POLITECNICO DI TORINO

Corso di Laurea Magistrale in ICT for Smart Society
Dipartimento di Electronics and Telecommunications


POLITECNICO
DI TORINO

# Machine Learning for Self-organizing Networks

**Tesi di:**
**Chongshun Wang**
**matricola 231135**

**Relatore:**
**Prof.ssa. VISINTIN MONICA**
**Prof. GARELLO ROBERTO**
**Mr. BULDORINI ANDREA (TIM)**

**Anno Accademico 2017-2018**

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Traditionally, the evolution of mobile networks from one generation to another has been driven by the hardware technology improvement. While the 5G revolution is different, the improvement of software technology becomes more important than the improvements of hardware, especially in terms of network management system. With the explosion of software improvements, the management of autonomic network can be put into practice taking advantage of also other cross-disciplinary knowledge advancements in the area of Machine Learning (ML). Self-organizing Network (SON) is used to refer to mobile network automation and human intervention minimization in the cellular network management.

Troubleshooting in traditional mobile networks is a procedure that has been manual. It has 3 stages 1). Detection: to identify the sectors with failure/abnormal performance. 2). Determination: to determine the cause of the problem. 3). Fix: to take actions to solve the problem (repair HW, configuration corrections, etc).

This project is a telecommunications field work developed by computer scientists. For this reason, it is important to notice that the back-

ground chapter was carefully balanced between electrical engineering and computer science aspects.

The remainder of the thesis is organized as follows: the second Second chapter explains the backgrounds, the goal of this work, why it is important and hard and also outlined the methodology. The third chapter provides the information of data preparation and cleaning. The forth chapter describes how we get the cell behavior pattern clusters and the fifth chapter describes how we classified the new data into our existed models.

# Chapter 2

# Backgrounds and Related Works

## 2.1 Self-Organizing Networks

**SON** is a main driving technology to improve efficiency, simplify management procedure and reduce the operation costs for next generation Radio Access Networks (RANs). The Main objective of SONs can be roughly divided into 3 main parts:

1. Bring intelligence and autonomous adaptability into cellular network,
2. Reduce capital and operation cost,
3. Enhance performance of network in terms of its capacity, coverage and improve the quality of service/experience.

In telecommunication field, SON can be useful to achieve real time autonomous network management.However, the SON solutions in current market are limited in various aspects:

1. SONs are mainly based on heuristic algorithms
2. SONs are implemented with low complexity solutions such as triggering for the automated control
3. Some operations are still done manually eg. engineers manually fix networks faults
4. Do not really take advantage on the huge amount of information that is available in the network

As we can observe in *Big Data Empowered Self Organized Networks* [10], huge amounts of data have been generated in current 4G networks during normal operations by control and management functions. It is reasonable to expect more will come in 5G networks due to the emergence of new applications and services, for example Internet of things (IoT) paradigms. So 5G network management is expected to face much more complex challenges.

A key concept in SONs is that the collections of data should be used to inform the system about what is going on in the network at any given time. Usually, this data collections are counters or key performance indicators (KPIs). As an example of KPIs, *pmreceived* is the number of paging messages received per minute (for a given cell).

## 2.2 Load Balancing of Antennas

So far, we said about the goal of this project is to implements machine learning on SONs, but the SONs settings, which is the main object are still abstract. Actually, we are talking about physically tilting the antennas. One of the technology in the field is *remote electric tilt* (RET). This tilting allows antennas to be controlled either manually

or remotely, but with SONs application, the remote control is necessary. When we apply the tilt to an antenna, we improve or reduce the signal coverage in areas of that antenna. In other words, when we're adjusting the tilt we seek a signal as strong as possible in areas of interest (where the traffic must be), or similarly, a signal the weakest as possible beyond the borders of the cell.

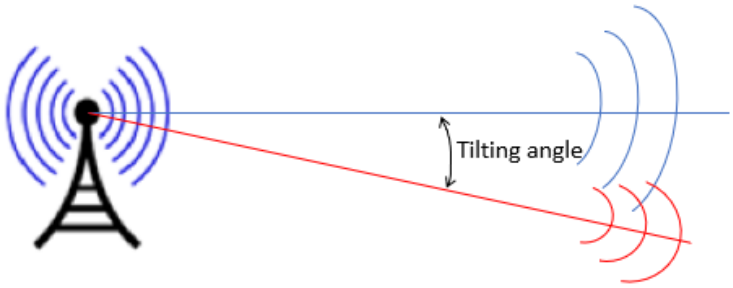



*Figure 2.1: Antenna inclination*

As discussed in *Antenna Tilt Load Balancing in Self-Organizing Networks* [5], the purpose of load balancing is to transfer loads from a cell that is experiencing heavy load to a neighboring cell that is experiencing less. Due to the unpredictable nature of user behavior, it is not easy without proper tools.

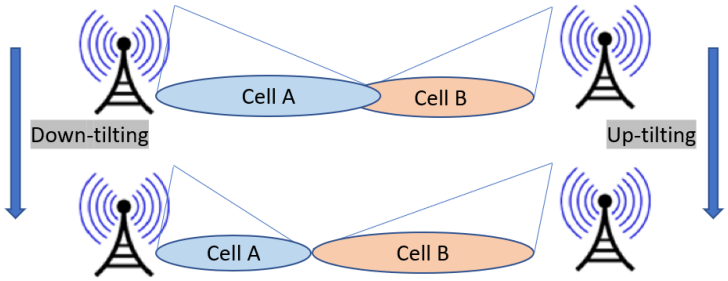



*Figure 2.2: Antenna Load Balancing. Illustration of load balancing. Down-tilting antenna A while up tilting antenna B, the coverage area for respective cells are changed.*

## 2.3 Data Mining and Machine Learning

We believe that Machine Learning is a promising technology in telecommunication network management. It allows the network to learn from experience so as to improving performance. In particular, big data analytics can pursue the self-awareness by driving the network management from reactive to **predictive**. That is why big data analytics are receiving big attention in research and market. The term of machine learning (ML) was originally introduced in 1959 by Arthur Samuel, when he defined ML as a field that gives computers the ability to learn without being explicitly programmed. And according to *Machine Learning* [9], *Using Output Codes to Boost Multi-class Learning Problems* [13], *Big Data: the next frontier for innovation, competition and productivity* [8], The primary goal of ML is to develop efficient algorithms, where time and the amount of data required for learning are one of the most important efficiency indicators. Learning algorithms should also be as general as possible, but also, those decisions/predictions have to be easily understandable by experts.

ML taxonomy is traditionally organized into 3 parts: (see Table 2.1)

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

**Programming Tools** The programming language in this work is Python which is very famous programming language in Scientific Computing and Machine learning fields(*Python for Scientific Computing* [11]) as well as some scientific packages from Python community, such

| Supervised Learning | |
|---|---|
| Classification | K-Nearest Neighbors<br>Support Vector Machine<br>Naive Bayes<br>Neural Network |
| Regression | K-Nearest Neighbors<br>Support Vector Machine<br>Naive Bayes<br>Neural Network<br>Decision Trees |
| Unsupervised Learning | |
| Clustering | Non-overlapping clustering<br>Hierarchical clustering<br>Overlapping clusterings |
| Dimensionality Reduction | Non-overlapping clustering<br>Hierarchical clustering<br>Overlapping clusterings |
| Anomaly Detection | Pruning techniques<br>Rule based systems |
| Reinforcement Learning | |
| Model-based | Dynamic Programming<br>Monte Carlo |
| Model-free | Temporal Difference |

*Table 2.1: Machine Learning Brief Summary*

as **Numpy**, **Matplotlib**, **Pandas**, **TensorFlow**, **Scikit-Learning** and **Kears** Table 2.2.

## 2.4 Related Work

The idea of using machine learning tools to telecommunication networks is not new at all. For example, in Unsupervised Learning of Traffic Patterns in Self-Optimizing 4th Generation Mobile Networks [2], the authors proposed a standard unsupervised method to aggregate

| *Tools* | *Name* |
|---|---|
| Numpy | "NumPy is the fundamental package for scientific computing with Python" |
| Pandas | "Powerful data structures for data analysis, time series,and statistics" |
| TensorFlow | A open-source machine learning framework from Google |
| Scikit-Learning | "Simple and efficient tools for data mining and data analysis" |
| Keras | "A high-level neural networks API, written in Python and capable of running on top of TensorFlow." |

*Table 2.2: Programming Language and Tools in this project*

user behavior in order to find the most suitable pattern of cell to match the user behavior. More general research (non-telecommunications related) about time series clustering in *Probabilistic discovery of time series motifs* [4], the main idea of time series clustering is to cluster intervals in a time series that show similarity. However it was realized we are more interested in finding a large clusters with long time series rather than small, very distinctive ones.

# Chapter 3

# Data Processing

Mobile networks generate a huge amount of traffic data which must be analyzed with proper tools, in order to bring intelligent decision making mechanism to the network management. In this chapter, we will explain the *data preparation* and *cleaning*, which is the first procedure of how machine learning tools can specifically be applied to SONs.

## 3.1 Data Description

The dataset consists of 100 different base stations (cells), and the 100 also including cases of same base station but using different technologies 2G/3G/4G. The duration of whole dataset is from 14, November, 2016 to 15, October, 2017. Each counter / KPI is measured every 15 minutes. The number of values for each counter is $N = 4records/hour \times 24hours = 96samples$. Full descriptions of the meaning of each features (measurements) are listed in the Table 3.1.

For example, a typical record for a day may look like as Table 3.2. Notice that, in order to make the records more synchronize to human behavior, the day start-end interval is shifted. The starting time of a

| List of Parameters | | |
|---|---|---|
| Record ID | identifier | eg. "AT" |
| | date and time | eg. 2017-02-15_02:15:00 |
| Measurements | pagreceived | paging message received per minute |
| | raatt | radio access attempt connection per minute |
| | rasucc | radio access successful connection per minute |
| | rrcconnestabatt | number of RRC connection requests per minute |
| | rrcconnestabsucc | number of RRC connection established per minute |
| | activeuedl | number of user equipment in the down-link |
| | activeueul | number of user equipment in the up-link |
| | dpcpvoldl | traffic volume in the down-link |
| | dpcpvolul | traffic volume in the up-link |
| KPIs | accessibilitatotale | total accessibility (percentage) |
| | accessibilitarrc | radio control accessibility (percentage) |
| | accessibilitas1signaling | signal accessibility (percentage) |
| | accessibilitaerab | radio access bearer accessibility (percentage) |

*Table 3.1: Parameters in the datasets*

day is shifted from 00:00 to 04:00. In other words, the first record of a day is at 04:00, correspondingly the last record of a day is at 03:45 instead of 23:45.

| key | Cell ID | Date | Time | activeuedl | More Parameters |
|---|---|---|---|---|---|
| LL1_2017-02-03_04:00:00 | LL1 | 2017-02-03 | 04:00 | 1725 | ... |
| LL1_2017-02-03_04:15:00 | LL1 | 2017-02-03 | 04:15 | 2479 | ... |
| LL1_2017-02-03_04:30:00 | LL1 | 2017-02-03 | 04:30 | 2443 | ... |
| ... | ... | ... | ... | ... | ... |
| LL1_2017-02-04_03:30:00 | LL1 | 2017-02-04 | 03:30 | 367 | ... |
| LL1_2017-02-04_03:45:00 | LL1 | 2017-02-04 | 03:45 | 472 | ... |

*Table 3.2: A snippet of the dataset*

Records of one base station shows the cellular network usage inside the coverage of that cell. But it is also very interesting to have a aspect from a larger geographic region. Those large virtual base stations are obtained by merging adjacent cells together, Table 3.3, and the records of them are calculated by simply summing the records of each small stations inside this large virtual station.

| Group | Cells |
|---|---|
| Group01 | KL1, KT1 |
| Group02 | DT1, DT2, DL1, DL2, ET1, ET2, EL1, EL2 |
| Group03 | DT1, DT2, DL1, DL2, EL1, EL2, ET1, ET2, HT2, HE2, KL1, KT1 |
| Group04 | CL1, CT1, DL1, DL2, DT2, DT1, ET1, ET2, EL1, EL2, LL1, LL3, LT1, LT3, KT1, KL1 |

*Table 3.3: Cell Mergence*

## 3.2 Data Selection and Clean

**Data selection** From measurements correlation matrix in figure 3.1, it is obvious that some measurements are highly correlated, for example the number of RRC (Radio Resource Control) connection requests per minutes (rrcconnestabatt) and the radio access attempt connection per minute (raatt). So this redundant features maybe hurt in machine learning aspects, it is necessary we reduce the number of features to be considered.

There are 9 measurements in the collections of data. In the experiments, we selected features in two ways. One way is to choose 4 parameters but 2 of them are added, so at last we have 3 features **down-link traffic** (dpcpvoldl), **up-link traffic** (dpcpvolul), **connected user** (activeueul + activeuedl). And second way is to choose 2 of them and add them, **total traffic** (dpcpvoldl + dpcpvolul).

**Missing Data** The collected data have to be cleaned for various reasons, for example, some machine learning algorithm are not applicable

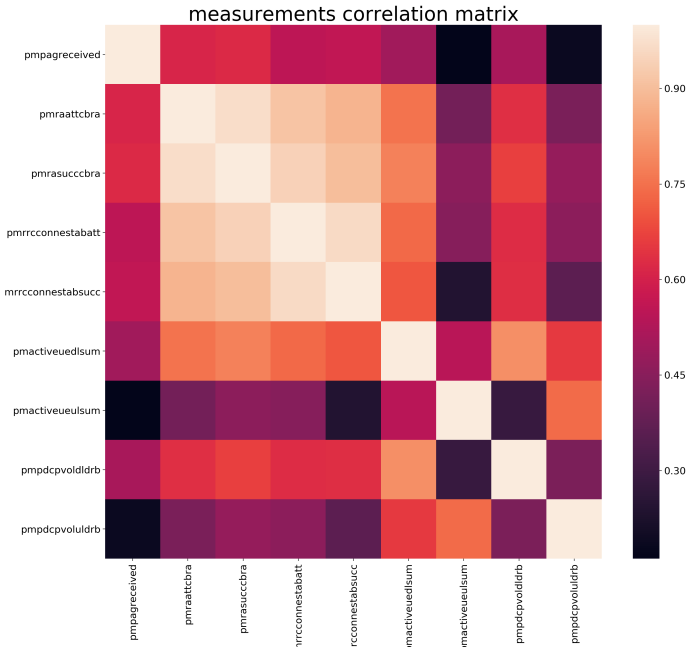



*Figure 3.1: Correlation Matrix of Measurements in Dataset. Bright color represents highly correlated measurements and vice-versa.*

with missing values in the dataset. In general,the step of data cleaning is necessary to **structure** the data to facilitate the following data analysis.

Dataset might have missing values for different reasons such as system management maintenance or simply system error. Handling missing data is very important as many machine learning algorithms do not support dataset with missing values and data is a kind resource that we should take advantage of, as much as possible.

In the experiments, a simple strategy was adopted for handling missing data that is removing that day if the record of it contains any missing value. But in order to maintain more records, we use the criteria that only the day with more than 3 hours missing records are **removed**, which means if a day contains more than 12 miss-

ing values ($3\ hours \times 4\ records/hour = 12\ records$) over 96 values ($24\ hours \times 4\ records/hour = 96\ records$) it will be discarded.

**Interpolation** After the removing process, the remaining missing values are **interpolated**. Traditionally, interpolation of missing value can be done by using one of the following strategy.

- Constant value that has meaning within the domain, for example 0 if the column consists of numeric value or categorical value if it is a categorical represented feature.
- Mean, median of that feature.
- A value estimated by another predictive model.

As a starting point, we choose to use **mean** value to substitute the missing data. Formally, if a records is missing $a_{i,j}^{d,t}$ for base station $i$ of measurement $j$ at day $d$ time $t$, then we assign

$$a_{i,j}^{d,t} = Mean(a_{i,j}^{x,t}) \qquad for\, x \in D \tag{3.1}$$

where $a_{i,j}^{x,t}$ is the non-missing record in the dataset of same base station and same time of a day but in different days, the mean value of such records are used to interpolate the missing record.

**Restructure** The last step of data cleaning is to restructure the data format into desired shape. In machine learning convention, datasets are organized in 2D matrix (For some applications data are organized into 3D, such as imagine processing, but in our applications 2D matrix

is enough). Each row is a sample and each column represents a feature. It is important to say that because our samples are time series, one value of measurement is a feature. To be clear, if we choose 3 features, then our datasets will contain 288 features ($96 \times 3 = 288$). The output of this step is to generate such 2D matrix for each cell.

## 3.3 Normalization

Normalization is a very common and necessary procedure before applying any machine learning algorithms on the dataset, basic motivations for normalization is described as follows:

- The range of values in different features often varies widely. For example, one major classifier uses the criteria of calculating the distance between two points by the Euclidean distance, normalization is necessary so that each feature approximately contributes proportionately to the final distance.
- The gradient descent converges much faster with normalization than without it.

In general case, the aim of normalization is to make each feature in the data have **zero-mean** and **unit-variance**. $x_i$ is normalized into $y_i$ in two steps. First, distribution mean and standard deviation for each feature are determined. Next the mean is subtracted from each of the features in the datasets and resulting values are divided by the standard deviation eq.3.4.

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.2}$$

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{N}(x_i - \overline{x})} \tag{3.3}$$

$$y_i = \frac{x_i - \overline{x}}{\sigma} \tag{3.4}$$

First of all, let me introduce the terminology that we used in this part. $N_d$ is the number of days and $N_{d,c}$ is the number of days for *cell c*. $N_f$ is the number of KPIs that we considered. After data reconstructed, records of a day, called a sample in machine learning field, is reshaped to a row vector with the shape $[1,\ N_f \times 96]$ where $N_f$ is number of measurements used, then samples of $N_{d,c}$ days for each base station in $N_c$, total number of base station, are concatenated. The shape of a final data set $[\mathbf{A}]$ is eq.3.5.

$$[\mathbf{A}].shape = [N_d,\ N_f \times 96] \tag{3.5}$$

$$N_f = \text{number of considered features}$$
$$N_d = \sum_{c=0}^{N_c} N_{d,c} = \text{total samples} = 23264$$
$$N_c = \text{number of cells} = 104$$

The normalization is made cell independently which means the

dataset is divided into different parts for different cells as it shows in table3.4.

| ID | connected rrc users | | | | traffic volume Mb | | | |
|---|---|---|---|---|---|---|---|---|
| AL1_2016-12-03 | (1) | (2) | ..... | (96) | (1) | (2) | ..... | (96) |
| AL1_2016-12-04 | (1) | (2) | ..... | (96) | (1) | (2) | ..... | (96) |
| AL1_2016-12-05 | (1) | (2) | ..... | (96) | (1) | (2) | ..... | (96) |
| *i-th* station *j-th* day | $x_{i,j}^{(1),1}$ | $x_{i,j}^{(2),1}$ | $x_{i,j}^{(k),1}$ | $x_{i,j}^{(96),1}$ | $x_{i,j}^{(1),2}$ | $x_{i,j}^{(2),2}$ | $x_{i,j}^{(k),2}$ | $x_{i,j}^{(96),2}$ |
| AL1_2016-10-12 | (1) | (2) | ..... | (96) | (1) | (2) | ..... | (96) |

*Table 3.4: Dataset for i-th base station*

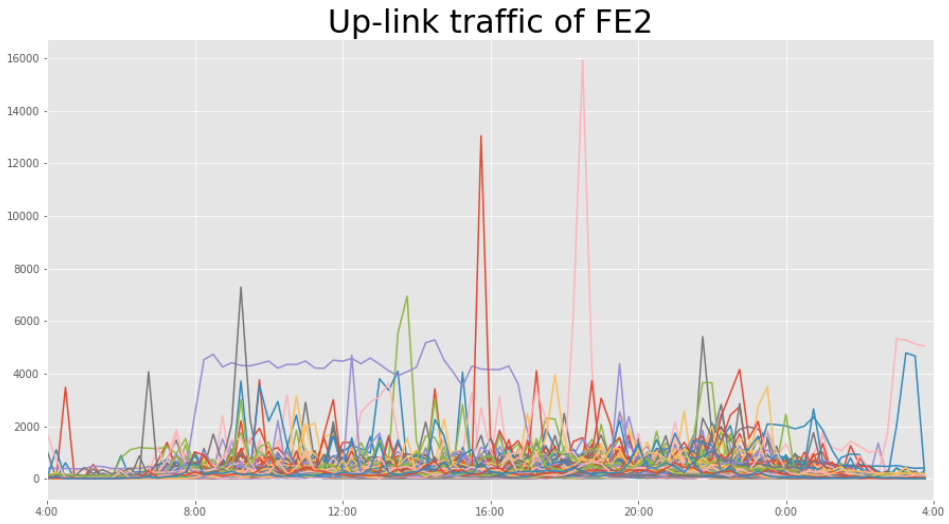



*Figure 3.2: Data after interpolation*

Normalization can be made in 3 different ways. In this part, the detail procedure of this 3 normalization criteria are introduced. For the first and second criteria, each cell is normalized independently, while in the third criterion we consider all cells together.

1. Normalization with all days

2. Normalization day by day
3. Normalization with all cells

### 3.3.1 Normalization with All Days

**Normalization with All Days** means to normalize our data set as normal normalization in machine learning field, gathering all numeric values in one feature and scaling them.
For $i$-th base station, the procedure of normalization is shown in eq.3.6. Notice that we assume to have *2* features, *connected rrc users* and *traffic volume Mb*, where *traffic volume mb* is the sum of traffic of down-link and up-link, and for each feature, we have *96* records in a day.

$$\begin{aligned} \overline{x_i^l} &= \frac{1}{N_{d,c} \times 96} \sum_{j=1}^{N_{d,c}} \sum_{k=1}^{96} x_{ij}^{kl} \\ \sigma_i^l &= \sqrt{\frac{1}{N_{d,c} \times 96} \sum_{j=1}^{N_{d,c}} \sum_{k=1}^{96} (x_{ij}^{kl} - \overline{x_i^l})} \\ l &\in (1, 2) \end{aligned} \tag{3.6}$$

Where $\overline{x_i^l}$ and $\sigma_i^l$ represent mean value and variance of $l$-th feature records in the $i$-th base station. $N_{d,c}$ is the number of available days for the $i$-th cell. Then $x_{ij}^{kl}$ in original dataset should be substituted by $x_{ij}^{kl}\prime$ as eq.3.7.

$$x_{ij}^{kl}\prime = \frac{x_{ij}^{kl} - \overline{x_i^l}}{\sigma_i^l} \tag{3.7}$$

An example of normalized data is shown in Fig.3.3

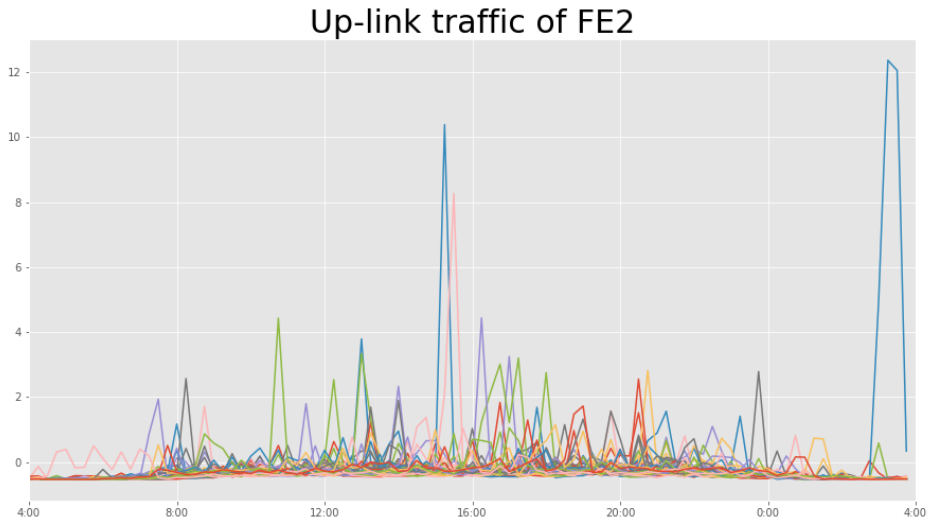



*Figure 3.3: Data after all days normalization*

### 3.3.2 Normalization Day by Day

**Normalization Day by Day** means perform normalization independently not only for different base stations but also different days.

For the $i$-th base station, the procedure of normalization is shown in eq.3.8 and eq.3.9. Notice that we assume to have *2* features, *connected rrc users* and *traffic volume Mb*, where *traffic volume mb* is the sum of traffic of down-link and up-link, and for each feature, we have *96* records in a day.

$$\begin{aligned} \overline{x_{ij}^{l}} &= \frac{1}{96}\sum_{k=1}^{96} x_{ij}^{kl} \\ \sigma_{ij}^{l} &= \sqrt{\frac{1}{96}\sum_{k=1}^{96}(x_{ij}^{kl} - \overline{x_{ij}^{l}})} \\ l &\in (1,2) \quad j \in N_{d,c} \end{aligned} \tag{3.8}$$

Here $\overline{x_i^l}$ and $\sigma_i^l$ represent mean value and variance of $l$-th feature records in the $i$-th base station at $j$-th day. Also $N_{d,c}$ is number of available days in $i$-th cell. Then $x_{ij}^{kl}$ in original dataset should be substituted by $x_{ij}^{kl}\prime$ as eq.3.9.

$$x_{ij}^{kl}\prime = \frac{x_{ij}^{kl} - \overline{x_{ij}^{l}}}{\sigma_{ij}^{l}} \tag{3.9}$$

An example of normalized data is shown in Fig.3.4

### 3.3.3 Normalization with All Cells

**Normalization with All Cells** means perform normalization assuming all base stations form into a large virtual one. The samples in different base station are normalized together.

For $i$-th base station, the procedure of normalization is shown in eq.3.10. Notice that we still assumed to have *2* features, *connected rrc users* and *traffic volume Mb*, where *traffic volume mb* is the sum

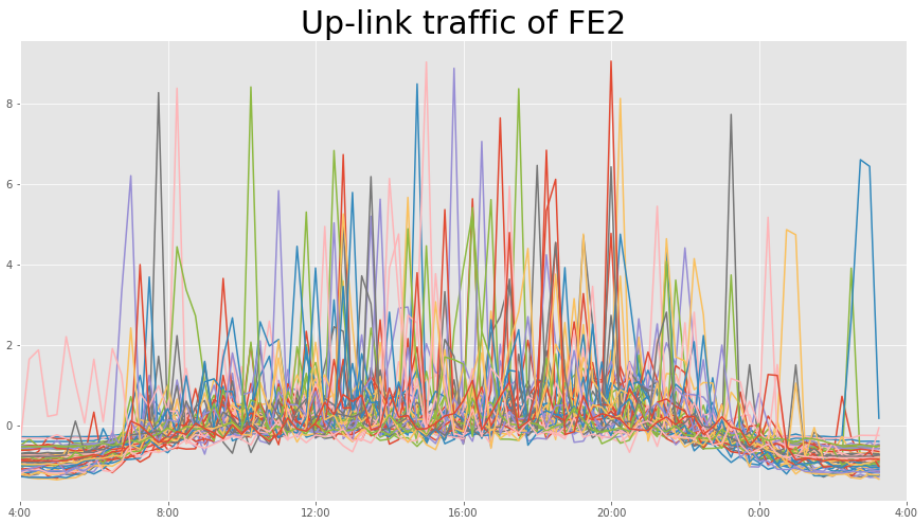



*Figure 3.4: data after day by day normalization*

of traffic of down-link and up-link, and for each feature, we have *96* records in one day.

$$
\begin{aligned}
\overline{x^l} &= \frac{1}{N_c \times N_{d,c} \times 96} \sum_{i=1}^{N_c} \sum_{j=1}^{N_{d,c}} \sum_{k=1}^{96} x_{ij}^{kl} \\
\sigma^l &= \sqrt{\frac{1}{N_c \times N_{d,c} \times 96} \sum_{i=1}^{N_c} \sum_{j=1}^{N_{d,c}} \sum_{k=1}^{96} (x_{ij}^{kl} - \overline{x^l})} \\
& i \in N_c, \quad j \in N_{d,c}, \quad l \in (1,2)
\end{aligned}
\tag{3.10}
$$

Here $\overline{x_i^l}$ and $\sigma_i^l$ represent mean value and variance of $l$-th feature records in all base stations all days. Also $N_{d,c}$ is number of available days in $i$-th cell and $N_c$ is the number of base stations that we have (also included virtual base station). Then $x_{ij}^{kl}$ in original dataset should

be substituted by $x_{ij}^{kl}\prime$ as eq.3.11.

$$x_{ij}^{kl}\prime = \frac{x_{ij}^{kl} - \overline{x^l}}{\sigma^l} \tag{3.11}$$

An example of normalized data is shown in Fig.3.5

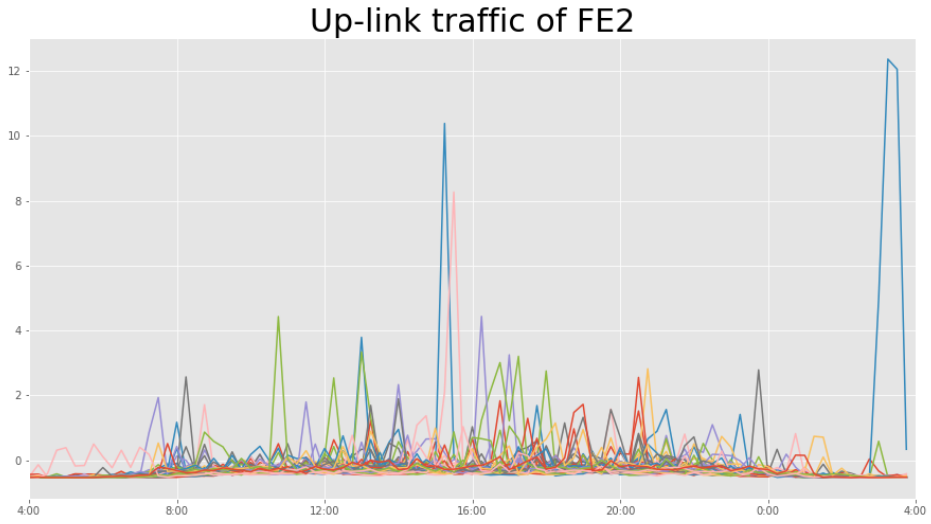



*Figure 3.5: data after all cells normalization*

## 3.4 Removal of Mean

Next procedure after normalization, the value of records are converted into the distance between the value of records and average value as showed in eq.3.12, where $x_{ij}^{kl}\prime$ represent new samples.

$$\begin{aligned} \overline{x_i^{kl}} &= \frac{1}{N_{d,c}} \sum_{j=1}^{N_{d,c}} x_{ij}^{kl} \\ x_{ij}^{kl\prime} &= x_{ij}^{kl} - \overline{x_i^{kl}} \\ j &\in N_{d,c} \end{aligned} \tag{3.12}$$

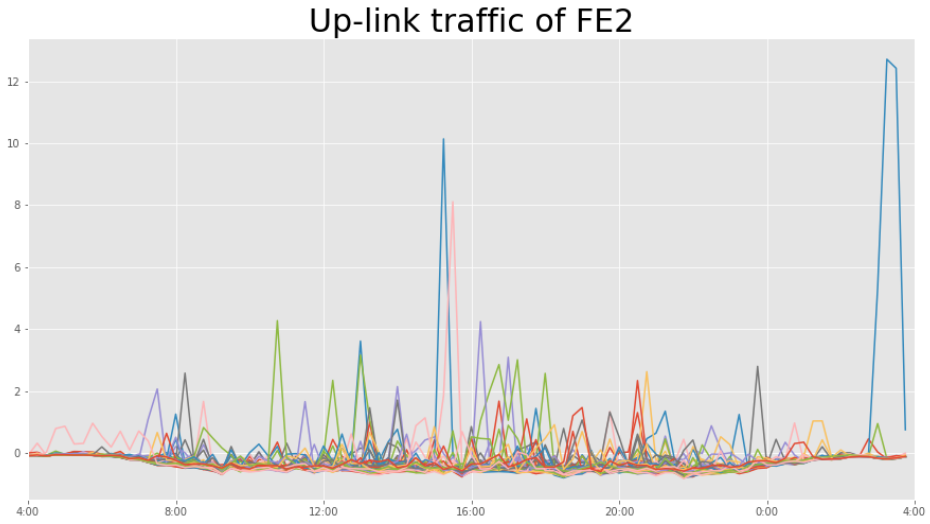



*Figure 3.6: data after removal of mean*

## 3.5 Smoothing

The results after removal is noisy this is evident in figure 3.6. **EWMA** algorithm is chosen to smooth the data series. EWMA represents **exponential weighted moving average**. Moving averages come from statistical analysis. The most basic aim of moving average is to create a series of average values of different subsets of the full data set.

Moving average technique can smooth out the noise of random outliers and emphasize long-term trends or cycles so that it is quite often used in stock analysis.

The Exponentially Weighted Moving Average (EWMA) is an advanced version of moving average for monitoring the process. EWMA averages data in a way that gives less and less weight to older data. In EWMA, the parameter we need to define is $\alpha$ which is called *smoothing factor* or *decay factor*. The smoothing scheme begins by setting $y_i$ to $x_i$, where $y_i$ stands for the observation after smoothing, and $x_i$ stands for the original observation. And the subscripts refer to the timestamps, $1, 2, \ldots, n$. In general, the EWMA is calculated as eq.3.13.

$$y_t = \frac{\sum_{i=0}^{t} w_i x_{t-i}}{\sum_{i=0}^{t} w_i} \tag{3.13}$$

There are also two variants of EWMA, the different is in choosing weights $w_i$ one is called *adjusted EWMA*, and the other called *non-adjusted EWMA* as eq.3.14.

$$\begin{aligned} &\text{Adjust = True:} && w_i = (1-\alpha)^i \\ &\text{Adjust = False:} && w_i = \begin{cases} \alpha(1-\alpha)^i & if\ i < t \\ (1-\alpha)^i & if\ i = t \end{cases} \end{aligned} \tag{3.14}$$

In the experiments, we used adjusted EWMA, An example of data after adjusted EWMA is shown in Fig.3.7

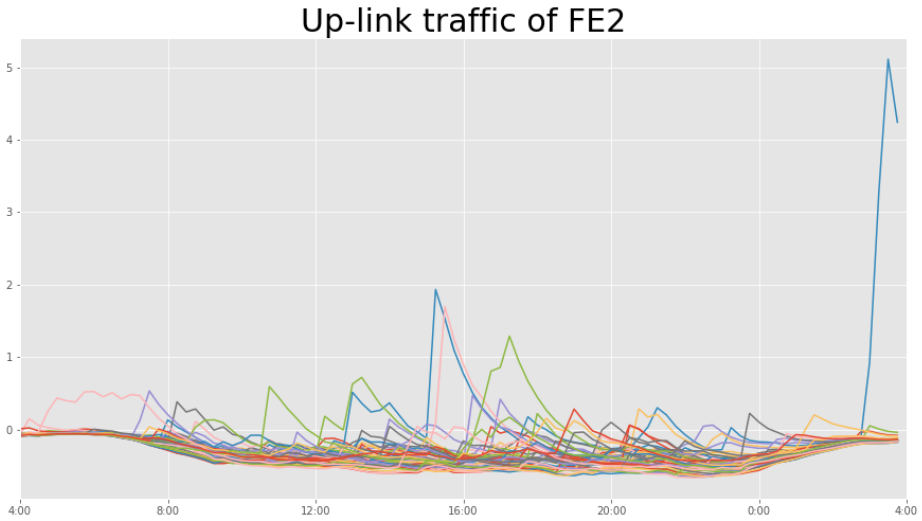



*Figure 3.7: data smoothed by adjusted EWMA*

# Chapter 4

# Clustering

In order to create a model from current data, we counted on *Unsupervised learning*. We desired to have a model that contains the traffic patterns for each cell. In this chapter, we will discuss about applying machine learning *Clustering* onto this application, but first let us discuss about some basic view of **clustering**.

## 4.1 K-means Introduction

**Clustering** is an **unsupervised** technique. The known data set is made just of the $M_{features}$ vectors $\mathbf{y}(n)$, which means each $\mathbf{y}(n)$ is a $M$ dimension vector. The machine learning algorithm, typically using the distances between couples of vectors, finds groups of close vectors and decides that they form a cluster. In the example of the blood analysis, the clustering algorithm might cluster the patients not according to their cardiovascular disease risk but according to the fact that they are male or female, if the gender were such that the vectors are clearly separated.

In the telecommunication field, an example of clustering may be

found in vector quantization. In this case, the probability density function $f_{\mathbf{y}}(\mathbf{u})$ of the $N$-dimensional random vector $\mathbf{y}$ is known and we want to quantize $\mathbf{y}$ according to the rule: if $\mathbf{y} \in \mathcal{R}_k$, then $\mathbf{y}$ is represented (substituted) by vector $\mathbf{x}_k$ which is the centroid of region $\mathcal{R}_k$. In other words, if $\mathbf{y}(n)$ is the $n$-th input of the quantizer, its corresponding output is $\mathbf{y}_Q(n)$, found according to the rule

$$\text{if } \mathbf{y}(n) \in \mathcal{R}_k \quad \text{then} \quad \mathbf{y}_Q(n) = Q(\mathbf{y}(n)) = \mathbf{x}_k, \qquad k = 1, \ldots, K, n \in \mathbb{Z} \tag{4.1}$$

The centroid of the region is mathematically defined as

$$\mathbf{x}_k = \int_{\mathcal{R}_k} \mathbf{u} f_{\mathbf{y}}(\mathbf{u})\, d\mathbf{u}.$$

Regions $\mathcal{R}_k$ must again be Voronoi regions, i.e. $\mathcal{R}_k \cap \mathcal{R}_h = \Phi$ for all $k \neq h$, and $\mathcal{R}_1 \cup \mathcal{R}_2 \cup \cdots \cup \mathcal{R}_K = \mathbb{R}^M$, because one and only one vector must be present at the output of the quantizer.

The Lloyd algorithm finds the optimum regions $\mathcal{R}_k$ (and therefore the optimum vectors $\mathbf{x}_k$, which are the centroids of these regions) that minimize the quantization error

$$e = \sum_{k=1}^{K} \int_{\mathcal{R}_k} \|\mathbf{x}_k - \mathbf{u}\|^2 f_{\mathbf{y}}(\mathbf{u})\, d\mathbf{u}.$$

Once the regions are found by the algorithm, the quantization is performed in a straightforward manner, according to the rule in (4.1). Note that the Lloyd algorithm can be used for lossy compression, in that vector $\mathbf{y}(n)$ is mapped into number $k$ (the index of the region it belongs to), and only $\log_2 K$ bits are sufficient to represent $k$.

This example shows that the values of $\mathbf{x}_k$ are not known in the

quantization problem and the algorithm has to find them, whereas in the previous classification/detection problem of the receiver they are known. Moreover, it is clear from this example that, for a given probability density function $f_{\mathbf{y}}(\mathbf{u})$, the shapes of the regions depend on the value of $K$, which is a design parameter.

**The hard k-means algorithm** Let $\mathbf{y}(n)$ be the $n$-th measured vector with $n = 1, \ldots, N_{samples}$, and assume that $\mathbf{y}(n)$ is a point in the $M$-dimensional space $\mathbb{R}^M$. The task of the k-means algorithm is to cluster together close vectors $\mathbf{y}(n)$. The assumption is that

$$\mathbf{y}(n) = \mathbf{x}(n) + \boldsymbol{\nu}(n)$$

where $\mathbf{x}(n)$ is one among the set $\{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$ and $\boldsymbol{\nu}(n)$ is a zero-mean random Gaussian vector. This assumption is clearly not correct in many cases, but it allows to understand how the algorithm works. The task of the algorithm is to find the estimates of $\mathbf{x}_1, \ldots, \mathbf{x}_K$, knowing $\mathbf{y}(n)$ for $n = 1, \ldots, N_{features}$. If we knew the set $\mathcal{S}_k$ of indexes $n$ defined as

$$\mathcal{S}_k = \{n \in [1, N_{features}] : \mathbf{y}(n) = \mathbf{x}_k + \boldsymbol{\nu}(n)\},$$

(i.e. the set of indexes $n$ such that $\mathbf{y}(n)$ was originated by $\mathbf{x}_k$) then we could estimate $\mathbf{x}_k$ as

$$\hat{\mathbf{x}}_k = \frac{1}{|\mathcal{S}_k|} \sum_{n \in \mathcal{S}_k} \mathbf{y}(n)$$

exploiting the fact that $\boldsymbol{\nu}(n)$ has zero mean; in the previous equation $|\mathcal{S}_k|$ is the cardinality of $\mathcal{S}_k$ (the number of its elements). Point $\hat{\mathbf{x}}_k$ is also called the centroid/barycenter. Of course we don't know $\mathcal{S}_k$ and the algorithm has to find it out.

The algorithm is described as follows:

1. We start with an initial guess of $\hat{\mathbf{x}}_k(0)$, $k = 1, \ldots, K$; this initial guess can be obtained by generating $K$ random vectors of dimension $M$ (typically Gaussian), or by randomly selecting $K$ of the $N_{features}$ vectors $\mathbf{y}(n)$. We set $i := 0$.

2. At the $i$-th step we associate to $\hat{\mathbf{x}}_k(i)$ the points $\mathbf{y}(n)$ which are closer to $\hat{\mathbf{x}}_k(i)$ than to the other points $\hat{\mathbf{x}}_h(i)$ (**assignment step**):

$$n \in \mathcal{S}_k \text{ if } \|\mathbf{y}(n) - \hat{\mathbf{x}}_k(i)\|^2 \leq \|\mathbf{y}(n) - \hat{\mathbf{x}}_h(i)\|^2 \quad \forall h \neq k \qquad n = 1, \ldots, N_{features}$$

3. We evaluate the mean value $\mathbf{m}_k(i)$ of the points $\mathbf{y}(n)$ (or the centroid of the points $\mathbf{y}(n)$) that have been associated with $\hat{\mathbf{x}}_k(i)$ (**update step**):

$$\mathbf{m}_k(i) = \frac{1}{|\mathcal{S}_k|} \sum_{n \in \mathcal{S}_k} \mathbf{y}(n), \quad k = 1, \ldots, K$$

4. We define $\hat{\mathbf{x}}_k(i+1) = \mathbf{m}_k(i)$, we set $i := i+1$, we go back to step 2 until the algorithm converges.

In the assignment step, instead of the Euclidean distance

$$d_{Euc}(\mathbf{y}, \mathbf{x}) = \|\mathbf{y} - \mathbf{x}\|^2 = \sum_{m=1}^{M} |y_m - x_m|^2$$

other distances can be used, for example the Manhattan distance

$$d_{Man}(\mathbf{y}, \mathbf{x}) = \sum_{m=1}^{M} |y_m - x_m|.$$

**The soft k-means algorithm** The soft k-means algorithm differs from the hard k-means algorithm in that the hard assignment step is substituted by a soft assignment step: vector $\mathbf{y}(n)$ is not given to one and only one cluster, but the algorithm evaluates the $K$ probabilities that it belongs to cluster 1, to cluster 2, etc., or, better, it evaluates the responsibility level $r_k(\mathbf{y}(n))$ that each cluster has on vector $\mathbf{y}(n)$, and this responsibility is a real number instead of being either 0 or 1, as in the case of the hard k-means algorithm.

The assumptions of the soft k-means algorithm are the following:

- $\mathbf{y}(n) = \mathbf{x}_k + \boldsymbol{\nu}_k(n)$ with probability $\pi_k$, for $k = 1, \ldots, K$; of course $\sum_{k=1}^{K} \pi_k = 1$, but $\pi_k$ might not be equal to $1/K$ (the hypotheses are not equally likely)
- the dimension of $\mathbf{y}(n)$ is $M$
- $\nu_k(n)$ is an $M$-dimensional, zero-mean, Gaussian random vector and each of its $M$ components have variance $\sigma_k^2$ (the noise variance depends on the cluster)
- parameters $\pi_k$ and $\sigma_k^2$ must be estimated

The algorithm is described as follows:

1. Start from an initial set of values $\mathbf{x}_k(0)$, $\sigma_k^2(0) = 1$, $\pi_k(0) = 1/K$, $k = 1, \ldots, K$. Set $i = 0$

2. **(Assignment step)** for $k = 1, \ldots, K$ and $n = 1, \ldots, N$:

$$r_k(\mathbf{y}(n); i) = \frac{\pi_k(i) \frac{1}{(2\pi\sigma_k^2(i))^{M/2}} e^{-\frac{1}{2\sigma_k^2(i)}\|\mathbf{y}(n)-\mathbf{x}_k(i)\|^2}}{\sum_{s=1}^{K} \pi_s(i) \frac{1}{(2\pi\sigma_s^2(i))^{M/2}} e^{-\frac{1}{2\sigma_s^2(i)}\|\mathbf{y}(n)-\mathbf{x}_s(i)\|^2}}$$

3. **(Update step)**

$$\mathbf{x}_k(i+1) = \frac{\sum_{n=1}^{N} r_k(\mathbf{y}(n); i)\mathbf{y}(n)}{\sum_{n=1}^{N} r_k(\mathbf{y}(n); i)}, \quad k = 1, \ldots, K$$

$$\sigma_k^2(i+1) = \frac{\sum_{n=1}^{N} r_k(\mathbf{y}(n); i)\|\mathbf{y}(n) - \mathbf{x}_k(i+1)\|^2}{M \sum_{n=1}^{N} r_k(\mathbf{y}(n); i)}, \quad k = 1, \ldots, K$$

$$\pi_k(i+1) = \frac{\sum_{n=1}^{N} r_k(\mathbf{y}(n); i)}{\sum_{k=1}^{K} \sum_{n=1}^{N} r_k(\mathbf{y}(n); i)}, \quad k = 1, \ldots, K$$

4. set $i := i + 1$, go back to step 2 until a convergence condition is met.

This soft version of the k-means algorithm is more flexible, but it might suffer from numerical and convergence problems. Its use is recommended if the results of the more stable hard k-means algorithm are not satisfactory.

**Local Minimum in K-means** Notice that in K-means algorithm, if enough time is given, the results will always converge, however this may be to a local minimum. This is highly caused by the **difference of initialization** of the centroids. As a result, in our programs, the computation is often done several times, with different initializations of the centroids. One method to help address this issue is the k-means++

initialization scheme, which has been implemented in scikit-learn (use the init='k-means++' parameter). This scheme initializes the centroids to be (generally) distant from each other, leading to provably better results than just a random initialization.

## 4.2 DBSCAN Introduction

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) was originally proposed in the paper *A density-based algorithm for discovering clusters in large spatial databases with noise* [7] in 1996. The **density based** clustering algorithm, is used to identify clusters of any shape in data set containing noise and outliers that lie alone in low-density regions. The advantages of this algorithm are that:

- Unlike k-means, it dose not require the user to specify the number of clusters to be generated
- It can discover arbitrary shape clusters
- It can identify outliers

As illustrated in Figure.4.1, clusters are made of dense region in data space and separated by low dense regions. DBSCAN is based on the intuitive notion of "clusters" and "noise". The density of points in a cluster is comparatively higher than the density of non-clustered points ("area of noise", outliers).

While the disadvantage is that it can not handle varying densities data and the results are sensitive to parameters, and bottleneck of DB-

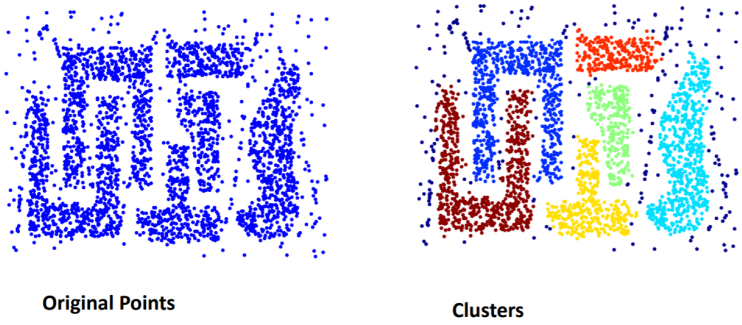



*Figure 4.1: When DBSCAN works well*

SCAN is that it is not easy to determine the correct set of parameters, the detail of parameter will be discussed in next part. Figure. See two examples of clustering performed by DBSCAN in Figs 4.1 and 4.2

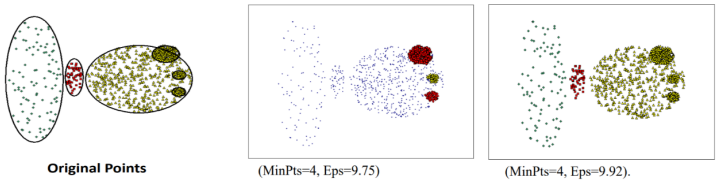



*Figure 4.2: When DBSCAN works bad*

### 4.2.1 Parameters and Definitions

According to *A density-based algorithm for discovering clusters in large spatial databases with noise* [7], there are 2 parameters and several definitions in this algorithm.

Parameter 1: $\epsilon$ **Neighborhood** Points within a radius $\boldsymbol{\epsilon}$ from point $p$, denoted by $\boldsymbol{N_\epsilon(p)}$ is defined as eq.4.2.

$$\boldsymbol{N_\epsilon(p) : \{q \in D \mid dist(p,q) \leq \epsilon\}} \tag{4.2}$$

Parameter 2: **MinPts (Minimal Reachable Points)** For a point $p$, if the number of points in p's reachable neighborhood is more than MinPts, than this $p$ is considered as a core point.

Definition 1: **Directly Density Reachable** A point $p$ is directly reachable from a point $q$ if

$$\begin{aligned} &1. \quad \boldsymbol{p \in N_\epsilon(q)} \\ &2. \quad \boldsymbol{|N_\epsilon(q)| \geq MinPts} \text{ (core point condition)} \end{aligned} \tag{4.3}$$

Definition 2: **Density Reachable** A point p is density reachable from a point q if there is a chain of points $\boldsymbol{p_1, \ldots, p_n}$ and $\boldsymbol{p_1 = p, /; , p_n = q}$ such that $\boldsymbol{p_{i+1}}$ is directly density reachable from $\boldsymbol{p_i}$.

Definition 3: **Cluster** Let D be a set of points. A cluster C is a non-empty subset of D satisfying the following conditions.

1. ∀p, q: if p ∈ C and q is density-reachable from p, then q ∈ C
2. ∀p, q ∈ C, p density-connected to q

(4.4)

Definition 4: **Noise** Let $\boldsymbol{C_1, \ldots, C_k}$ be the clusters of the database D, $\boldsymbol{i = 1, \ldots, k}$. Then we define the *noise* as the set of points in the database D not belonging to any cluster $\boldsymbol{C_i}$

$$\boldsymbol{noise = \{p \in D | \forall i : p \notin C_i\}} \tag{4.5}$$

Given $\boldsymbol{\epsilon}$ and **MinPoints**, a point is a *core point* if it has more than **MinPoints** number of points within $\boldsymbol{\epsilon}$, these points are the inte-

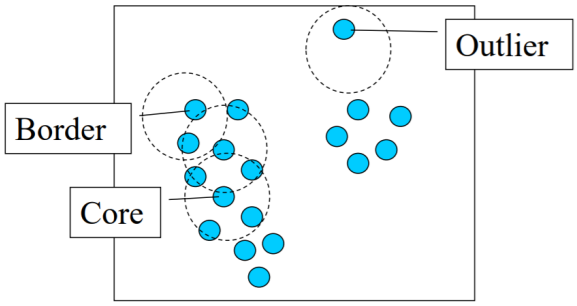



*Figure 4.3: Illustration of DBSCAN Structure*

rior of a cluster. A **border** point has fewer than **MinPoints** within $\epsilon$, but is in the neighborhood of a core point. A **noise** point is any point that is not a **core** point nor a **border** point. In Fig.4.3 the definitions are clarified.

## 4.3 Clustering Performance

Suppose that, until now, we already had some preliminary results from machine learning model by using k-means algorithm (or other clustering algorithms), we still have to find a quantitative method to evaluate if our results as reasonable. In this section, we will introduce some well known unsupervised learning evaluation metrics. In general, the performance evaluation of a clustering algorithm is not as trivial as counting the number of errors or the precision and recall of a supervised classification algorithm. In practice, the knowledge of the ground truth classes, which requires manual assignment by human annotators, is almost never available. Also, in general any evaluation method should never take the absolute values of the cluster labels into account, the reason is that the labels in clustering is randomly assigned, but since in our application, each cluster has its own meaning, it will be useful

to *re-order* the numerical value of the cluster's labels.

Considering, the output of a k-means clustering is

$$\mathbf{C} = \{\boldsymbol{C_1}, \boldsymbol{C_2}, ..., \boldsymbol{C_k}\} \tag{4.6}$$

Firstly we define some parameter as following, and notice that $\mathbf{x}$ represents a sample which is a row vector with the shape $[\mathbf{1}, \boldsymbol{N_f} \times \mathbf{96}]$:

1. *dist*$(\mathbf{x}, \mathbf{y})$ computes the distance between two samples($\mathbf{x}$, $\mathbf{y}$), here the distance means *euclidean distance*

$$\boldsymbol{dist}(\mathbf{x}, \mathbf{y}) = \sqrt{(\boldsymbol{x_1} - \boldsymbol{y_1})^2 + (\boldsymbol{x_2} - \boldsymbol{y_2})^2 + \cdots + (\boldsymbol{x_n} - \boldsymbol{y_n})^2} \tag{4.7}$$

2. $\boldsymbol{\mu_i}$ represents centroid of a cluster $\boldsymbol{C_i}$ and $\boldsymbol{N_i}$ represents the number of samples in the cluster $\boldsymbol{C_i}$

$$\boldsymbol{\mu_i} = \overline{\boldsymbol{x_i}} = \frac{\mathbf{1}}{\boldsymbol{N_i}} \sum_{\mathbf{1} \le i \le \boldsymbol{N_i}} \mathbf{x_i} \tag{4.8}$$

3. *avg*$(\boldsymbol{C_i})$ computes the average distance of each sample-pair in cluster $\boldsymbol{C_i}$

$$\boldsymbol{avg}(\boldsymbol{C_i}) = \frac{\mathbf{2}}{\boldsymbol{N_i}(\boldsymbol{N_i} - \mathbf{1})} \sum_{\mathbf{1} \le i \le j \le \boldsymbol{N_i}} \boldsymbol{dist}(\mathbf{x_i}, \mathbf{x_j}) \tag{4.9}$$

where $\mathbf{x_i}, \mathbf{x_j} \in \boldsymbol{C_i}$

4. *diam*$(\boldsymbol{C_i})$ is largest distance of sample-pair in cluster $\boldsymbol{C_i}$

$$\begin{aligned} &\boldsymbol{diam}(C_i) = \max_{1 \le i \le j \le N_i} \boldsymbol{dist}(\mathbf{x_i}, \mathbf{x_j}) \\ &\text{where } \mathbf{x_i}, \mathbf{x_j} \in \boldsymbol{C_i} \end{aligned} \tag{4.10}$$

5. $\boldsymbol{d_{min}}(\boldsymbol{C_i}, \boldsymbol{C_j})$ is the smallest distance for pair ($\mathbf{x}$,$\mathbf{y}$) where $\mathbf{x} \in \boldsymbol{C_i}$ ,$\mathbf{y} \in \boldsymbol{C_j}$

$$\boldsymbol{d_{min}}(C_i, C_j) = \min_{\mathbf{x} \in C_i, \mathbf{y} \in C_j} \boldsymbol{dist}(\mathbf{x}, \mathbf{y}) \tag{4.11}$$

6. $\boldsymbol{d_{cen}}(\boldsymbol{C_i}, \boldsymbol{C_j})$ is the centroids distance between two clusters $\boldsymbol{C_i}$ and $\boldsymbol{C_j}$

$$\boldsymbol{d_{cen}}(C_i, C_j) = \boldsymbol{dist}(\boldsymbol{\mu_i}, \boldsymbol{\mu_j}) \tag{4.12}$$

### 4.3.1 Davies-Bouldin Index

The Davies-Bouldin Index(DBI) is defined in eq.4.13

$$\boldsymbol{DBI} = \frac{1}{\boldsymbol{k}} \sum_{i=1}^{k} \max_{j \ne i} \left( \frac{\boldsymbol{avg}(C_i) + \boldsymbol{avg}(C_j)}{\boldsymbol{d_{cen}}(C_i, C_j)} \right) \tag{4.13}$$

Since K means algorithm tends to produce clusters with low intra-cluster distances (*high intra-cluster similarity*) and high inter-cluster distances (*low inter-cluster similarity*), the clustering implementation that produces a collection of clusters with the smallest Davies-Bouldin index is considered the best algorithm based on this criterion.

### 4.3.2 Dunn Index

The Dunn Index(DI) is defined as eq.4.14

$$\boldsymbol{DI = \min_{1 \leq i \leq k} \{ min_{j \neq i} ( \frac{d_{min}(C_i, C_j)}{\max_{1 \leq l \leq k} diam(C_l)} ) \}} \tag{4.14}$$

The Dunn index aims to identify how dense and how well-separated clusters are. Since internal criterion seeks clusters with high intra-cluster similarity and low inter-cluster similarity (same criterion when we choosing best DBI), algorithms that produce clusters with high Dunn index are more desirable.

### 4.3.3 Silhouette Score

The Silhouette Coefficient(SI) for a datum $i$ is calculated using the mean intra-cluster distance $\boldsymbol{a_i}$ (eq.4.15) and the mean nearest-cluster distance $\boldsymbol{b_i}$ (eq.4.16). The Silhouette Coefficient (eq.4.18) for a sample $i$ is (b - a) / max(a, b).

Just to clarify, nearest-cluster distance is the distance between a sample and centroid of nearest cluster that the sample is not a part of (eq.4.17). The best value of $\boldsymbol{s_i}$ is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. A negative value generally indicates that a sample has been assigned to the wrong cluster, as a different cluster is more similar.

$$\boldsymbol{a_i} = \frac{\boldsymbol{1}}{\boldsymbol{N_i - 1}} \sum_{\boldsymbol{j=1, j \neq i}}^{\boldsymbol{N_i}} \boldsymbol{d(i, j)} \qquad \forall(i,\, j) \text{ in the same cluster} \tag{4.15}$$

$$\boldsymbol{b_i} = \frac{\mathbf{1}}{\boldsymbol{K-1}} \sum_{\boldsymbol{k,k\neq c_i}}^{\boldsymbol{K}} \boldsymbol{dmin(i, C_k)} \qquad \boldsymbol{c_i} \text{ is cluster index for } i \tag{4.16}$$

$$\boldsymbol{dmin(i, C_k)} = \min_{\boldsymbol{j \in C_k}} \boldsymbol{d(i, j)} \tag{4.17}$$

$$\boldsymbol{s_i} = \frac{\boldsymbol{b_i - a_i}}{\mathbf{max}\{\boldsymbol{a_i, b_i}\}} \tag{4.18}$$

$$\boldsymbol{SI} = \frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{i}}^{\boldsymbol{N}} \boldsymbol{s_i} \tag{4.19}$$

## 4.4 K-means Clustering Implementation

In this section, the detail of procedure of K-means clustering implementation will be explained. First of all, since we have 103 base stations in total, it is not possible to analyze all of them separately, so we pick some of them as typical samples and to see the performance of algorithms.

This work will show the clustering result for 7 base station which 3 of them are real base stations and 4 are virtual stations. Namely they are *BL2, GT1, Group1, Group2, Group3, Group4, SC*. And those 4 virtual base stations are a relation of inclusion, Group1 $\subset$ Group2 $\subset$ Group3 $\subset$ Group4.

First, we start K-means clustering with different $\boldsymbol{k}$ (number of clusters we desired to generate), $\boldsymbol{k = 3, \ldots, 9}$. And in k-means algorithm, the procedure is initialized randomly so it is common that we have different results in different clustering trial, thats explained that in the

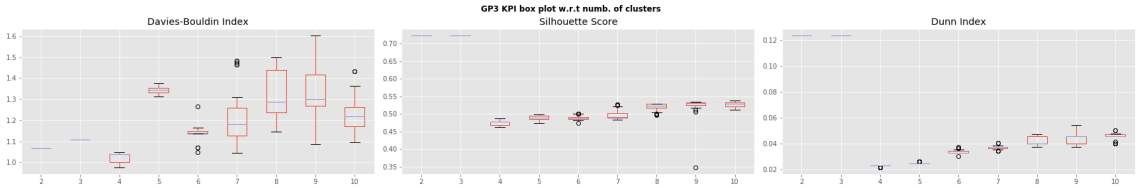



*Figure 4.4: Performance Index Box plot for GP3*

plot 4.4, we have different performance index values for same k in most cases (while we still have some cases that the performance index values are consistent in 50 time trials, eg, k=2 and 3 for base station GP3). After the performance index values are calculated for each trail, we must compare the performance score for each trail and we would say this result is a **better** one if it has larger Davies-Bouldin Index values, lower Dunn Index values and higher Silhouette Score values. A naive equation to combine all 3 performance index in eq.4.20

$$\begin{aligned} \boldsymbol{weights} &= [\boldsymbol{w_1 = -0.5, w_2 = 1.4, w_3 = 1}] \\ \boldsymbol{Score} &= \boldsymbol{w_1 \times DBI + w_2 \times DI + w_3 \times SI} \end{aligned} \tag{4.20}$$

## 4.5 DBSCAN Implementation

As we discussed in chapter.4.2, DBSCAN is a **density based** clustering algorithm but we do not need to specify the number of clusters as we did in K-means. The input parameters are $\boldsymbol{\epsilon}$ **Neighborhood** and **Minimal Points** and the algorithm based on the pairwise distance between samples.

### 4.5.1 Distance Metric

Finding a good distance metric in featured space is crucial especially in this project since we have a very high dimensions in machine learning perspective. Some distance metric learning algorithms are proposed in *Distance Metric Learning: A Comprehensive Survey* [14]. While in this project, we used 2 distance metric *Euclidean distance* and *Dynamic Time Wrapping distance* so that we can have a sort of comparisons of results.

#### Euclidean Distance

The *Euclidean distance* between time sequences $\mathbf{X} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n})$ and $\mathbf{Y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_n})$ is given by the extension of *Pythagorean formula*:

$$\begin{aligned} \boldsymbol{dist}(\mathbf{X}, \mathbf{Y}) = \boldsymbol{dist}(\mathbf{Y}, \mathbf{X}) &= \sqrt{(\boldsymbol{x_1} - \boldsymbol{y_1})^2 + (\boldsymbol{x_2} - \boldsymbol{y_2})^2 + \cdots + (\boldsymbol{x_n} - \boldsymbol{y_n})^2} \\ &= \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} (\boldsymbol{x_i} - \boldsymbol{y_i})^2} \end{aligned} \tag{4.21}$$

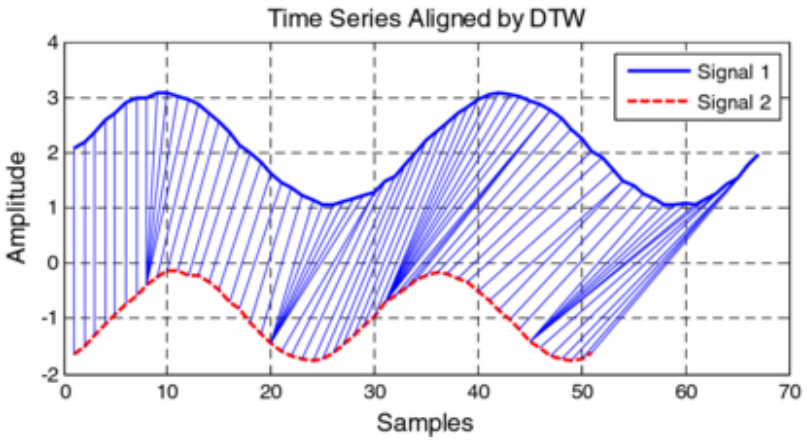



*Figure 4.5: Time Series Aligned by DTW*

### DTW

Dynamic time warping ($DTW$) is one of the most popular algorithm in the time series analysis. It measures the similarity between two temporal sequences regardless it might vary in speed. For example, the similarities in speaking could be measured by DTW, even if one person was talking faster then the other or they are talking in different tunes during the recording. DTW algorithm has been applied widely in video, audio and graphics data analysis and recognition. A very significant feature of DTW is that it is able to automatically cope with time deformations and different speeds associated with time-dependent data which suitable in our application where the interesting event in a day might get affected by the event length and event scale.

In general, comparing 2 sequences $\mathbf{X} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_{N1}})$ of length N1 $\in \mathbb{N}$ and $\mathbf{Y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots, \boldsymbol{y_{N2}})$ of length N2 $\in \mathbb{N}$. These discrete time sequences are sampled at equidistant points in time domain. But actually, in our application the sequence is *multi-dimensional time series* (MDT). MDT consist of $M$ individual time

series ($\boldsymbol{M \geq 2}$) and each time series has $\boldsymbol{N}$ observations, but for sake of complexity, we assumed that all $\mathbf{M}$ are the measurements for same KPI:

$$\begin{aligned}
\mathbf{X_1} &= (\boldsymbol{x_{1,1}}, \boldsymbol{x_{1,2}}, \ldots, \boldsymbol{x_{1,N}}) \\
\mathbf{X_2} &= (\boldsymbol{x_{2,1}}, \boldsymbol{x_{2,2}}, \ldots, \boldsymbol{x_{2,N}}) \\
&\cdots \\
\mathbf{X}_M &= (\boldsymbol{x_{M,1}}, \boldsymbol{x_{M,2}}, \ldots, \boldsymbol{x_{M,N}}) \\
&\Downarrow \\
\mathbf{X} &= (\boldsymbol{x_{1,1}}, \ldots, \boldsymbol{x_{1,N}}, \boldsymbol{x_{2,1}}, \ldots, \boldsymbol{x_{2,N}}, \boldsymbol{x_{3,1}}, \ldots, \boldsymbol{x_{3,N}})
\end{aligned} \tag{4.22}$$

To calculate DTW distance, we have to build *cost matrix* $\boldsymbol{C} \in \mathbb{R}^{\boldsymbol{N_1} \times \boldsymbol{N_2}}$, by calculating each pair of elements in the sequences $\mathbf{X}$ and $\mathbf{Y}$. The cost matrix is defined by eq.4.23, and the distance here is *Euclidean Distance*.

$$\mathbf{C}(\boldsymbol{n}, \boldsymbol{m}) = \boldsymbol{dist}(\boldsymbol{x_n}, \boldsymbol{y_m}) \tag{4.23}$$

In the cost matrix, an (N,M)-wraping path $\boldsymbol{p} = (\boldsymbol{p_1}, \ldots, \boldsymbol{p_L})$ is an alignment between sequence $\mathbf{X}$ and $\mathbf{Y}$. The *total cost* $\boldsymbol{c_p}(\mathbf{X}, \mathbf{Y})$is defined as the summation of costs along the wraping path.

$$\boldsymbol{c_p}(\mathbf{X}, \mathbf{Y}) = \sum_{\boldsymbol{l=1}}^{\boldsymbol{L}} \boldsymbol{c}(\boldsymbol{x_{n_l}}, \boldsymbol{y_{m_l}}) \tag{4.24}$$

Then the DTW distance is to the minimum cost path within the cost

matrix.

$$\begin{aligned} \boldsymbol{DTW}(\mathbf{X},\mathbf{Y}) &= \boldsymbol{c_{p^*}}(\mathbf{X},\mathbf{Y}) \\ &= \boldsymbol{min}\{\boldsymbol{c_p}(\mathbf{X},\mathbf{Y}) \mid \boldsymbol{p} \text{ is an (N,M) warping path}\} \end{aligned} \tag{4.25}$$

Typically,if x and y are similar to each other, c(x, y) is small (low cost), and otherwise c(x, y) is large (high cost). And because DTW is able to compare similarity between sequences with different length, one further work could be we simulate some event behavior (just for event instead of 24-hours data) and try DBSCAN and assign this event as core point.

## 4.6 Results

In this section, we will show the results of implementation of clustering and discuss them. After comparing the results, we see there is no much difference in selecting 3 features *down-link traffic, up-link traffic, connected user* and just 1 feature *total traffic*. So the results shown below are mixed by 3 features and 1 feature.

### 4.6.1 K-means Results

**Different results in 50 trials** The K-Means algorithm leads to different results even if each trial is executed under same parameters setting (caused by random initialization). In order to overcome this phenomenon, the K-Means clustering algorithm is executed for $\boldsymbol{k = 2, 3, \ldots, 10}$, and 50 times for each value of k, and take 3 features: *uplink traffic,downlink traffic, connected user*, or just 1 feature: *total traffic*, where *total traffic* = *uplink traffic* + *downlink traffic*.

One interesting point is that, the number of different results can be viewed as the base station complexity. Because, in some how, it represents the stability of the base station. To be more clear, as the Table.4.1 indicates the number of different results of GP2 (55 & 56) is much less than the different results of GT1(316 & 305) out of 500 trials ($= \mathbf{50 \times 10}$) in total.

| cell name | BL2 | GT1 | GP1 | GP2 | GP3 | GP4 | SC |
|---|---|---|---|---|---|---|---|
| 3–features | 295 | 316 | 164 | 55 | 142 | 152 | 184 |
| 1-feature | 289 | 305 | 215 | 66 | 156 | 107 | 211 |

*Table 4.1: Number of different clustering outputs received out of 500 trials*

Fig.4.6 shows 3 clustering trails, one for GP2 and two for GT1, fig.4.6b and fig.4.6c represent different clustering results for cell GT1 when k=6. As it shows, the daily behavior's variation in cell GP2 is much simpler than in GT1. For the more complexer behavior, the performance weights tuning is more necessary, because it is harder for this cell to select better clustering among all the results.

Fig.4.7 shows the clustering KPIs and its performance index are plot. The 3 dimensions in plots represent Davies-Bouldin Index, Silhouette Score and Dunn Index, and color represents the performance score, each scatter point represents one trial in clustering with different number of clusters. The brighter color of the point indicates the better clustering performance, the performance is evaluated as last section discussed eq.4.26.

$$\begin{aligned} \boldsymbol{weights} &= [\boldsymbol{w_1 = -0.5, w_2 = 1.4, w_3 = 1}] \\ \boldsymbol{Score} &= \boldsymbol{w_1 \times DBI + w_2 \times DI + w_3 \times SI} \end{aligned} \tag{4.26}$$

Every results from clustering trials are saved and the performance scores are calculated according to eq.4.13, eq.4.19, eq.4.14 and eq.4.26. Then select highest score as the final clustering results of that base station.

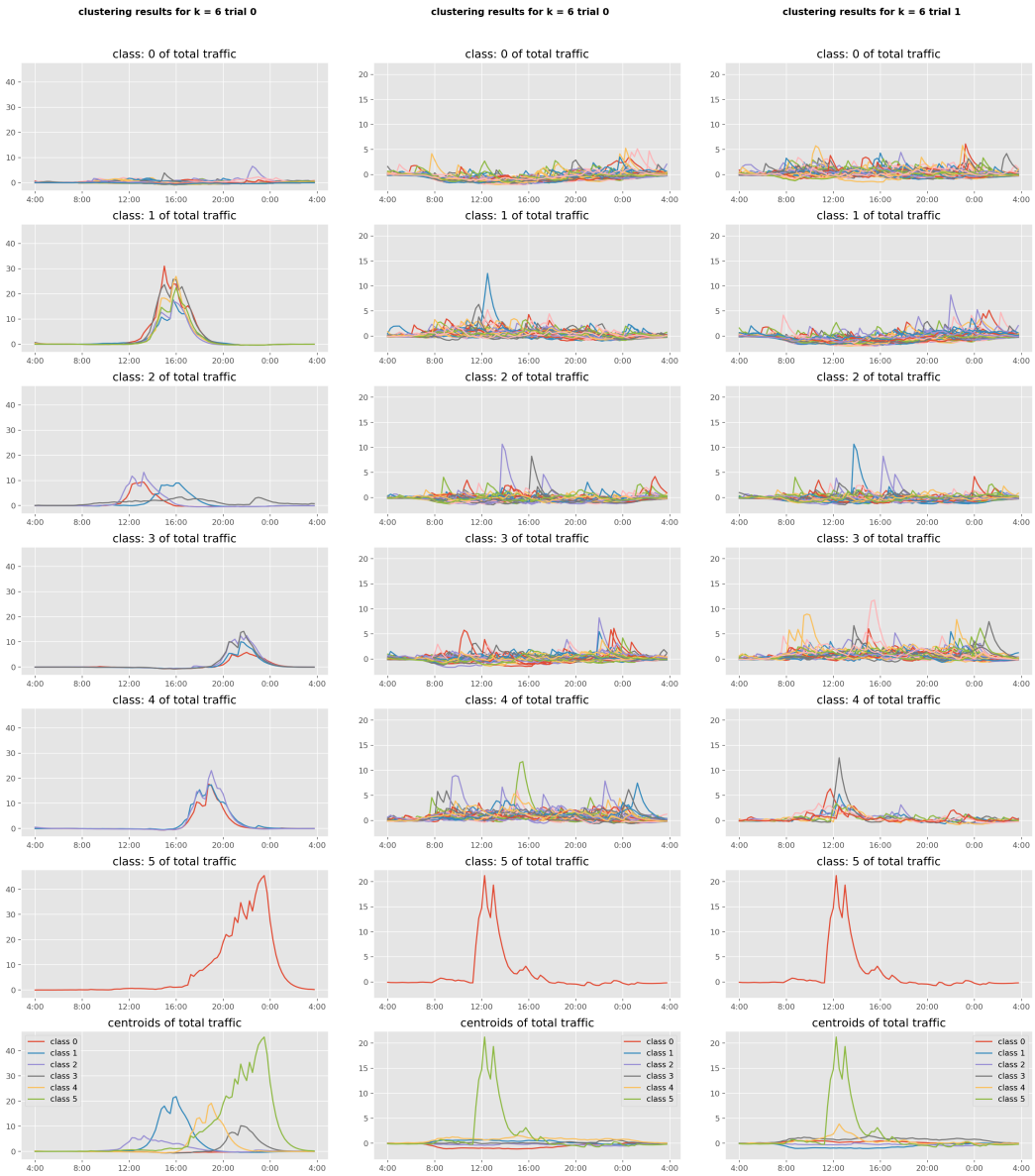
The final results for clustering phase is a set of labels for each pair of cell-day and one heat map shown in Fig4.9. Each column represents a day and each row represents a cell. The dark color represents the abnormal day and light color represents normal day, gray color is the day of no-data day. From the heat map, it is obvious to see the periodicity in horizontal direction, and actually the periodicity is 7 (days).

In vertical direction, we can see that the geographic adjacent cell are more similar than those far away.

### 4.6.2 DBSCAN Results

The DBSCAN clustering are executed by setting $\boldsymbol{\epsilon}$ and $\boldsymbol{minPoints}$ parameters as discussed in chapter4.2. In fig.4.10, it compare the results of DBSCAN clustering with $\boldsymbol{\epsilon = 10}$ $\boldsymbol{minPoints = 5}$ and $\boldsymbol{\epsilon = 1}$ $\boldsymbol{minPoints = 5}$. Theoretically, with $\boldsymbol{\epsilon}$ decreased, we will have more samples labeled as outliers (*class -1*). Although it is not so apparent in the fig.4.10a and fig.4.10b, but the lines (samples) in class 0 in first figure is more than the lines in class 0 in 2nd figure.

Fig.4.11 and Fig.4.12 shows clustering heat map produced by DBSCAN. From the figure we can see that although the output are not identical, the DTW distance metric has more samples in cluster-0 and cluster-1 (low event day), but the results almost same. The detail of the difference among the different clusters will be discuss in next chapter.

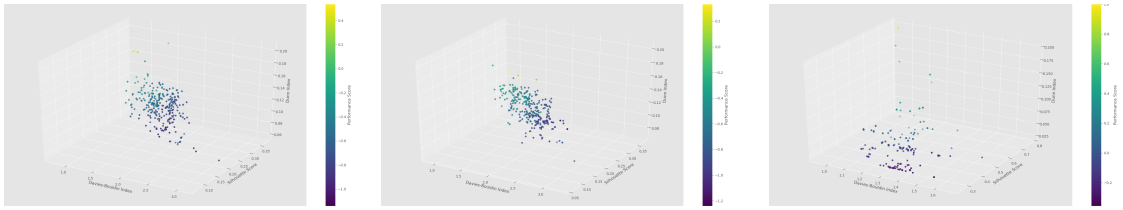



*(a) Clustering result GP2 k = 6 (total traffic)*

*(b) Clustering result GT1 k = 6, 1st trial (total traffic)*

*(c) Clustering result GT1 k = 6, 2nd trial (total traffic)*

*Figure 4.6: Base station complexity comparison between GP2 and GT1.*



(a) KPIs plot BL2 (b) KPIs Plot GT1 (c) KPIs Plot GP4

*Figure 4.7: KPIs and Performance Score which 3 dimensions are Davies-Bouldin Index, Silhouette Score and Dunn Index and color represents performance score.*

(a) K-means Results BL2 (b) K-means Results GT1 (c) K-means Results GP4

*Figure 4.8: Best Clustering Results whose gives highest Performance Score in the Fig.4.7*

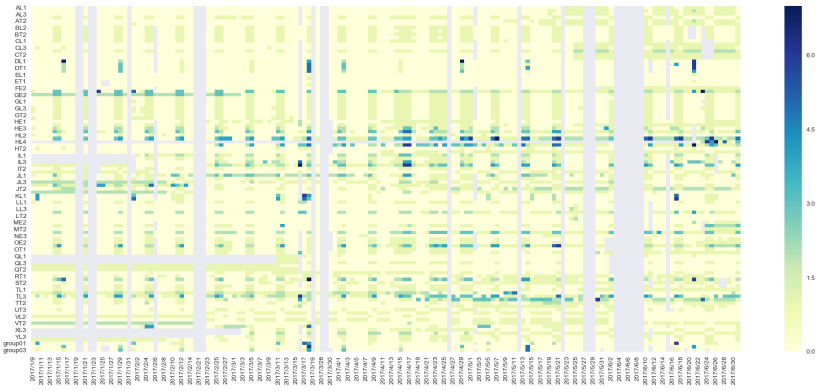
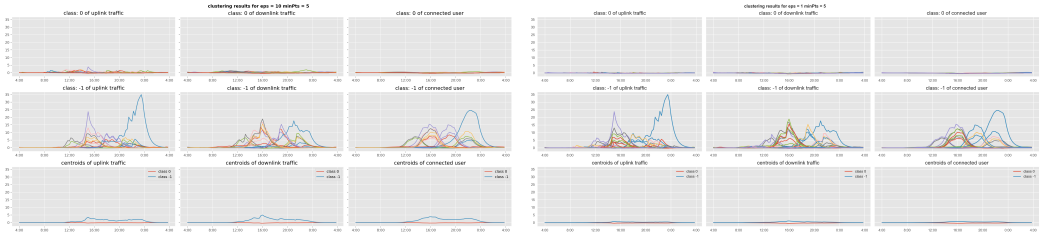


*Figure 4.9: Clustering Heat map*



*(a) DBSCAN Clustering GP2 with setting parameter eps=10 minPts=5*

*(b) DBSCAN Clustering GP2 with setting parameter eps=1 minPts=5*

*Figure 4.10: DBSCAN Clustering Results*

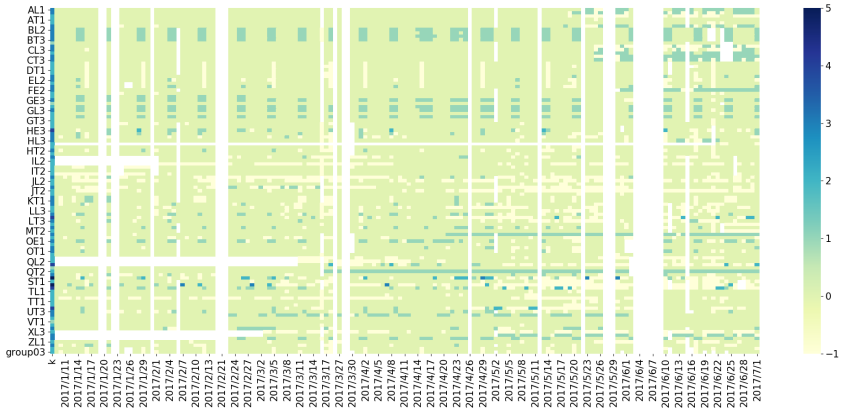


*Figure 4.11: Clustering Heat map by using DBSCAN DTW distance*

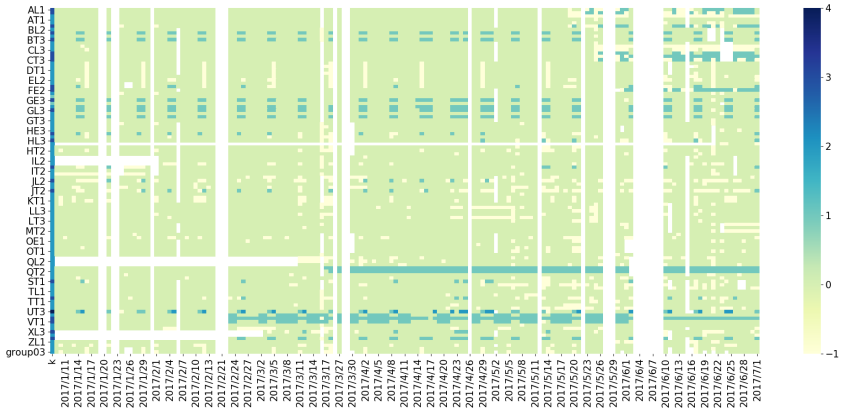


*Figure 4.12: Clustering Heat map by using DBSCAN Euclidean distance*

# Chapter 5

# Classification and Self-optimization

In SON network, classification can be applied to autonomously detect cells that are not operating properly due to possible failures. Some solutions for this problems have been proposed in *Detection of Sleeping Cells in LTE Networks Using Diffusion Maps* [3] and *Diagnosis based on genetic fuzzy algorithms for LTE Self-Healing* [6]. In particular, in [3], the authors propose a solution based on diffusion maps, by means of clustering schemes so as to detect abnormal behaviors from a "sleeping" base station. [6] presents a solution based on fuzzy logic for the automatic diagnosis of troubleshooting system. In order to determine if a failure occurs, a controller is designed to receive as an input a set of representative KPIs.

In this experiments, we tried different popular classifiers, *Minimum Distance Classifier*, *Gaussian Navies Bayes Classifier*, *Convolution Neural Network*, in ML field. And we will benchmark the performances of those classifiers. First let us introduce the evaluation index in classification fields.

## 5.1 Classification

**Classification** is a **supervised** technique. The known data set is made of $\boldsymbol{N_{meas}}$ vectors $\mathbf{y}(\boldsymbol{n})$ (with $\boldsymbol{M}$ elements) and their corresponding class $\boldsymbol{c}(\boldsymbol{n})$. The class $\boldsymbol{c}(\boldsymbol{n})$ of $\mathbf{y}(\boldsymbol{n})$ is known because typically it is given by the human brain. An example can be the following: a medical doctor analyses the $\boldsymbol{M}$ blood parameters of a patient and classifies him/her as a patient with high or low risk of cardiovascular diseases; there are two classes (high or low risk) and the medical doctor performs the classification, using his/her experience. Machine learning algorithms exist that, using a known data set $\{\mathbf{y}(\boldsymbol{n}), \boldsymbol{c}(\boldsymbol{n})\}_{\boldsymbol{n=1}}^{\boldsymbol{N_{meas}}}$, estimate the class $\hat{\boldsymbol{c}}(\boldsymbol{n})$ of a new vector $\mathbf{y}(\boldsymbol{n})$. Note that, whatever the classification algorithm is, the decision/classification rule can only be:

$$\hat{\boldsymbol{c}}(\boldsymbol{n}) = \boldsymbol{k} \quad \text{if} \quad \mathbf{y}(\boldsymbol{n}) \in \boldsymbol{\mathcal{R}_k}$$

where $\boldsymbol{\mathcal{R}_k}$ is the decision region for class $\boldsymbol{k}$ (a subset of $\mathbb{R}^{\boldsymbol{N}}$), and, since we typically want a unique decision, the regions must be such that $\boldsymbol{\mathcal{R}_k} \cap \boldsymbol{\mathcal{R}_h} = \boldsymbol{\Phi}$ for all $\boldsymbol{k} \neq \boldsymbol{h}$, and $\boldsymbol{\mathcal{R}_1} \cup \boldsymbol{\mathcal{R}_2} \cup \cdots \cup \boldsymbol{\mathcal{R}_K} = \mathbb{R}^{\boldsymbol{M}}$.

In the telecommunication field, the detector of the receiver has an input vector $\mathbf{y}(\boldsymbol{n})$ and must decide which of the possible symbols $\mathbf{x}_{\boldsymbol{k}}$, $\boldsymbol{k} = \boldsymbol{1, \ldots, K}$ has been transmitted, knowing that $\mathbf{y}(\boldsymbol{n}) = \mathbf{x}(\boldsymbol{n}) + \boldsymbol{\nu}(\boldsymbol{n})$, being $\mathbf{x}(\boldsymbol{n})$ in the set $\{\mathbf{x}_{\boldsymbol{k}}\}_{\boldsymbol{k=1}}^{\boldsymbol{K}}$. This problem is typically referred to as a hypothesis testing problem, since the receiver tests the hypotheses:

$$\boldsymbol{\mathcal{H}_1} : \mathbf{y}(\boldsymbol{n}) = \mathbf{x_1} + \boldsymbol{\nu}(\boldsymbol{n})$$

$$\boldsymbol{\mathcal{H}_2} : \mathbf{y}(\boldsymbol{n}) = \mathbf{x_2} + \boldsymbol{\nu}(\boldsymbol{n})$$

$$\vdots$$

$$\boldsymbol{\mathcal{H}_K} : \mathbf{y}(\boldsymbol{n}) = \mathbf{x}_{\boldsymbol{K}} + \boldsymbol{\nu}(\boldsymbol{n})$$

and selects the "best" one according to some criterion (minimum error probability, for example). In the machine learning jargon, this hypothesis testing problem is a classification problem and the estimated class is $\hat{\boldsymbol{c}}(\boldsymbol{n}) = \boldsymbol{k}$ if the detector selects the $\boldsymbol{k}$-th hypothesis $\boldsymbol{\mathcal{H}_k}$.

## 5.2 Performance Evaluation Index

**Precision & Recall** Precision is also called *positive predictive value* and recall also known as *sensitivity* or *True Positive Rate* eq.5.1. The *precision* is the number of true positives divided by the total number of elements labeled as the positive class; in another word, it is the number of positive predictions divided by the total number of positive class values predicted. Recall is defined as the number of true positives divided by the total number of elements that actually belong to the positive class.

$$\begin{aligned} \boldsymbol{Precision} &= \frac{\boldsymbol{tp}}{\boldsymbol{tp} + \boldsymbol{fp}} \\ \boldsymbol{Recall} &= \frac{\boldsymbol{tp}}{\boldsymbol{tp} + \boldsymbol{fn}} \end{aligned} \tag{5.1}$$

$\boldsymbol{F_1}$ **Score** is the harmonic mean of precision and recall defined in eq.5.2:

$$\boldsymbol{F_1} = \frac{\mathbf{2}}{\frac{\mathbf{1}}{\boldsymbol{Recall}} + \frac{\mathbf{1}}{\boldsymbol{Precision}}} = \mathbf{2} \times \frac{\boldsymbol{Precision} \times \boldsymbol{Recall}}{\boldsymbol{Precision} + \boldsymbol{Recall}} \tag{5.2}$$

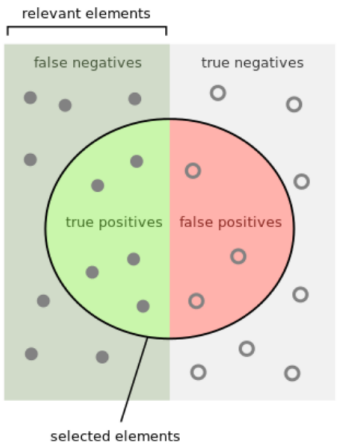



*Figure 5.1: Classification Terminology*

## 5.3 Classifiers

This section is devoted to discuss the classifiers that we implemented during the experiments. We use the *unlabeled data* and the clustering results (*labels*) to make up of necessary labeled data (*training data* and *testing data*) in supervised learning. We designed a couple of different classifiers in the experiments to compare the potential capability in SON field for future applications.

### 5.3.1 Minimum Distance Classifier

**Brief Introduction** The MD classifier *Minimum Distance Classifier* is only based on the relative distance of samples and each representative points of clusters, we do not consider the probability or the samples distribution function in the statistical point of view. The detail information about MD classifier are shown as following:

1. Random splitting datasets into training data and testing data.
2. Retrieving the labels of training data (clustering results) and

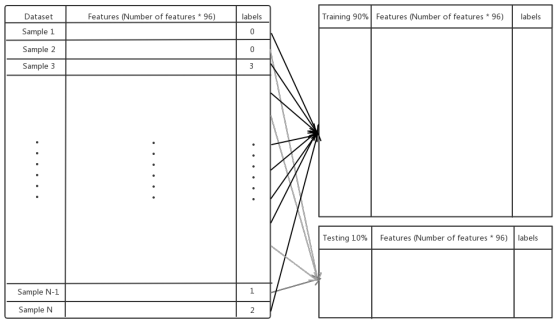



*Figure 5.2: Random dataset splitting*

calculating centroids for the clusters $\boldsymbol{C = 1, 2, \ldots, k}$ where $\boldsymbol{k}$ represents the number of clusters.

3. Calculating distance between samples and every centroid, the distance metric is decided to be *Euclidean Distance*, eq.5.3.

$$\boldsymbol{d(\mathrm{x}, \mathrm{Centroid_C}) = \{(x_1 - Centroid_{(C,1)})^2 + (x_2 - Centroid_{(C,2)})^2 + \cdots + (x_{(N_f \times 96)} - Centroid_{(C,N_f \times 96)})^2\}^{(1/2)}} \tag{5.3}$$

4. Assign the sample to that cluster if the distance between the sample and that centroid is minimum one among all pairs.

| | Class 0 | Class 1 | Class 2 | Class 3 | Class 4 | Class 5 |
|---|---|---|---|---|---|---|
| Time | 24 hours | 20:00 - 01:00 | 09:00 - 21:00 | 20:00 - 01:00 | 13:00 - 19:00 | 20:00 - 01:00 |
| Event scale | / | low | low | medium | low/medium/high | high |

*Table 5.1: Class Description*

**Implementations** We chose the labels from K-Means clustering result for the 6 cells *BL2,GT1,GP1,GP2,GP3,GP4*, Fig.5.3. Class 0 is the group of *no event day*, the traffic is flat and nearly 0 in the whole day. Class 1 is the group of days with *low volume events at night*. Class

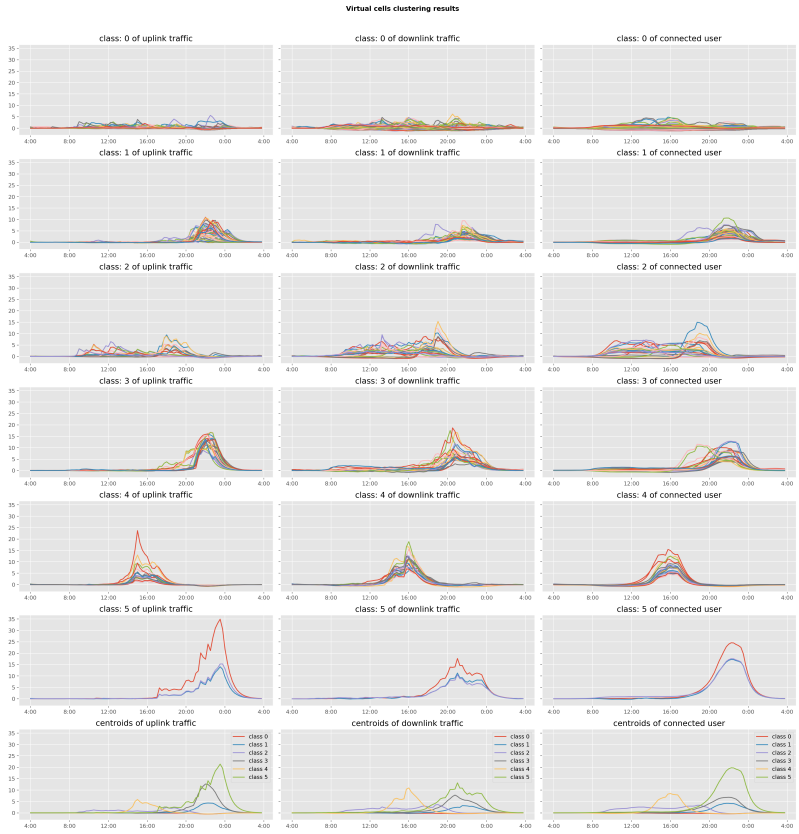



*Figure 5.3: Used classification labels*

2 is the group of days with *low volume events during daytime* (09:00 to 21:00). Class 3 is the group of days with *medium volume events at night*. Class 4 is the group of days with *events at afternoon* (13:00 to 18:00). Class 5 is the group of days with *high volume events at night* (13:00 to 18:00).

### 5.3.2 GNB Classifier

**Brief Introduction** *Naive Bayes Classifier* is a supervised learning algorithms based on *Bayes' theorem* eq.5.4 with the "naive" assump-

tion that each features are independent of others.

$$\boldsymbol{P(h|d)} = \frac{\boldsymbol{P(d|h)} \times \boldsymbol{P(h)}}{\boldsymbol{P(d)}} \tag{5.4}$$

And in the experiment, we implemented the *Gaussian Naive Bayes* (GNB) algorithm for the classification where the likelihood of the features is assumed to be Gaussian(normal distribution).

**Implementation** We chose same labels from K-Means clustering results for the 6 cells as MD classifier, and selected 3 features as clustering dataset. Class 0 is the group of *no event day*, the traffic is flat and nearly 0 in the day. Class 1 is the group of days with *low volume events at night*. Class 2 is the group of days with *low volume events during daytime* (09:00 to 21:00). Class 3 is the group of days with *medium volume events at night*. Class 4 is the group of days with *events in the afternoon* (13:00 to 18:00). Class 5 is the group of days with *high volume events at night* (13:00 to 18:00).

**Drawbacks of GNB Classifier** As explained in *Smoothness without Smoothing: Why Gaussian Naive Bayes Is Not Naive for Multi-Subject Searchlight Studies* [12], GNB classifier takes each data point, and assigns it to whichever class it is nearest, but unlike MD classifier calculating the nearness by using the Euclidean distance from the centroids (class-means), the GNB classifier takes into account not only the distance between the samples and centroids but also how this samples located with respect to the variance of the class. So the weaknesses of GNB, but also some surprising strengths, come from the "Naive" assumption of the Gaussian Naive Bayes. The Naive aspect treats all input dimensions as independent of each other. In our application, the

features are total dimension, $\boldsymbol{N_f \times 96}$, and since it is a time sequence, of course there is high covariance between adjacent dimensions (what happens now is somehow related what happened 15 minutes before), but the GNB classifier does not model it.

### 5.3.3 CNN

**Neural Networks** are modeled by collections of neurons that are connected in an acyclic graph. In other words, the outputs of some neurons are the inputs of other neurons. Acyclic property are emphasized that in the network, otherwise, it would imply an infinite loop in the forward pass of a network.

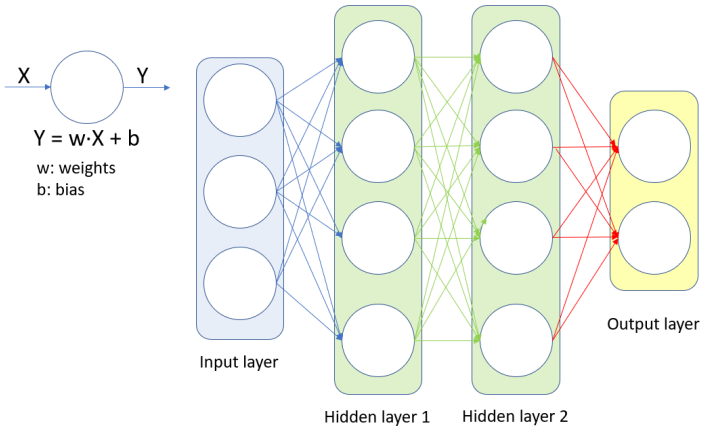



*Figure 5.4: Structure of a 3-layer neural network, with an input layer of 3 inputs, two hidden layers of 4 neurons and one output layer with 2 outputs. Two adjacent layer are fully connected but no connection exists within a layer.*

Neural Network models are often organized into several distinct layers of neurons. The most common layer type of NN model is *fully-connected* layer in which neurons between two adjacent layers are fully pairwise connected, but neurons within a single layer share no connections, Fig.5.4. The neurons have learn-able weights and biases. Each

neuron receives some inputs, performs a dot product and (optionally) allows the non-linearity functions.

**Convolutional Neural Networks** are very similar to ordinary Neural Networks as described in the previous paragraph. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. as shown in Fig.5.5 (Note that the word depth here does not refer to the depth of a full Neural Network. It refers to the third dimension of an activation volume. The depth of a full Neural Network can refer to the total number of layers in a network.)

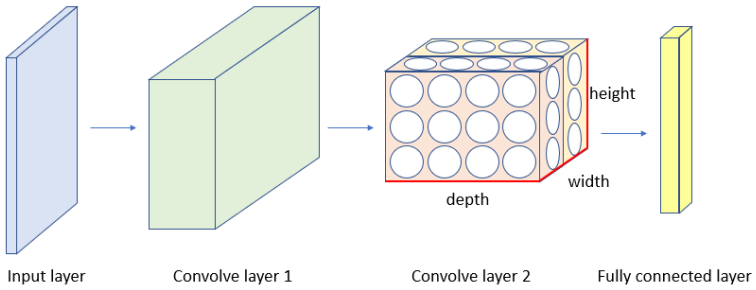



*Figure 5.5: A common structure of a convolutional neural network (CNN). A ConvNet arranges 3 dimensions dataframe as visualized.*

There are several different kinds of layers in the ConvNet architecture: **Convolutional Layer**, **Pooling Layer**, and **Fully-Connected Layer**. Those layers are stacked to form a full ConvNet architecture.

**Convolutional Layer** The Convolutional Layer computes the convolution of the previous layer. The number of dimensions is a property of the problem being solved. For example, 1D convolution for time series signals, 2D convolution for images, 3D convolution for movies. Thus in our application, 1D convolutional computing was adopted.

Suppose we have input as shape "$[\boldsymbol{N_s} \times \mathbf{96}]$", we use an "$[\boldsymbol{m} \times \mathbf{1}]$" filter and use no stride step, our convolutional layer output will be of size "$[\boldsymbol{N_s} \times \mathbf{96} - \boldsymbol{m}]$"

**Pooling Layer** There are two types of pooling layers, max-pooling and average-pooling. The pooling layer takes small rectangular blocks from the convolutional layer and subsamples it to produce a single output from that block. Our pooling layers are applied in layer-1 and layer-2. In layer-1 it takes the maximum of the block and in layer-2 it takes the average of the block.

**Fully-Connected Layer** Finally, after several convolutional and max pooling layers, there is fully-connected layer as it is in normal neural networks. A fully connected-layer takes all neurons in the previous layer and connects it to every single neuron it has. Fully connected layers are not spatially located anymore (you can visualize them as one-dimensional), so there can be no convolutional layers after a fully connected layer.

**Convolutional Neural Network Classifier** uses a trained ConvNet as classifier to "predict" the label of samples. In our experiment, the ConvNet is a fairly small network with 4 layers and there are 1862 trainable parameters. The detail of the ConvNet is described in the Table.5.2.

Apart from some layers we have discussed in the last paragraph, there are some more functional layers that we had in the network. *Zero padding* is used to add to the border of the time series, the border is extended by a couple of zeros (10 in the experiments, but can be tuned

| | *Layer* | *Output Shape* | *Num. Parameters* |
|---|---|---|---|
| Layer 0 | Input Layer | ($\boldsymbol{N_s}$,288,1) | 0 |
| | Zero Padding | ($\boldsymbol{N_s}$,298,1) | 0 |
| Layer 1 | Conv 1D | ($\boldsymbol{N_s}$,295,8) | 40 |
| | Relu(Activation) | ($\boldsymbol{N_s}$,295,8) | 0 |
| | Max Pooling 1D | ($\boldsymbol{N_s}$,147,8) | 0 |
| Layer 2 | Conv 1D | ($\boldsymbol{N_s}$,73,8) | 200 |
| | Relu(Activation) | ($\boldsymbol{N_s}$,73,8) | 200 |
| | Average Pooling 1D | ($\boldsymbol{N_s}$,36,8) | 0 |
| Layer 3 | Conv 1D | ($\boldsymbol{N_s}$,17,8) | 200 |
| | Relu(Activation) | ($\boldsymbol{N_s}$,17,8) | 200 |
| Layer 4 | Flatten | ($\boldsymbol{N_s}$,136) | 200 |
| | Fully Connected | ($\boldsymbol{N_s}$,6) | 822 |

*Table 5.2: Convolutional Neural Network Structure*

as a hyper-parameter). *Activation function* helps to add nonlinearity to the network as a pure convolution is a linear operation in Mathematics point of view. The activation function that we use **Relu** which is defined as 5.5. The *categorical labels* are adopted since this is a multi-label classification problem.

$$\boldsymbol{Relu(x) = max(x, 0)} \tag{5.5}$$

## 5.4 Results and Figures

In this section, we will present the results obtained from the experiments. In particular, using the 3 learning classifiers described in Section 5.3.1, 5.3.2, 5.3.3, we tested how each of the classifier performed by calculating the performance evaluation metrics mentioned in Section 5.2.But one thing have to be clearly discussed, those performance

evaluation metrics are applicable only for binary classification which you only have 2 class as candidates, but in our experiment since class 0 contains no-activity days, remaining class contains activity days with different activity level. Here we assumed that our classification is binary: the class 0 vs classes 1-6. The reason is that in the classification outputs, there is no error happens in classes 1-6 (for example, no sample belongs to class 6 are labeled as class 1). So this assumption works well and makes the evaluation metric stay simple.

| *Min Distance Classifier* | Precision | Recall | F1 |
|---|---|---|---|
| Training Set | 1 | 0.935 | 0.96 |
| Testing Set | 1 | 0.75 | 0.857 |

| *GNB Classifier* | Precision | Recall | F1 |
|---|---|---|---|
| Training Set | 0.650 | 1 | 0.788 |
| Testing Set | 0.571 | 1 | 0.727 |

| *CNN Classifier* | Precision | Recall | F1 |
|---|---|---|---|
| Training Set | 1 | 1 | 1 |
| Testing Set | 0.923 | 1 | 0.96 |

*Table 5.3: Classifiers Evaluation KPIs. The 3 KPIs are evaluated* Precision, Recall, F1 score. *For each KPIs, 1 is optimal results.*

The *Minimum Distance Classifier* is a very simple classifier as KNN classifier (when it is proposed in 1992, *An introduction to kernel and nearest-neighbor nonparametric regression* [1]). In KNN, the classification decision making is based on the distance from the unlabeled sample to *closest neighbor* in the feature space. But in our MD classifier, the considered distance is between the unlabeled sample and centroids. The first advantage of MD classier is that it does not need a real training phase, the model is established as long as the training data confirmed, second advantage is that the performance is good in

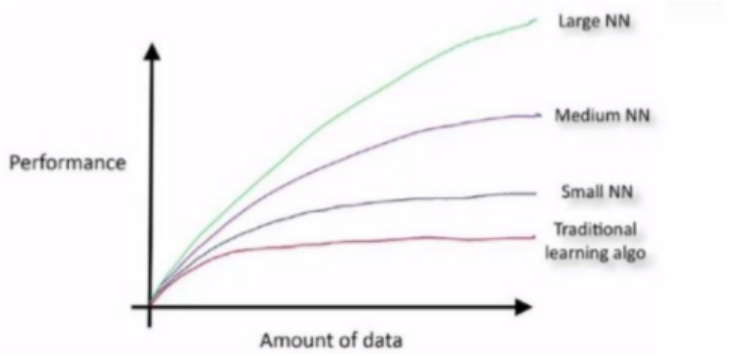



*Figure 5.6: The potential of deep learning*

*Precision* aspect. The disadvantage of MD classifier is that we can't have a better accuracy by increasing the size of the dataset as we expected in neural networks.

The *Gaussian Naive Bayes Classifier* can be viewed as MD classifier with Bayes theorem, the distribution of samples in feature space is assumed to be Gaussian(normal) distribution. As the drawbacks discussed in Section 5.3.2, the performance is worse than the MD classifier, 5.3. But since the GNB classifier is a powerful and famous classifier in machine learning field, this is worth to try.

The *Convolutional Neural Network Classifier*, during the whole experiments, is the most stable and has the best performance. Besides the current classification accuracy, the ConvNet will benefit from a larger size of training data.

## 5.5 Outliers Detection

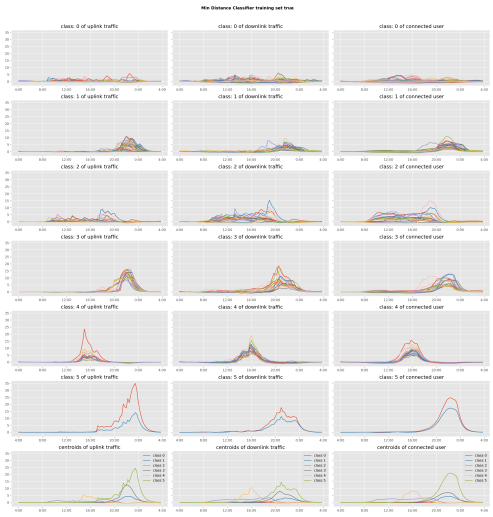
In order to find a way to identify a new pattern of events (which can be called outliers because it does not belong to any already analyzed clusters), we have to find a distance threshold as a boundary of each cluster.

The methodology of *Outliers Detection* is to use the labels from clustering to calculate $\boldsymbol{\sigma_c}$ standard deviation of distance between samples and centroids $\boldsymbol{\mu_i}$ of $\boldsymbol{cluster_c}$. For example, If the distance from new data to every cluster is larger then $\mathbf{3}\boldsymbol{\sigma}$, the new data is labeled as an outlier.

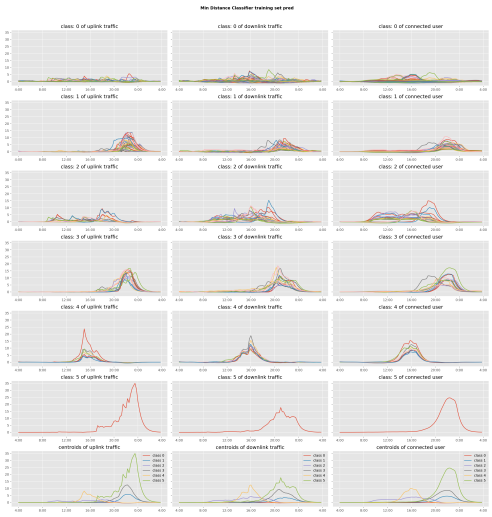
$$\begin{aligned} \boldsymbol{\mu_c} = \overline{\boldsymbol{x_i}} &= \frac{\mathbf{1}}{\boldsymbol{N_i}} \sum_{\mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{N_i}} \mathbf{x_i} \qquad \boldsymbol{x_i} \in \boldsymbol{cluster_c} \\ \boldsymbol{\sigma_c} &= \sqrt{\frac{\sum_i^N \boldsymbol{dist}(\boldsymbol{x_i}, \boldsymbol{\mu_c})^{\mathbf{2}}}{\boldsymbol{N} - \mathbf{1}}} \qquad \boldsymbol{N} = |\boldsymbol{cluster_c}| \end{aligned} \tag{5.6}$$

The method of the mean plus or minus three Standard Deviation is based on the characteristics of a normal distribution for which 99.87% of the data appear within this range.

While this method has several things to be discussed. Firstly, this method assumes that the distribution is normal (outliers included). Secondly, the mean and the standard deviation are strongly impacted by the outliers. Thirdly, this method is very unlikely to detect outliers in small samples.

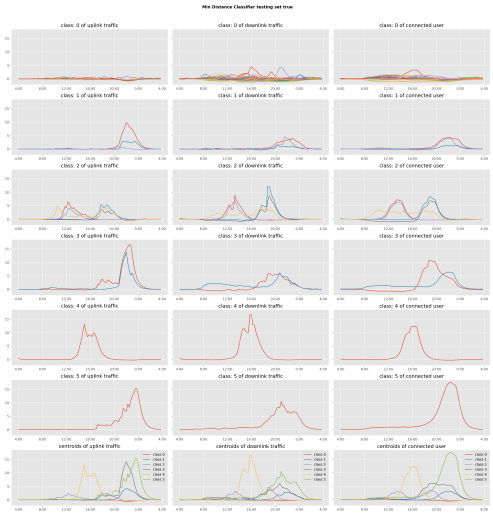


*(a) Minimum Distance Training Set True label*

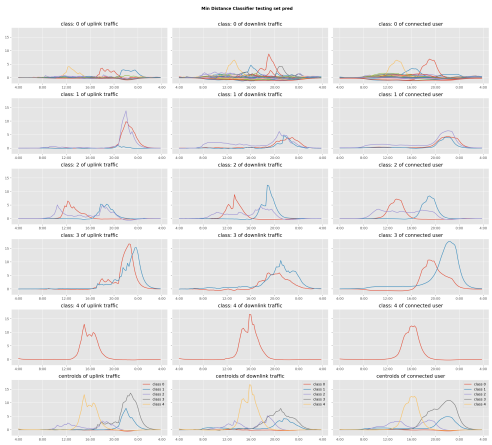


*(b) Minimum Distance Training Set Prediction Label*

*Figure 5.7: Minimum Distance Classifier Training Set*

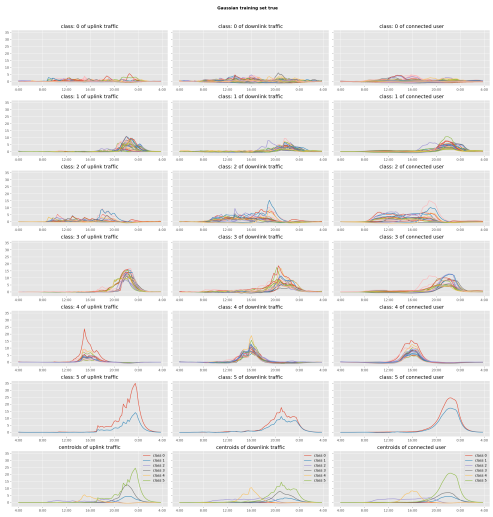


*(a) Minimum Distance Classifier Testing Set True label*

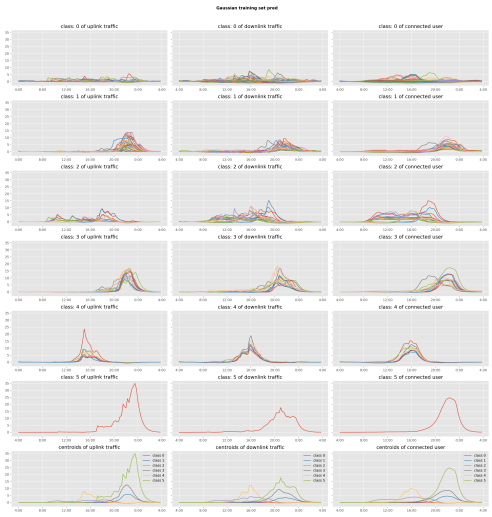


*(b) Minimum Distance Classifier Testing Set Prediction Label*

*Figure 5.8: Minimum Distance Classifier Testing Set*

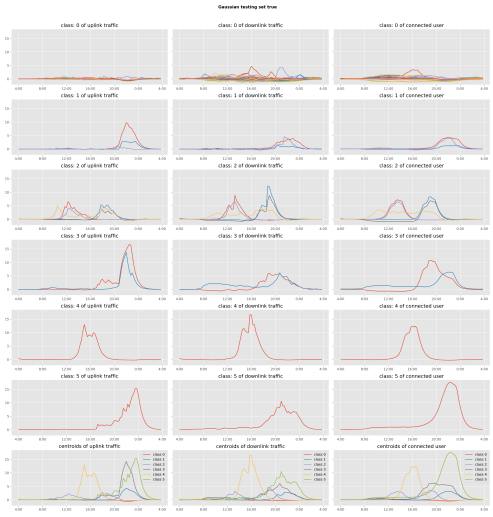


*(a) GNB Classifier Training Set True label*

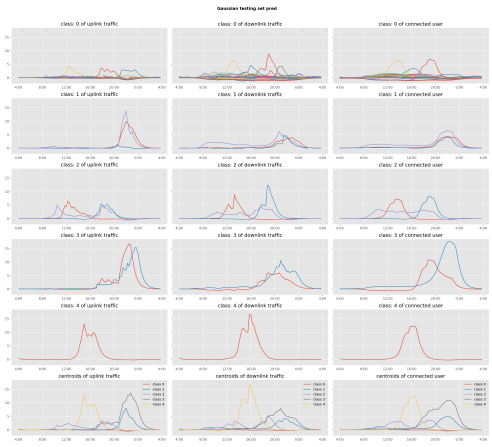


*(b) GNB Classifier Training Set Prediction Label*

*Figure 5.9: GNB Classifier Training Set*

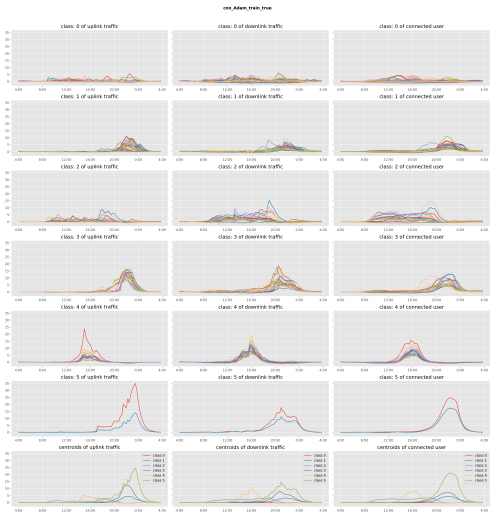


*(a) GNB Classifier Testing Set True label*

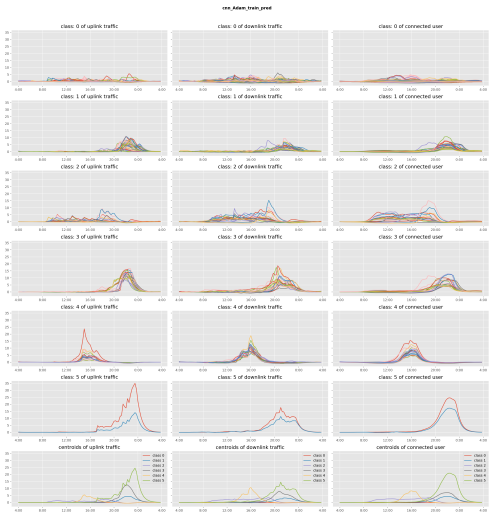


*(b) GNB Classifier Testing Set Prediction Label*

*Figure 5.10: GNB Classifier Testing Set*

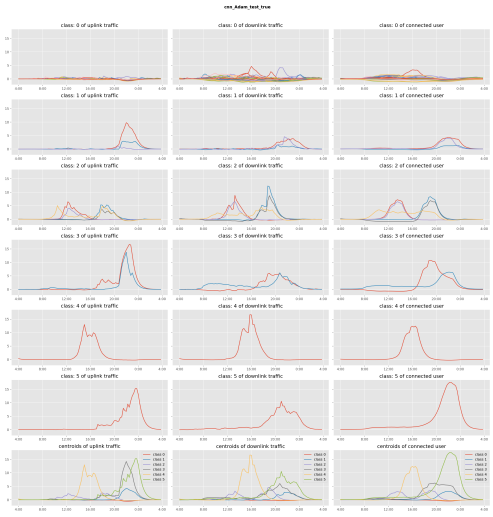


*(a) CNN Classifier Training Set True label*

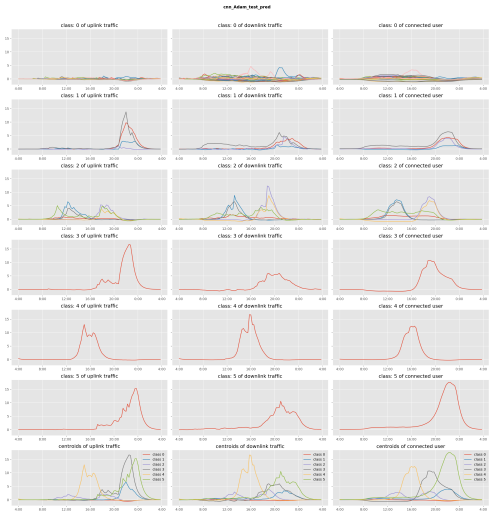


*(b) CNN Classifier Training Set Prediction Label*

*Figure 5.11: CNN Classifier Training Set*



*(a) CNN Classifier Testing Set True label*



*(b) CNN Classifier Testing Set Prediction Label*

*Figure 5.12: CNN Classifier Testing Set*

# Chapter 6

# Conclusions and Future Works

In this thesis, we described the challenges in the new generation of communication network and how machine learning could make the difference. In the second chapter, we covered the basic background knowledge of both telecommunication field (SONs) and machine learning science field because this is an inter-disciplinary project, also the used tools and packages are introduced.

In the third chapter, We described the data processing, the feature selection procedure, and data cleaning. The most noticeable point in this chapter is that we removed the means of samples (field counters), so we transfer absolute values to the deviation to the daily means. models for different telecommunication cells via clustering and made classifiers to predict unlabeled days.

The fourth and fifth chapter are devoted to describing the procedure of clustering and classification. In particular, in the fourth chapter, we described how to derive a model that contains the traffic patterns for each cell by unsupervised learning. And in fifth chap-

ter, we combine the new unlabeled data with the model we created in chapter 4.
In order to complete this experiment, there are several possible activities.

1. Distance metric learning, the distance metrics we used in this experiments are *Euclidean distance* and *Dynamic time wrapping distance*. But for time series sequences, we could also derive this distance by learning algorithms [14].
2. Different measurements selection. In the experiments, we selected the KPIs to be considered *traffic volume(Mb)* and *the number of connected users*, but there are also many other KPIs available as we listed in chapter 2. For sure we can take advantage of them somehow.
3. Dimension reduction for better GNB classifier adoption.

And the future development could be *Outliers detection* in real time. Also, it is interesting to take geographic information into consideration, in current experiment, we just derive models from the field measurements *KPIs* without knowing the cell type (residence, industry, school, etc.) By adding geographic information, we can group our cells in a more reasonable way so we could have just one pattern for each cell type.

# Bibliography


[1] N.S. Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician.*

[2] Emil Bergner. Unsupervised learning of traffic patterns in self-optimizing 4th generation mobile networks. 2012.

[3] F. Chernogorov, J. Turkka, T. Ristaniemi, and A. Averbuch. Detection of sleeping cells in lte networks using diffusion maps. *in proc. of the IEEE 73rd Vehicular Technology Conference (VTC Spring)*, 2011.

[4] Keogh E. Chiu, B. and S Lonardi. Probabilistic discovery of time series motifs. *Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining*, 2003.

[5] V. Bratu Claes Beckman. Antenna tilt load balancing in self-organizing networks. *Antennas and Propagation (EuCAP), 2013 7th European Conference on*, 2013.

[6] A. Gomez-Andrades I. Serrano E. J. Khatib, R. Barco. Diagnosis based on genetic fuzzy algorithms for lte self-healing. *IEEE Transactions on Vehicular Technology)*, 2015.

[7] Sander Jorg-Xu Xiaowei Ester Martin, Kriegel Hans-Peter. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 1996.

[8] Global Institute McKinsey. Big data: the next frontier for innovation, competition and productivity, 2011.

[9] Thomas M. Mitchell. *Machine Learning*. 1997.

[10] Josep Mangues-Bafalluy Nicola Baldo, Lorenza Giupponi. Big data empowered self organized networks. *in proc. of the 20th IEEE European Wireless*, 2014.

[11] Travis Oliphant. Python for scientific computing. *Computing in Science and Engineering 9(3):10-20*, 2007.

[12] Yune-Sang Lee Rajeev D. S. Raizada1. Smoothness without smoothing: Why gaussian naive bayes is not naive for multi-subject searchlight studies. *PLoSONE 8(7): e69566. doi:10.1 371/journal.pone.0069566*, 2013.

[13] R.E. Schapire. Using output codes to boost multiclass learning problems. *14th International Conference on Machine Learning*, 1996.

[14] Liu Yang. Distance metric learning: A comprehensive survey. *Michigan State University*, 2006.