stacked RNN and stacked LSTM network with the Tuckey's biweight loss. Except for a small difference, both the RNN and LSTM model creates almost the same predictions on rotor speed and grid power values. The third figure includes generated power curves for all experimental setups. In all figures, blue dots represent the observations while red dots show the model outputs. In Figure 4.8, one can detailly analyze the performance of the 1-layer models with a different setting. Similarly, in Figure 4.9, we share the performance of stacked (2-layered) networks. Despite their small differences, all models produced acceptable results and found almost the same anomaly result. The anomaly scores are shown in Figure 4.10 and Figure 4.11. Black vertical lines in the figures represent the malfunction in turbines, and the system generates warning before the malfunction. Additionally, we know that at the February there is an iciness on the turbines. Models successfully detect these anomalies in all

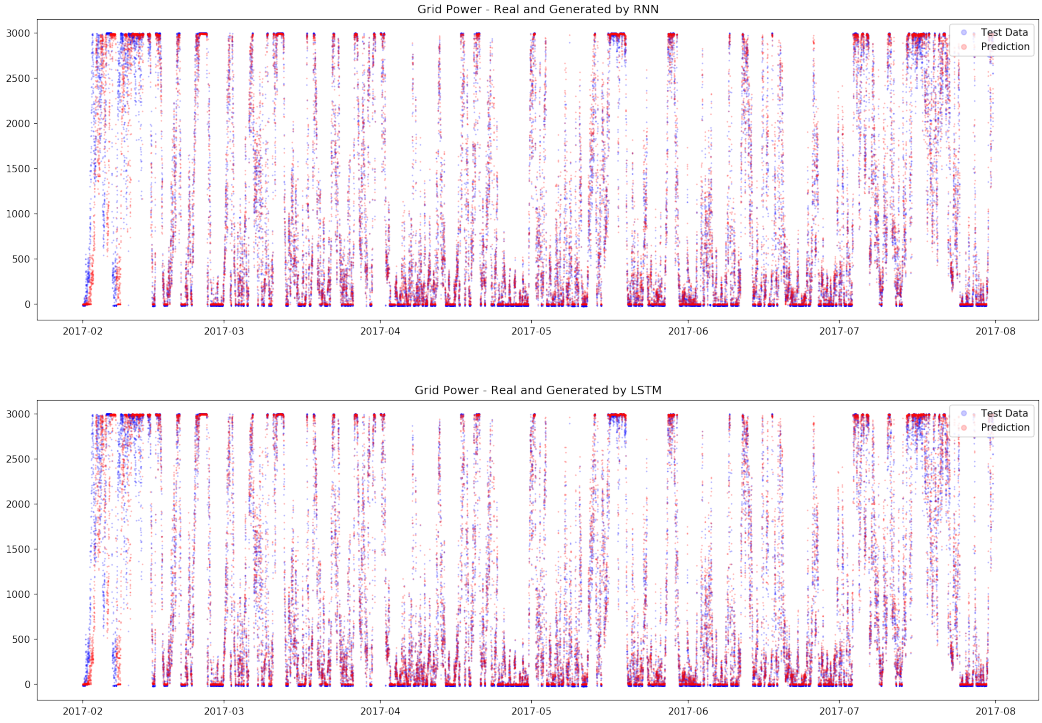
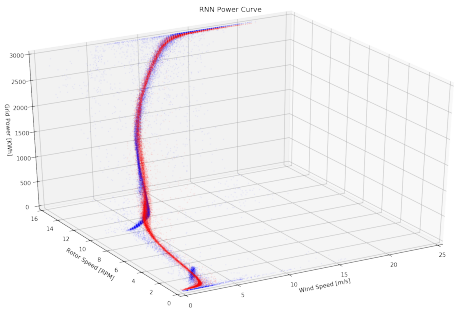
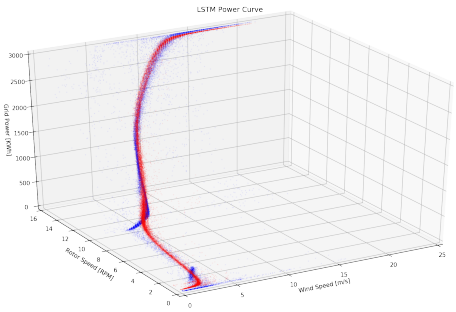



Figure 4.7. Predicted generated power observations with RNN and LSTM models which use Tukey's biweight loss

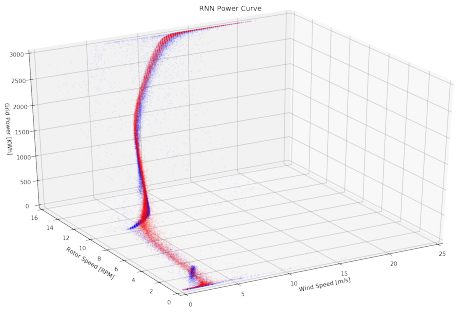
experiments. These figures show that the results are very similar to each other and also similar to the ones that we find with probabilistic models in Figure 4.5. Detailed analysis of the 1-layer networks can be found in Figure 4.10, while Figure 4.11 includes detailed analysis of the 2-layer (stacked) networks. In both experiments, blue dots represent the observations while red dots represent the predictions.



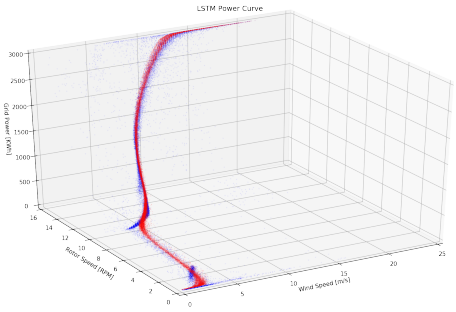

(a) RNN with MSE loss



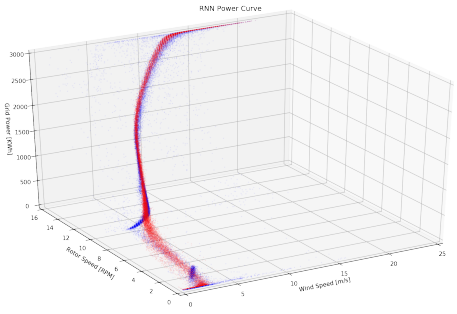

(b) LSTM with MSE loss



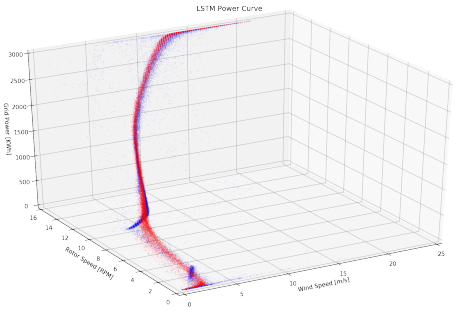

(c) RNN with L1 loss




(d) LSTM with L1 loss

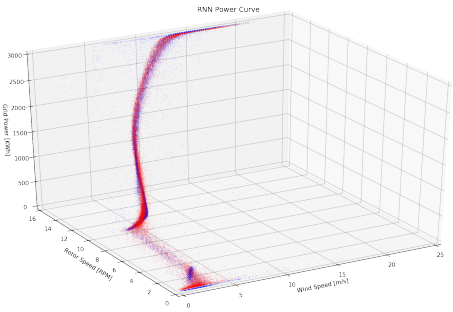



(e) RNN with Tukey's biweight loss

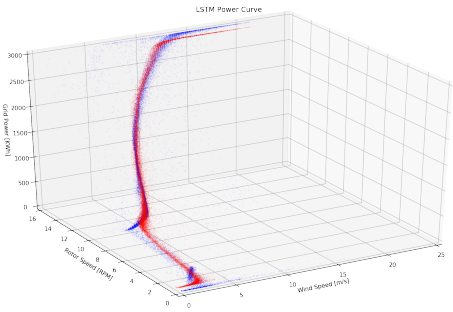



(f) LSTM with Tukey's biweight loss

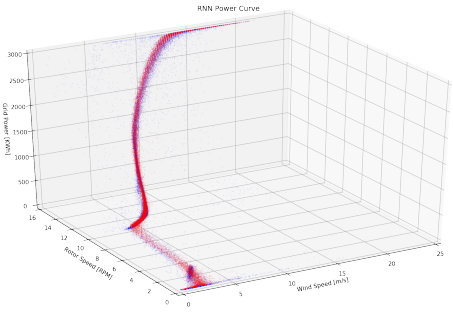
Figure 4.8. Generated from 1-layer network and observed power curves



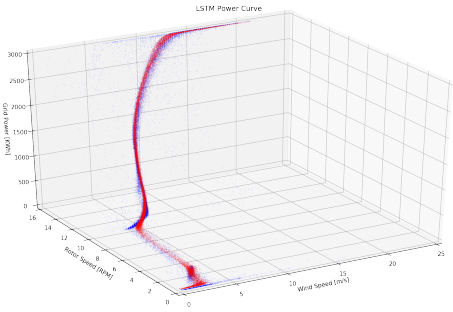

(a) RNN with MSE loss



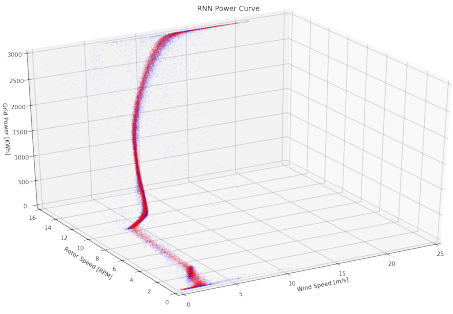

(b) LSTM with MSE loss



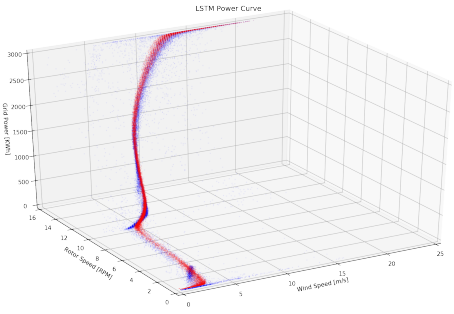

(c) RNN with L1 loss




(d) LSTM with L1 loss

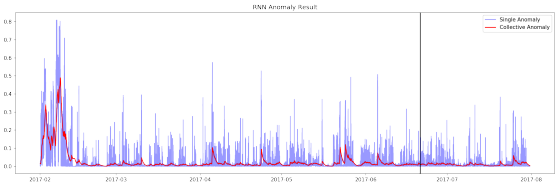



(e) RNN with Tukey's biweight loss

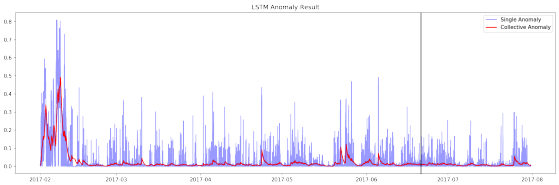



(f) LSTM with Tukey's biweight loss

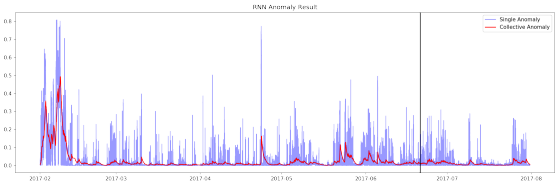
Figure 4.9. Generated from 2-layer network and observed power curves



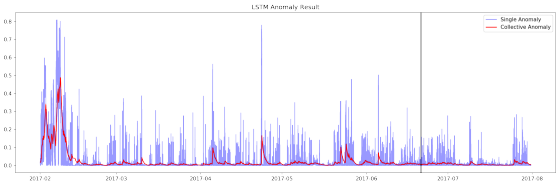

(a) RNN with MSE loss



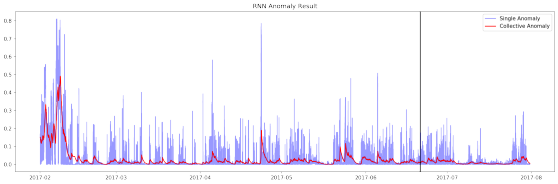

(b) LSTM with MSE loss



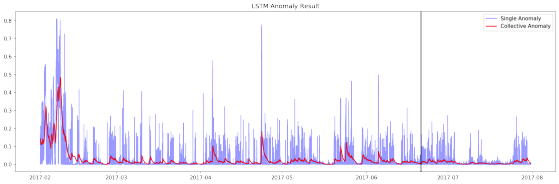

(c) RNN with L1 loss




(d) LSTM with L1 loss

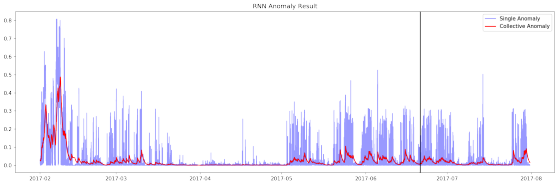



(e) RNN with Tukey's biweight loss

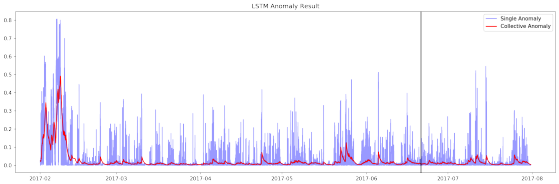



(f) LSTM with Tukey's biweight loss

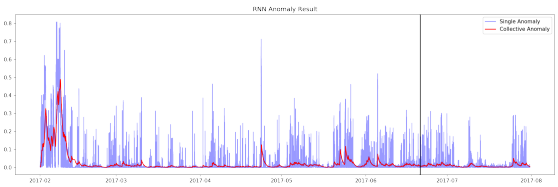
Figure 4.10. Anomaly scores of the 1-layered networks on test data.



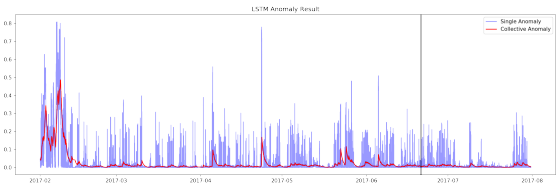

(a) RNN with MSE loss



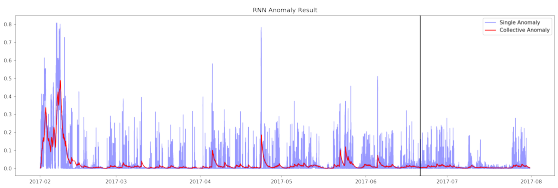

(b) LSTM with MSE loss



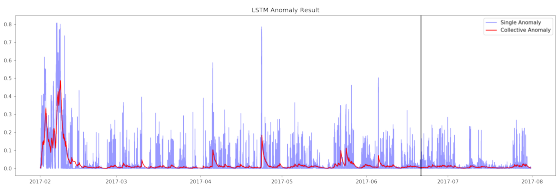

(c) RNN with L1 loss




(d) LSTM with L1 loss




(e) RNN with Tukey's biweight loss




(f) LSTM with Tukey's biweight loss

Figure 4.11. Anomaly scores of the 2-layered (stacked) networks on test data.

# 5. CONCLUSION AND FUTURE WORK

In this work, we developed models and algorithms for anomaly detection in time series data. We conducted our work in two different approaches, which are statistical algorithms and deep learning based sequence models. As a result of this work, the models that we developed can perform on the same datasets, and that can create a similar anomaly score.

As a statistical approach, we developed a special kind of hidden Markov model which can handle anomaly detection task on noisy real-world data. We assign one additional hidden state, 'outlier state' to the latent space to explain noisy, unwanted observations. This is similar to what we did in the GMM. In GMM, we defined Gaussian mixtures, which correspond to latent states of the system, and we represented an additional Gaussian mixture with has high variance, which corresponds to 'outlier state' in HMM. However, in GMM, we can track anomalies only from the outlier state. If an observation falls into an outlier state, then the model arises an anomaly error. This approach has the following main drawbacks which we solved with HMM;

(i) The observations in the outlier state are often not outliers of the system we seek to detect. The observations in this state are mostly composed of errors in the observations.
(ii) GMM can not track the state transitions. Therefore, if unexpected observations occur, GMM cannot detect it.

The deep learning based models we proposed are constructed with RNN and LSTM. This part of our study emerged by combining specific parts of two studies [23, 27]. Malhotra et.al. in [23] performed anomaly detection method on cyclic data; on the other hand, Bontemps et.al. in [27] tried to identify the collective anomalies. In the light of these studies, we developed an anomaly detection model which first learns the patterns of the system and then evaluates the new coming observations considering the previous observations. This model, similar to HMM, track more complex state

transition, and therefore, it can catch the collective anomalies. We were able to observe the effects of the degree of past dependence on the system. Since LSTMs are very successful in capturing past links, we could have achieved much more successful results with LSTMs when the system has such a dynamic.

As we demonstrated in our experiments, the proposed models are powerful, flexible, and yield good results in an arguably challenging problem. Nonetheless, both models and applications can be further improved in many respects. Possible future research directions are as follows:

(i) Although HMM model is successful in detecting anomaly states, it is not equally successful in capturing the change of system over time. Therefore, over time, its performance may decline.

(ii) LSTM model is successful in capturing collective anomalies but can be sensitive to missing data, which in some cases may adversely affect system performance.

(iii) The application areas of the models could be expanded with new datasets, which requires the track of the changes in normal behavior.

(iv) To solve the two problems mentioned above, a more complex model can be created including the combination of LSTM and HMM or the particle filter.

(v) Coupled anomaly detection is planned to be developed. Conduction of anomalies in the systems by looking at other parallel systems will improve the analysis and anomaly warnings.

# REFERENCES


1. Chandola, V., A. Banerjee and V. Kumar, "Anomaly detection: A survey", *ACM computing surveys (CSUR)*, Vol. 41, No. 3, p. 15, 2009.

2. Hodge, V. and J. Austin, "A survey of outlier detection methodologies", *Artificial intelligence review*, Vol. 22, No. 2, pp. 85–126, 2004.

3. Mehra, R. K. and J. Peschon, "An innovations approach to fault detection and diagnosis in dynamic systems", *Automatica*, Vol. 7, No. 5, pp. 637–640, 1971.

4. Aleskerov, E., B. Freisleben and B. Rao, "Cardwatch: A neural network based database mining system for credit card fraud detection", *Proceedings of the IEEE/IAFE 1997 computational intelligence for financial engineering (CIFEr)*, pp. 220–226, IEEE, 1997.

5. Spence, C., L. Parra and P. Sajda, "Detection, synthesis and compression in mammographic image analysis with a hierarchical image probability model", *Proceedings IEEE Workshop on Mathematical Methods in Biomedical Image Analysis (MMBIA 2001)*, pp. 3–10, IEEE, 2001.

6. Barford, P., J. Kline, D. Plonka and A. Ron, "A signal analysis of network traffic anomalies", *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pp. 71–82, ACM, 2002.

7. Teng, H. S., K. Chen and S. C. Lu, "Adaptive real-time anomaly detection using inductively generated sequential patterns", *Proceedings. 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 278–284, IEEE, 1990.

8. Rousseeuw, P. J. and A. M. Leroy, *Robust regression and outlier detection*, Vol. 589, John wiley & sons, 2005.

9. Markou, M. and S. Singh, "Novelty detection: a review—part 1: statistical approaches", *Signal processing*, Vol. 83, No. 12, pp. 2481–2497, 2003.

10. Markou, M. and S. Singh, "Novelty detection: a review—part 2: neural network based approaches", *Signal processing*, Vol. 83, No. 12, pp. 2499–2521, 2003.

11. Ranshous, S., S. Shen, D. Koutra, S. Harenberg, C. Faloutsos and N. F. Samatova, "Anomaly detection in dynamic networks: a survey", *Wiley Interdisciplinary Reviews: Computational Statistics*, Vol. 7, No. 3, pp. 223–247, 2015.

12. Song, X., M. Wu, C. Jermaine and S. Ranka, "Conditional anomaly detection", *IEEE Trans. Knowl. Data Eng.*, Vol. 19, No. 5, pp. 631–645, 2007.

13. Chandola, V., A. Banerjee and V. Kumar, "Anomaly detection for discrete sequences: A survey", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 24, No. 5, pp. 823–839, 2012.

14. Gupta, M., J. Gao, C. C. Aggarwal and J. Han, "Outlier detection for temporal data: A survey", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 26, No. 9, pp. 2250–2267, 2014.

15. Chandola, V., V. Mithal and V. Kumar, "Comparing anomaly detection techniques for sequence data", *Technical Report 08-021*, 2008.

16. Harvey, A. C., *Forecasting, structural time series models and the Kalman filter*, Cambridge university press, 1990.

17. Das, S., *Time series analysis*, Princeton University Press, Princeton, NJ, 1994.

18. Brockwell, P. J., R. A. Davis and M. V. Calder, *Introduction to time series and forecasting*, Vol. 2, Springer, 2002.

19. Barber, D., A. T. Cemgil and S. Chiappa, *Bayesian time series models*, Cambridge University Press, 2011.

20. Barber, D. and A. T. Cemgil, "Graphical models for time-series", *IEEE Signal Processing Magazine*, Vol. 27, No. 6, pp. 18–28, 2010.

21. Rabiner, L. R., "A tutorial on hidden Markov models and selected applications in speech recognition", *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257–286, 1989.

22. Längkvist, M., L. Karlsson and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling", *Pattern Recognition Letters*, Vol. 42, pp. 11–24, 2014.

23. Malhotra, P., L. Vig, G. Shroff and P. Agarwal, "Long short term memory networks for anomaly detection in time series", *Proceedings*, p. 89, Presses universitaires de Louvain, 2015.

24. Srivastava, A., A. Kundu, S. Sural and A. Majumdar, "Credit card fraud detection using hidden Markov model", *IEEE Transactions on dependable and secure computing*, Vol. 5, No. 1, pp. 37–48, 2008.

25. Ouyang, T., A. Kusiak and Y. He, "Modeling wind-turbine power curve: A data partitioning and mining approach", *Renewable Energy*, Vol. 102, pp. 1–8, 2017.

26. Romero, A., Y. Lage, S. Soua, B. Wang and T.-H. Gan, "Vestas V90-3MW wind turbine gearbox health assessment using a vibration-based condition monitoring system", *Shock and Vibration*, Vol. 2016, 2016.

27. Bontemps, L., J. McDermott, N.-A. Le-Khac *et al.*, "Collective anomaly detection based on long short-term memory recurrent neural networks", *International Conference on Future Data and Security Engineering*, pp. 141–152, Springer, 2016.

28. Hawkins, S., H. He, G. Williams and R. Baxter, "Outlier detection using replicator neural networks", *International Conference on Data Warehousing and Knowledge Discovery*, pp. 170–180, Springer, 2002.

29. Tan, P.-N., *Introduction to data mining*, Pearson Education India, 2018.

30. Hoadley, B., "Asymptotic properties of maximum likelihood estimators for the independent not identically distributed case", *The Annals of mathematical statistics*, pp. 1977–1991, 1971.

31. Barber, D., *Bayesian reasoning and machine learning*, Cambridge University Press, 2012.

32. Rauch, H. E., C. Striebel and F. Tung, "Maximum likelihood estimates of linear dynamic systems", *AIAA journal*, Vol. 3, No. 8, pp. 1445–1450, 1965.

33. Viterbi, A., "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", *IEEE transactions on Information Theory*, Vol. 13, No. 2, pp. 260–269, 1967.

34. Forney, G. D., "The viterbi algorithm", *Proceedings of the IEEE*, Vol. 61, No. 3, pp. 268–278, 1973.

35. Gauvain, J.-L. and C.-H. Lee, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains", *IEEE transactions on speech and audio processing*, Vol. 2, No. 2, pp. 291–298, 1994.

36. Dempster, A. P., N. M. Laird and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society: Series B (Methodological)*, Vol. 39, No. 1, pp. 1–22, 1977.

37. Kullback, S. and R. A. Leibler, "On information and sufficiency", *The annals of mathematical statistics*, Vol. 22, No. 1, pp. 79–86, 1951.

38. Tu, S., "Derivation of Baum-Welch Algorithm for Hidden Markov Models", *Citeseer*, 2015.

39. Bishop, C. M., *Pattern recognition and machine learning*, springer, 2006.

40. Goodfellow, I., Y. Bengio, A. Courville and Y. Bengio, *Deep learning*, Vol. 1, MIT press Cambridge, 2016.

41. Graves, A., "Supervised sequence labelling", *Supervised sequence labelling with recurrent neural networks*, pp. 5–13, Springer, 2012.

42. Hochreiter, S., "The vanishing gradient problem during learning recurrent neural nets and problem solutions", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 6, No. 02, pp. 107–116, 1998.

43. Hochreiter, S. and J. Schmidhuber, "Long short-term memory", *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

44. Olah, C., *Understanding lstm networks*, 2015, `http://colah.github.io/posts/2015-08-Understanding-LSTMs`, accessed at May 2019.

45. Greff, K., R. K. Srivastava, J. Koutník, B. R. Steunebrink and J. Schmidhuber, "LSTM: A search space odyssey", *IEEE transactions on neural networks and learning systems*, Vol. 28, No. 10, pp. 2222–2232, 2017.

46. Werbos, P. J. *et al.*, "Backpropagation through time: what it does and how to do it", *Proceedings of the IEEE*, Vol. 78, No. 10, pp. 1550–1560, 1990.

47. Poyraz, O., M. B. Kurutmaz, A. T. Cemgil and S. Selamoğlu, "Anomaly detection on wind turbines", *2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, IEEE, 2018.

48. Graves, A., A.-r. Mohamed and G. Hinton, "Speech recognition with deep recurrent neural networks", *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649, IEEE, 2013.

49. Pascanu, R., C. Gulcehre, K. Cho and Y. Bengio, "How to construct deep recurrent neural networks", *arXiv preprint arXiv:1312.6026*, 2013.

50. Breheny, P., *Robust regression*, 2012, `http://web.as.uky.edu/statistics/users/pbreheny/764-F11/notes/12-1.pdf`, accessed at May 2019.

51. Belagiannis, V., C. Rupprecht, G. Carneiro and N. Navab, "Robust optimization for deep regression", *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2830–2838, 2015.

52. Black, M. J. and A. Rangarajan, "On the unification of line processes, outlier rejection, and robust statistics with applications in early vision", *International Journal of Computer Vision*, Vol. 19, No. 1, pp. 57–91, 1996.

# APPENDIX A: COMPARISON OF THE LOSS FUNCTIONS

Neural networks are trained under loss functions. So, for this purpose, we will compare the MSE loss, L1 loss, and Tukey's Biweight loss.

During the analysis, let $r_t = y_t - \hat{y}_t$ is residual for single output and let $\rho(.)$ corresponds to error for given residual. The loss of individual residuals are as follows:

$$\rho(r_t) = \frac{1}{2} r_t^2 \tag{A.1}$$

$$\rho(r_t) = |r_t| \tag{A.2}$$

$$\rho(r_t) = \begin{cases} \frac{c^2}{6}\left[1 - \left(1 - \left(\frac{r_t}{c}\right)^2\right)^3\right] & , \text{ if } |r_t| \leq c \\ \frac{c^2}{6} & , \text{ if } |r_t| > c \end{cases} \tag{A.3}$$

Where Equation A.1 corresponds to MSE loss, Equation A.2 refers to L1 loss and Equation A.3 corresponds to Tukey's biweight loss functions. The results and figures of these losses are shown in the Figure A.1. The gradients of these loss functions are shown in Figure A.2. Mathematically, these gradients are as follows, respectively:

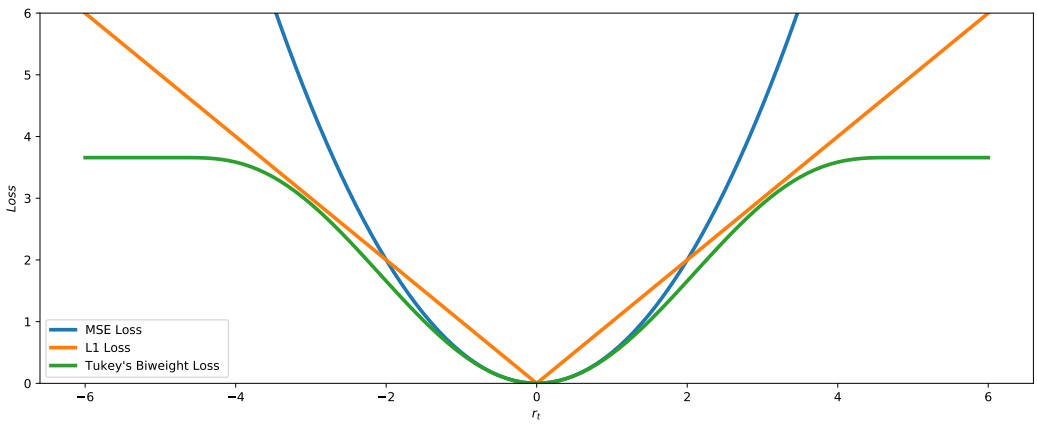



Figure A.1. Comparison of the loss functions

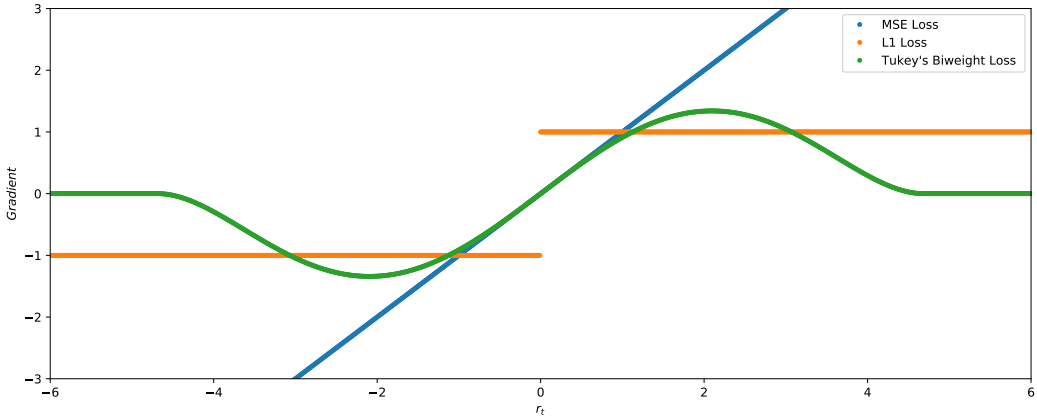



Figure A.2. Comparison of the gradients of loss functions

$$\rho'(r_t) = r_t \tag{A.4}$$

$$\rho'(r_t) = \begin{cases} -1 & , \text{ if } r_t < 0 \\ 1 & , \text{ if } r_t > 0 \end{cases} \tag{A.5}$$

$$\rho'(r_t) = \begin{cases} r_t \left(1 - \left(\frac{r_t}{c}\right)^2\right)^2 & , \text{ if } |r_t| \leq c \\ 0 & , \text{ if } |r_t| > c \end{cases} \tag{A.6}$$

The choice of $c$ of the 'Tukey's biweight' loss depends on the 'asymptotic efficiency'. It sets to 4.685to provides an asymptotic efficiency 95% that of linear regression for the normal distribution. In the use case, one should calculate the *median absolute deviation* (MAD) of the residuals and set the new residuals as follows:

$$r_t^{\text{MAD}} = \frac{r_t}{1.4826 \times \text{MAD}_t} \tag{A.7}$$

*Median Absolute Deviation* (MAD) measures the variability in residuals and calculated as follows:

$$\text{MAD}_t = \text{median}_{k \in \{1,\ldots,\mathcal{P}\}} \left( \left| r_{t,k} - \text{median}_{j \in \{1,\ldots,\mathcal{P}\}} \left( r_{t,j} \right) \right| \right) \tag{A.8}$$