Machine Learning with stream processing engines for IoT applications

A Degree Thesis

In partial fulfillment of the requirements for the degree in Audiovisual Systems Engineering

Author: Mariona Carós Roca
Advisors: Stéphan Clémençon

Universitat Politècnica de Catalunya (UPC)
June 2017
Abstract

The Internet of Things (IoT) enables to connect multiple devices for providing a certain service, consequently huge amount of data is generated in time, known as time series. This phenomenon presents unique challenges in defining the data behavior and detecting anomalies.

In this thesis, we present an appropriate method for defining the normal behavior of time series data and detection of anomalies.

We generate a daily periodic data set of time series based on the analysis of an energy consumption real data.

Then, by observing the input data, assumed to be independent from an unknown probability distribution, we define the normal behavior. The description of the data distribution is obtained by certain statistics and a Marked Point Process of change points.

We develop techniques for detecting the anomalies and providing the type of anomaly as well, using a Multiple Hypothesis Testing.

Finally, we present some experiments with the synthetic and real time series.
Resum

L’internet de les coses, més conegut com Internet of Things o IoT permet interconnectar diferents fonts amb la finalitat de cobrir una sèrie de necessitats. Per consegüent, el tractament de la gran quantitat de dades generada en temps real suposa un repte actualment.

En aquesta línia, un dels desafiaments més importants es basa en la definició del comportament de les sèries temporals així com la detecció de possibles anomalies, el qual és objecte d’aquesta tesi.

Per generar l’algorisme presentat en aquest treball s’han observat les dades obtingudes de diversos sensors que detecten consum d’energia, de les quals desconeixem la seva funció de distribució de probabilitat. El comportament normal de les dades el definim per mitjà d’un mètode d’acumulació d’esdeveniments conegut com Marked Point Process i l’estimació de certs paràmetres estadístics.

A continuació, s’han implementat una sèrie de tècniques per a la detecció d’anomalies i la seva posterior identificació mitjançant el mètode Multiple Hypothesis Testing.

Finalment, es presenten evidències dels bons resultats obtinguts tant per a les dades observades com per a un conjunt de mostra sintètica.
Resumen

El internet de las cosas, más conocido como Internet of Things o IoT permite interconectar distintas fuentes con el fin de cubrir una serie de necesidades. Por consiguiente, el tratamiento de la gran cantidad de datos generada en tiempo real supone un reto actualmente.

En esta línea, uno de los desafíos más importantes se basa en la definición del comportamiento de las series temporales así como la detección de posibles anomalías, el cual es objeto de esta tesis.

Para generar el algoritmo presentado en este trabajo se han observado los datos obtenidos de varios sensores que detectan consumo de energía, de los cuales desconocemos su función de distribución de probabilidad, para definir su comportamiento normal. Este proceso se lleva a cabo mediante un método de acumulación de eventos conocido como Marked Point Process y la estimación de ciertos parámetros estadísticos.

A continuación, se han implementado una serie de técnicas para la detección e identificación de anomalías mediante el método Multiple Hypothesis Testing.

Finalmente, se presentan evidencias de los buenos resultados obtenidos tanto para los datos observados como para un conjunto de muestra sintética.
Acknowledgements

First of all, I want to thank my advisor Stéphan Clémençon, for making possible my collaboration in this project, as well as guiding me during the project development.

I would also like to thank masters intern Safa Boudabous for their help in project which made possible that we could develop all the solutions presented in this report.

Finally, I would like to thank Ons Jelassi and Xavier Giro-i-Nieto for their support and help whenever they could.
Revision history and approval record

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>22/05/2017</td>
<td>Document creation</td>
</tr>
<tr>
<td>1</td>
<td>22/06/2017</td>
<td>Document revision</td>
</tr>
<tr>
<td>2</td>
<td>29/06/2017</td>
<td>Document approbation</td>
</tr>
</tbody>
</table>

DOCUMENT DISTRIBUTION LIST

<table>
<thead>
<tr>
<th>Name</th>
<th>e-mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mariona Carós Roca</td>
<td><a href="mailto:mariona.caros@alu-etsetb.upc.edu">mariona.caros@alu-etsetb.upc.edu</a></td>
</tr>
<tr>
<td>Stéphan Clémençon</td>
<td><a href="mailto:stephan.clemencon@telecom-paristech.fr">stephan.clemencon@telecom-paristech.fr</a></td>
</tr>
<tr>
<td>Ons Jelassi</td>
<td><a href="mailto:ons.jelassi@telecom-paristech.fr">ons.jelassi@telecom-paristech.fr</a></td>
</tr>
<tr>
<td>Xavier Giró i Nieto</td>
<td><a href="mailto:xavier.giro@upc.edu">xavier.giro@upc.edu</a></td>
</tr>
</tbody>
</table>

Written by: Date 22/06/2017
Name: Mariona Carós Roca
Position: Project Author

Reviewed and approved by: Date 22/06/2017
Name: Stéphan Clémençon
Position: Project Supervisor

Reviewed and approved by: Date 30/06/2017
Name: Xavier Giró i Nieto
Position: Project Supervisor
Contents

1 Introduction 11

1.1 Statement of purpose ........................................... 11
1.2 Requirements and specifications ............................. 12
1.3 Methods and procedures ..................................... 12
1.4 Work Plan .................................................... 13
    1.4.1 Gantt Diagram .......................................... 13
    1.4.2 Work Packages ........................................ 14
    1.4.3 Incidents and Modification ........................... 14

2 State of the art .................................................. 15

2.1 Time Series Modeling ......................................... 15
2.2 Change Points ................................................ 16
2.3 Anomaly Detection .......................................... 16
    2.3.1 Hypothesis Testing ..................................... 17
    2.3.2 Critical value approach ............................... 18
    2.3.3 P-value approach ...................................... 18
    2.3.4 Test statistics .......................................... 19

3 Methodology ..................................................... 21

3.1 Framework .................................................... 21
3.2 Characteristics of data set .................................. 21
    3.2.1 Real data ............................................... 22
    3.2.2 Synthetic data ........................................ 22
3.3 Estimation of data distribution .............................. 23
3.4 Adding anomalies into the data stream ...................... 26
3.5 Detection and classification of anomalies .................... 28
3.5.1 Oscillation ......................................................... 28
3.5.2 Mean Shift ....................................................... 30
3.5.3 Lost Signal ....................................................... 32
3.5.4 Outliers ......................................................... 32

4 Results .............................................................. 33
  4.1 Testing setup .................................................... 33
  4.2 Evaluation Criteria ............................................. 34
  4.3 Qualitative results .............................................. 34

5 Budget .......................................................... 38

6 Conclusions ...................................................... 39
List of Figures

1.1 Gantt Diagram of the Thesis ........................................... 13

2.1 Real Time series data from an elevator power consumption in KWh ........ 15
2.2 Time Series with steps (change points) .................................. 16
2.3 P-value of a Student t distribution. The p-value is the area under the probability density once we observed the data point. ........................................ 18
2.4 Comparison between cumulative functions $F_0$ and $F_1$ .................... 20

3.1 Scheme of the algorithm behavior ......................................... 21
3.2 Time series data of a group of devices power consumption in Kwh ........ 22
3.3 Synthetic time series ...................................................... 23
3.4 Steps of the distribution estimation phase ................................ 23
3.5 Steps detected for the given time series .................................. 24
3.6 T-scores computed from the previous time series .......................... 24
3.7 Marked Point Process of 2 point between the interval (a,b) ................. 25
3.8 PDF of steps detected for the given time series ........................... 25
3.9 CDF of rate of steps every 4 hours for a given time series ................. 26
3.10 Oscillation anomaly between January 20 and January 29 ................ 27
3.11 Time series with extreme noise between January 20th and January 29th .. 27
3.12 Mean shift on January 16th .............................................. 27
3.13 Time series with outliers on January 16th, 17th and 20th ................ 27
3.14 Lost signal anomaly from January 20th until January 26th ................. 28
3.15 Probability Mass Function of Poisson for a $\lambda = 4$ .................... 29
3.16 P-values of KS test for a given time series with an oscillation anomaly .. 30
3.17 Segment of Wilcoxon p-values of a time series with a mean-shift anomaly 30
3.18 Synthetic time series with detected mean-shift anomaly using Wilcoxon test 30
3.19 Segment of real time series with detected mean-shift anomaly

3.20 Log-Likelihood Ratio of the data mean with a mean-shift anomaly

3.21 *lost signal* anomaly detected after a fixed timeout

3.22 *Outlier* anomaly detected

3.23 SD values of the previous time series

4.1 Some time series from the real data set

4.2 False alarm mean shift detection

4.3 False alarm outliers

4.4 Detected steps (change points) for the oscillation anomaly detection

4.5 Detected oscillation

4.6 Detected change points
List of Tables

2.1 Table of hypothesis testing error types ................................................. 18
4.1 Performance using synthetic data ......................................................... 35
4.2 Performance using real data ............................................................... 35
5.1 Budget of the project .................................................................. 38
Chapter 1

Introduction

1.1 Statement of purpose

Internet is become more widely available, the cost of connecting is decreasing and more devices are being created with Wi-Fi capabilities. All of these circumstances give way to the Internet of Things. Also, referred to as smart devices, these objects with sensors and network connectivity are able to collect and exchange data. In the past decade IoT has spread into a wide range of fields, such as smart buildings and homes which is the application domain of this project. According to [15] the Internet of Things will include 26 billion units installed by 2020. The power of this paradigm is the ability to provide real time data from many different distributed sources to other machines or entities for a variety of services.

This phenomenon clearly requires tools and techniques for data streams monitoring. Under this vision, the detection and classification of unexpected changes in the data normal behavior is essential for the proper working of services.

In this thesis an unsupervised learning model is implemented, considering the trade-off that exists between accuracy and computation time, for the estimation of the incoming data distribution and anomaly detection. The aim of the algorithm is to detect the type of change in the distribution as quickly as possible to warn the user about the anomaly detected in the data.

Unsupervised learning has been chosen because of labeling large data is expensive and time consuming. Even we do not have a prior knowledge of the incoming data and algorithm parameters could change in time or differ depending on the user.

We focus on on-line processing time series data, specifically data with steps (abrupt changes in mean). The data points are assumed to be generated sequentially and independently by an unknown underlying probability distribution. The approach of the model is to define a region representing normal behavior and declare any observation in data which does not belong to this normal region as anomaly. One of the challenges is defining the normal region of the time series behavior because the boundary between normal and anomalous behavior is often not precise. In addition, normal behavior keeps evolving so the model should adjust itself. We distinguish between different types of anomalies: Oscillation, mean shift, outliers and lost signal, which are explained in detail within the document. For each type of anomaly, we use a different detection technique focused on different statistics.

Furthermore, we want to minimize the expected detection delay subject to a false alarm constraint. Hence, we want that our algorithm detects true changes with high probability (few false negatives), but we also want the test to notify us only if a true change has occurred (few false positives).
Based on the given context, the goals of the project are:

- Obtain the normal behavior of the incoming data by using statistics.
- Detect when the data distribution changes at the earliest.
- Recognize the identified anomaly.
- Work with real sets of sequential data.

1.2 Requirements and specifications

The requirements have been clearly defined from the beginning by the supervisors. The IoT sensors collect data in real time, so every certain period of time we receive a data sample, we want to analyze and detect if there is any anomaly in near real time, so the first requirement is to use an on-line sequential analysis.

In addition, we do not make assumptions about the income data distribution form. Thus, our second requirement lies on the implementation of a non-parametric machine learning algorithm.

Furthermore, we consider less available memory storage than data to process so the memory requirements should be independent of the dataset size.

The specifications for properly project development are the following:

First of all, the implementation using Python because it is a widely used high-level language for general-purpose programming; it has a design which emphasizes code readability, and a syntax that allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. Besides, it is an interpreted language. This means that Python codes run on a virtual machine that provides a layer between the code and the platform it runs on, making Python codes portable across different platforms.

The second specification is the knowledge of R programming language for research purpose because R has a much bigger library of statistical packages than Python. Specially packages in change point detection literature as *cpm* [16] or *changepoint* [12]

1.3 Methods and procedures

This project is a research internship performed in the framework of the Télécom Computer Science department. This internship is founded by *La Chair Machine Learning for Big Data*, which aims to produce a methodological research that responds to the challenge of statistical analysis of big data and to lead training in this field at Télécom ParisTech. Founded in September 2013 with the support of Fondation Télécom and financed by four partner companies: Safran,
PSA Peugeot Citroën, Criteo and BNP Paribas, the Chair is led by the mathematician Stéphan Clémençon, Professor in the Department of Signal and Image Processing of Télécom ParisTech.

This project has been developed from the scratch and will have continuity until September by Safa Boudabous, a Télécom master intern.

1.4 Work Plan

This project has followed the established work plan, with a few exceptions and modifications explained in the section 1.4.3.

1.4.1 Gantt Diagram

Figure 1.1: Gantt Diagram of the Thesis
1.4.2 Work Packages

- WP 1: Project proposal and work plan.
- WP 2: Environment preparation
- WP 3: Research and deep study in the field
- WP 4: Development of the algorithm
- WP 5: Analysis of the algorithm performance
- WP 6: Writing and presentation of the project

1.4.3 Incidents and Modification

First of all, we had problems with collecting data in real time by sensors, so we did not have as real data as we imagine at first. Consequently, we did not implement the Hadoop ecosystem as we considered it was not necessary for testing the algorithm.

Secondly, the algorithm implementation (W4) was started later than expected because the design of the algorithm was not clear and the research and study of the literature took more time than expected. The WP 6 was delayed as well because we spent more time developing and testing the algorithm.

Finally, we divided some of the big tasks to provide more information about the steps we followed for the algorithm development, such as the first step of generating data or specific techniques as defining change point distribution.
Chapter 2

State of the art

2.1 Time Series Modeling

The IoT growth gives rise to the deployment of massive numbers of sensors in various fields. These sensors are continuously producing a huge amount of time series data, which creates a correspondingly huge demand for time series data analysis, such as pattern recognition.

Time series is a sequence of data points, measured typically at uniform time intervals. A common assumption in many time series techniques is that data is stationary. A stationary process has the property that the mean, variance and autocorrelation structure do not change over time. Unfortunately, IoT data may come from a variety of domains, with different properties and ranges of data, so the underlying process that generates the data stream usually changes over time [6]. For this reason it is important to develop an adaptive algorithm which changes its behavior at the time it is run, based on the information available.

Another common property is seasonality, which means that the same patterns are repeated every certain period of time. It is very common for example in energy consumption time series, which is the type of data we use, where the same activities are repeated daily or weekly and we can find a pattern in the data.

On the following picture we can see an example of a repeated pattern in the same day of the week, every week.

![Real Time series data from an elevator power consumption in KWh](image)

Figure 2.1: Real Time series data from an elevator power consumption in KWh

There are many methods of model fitting. In order to fit this kind of models, we can distinguish two groups: parametric and non-parametric methods. The parametric approach assumes that the stochastic process has a certain structure which can be described using a number of parameters (for example, using an autoregressive model [26]). In these approaches, the task is to estimate the parameters of the model that describes the stochastic process in time. A deep study of parametric techniques in time series modeling can be found in [1]. By contrast, non-parametric approach explicitly estimates specific statistics without assuming that the process has any particular structure [7], [9]. This avoids the need of data to follow a specific parametric function. Therefore, this report is entirely focused on the non-parametric approach.
2.2 Change Points

The point in a data stream where the statistical properties of an underlying process change is called "change point". In practice, this changes manifest in a shift in mean, variance, correlation or spectral density of the process.

There exists different methods for estimating change points. Some examples are CUSUM, Binary Segmentation and PELT, used and described in [10] [11] which implementation can be found on [12]. Although in this project we use the Student T-test in a sliding window approach, explained in detail at sections 2.3.4 and 3.3.

The majority of the change point literature is focused on the off-line setting, where the entire signal is processed and then the algorithm identifies the times when the changes occurred [9]. However, the requirement for detecting change points quickly and accurately is of interest to a wide range of disciplines where the detection must be performed when the data arrives. The application of the retrospective testing procedure to the sequential setting is first attributed to Abraham Wald [22]. The interest of this procedure is that the sample size is not fixed in advance and data is evaluated as it is collected.

Our project focuses on sequential analysis, specifically in step detection, which is the process of finding abrupt changes (steps, jumps, shifts) in the mean level of a time series. This way we can characterize the data behavior by the counting and position of steps. An example of this kind of time series is shown on the following figure.

![Figure 2.2: Time Serie with steps (change points)](image)

2.3 Anomaly Detection

Detecting anomalies in time series data is a problem of practical interest in many signal processing applications. This problem can be viewed as finding deviations of a characteristic property in the system of interest. This is achieved by continuous monitoring of a system for significant deviations from the normal behavior patterns.

There have been a number of techniques suggested in the literature for detecting anomalies [2][14], such as control-charts, regression models, cluster analysis, Hidden Markov Models [23], Bayesian networks, Principal Component Analysis, sliding window method, etc. Though, all anomaly detection systems follow the same steps. First, data is observed and the system creates a baseline profile of the normal behavior, thereafter any data that deviates from the baseline is treated as a possible anomaly.
We focus on the sliding window method as it is the technique we use to convert the sequential learning problem into a comparison of two data samples. The sliding window method constructs a window classifier that maps an input window of width $W$ into an individual output value. This method has been successfully used in a number of machine learning based anomaly detection techniques [3] [4] [23]. Warrender et al. [23] proposed a method that utilized sliding windows to create a database of normal sequences for testing against test instances. Eskin et al. [4], improved the traditional sliding window method by proposing a modeling methodology that uses dynamic length of a sliding window dependent on the context of the system-call sequence.

Most of the anomaly detection methods have been specifically developed for certain application domains, such as fraud detection, health care[2] or intrusion detection [3] [4], among many others. Even though the methodology is the same for most of the anomaly detection systems, the exact notion of an anomaly is different for each application domain. Thus, applying our algorithm in another domain would not work as expected.

In this project, Hypothesis Testing is the main background of anomaly detection.

2.3.1 Hypothesis Testing

Hypothesis testing is a powerful method to compare a data set obtained by sampling against a model and find its similarity. Hypothesis Testing is well known and used in the statistical theory [20] [17]. This method consists in making an initial assumption and based on the available evidence (the current data), deciding whether to reject or not the initial assumption. The hypothesis we assume is true is called the null hypothesis (denoted $H_0$) and the opposite is called alternative hypothesis (denoted $H_1$).

The general idea of Hypothesis Testing involves:

1. Making an initial assumption, the null hypothesis.
2. Collecting evidence (data).
3. Based on the available evidence and by using a test statistic, deciding whether to reject or not reject the initial assumption.

Whenever the resulting value of the test statistic falls in the critical region, we reject the null hypothesis. The critical region is designed using the next sentence: if the $H_0$ is true, the probability that the test statistic will take a value in the critical region is less than a pre-defined constant, denoted $\alpha$.

When one of the hypothesis is selected, it can be mistakes of two types [24] [20]: Type I error (False Positive) and Type II error (False Negative). Type I error is the false rejection of the null hypothesis and type II error is the false acceptance of the null hypothesis. The Table 2.1 shows the relationship between power and error in Hypothesis Testing.

There are two processes to determine whether the evidence is likely or unlikely given the initial assumption. They are the “critical value approach” and the “P-value approach”
<table>
<thead>
<tr>
<th>Accept $H_0$</th>
<th>Reject $H_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>$H_0$ TRUE</strong></td>
<td><strong>Correct decision</strong></td>
</tr>
<tr>
<td><strong>$H_0$ FALSE</strong></td>
<td><strong>Type II error (False Negative)</strong> Probability $= \beta$</td>
</tr>
</tbody>
</table>

Table 2.1: Table of hypothesis testing error types

### 2.3.2 Critical value approach

The critical value approach consists in comparing the observed test statistic to some cutoff value, called the *critical value* to build a confident interval \[5\]. If the test statistic is more extreme than the critical value, then the null hypothesis is rejected in favor of the alternative hypothesis. If the test statistic is lower than the critical value, then the null hypothesis is not rejected. The threshold is typically chosen to bound the rate of false alarm.

### 2.3.3 P-value approach

The p-value is the probability of finding the more extreme results when the null hypothesis of a study question is true. The idea is to find the minimum level at which the test rejects $H_0$.

As long as $H_0$ is true, the p-value is uniformly distributed in the interval $(0,1)$. Whereas $H_1$ is true, the distribution of p-value will concentrate closer to zero. This implies that if we reject the null hypothesis when the corresponding p-value is less than a significance level $\alpha$, then the probability of a false alarm is $\alpha$.

Unlike the p-value, the $\alpha$ level is not derived from any observational data and does not depend on the underlying hypothesis; the value of $\alpha$ is instead set by the researcher before examining the data. The choice of the significance level $\alpha$ at which you reject $H_0$ is arbitrary. Conventionally the 5%, 1% and 0.1% levels are used \[21\]. Most authors refer to statistically significant as $P<0.05$ and statistically highly significant as $P<0.001$ \[24\].

![Figure 2.3: P-value of a Student t distribution. The p-value is the area under the probability density once we observed the data point.](image-url)
2.3.4 Test statistics

In this section, a brief description of tests used in our algorithm is introduced.

**T-test**

The T-test is used to determine if the means from two data sets are significantly different. It follows a Student t-distribution under the null hypothesis. The particularity of the Student t-distribution is that describes samples drawn from a full population whereas a normal distribution describes a full population. Although, it can be used in situations where the data is generated from other distributions, such as binomial and Poisson. This is thanks to properties of maximum likelihood estimators [19].

The formula for the T-test is a ratio \( T \). The top part of the ratio is just the difference between the window mean and the hypothesized population mean. The bottom part is a measure of dispersion, the standard deviation divided by the square root number of samples. This formula is essentially another example of the signal-to-noise metaphor in research.

\[
T = \frac{\overline{X} - \mu}{SD/\sqrt{n}} \tag{2.1}
\]

**Wilcoxon test**

The Wilcoxon test [25] is a statistical test that measures the tendency of one sample (data set of observations) to contain values that are larger than those in the other sample. It is considered a paired difference test as it needs two data sets to compare and it assumes both of them to come from the same population. In particular, it tests whether the distribution of the differences of two groups of data is symmetric about zero.

The procedure of the test is the following: There are a total of \( 2N \) data points. For pairs \( i = 1, \ldots, N \) let \( x_i = (x_1, x_2, \ldots, x_n) \) and \( y_i = (y_1, y_2, \ldots, y_n) \) denote the observations from the samples. We count the number of times an \( x_i \) from sample 1 is greater than a \( y_i \) from sample 2 (this number is the rank \( R_i \)). Then we obtain the Wilcoxon score, as we can see on the formula:

\[
W = \sum_{n=1}^{N} [\text{sgn}(x_i - y_i) \cdot R_i] \tag{2.2}
\]

Under the null hypothesis we would expect the difference between the pairs follows a symmetric distribution around zero.

As we said, this test only permits the detection of certain limited types of change, such as change in the mean [9]. Thus, we need another test to compare other statistics: The Kolmogorov-Smirnov test.
Kolmogorov-Smirnov test

The Kolmogorov-Smirnov test (K-S test) is a nonparametric test of the equality of continuous, one-dimensional probability distributions used to compare a sample with a reference probability distribution [18] [8]. The K-S test is sensitive to more general distributional changes not limited to scale shifts, like Wilcoxon test. In particular, the K-S test quantifies a distance between the empirical cumulative distribution function of the observed sample ($F_1$) and the cumulative distribution function of the reference distribution ($F_0$) as we can see on the Figure 2.4.

![Figure 2.4: Comparison between cumulative functions $F_0$ and $F_1$](image)

Given a sample $x = (x_1, x_2, ..., x_n)$ of independent random variables with distribution function $F_1$, consider the problem of testing $H_0 : F_1 = F_0$ versus $H_1 : F_1 \neq F_0$.

The K-S test is defined as the maximum difference between the distributions $F_1$ and $F_0$:

$$K = \sup_x |F_0(x) - F_1(x)|$$  \hspace{1cm} (2.3)

If the K-S statistic is small or the obtained p-value is high, we cannot reject the hypothesis that the distributions of the two samples are the same.

The two-sample K-S test is one of the most useful and general nonparametric methods for comparing two samples, as it is sensitive to differences in both location and shape of the empirical cumulative distribution functions of the two samples [13].

In section 3.5 we explain the application of these tests on our data sets. In section 4.3 the performance of these tests is presented with some experiments.
Chapter 3

Methodology

This chapter presents the methodology used to develop this project and the process followed to achieve our final results.

3.1 Framework

![Scheme of the algorithm behavior](image)

Figure 3.1: Scheme of the algorithm behavior

The Figure 3.1 summarizes the behavior of our algorithm in the two main phases of a machine learning system: Learning and classification.

As we can see on the first block, when the engine starts to receive the time series data, the algorithm generates a profile to represent the normal behavior. This is done by computing the incoming data statistics and counting the number of change points. The model keeps updating itself during a training period of time $T$.

After the period of time $T$, the second block takes place, in this block the method to detect anomalies is implemented. The method uses test statistics and functions based on ranks of observations for the detection of any change in the distribution. With every incoming sample a score for each test is computed.

Whenever the computed score exceeds a threshold, an alarm is sent. Otherwise, the algorithm keeps computing the tests and updating the model.

3.2 Characteristics of data set

Since the aim of the thesis is mining time series data to detect anomalies, we should firstly set the characteristics of the data.

As we said before, our approach assumes that data is independently generated, non-parametric and makes no assumption about its distribution. Nevertheless the input data should be daily or
weekly periodic so that the algorithm can get a good description of the daily data behavior and detect any alteration of the expected distribution. Although, if the data present a progressive change in its periodicity, the algorithm is able to readjust itself because it is continuously incorporating new knowledge.

In addition, the algorithm is devised for time series with steps, as one of the main properties of the data behavior modeling is the number and location of its change points. It is important to point out that throughout all the project non-stationary data is assumed because real-world processes do not exhibit a well-defined behavior. This is one of the major challenges as our understanding about what is normal and anomalous changes over time. So the time series have a time-varying mean or a time-varying variance or both. To face this assumption, the model is continuously updating its statistics. Even during the anomaly analysis, dismissing any sample detected as anomaly to avoid corrupting the model.

Based on our experience, to obtain the best results a sampling interval between 1 and 4 samples per hour is desired. Out of this range we cannot guarantee good results, as we have tested the algorithm with a sampling rate of 1 sample/hour with the real data and sampling rates of 2 and 4 samples/hour with synthetic data. As more samples we get from the data stream, more information is given to the algorithm so it is able to fit better the model. A more deep explanation about the importance of the sampling frequency in the algorithm performance is given at the section 4.3 with practical experiments.

3.2.1 Real data

The real data set used for testing the algorithm consists of energy consumption measurements from building sensors located in Paris. The data sets were recorded every hour during 2015 and stored in form of multiple time series.

Figure 3.2: Time series data of a group of devices power consumption in Kwh

3.2.2 Synthetic data

In order to compare the data statistics for the implementation of a non-parametric anomaly detector, it is necessary to use simulated data so the time of a change in the distribution is known in advance.

The simulated data sets were generated after the analysis of real data to make them similar (Figure 3.3), with the advantage of being able to adjust parameters like noise, amplitude, sample frequency or its form. This data was used to test the algorithm and set the thresholds, before test it with real data. The different time series basically consist in a combination of pulses with weekly periodicity (i.e. same form every Monday) with some gaps within the week as the real data have.
3.3 Estimation of data distribution

In this section we describe our algorithm for the estimation of the time series distribution. We consider a data stream with the characteristics described before. We also consider that the size of data available is much larger than the amount of available memory. During a fixed period of time, the engine finds out the data distribution while is not looking for anomalies.

The steps followed for the implementation of the learning phase are summarized in the following diagram:

We base our algorithm on a sliding window paradigm, introduced at the State of the art, which is a non-parametric way to estimate the probability density function of a random variable.

The window has a specific number of samples, fixed by the user, and it slides forward with each incoming data point, maintaining its size. The length of the window is a free parameter which exhibits a strong influence on the resulting estimate. We have chosen it according to the data sample frequency by testing. We use a window length of 15 samples (15h) for a sample rate of 1s/h and a length of 20 samples (5h) for a 4s/h rate. We can see that the window period of time differs a lot depending on the sample frequency. This is because we need a minimum number of samples (15s) to get solid statistics and to obtain reliable results using the tests.

The first week of data, the algorithm calculates the mean and the standard deviation, as well as the minimum value and the area inside the window. We selected these statistics as we saw they were the ones that gave more information in comparison with others. The algorithm computes the statistics every time a new data point appears on the stream. Then, it stores them by day of the week, thus initializing a particular model for each day. Additionally, for every incoming sample it computes and updates the global statistics mean and standard deviation by using the Welford’s method.
The Welford’s method is a simple streaming approach for computing sample mean and variance, without having to store large numbers of values. The following formulas are used to update these statistics:

\[
\mu_n = \mu_o + \frac{(x - \mu_o)}{(k + 1)}
\]

\[
S = S + (x - \mu_n)(x - \mu_o)
\]

\[
SD = S/(k + 1)
\]

Where \( \mu_n \) is the new computed mean, \( \mu_o \) is the old mean, \( x \) is the new data point, \( k \) is the number of samples and \( S \) is used to calculate the population standard deviation \( SD \).

After the first week, the algorithm keeps calculating the statistics mentioned before and updating the ones stored in the model by doing the average between these ones and the new ones. Likewise, it computes the T-test using the global and the current mean and variance calculated so far.

The T-test assesses whether the means of two groups are statistically different from each other, as it is explained in the State of the art [2,3,4]. In our work we use this approach to detect the steps of the time series, as we consider a step a significantly change in mean. For the calculation of the T-test, we use the mean standard error \( SE \) (SD divided by the squared size of the window \( W \)), the mean of the global stream \( \mu_g \) (the entire signal we have seen so far) and the current mean of the window \( \mu_w \). The formula to compute the test is the following one:

\[
T = \frac{\mu_w - \mu_g}{SE}
\]

\[
SE = \frac{SD}{\sqrt{W}}
\]

We detect a step every time the trend of the computed T-scores change its direction, as it means there is an abrupt change in mean. This methodology enables to detect the change points in real time. We can see the performance of this procedure on the following pictures.
For each jump detected, the amplitude and date of the change point are stored and the step counter is increased. The step counter, as its name suggests, counts the number of times a change is detected during the same day. The distribution of change points in time is created following a Marked Point Process.

A Marked Point Process is a collection of points randomly located on some mathematical space such as time. It is a useful model for the description of a sequence of events randomly located in time, as our case. We use two ways to build a Marked Point Process.

The first one is by counting the number of points arriving at a definite interval of time (day in our case) as it is shown in the Figure 3.7.

![Figure 3.7: Marked Point Process of 2 point between the interval (a,b)](image)

The second one is by accumulating the points \( x_i \) arriving up to a fixed time \( t \). Creating a cumulative distribution function (CDF) of points in time.

\[
N_t = \sum_{i=1}^{\infty} x_i \leq t
\]  

(3.3)

In our algorithm, the CDF of change points is created by storing the number of change points every 4 hours. This way, we have a precise distribution of jumps in a day. We set this period of time as based on our experiments it was enough to have some change points in the same range for giving information in time.

At the following pictures we can see the probability density function (PDF) of change points in the week and the CDF of change points for each day of the week of the previous synthetic time series presented in Figure 3.5.

![Figure 3.8: PDF of steps detected for the given time series](image)
As we can see on the Figure 3.9, the time series (Figure 3.5) has a different CDF for each day. Excluding Monday, Tuesday and Sunday where it is 0 as they do not have any change point during the day.

The CDF enables to have knowledge of the steps distribution in time, so we have more specific information than just storing the number of steps in day. Moreover, this information will permit to detect anomalies that PDF will not.

3.4 Adding anomalies into the data stream

One of the goals of our work is the detection of specialized types of change in distribution, as mentioned before. In this section we explain the anomalies we want to detect and its generation.

To create a time series with an anomaly, we select a period of time from the data stream and we add the anomaly within this period of time.

The anomalies we aim to detect are: oscillation, mean shift, outliers and lost signal.

Oscillation

The oscillation anomaly is basically white noise and it is created as a random signal with a constant mean and a finite variance.

We create two types of signal which should be detected as oscillation anomaly. The first one is created by replacing the segment of the time series defined as anomaly by a uniform white noise, as we can see on the following picture.
The second one is created by adding a white noise to the signal. This type of anomaly represents the signals with too much noise to get useful information from them. So we consider this change in distribution must be reported to the user.

Mean shift

The mean shift anomaly, as the name suggests, is an increase of mean in the whole data. We generate it by adding a constant to the selected period of time series to increase its amplitude.

Outliers

Outliers are observation points that are distant from the rest of observations. We create them by adding a much higher value than the data values to specific data points of the time series.
Lost Signal

We refer *lost signal* as to stop receiving data. This anomaly is generated by multiplying the time series period of time by zero and adding a little noise.

![Figure 3.14: Lost signal anomaly from January 20th until January 26th](image)

3.5 Detection and classification of anomalies

In this section we compare the stored distribution, which defines the behavior of the data, and the current data distribution of the window. Our goal is to design several techniques that examine the samples from both probability distributions and decide whether these distributions are enough different for reporting an anomaly.

We consider the data set and the sliding window of the previous section 3.3. After the training time, whenever a new item appears on the stream, the amount of data points inside the window are considered as a sample that is compared with the stored sample (the normal behavior of statistic distribution). Then, the algorithm applies different functions $F_i$ to measure the discrepancy between particular statistics of both samples. Therefore, the Hypothesis Testing of each anomaly type is checked. When the computed statistic of a test exceeds the set threshold $\alpha_i$, the anomaly related to the test is reported.

The key of the system is the choice of the test, to detect each type of anomaly, and the constants $\alpha_i$ which determine what is considered an anomaly. The selection of the parameter $\alpha_i$ defines our balance between sensitivity and robustness of the detection. The smaller $\alpha_i$ is, the more likely the algorithm will detect changes in the distribution, but the larger is the risk of false alarm. The $\alpha_i$ is set according to the application requirements.

We now discuss the techniques we use for computing the difference between the distributions depending on the desired anomaly to be detected. The results are reported in Section 4.3.

3.5.1 Oscillation

First of all, we consider the *oscillation* anomaly. To detect this anomaly we use two techniques.

The first technique consists in using the stored information of the time series PDF (Figure 3.9) as the parameter $\lambda$ of the Poisson distribution 3.15 and compare it with the number of incoming change points. We use this technique with time series of 1s/h sample rate which is our real data set. This modeling approach is known as a *Poisson Point Process* which is mainly a
way to model random events happening in time with the particularity that the points are related
to the Poisson distribution:

\[ P(x) = \frac{e^{-\lambda} \lambda^x}{x!} \]  

Whenever the probability mass function of the incoming change points differs too much from
our Poisson distribution we report an anomaly. Based on our experience, we set the critical value \( \alpha < 0.01 \), as explained in section \( \text{2.3.3} \) this value is considered little enough to discard the null hypothesis.

As we said before, there is a chance that the change points of the normal behavior are
distributed throughout the day. Nevertheless suppose that a specific day all change points are
detected in the same time slot. In this case the engine will not be able to detect the anomaly
because the average of the day would be the same. For this reason, we developed another
technique for detecting a high number of change points using the CDF(cp). This function gives
a more detailed definition of steps distribution, the drawback is we need more samples and more
steps in the time series to get good results. Because of this, we only used this technique with
the synthetic data of \( 4 \) s/h.

The technique consists in applying the K-S test (introduced in section \( \text{2.3.4} \)) at the end of
the day. We apply this test to compare both CDF(cp), setting the stored CDF(cp) as the null
hypothesis and the current CDF(cp) as the alternative hypothesis. We execute the test by the
end of the day because this way we have the distribution of change points of the whole day.
We set \( \alpha < 0.05 \) because this test is very precise and gives good results using cumulative density
functions as we do. Therefore, as we have a good description of the information and an accurate
test we can increase the cutoff probability. as the probability of false alarms is low.

As we can see on the Figure \( \text{3.16} \) for a given time series with an oscillation anomaly, the
p-value of the K-S test remains between 0.8 and 1 until the anomalous samples, then the p-value
decreases to 0.013.
3.5.2 Mean Shift

The detection of mean shift anomaly is achieved by using several hypothesis test. The first one is the Wilcoxon test, well-known for detecting changes in mean, as explained in section 2.3.4. Nevertheless, using few samples, the result of the test is an overlay sensitive notion of distance because the results oscillate between 0 and 1 with every incoming data point. It is difficult to have precise results with this behavior, so we wait a period of time before sending the alarm to ensure the value is reliable, instead of a fluctuation. Moreover, we only apply this test when we use a window length of minimum 20 points. We chose this value as Scipy Stats Library do not recommend to use it with a shorter window length.

To build the test we use the stored distribution of sample values ($H_0$) and the current samples in the window ($H_1$). We compute the test and whenever $\alpha<0.001$ we increase the counter which cumulates the amount of time. When the counter reaches 40 samples we send the alarm. We set the critical probability value of 0.1% because this test is very sensitive, we tried with higher values and the number of false alarms become very high. As we said, we prefer to be sure of the anomaly before raising the alarm.
The mean shift detection for a few samples window size ($W < 20$) is achieved by using the Log-Likelihood Ratio. We use it to calculate how many times the mean of the incoming data ($H_1$) differs from the stored mean of the day ($H_0$). We compare the result with a critical value to decide whether to report an anomaly or not.

We set the critical value at 2, as based in our experiments we considered it is enough high difference from the normal behavior to send an alarm. Additionally we use a counter, as in the Wilcoxon test, to ensure the ratio result is higher than the critical value during 10 samples, to avoid false alarms. In the next figures we can see a real time series with a detected mean-shift anomaly and the computed Log-Likelihood Ratio $3.5$ of the data mean.

$$LRT = -2 \ln \left( \frac{H_0}{H_1} \right) \quad (3.5)$$

![Figure 3.19: Segment of real time series with detected mean-shift anomaly](image)

![Figure 3.20: Log-Likelihood Ratio of the data mean with a mean-shift anomaly](image)

In addition, we use another ratio because we saw we were dismissing some anomalies using only the LRT of the mean. The second ratio consists in a ratio between the minimum value of the current data in the window and the normal behavior samples. We assume a daily periodic time series with jumps, so if the minimum value inside the window is very different from the one we have stored in the same period of time, it means the whole time series has increased in mean. Thus, using the same counter, we report the same anomaly when the current minimum value of the window is at least three times the stored minimum value.
3.5.3 Lost Signal

The detection of lost signal anomaly is done by using the area of the window. We define a timeout, which can be changed by the user, and we count the number of times the computed area of the window is less than 0.5. When the counter reaches the timeout the system sends an alarm.

Figure 3.21: lost signal anomaly detected after a fixed timeout

3.5.4 Outliers

The simple classical approach to detect outliers is to use the Standard Deviation (SD). The SD of a data set is the square root of its variance. It is a measure used to quantify the amount of variation or dispersion of a set of data values. A low SD indicates that the data points tend to be close to the mean, while a high SD indicates that the data points are spread out over a wider range of values. Therefore, when the computed SD becomes very high we know there exist the possibility to contain an outlier.

To identify outliers we compare the difference between the stored SD and the current SD with the stored SD. If the resulting score is higher than 4 we report the anomaly. This cutoff value is based on testing and it defines our perception of what values are considered outliers.

Figure 3.22: Outlier anomaly detected

Figure 3.23: SD values of the previous time series
Chapter 4

Results

This chapter presents the results for the experimental evaluation of anomaly detection techniques discussed in Chapter 3.

4.1 Testing setup

A simulation study is designed in order to assess the performance of the algorithm introduced in section 3. Two data sets are used for the purpose of the study, introduced in 3.2. One with real data and another one with synthetic data.

The real data set consists of 48 energy consumption time series with length of 1 year (8,730 samples) and a frequency rate of 1 s/h. The time series have different characteristics in terms of the nature of distribution. We added to each time series the three anomalies we generated in section 3.4 (oscillation, mean shift and outliers) obtaining a total of 144 different time series with anomalies. We did not use the lost signal anomaly in the experiments as we considered it not significant for the accuracy performance as it is always detected after the timeout set by the user.

Figure 4.1: Some time series from the real data set
The synthetic data set consist in 16 time series with anomalies of 1 year duration (35,136 samples) with a frequency rate of 4s/h. This data set was used during the development of the algorithm with the purposes of computing different statistics and selecting the ones which give more information, apart from testing and choosing the more accurate tests for our kind of data, as well as setting the window size and critical values.

4.2 Evaluation Criteria

We evaluate the techniques on different metrics, such as precision and recall to evaluate our classifier output quality and mean detection delay (\(ARL_1\)) to evaluate the time of detection.

Precision (\(P\)) is a measure of result relevancy as it gives a quantity of correct classifications. It is defined as the number of true positives (TP) over the number of true positives plus the number of false positives (FP).

\[
P = \frac{TP}{TP + FP}
\]  \hspace{1cm} (4.1)

Recall (\(R\)) is a measure of how many truly relevant results are returned. It is defined as the number of true positives (TP) over the number of true positives plus the number of false negatives (FN).

\[
R = \frac{TP}{TP + FN}
\]  \hspace{1cm} (4.2)

These quantities are also related to the \((F_1)\) score, which is an accuracy metric defined as the harmonic mean of precision and recall.

\[
F_1 = 2 \frac{P \times R}{P + R}
\]  \hspace{1cm} (4.3)

\(ARL_1\) is the number of expected samples or time until an anomaly is detected. It is desired a low level as possible to minimize the reaction time of the algorithm.

4.3 Qualitative results

We evaluate the criteria metrics for each data set and each type of anomaly and then we do the average of each data set to obtain the performance of the system for each case.

The obtained accuracy results of each type of anomaly are reported in Table 4.1 (for synthetic data) and Table 4.2 (for real data). The results reveal several interesting insights into the performance of the techniques which we are going to explain down below.

The algorithm, using the real data set as input, has obtained a precision of 71.8\%, a recall of 93.9\% and a \(F_1\) of 81.4\%. From our results we conclude that our system detects almost all anomalies but has a high rate of false alarms. This is because some of the analyzed time series have many sudden changes which the system detect as anomalies. In addition, some of the time series have seasonal changes which our algorithm does not take into account. A good improvement would be to include seasonal knowledge into the model.
The obtained $ARL_1$ is 17.4 samples (hours) which we consider is a high value for a streaming detection algorithm, but this ensures a low rate of false alarms. This value is because of the frequency rate of 1s/h and could be improved giving more information to the algorithm using data with higher sample frequency. We can see that the $ARL_1$ is better by using the synthetic data (4s/h) which obtained 10.8 hours of delay. This is because using data with high sample rate the algorithm is able to get more knowledge in the same period of time.

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>ARL (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outlier</td>
<td>100,00%</td>
<td>100,00%</td>
<td>100,00%</td>
</tr>
<tr>
<td>Mean shift</td>
<td>80,00%</td>
<td>100,00%</td>
<td>88,89%</td>
</tr>
<tr>
<td>Oscillation</td>
<td>100,00%</td>
<td>100,00%</td>
<td>100,00%</td>
</tr>
<tr>
<td>Lost signal</td>
<td>100,00%</td>
<td>100,00%</td>
<td>100,00%</td>
</tr>
<tr>
<td>System</td>
<td>95,00%</td>
<td>100,00%</td>
<td>97,22%</td>
</tr>
</tbody>
</table>

Table 4.1: Performance using synthetic data

We can see that in synthetic data all anomalies were detected. The 80% of precision in mean shift anomaly is because of false alarm as we used one signal with variations in time without periodicity and the algorithm was not be able to adjust itself the whole time.

The $ARL_1$ value of synthetic data is calculated without taking into account the lost signal anomaly timeout of 120 samples because this time is not required by the system, it is fixed in advance by the user.

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>ARL (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outliers</td>
<td>89,70%</td>
<td>88,19%</td>
<td>88,94%</td>
</tr>
<tr>
<td>Mean shift</td>
<td>65,75%</td>
<td>100,00%</td>
<td>79,34%</td>
</tr>
<tr>
<td>Oscillation</td>
<td>60,00%</td>
<td>93,75%</td>
<td>73,17%</td>
</tr>
<tr>
<td>System</td>
<td>71,82%</td>
<td>93,98%</td>
<td>81,42%</td>
</tr>
</tbody>
</table>

Table 4.2: Performance using real data

Comparing the tables we can see some differences between the obtained metrics. The results emphasize how difficult it is to detect anomalies in non-stationary time series. Additionally, a high percentage of real time series used in the experiments were not weekly periodic, as shown in the Figure 4.1.

On the following figures we provide the situations that cause false negatives and false alarms and we explain the reasons.

The shift mean and oscillation anomalies have a precision between 60%-66% because of the false alarms. In the following picture we can see that two mean shifts are detected because of the unexpected increasing mean average of the time series.
The outlier anomaly is detected when there is a sudden increase of the standard deviation. If the time series has no activity during a period of time (Figure 4.3) and then the amplitude increases, our algorithm can associate this increasing as an outlier because it was not expected in the normal behavior.

Regarding the oscillation anomaly, it is detected when there is a sudden increase of change points. Thus, if the time series behavior is not well defined, the algorithm tends to detect false positives. Whereas if the time series has similar distribution every week, as in Figure 4.4 the anomaly is well detected.
Finally, we can see that the algorithm always tries to apply the techniques, when the steps are unclear the algorithm keep segmenting the time series using the increase and decrease of mean.

Figure 4.6: Detected change points
Chapter 5

Budget

This project has been developed using the resources provided by Telecom ParisTech.

The hardware resources needed for the project were a CPU and sensors to collect data. The CPU was Intel W3670 3.20GHz 12G RAM which costs 300€. The sensors for receiving energy consumption data from buildings in Paris were provided by Zodianet without any additional cost.

Regarding software, everything we have used is open-source and thus there is not additional cost. The specific software required for the development of this thesis is: Python 2.7, Numpy 0.9.1, Scipy 0.15.1, Jupyter Notebook, GanttProject, Sublime and RStudio.

Therefore, the main costs of this projects comes from the salary of the researches and the time spent in it. The team for the development of this thesis is formed by two senior engineers as the advisors and myself as a junior engineer. I consider that the total duration of the project was 20 weeks as presented in the Gantt diagram 1.1

<table>
<thead>
<tr>
<th>Position</th>
<th>Amount</th>
<th>Wage/hour</th>
<th>Dedication</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Junior engineer</td>
<td>1</td>
<td>8,00 €/h</td>
<td>35 h/week</td>
<td>5,600 €</td>
</tr>
<tr>
<td>Senior engineer</td>
<td>2</td>
<td>20,00 €/h</td>
<td>1 h/week</td>
<td>800 €</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>6,400 €</strong></td>
</tr>
</tbody>
</table>

Table 5.1: Budget of the project
Chapter 6

Conclusions

The goal of this thesis was to detect and classify anomalies of non-stationary time series in streaming without prior-knowledge.

We described an analytical framework for IoT data, based in statistical anomaly detection, where the system first observes the activity of data and then generates profiles to represent their behavior. We focused on energy consumption data, in particular, time series with steps.

We proved our algorithm is able to detect anomalies despite the lack of prior knowledge on data. It is achieved by using hypothesis testing with distance functions to find the distribution changes while providing strong statistical guarantees with small sample size.

We focused on the detection of specialized types of anomalies rather than general definitions of change detection in distribution. We have examined and evaluated the different techniques we use to determine the nature of changes in the distribution.

We presented experiments showing the algorithm performance in terms of high detection rate and low false alarm rate, as well as mean detection delay.

Besides, developing this thesis we have seen it is very ambitious to model all kind of behaviors by using purely statistical methods. Moreover, it is difficult to set critical values that balance the likelihood of false positives with the likelihood of false negatives.

Overall, the non-parametric hypothesis testing procedure is found to perform well, and represents a proper approach when performing anomaly detection analysis on data of unknown distributional form.

As a future work, we want to incorporate seasonal knowledge because we think it would highly improve the accuracy and false alarm rate. Furthermore, we would like our algorithm to set the critical values automatically while analyzing the statistics behavior and to set the window size depending on the sample frequency and amount of information in each sample. Finally, we would like to detect more types of anomalies by adding more techniques to take into account more abnormal situations.
Bibliography


