Politecnico di Milano

Scuola di Ingegneria Industriale e dell'Informazione

Master of Science Degree in Telecommunication Engineering

Dipartimento di Elettronica, Informazione e Bioingegneria

Machine Learning-based traffic prediction and pattern extraction for dynamic optical routing in SDN mobile metro networks

Supervisor: **Prof. GUIDO ALBERTO MAIER**

Master's degree Thesis of:

SEBASTIAN TROIA

824039

Academic Year 2015-2016

Abstract

The high variability of the internet traffic and the static planning of the network resources, create overprovisioning outside the peak hours that leads to revenue decrease and network efficiency degradation. In this work we are focusing on machine learning methods that allow to build analytical models from particular inputs in order to make predictions or decisions. Using algorithms that iteratively learn from data and dynamic network operations, we are able to solve these problems.

In this work we used three datasets. The first one is a set of Call Detail Records (CDRs) in which are stored information of voice/sms/data of mobile users for each squared cell of the Milan area, measured during November and December 2013. The second dataset is a set of Base Station (BS) locations information, as the type of the BS (GSM, UMTS, LTE) and GPS position. The last dataset describes the land usage of the entire Lombardy region.

Machine learning methods have been applied to the first dataset to make predictions of CDRs and clustering, in which typical patterns are detected and analized in order to better manage the resources allocation of the metro network

in Milan. Then, thanks to the second dataset, the network topology and the bandwidth demand were estimated with the purpose of implementing a dynamic bandwidth allocation. In the end, we used a descrete event simulator to get optimal solutions of Routing and Wavelength Assignment (RWA) by solving Integer Linear Programming (ILP) problems.

The project builds a predictor and a pattern detector given the spatio-temporal fluctuations of the traffic in the mobile network. The results have been used as an input for the simulator to reproduce a metro network and optimaze dynamically the resource allocation. Thanks to the predictability of traffic, all optimization problems can be solved offline, and then the solutions found can be downloaded in the network at reconfiguration time points.

Keywords: machine learning, prediction, clustering, pattern recognition.

Contents

List of Figures

Chapter 1

Introduction

1.1 General introduction to the work

The work mainly focuses on the implementation of machine learning methods applied to the mobile and metro network, in order to ensure and optimize the resources allocation for Mobile Cloud Computing (MCC) services.

The volume of mobile data traffic is still smaller in comparison with its fixed counterpart, but this trend is rapidly changing. The popularity of smart-phones, the advent of HD-resolutions mobile-terminal screens, mobile cloud services and the Internet of things are major contributions to the expected mobile data traffic 1000-fold growth from 2015 to 2020 [1] [2].

In recent years, cloud computing enabled the outsourcing of computing resources such as IT infrastructure, software and service platform, with the wide application of ultra-fast 4G mobile networks and usage of highly featured mobile devices [3][4]. MCC is the combination of cloud and mobile computing, and wireless networks that could provide rich computational resources to mobile users. A new platform combines the mobile devices and cloud computing to create a new infrastructure, whereby cloud performs

the heavy lifting of computing-intensive tasks and storing massive amounts of data. In this new architecture, data processing and data storage are provided outside of mobile devices [4]. Figure 1.1 shows a model of MCC.

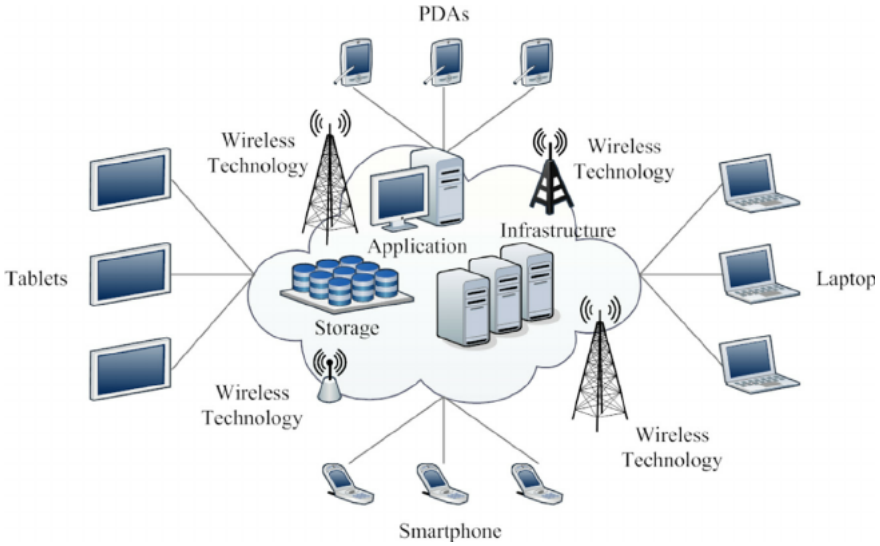


Figure 1.1: *MCC model*

As result, MCC has the following advantages:

- Break through limitation of the hardware
- Flexibility in data storage
- Smart balance of loads
- Improvement of management cost
- Cost reduction by on-demand services.

The significant growth of MCC services will surely become reality very soon. Applications targeted at mobile devices have started becoming abundant with applications in various categories such as entertainment, health, games, business, social networking, travel and news. For these reasons, it

is important to deal with network resource allocation problems related to MCC services, by exploiting intelligent methods.

Machine learning is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. In 1959, Arthur Samuel defined machine learning as a *Field of study that gives computers the ability to learn without being explicitly programmed*. Machine learning explores algorithms that can learn from and make predictions on data [5]. Such algorithms operate by building a model in order to make data-driven predictions or decisions, rather than following strictly static program instructions. Machine learning tasks are typically classified into three different categories [6]:

- Supervised learning: computer is presented with inputs and their desired outputs, and the goal is to learn a general rule that maps inputs to outputs.

- Unsupervised learning: no outputs are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself, as discovering hidden patterns in data.

- Reinforcement learning: a computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle), without being programmed to do so. Another example is learning to play a game facing an opponent.

The goal of this thesis is to apply aforementioned methods in order to build an *intelligent* network capable of selecting automatically the best resources allocation, and to study the statistical properties of the traffic data itself. By implementing learning algorithms for example in an SDN controller of a network, we can analize the data recorded dayly at each base

station, and forecast the traffic load in the short and long term. Furthermore, we are able to look for hidden behaviours that are not easy to detect.

1.2 Thesis structure and purpose

The core of this thesis is the development and improvement of prediction algorithms and clustering methods with the aim of extracting typical traffic load patterns. The work pursued four steps:

1. Processing of the datasets to make them easy to use;
2. Implementation of three prediction algorithms;
3. Adoption of two clustering methods to define the social function of the base stations;
4. Implementation of one method for the pattern extraction;
5. Planning experiment of the metro-core network of Milan, optimizing the reources allocation;

The chapters are organized as following:

Chapter 2 describes the state of the art of prediction and clustering methods. In literature there are interesting studies regarding time series forecasting and clustering of large data-set, principally focused on speech recognition, video processing, image detection and pattern recognition. What we have done in this work is to apply and tailor these machine learning methods in order to give the network the ability of configure itself.

Chapter 3 shows the input data-sets adopted for the project. The first one comes from a challenge that Telecom Italia launched in 2014, called BIG DATA CHALLENGE, that is a contest designed to stimulate the creation

and development of innovative technological ideas in the Big Data field. The data-set is the result of a computation over the Call Detail Records (CDRs) generated by the Telecom Italia cellular network over the city of Milan. CDRs log the user activity for billing purposes and network management. By aggregating the aforementioned records, the data-set was created providing SMSs, calls and Internet traffic activity during the months of November and December 2013. It measures the level of interaction of the users through the mobile phone network. For example, the greater is the number of SMS sent, the higher is the activity of the users. The second data-set collects information about the base stations of Milan. OpenCellID is the world's largest collaborative community project that collects GPS positions of cell towers for a multitude of commercial and private purposes. The last data-set is called DUSAF (Destinazione d'Uso dei Suoli Agricoli e forestali). In 2001, Lombardy region embarked on the creation of a tool for monitoring the use of the soil through the creation of a homogeneous database of the whole region. This database, which photographs the "function of the territory", is commonly designated by its acronym DUSAF, and help us to understand the *social function* of a single or group of base stations. All of these data-sets are open source and useful for big data analysis.

In Chapter 4 are listed all the methods used in this thesis to make predictions of internet traffic loads, hour by hour (short term) and 24 hours (long term). First of all, we tried common methods, as Autoregressive model and Artificial Neural Network, and later we built a machine learning method by creating a prediction formula.

Then, chapter 5 shows the clustering models designed to define the social function of the base stations. In literature there are plenty of methods, Kmeans is the most famous. In this work we tried also another method

called Spectral Clustering (SP), describing three different algorithms.

Chapter 6 explains how to detect the traffic load patterns automatically, by exploiting a famous tool in pattern recognition called Non Negative Matrix Factorization (NNMF).

Chapter 7 is dedicated to the simulation of the Milan network and the implementation of the prediction mentioned in chapter 4. In particular, a discrete event simulator has been used to get optimal routing and wavelength assignement in the optical metro network, showing the differences between the presence or not of the prediction. The goal is to use the predicted traffic data and the pattern extraction to optimize dinamically the network, reducing the blocking probability and the energy consumption. We assume that all the dynamic bandwidth allocation mechanisms are supported by SDN technology, therefore the controller collects global network informations.

Currently the project is in cooperation with the Pattern Recognition Intelligent System (PRIS) Lab of the Beijing University of Posts and Telecommunications. The development of this thesis is part of the EU FP7 IRSES Marie Curie MobileCloud project that has goal to foster research collaborations between Chinese and European university in Mobile Cloud Computing. The proposed joint exchange project aims to provide a stimulating and structured platform for the exchange of researchers and for the joint development of innovative ideas in the emerging areas within mobile cloud computing and beyond. To achieve this goal, a broad combination of eight universities and research institutions in Europe and China collaborate together and create a multidisciplinary and international environment for an innovative research experience and knowledge exchange in mobile cloud computing. The specific objectives of MobileCloud are to investigate inno-

vative methodologies and approaches to optimize mobile cloud computing resources and satisfy service requirements including energy efficiency and high resource utilization in the emerging cloud computing era.

Chapter 2

State of the art

The goal of this thesis is to investigate methods that allow us to make predictions on data traffic, and clustering the network. The third step is to evaluate a pattern extraction algorithm and then to simulate an optical metro network in order to plan the resource allocation. With this aim, we investigated the current status of the studies about machine learning methods that allow us to make learning algorithms on prediction and clustering. We subsequently investigated the studies in the field of pattern recognition in order to derive an effective method for the traffic pattern's extraction. In the end, we focused on the current state of the mobile cloud architecture and energy consumption of the underlying optical network components in order to use the results of the prediction, clustering and pattern extraction, and plan in advance the resource allocation and the power consumption.

Recent years have witnessed significant progress in machine learning field. In this state of the art we aim to achive two goals: make a comparison of different methods and sketch a vision for future work that can help motivate and guide readers that are interested in building a smart network.

Prediction is a core process for network optimization decisions and a

foundamental branch of machine learning. One of the most famous algorithms is described in [7] called Autoregressive Integrated Moving Average (ARIMA) model. ARIMA models are generally applied to stationary time series with no trends or seasonality informations making a regression on previous samples. Performances of prediction algorithms are based on the type of data input, so we can not know if a method is better than the other until we do not try different kind of input data-sets. In general a good approach is to mix different methods as in [8], where authors built a hybrid model. They concluded that using ARIMA and Artificial Neural Network (ANNs), for non-linear time series, is more efficient than using just one. The motivation comes from the following perspectives. First, it is difficult in practice to determine whether a time series is generated from a linear or nonlinear underlying process or whether one particular method is more effective than the other. Second, real-world time series are rarely pure linear or nonlinear; they often contain both linear and nonlinear patterns. Using three different datasets, the author concluded that a mixed approch to the prediction can achive very good results.

Ref. [9] proposed another method called Fractional ARIMA. They are time series models that generalize ARIMA by allowing non-integer values of the differencing parameter. Those experiments showed that FARIMA is a good traffic model and capable of capturing the property of actual traffic.

Artificial Neural Network (ANN) is inspired by biological neural networks, and try to emulate the conncetions that are typical of a brain. Ref. [10], presents an ANN approach in order to forecast the electric load in a period of 24 hours. The model is used to learn relationships among past, current and future temperatures and loads. The results shows that the ANN is suitable to interpolate the load and temperature pattern data of training

set to provide the future load. Compared to other regression methods, the ANN allows more flexible relationships and accuracy.

Ref. [11], describes a number of traffic predictors (such as ARIMA, FARIMA, ANN and wavelet-based predictors) and analyzes their computational complexity. Results showed significant advantages for the ANN technique. FARIMA could provide the accuracy prediction, but at the cost of computational complexity. Compared to other predictors, ARIMA and wavelet forecasting performed considerably worse.

The clustering problem has been studied for many years. The purpose is to group a set of objects such that in the same group are more "similar" to each other than to those in other groups. The main issue is to define the similarity that must be exploited looking at the database being studied. In [12] is treated a study on the K-means method, that is one of the most famous clustering algorithm. Under statistical point of view, it concludes that the results (sometimes strongly) depend on the order of the objects in the input file. This method represents an important drawback, and can be reduced running multiple times the algorithm with different inputs. Indeed, Ref. [13] demonstrates how the popular k-means clustering algorithm can be profitably modified to make use of particular informations of the dataset. In experiments with artificial constraints on six data-sets, the authors observed improvements in clustering accuracy. They also apply this method to the real-world problem of automatically detecting road lanes from GPS data and observe increases in performance. A study published on the journal of the Royal Statistical Society [14], illustrates how a K-means algorithm uses swops as well as transfers to try to overcome the problem of local optima, and give another method to set up the initial centers of the clusters. Results shows an improvement in time computation with a number of iterations

usually less than ten.

A promising method is the Spectral Clustering (SP). Ref. [15] is a tutorial of the method and shows different algorithms with similar performances. SP is simple to implement and can be solved efficiently by standard linear algebra software. In addition, outperforms traditional clustering algorithms such as k-means. The main trick is to change the representation of the data points and thanks to the properties of the graph Laplacians, this change of representation tend to be useful.

In [16] a number of open issues in spectral clustering are analyzed, like selecting the appropriate scale of analysis, handling multi-scale data, clustering with irregular background clutter, and finding automatically the number of groups. This leads to a new algorithm in which the final randomly initialized k-means stage is eliminated. The results of this new algorithm are promising, improving automatic image segmentation.

One of the main issue of the SP model is the expensive time computation in the derivation of eigenvectors. Ref. [17] introduces a new cost function based on an error measure between a given partition and a solution of the spectral relaxation of a minimum normalized cut problem. This leads to a more robust method instead of the irrelevant features. Ref. [18] proposed an efficient algorithm for reducing a large mixture of Gaussians into a smaller mixture while still preserving the component structure of the original model. This is achieved by clustering the components. The method minimizes a new easily computed distance measure between two Gaussian mixtures that can be motivated from a suitable stochastic model. The iterations of the algorithm use only the model parameters, avoiding the need for explicit resampling of datapoints. Results demonstrates the method by performing hierarchical clustering of scenery images and handwritten digits. In order

to evaluate the quality of the clustering in terms of correlation with the category information, the authors computed the mutual information (MI) between the clustering result (into k clusters) and the category affiliation of the images in a test set. A high value of mutual information indicates a strong resemblance between the content of the learned clusters and the hand-picked image categories.

Performances calculation of a clustering method is a crucial problem, mostly in unsupervised learning problem. Ref. [19] lists a lot of indeces that describe the quality of clustering itself.

In the field of pattern recognition, the Non Negative Matrix Factorization is one of the most used method, thanks to the ability to give basic flows or patterns. In [20] [21] and [22] this method is widely studied. They concluded that this kind of pattern extraction is usuful for a large group of data, from image to speech recognition. Therefore, thanks to the matrix decomposition, it is possible to detect hidden flows accentuating unknown behaviors. Ref. [23] provides systematic analysis and extensions of NNMF to the symmetric and the weighted matrices of spectral clustering. This paper shows that NNMF can be equivalent to Kernel K-means clustering and Laplacian-based spectral clustering.

The mobile carrier network (MCN) traffic can be adopted to figure out the movements of human beings in a specific area. In [24], a human mobility analysis from a MCN illustrates that there is a high predictability (from 80% to 93%) on the user mobility due to the inherent regularity of human behavior. A study from Google's WiFi network [25], showed that human mobility is powerfully related with time-dependent traffic fluctuations at radio access network, neglecting its spatial variations. The energy efficiency limits of networks that can adapt to traffic load variations in time was ana-

lyzed in [26]. Most of the works on energy efficiency takes into account only the temporal fluctuation of the traffic demand. Given that base stations are the most power-hungry devices in MCN architectures, the energy efficiency efforts in MCN focused mainly on the radio access segment [27] [28]. The combined effect of temporal and spatial traffic was first proposed for energy efficient operation of access networks. Starting from the radio access networks with a small cluster of MCN cell sites [29]. [30] proposed a energy efficient management for passive optical network. Recently, tidal traffic effect was considered in [31] [2] to propose energy efficiency in metropolitan networks. A limitation of this works, is that they assumed mainly two basic tidal traffic patterns: residential and business. However, the social composition of metropolitan areas is more complex than just residential and business, and multiple social functions or services can coexist in the same location.

Chapter 3

Data-sets

In this chapter we introduce three different data-sets adopted to develop this thesis. The first one is used to build prediction algorithms, clustering and pattern extraction. Instead, the other two have been processed together with the first data-set in order to obtain a suitable input for the network simulator in chapter 7.

3.1 Telecom Italia data-set

The data-set refers to the traffic of voice/sms/data for each squared cell of the Milan area, measured during November and December 2013. It provides information about telecommunication activities in the city. Squared cells are numbered through ID, instead cell geometry is expressed as GEOJSON format and projected in WGS84 (EPSG:4326). Below, we can see the coordinates of the four angles of a square:

'type': 'Polygon', 'coordinates': [[[9.0114910478323, 45.35880131440966], [9.014491488013135, 45.35880097314403], [9.0144909480813, 45.35668565341486], [9.011490619692509, 45.356685994655464], [9.0114910478323, 45.35880131440966]]]

The 5th coordinate is the same to the first one because of the old version of the GEOJSON format. Figures **??** and **??** shows how cells are arranged on the territory.

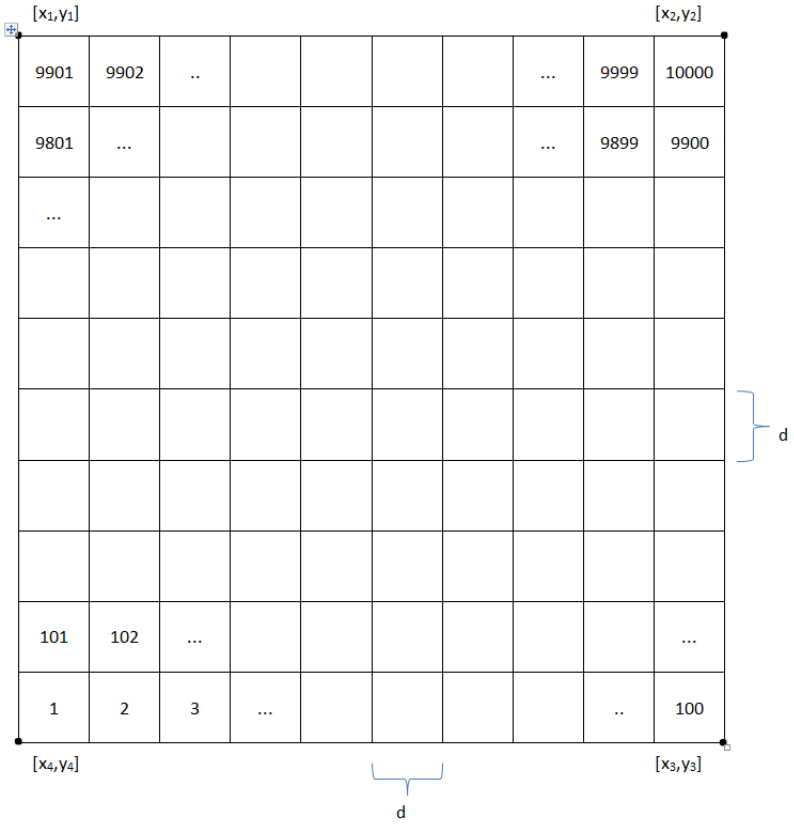


Figure 3.1: *Milan grid*

The data-set contains the following information:

- **Square ID**: the ID of the square that is part of the Milan GRID;
- **Time interval**: the beginning of the time interval expressed as the number of millisecond elapsed from the Unix Epoch on January 1st, 1970 at UTC. The end of the time interval can be obtained by adding 600000 milliseconds (10 minutes) to this value;
- **Country code**: the phone country code of a nation.

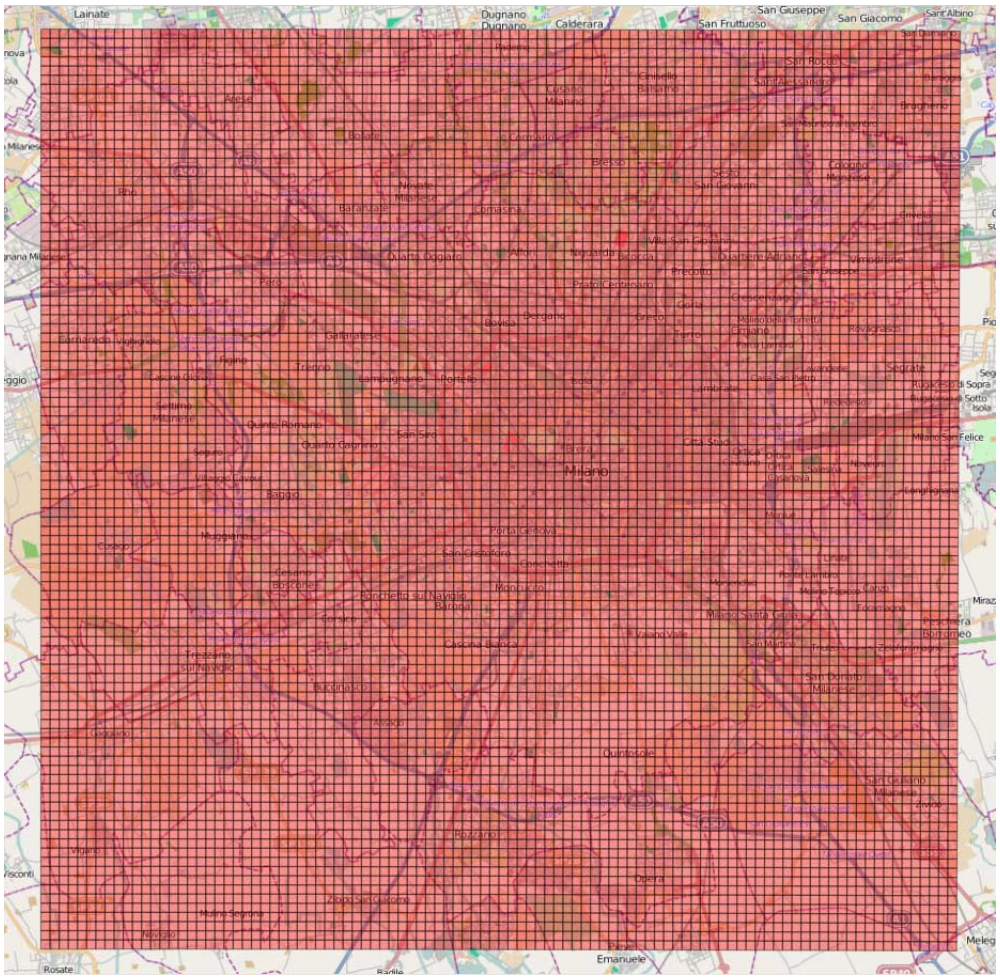


Figure 3.2: *Milan map*

- **SMS-in activity**: the activity in terms of SMS received inside the Square ID, during the time interval and sent from the nation identified by the country code;
- **SMS-out activity**: the activity in terms of SMS sent inside the Square ID, during the time interval and received by the nation identified by the country code;
- **Call-in activity**: the activity in terms of received calls inside the Square ID, during the time interval and issued from the nation iden-

tified by the country code;

- **Call-out activity**: the activity in terms of issued calls inside the Square ID, during the time interval and received by the nation identified by the country code;
- **Internet traffic activity**: the activity in terms of performed internet traffic inside the Square ID, during the time interval and by the nation of the users, performing the connection identified by the country code.

The data-set is the result of a computation over the Call Detail Records (CDRs) generated by the Telecom Italia cellular network over the city of Milan. CDRs log the user activity for billing purposes and network management every ten minutes, creating 144 records for each day. There are many kind of CDRs; for the generation of this data-set we considered those related to the following activities:

- **Received SMS**: a CDR is generated each time a user receives an SMS;
- **Sent SMS**: a CDR is generated each time a user sends an SMS;
- **Incoming Calls**: a CDR is generated each time a user receives a call;
- **Outgoing Calls**: CDR is generated each time a user issues a call;
- **Internet**: a CDR is generated each time:
  - a user **starts** an internet connection;
  - a user **ends** an internet connection;
  - during the same connection one of the following limits is reached: **15 minutes** from the last generated CDR or **5 MB** from the last generated CDR;

3.2 Social function data-set

To discover the social function of each squared cell of the first data-set, I used the DUSAF (Destinazione d'Uso dei Suoli Agricoli e forestali) map. It describes the land usage of the entire Lombardy to monitor the changes that take place in all the region. We got the last version of 2012 [32]. The database is in *shape* format, so I used the software QGIS to manipulate it, extracting the city of Milan and periphery according to the network of the first data-set. In figure 3.3 is shown the map of Lombardy.

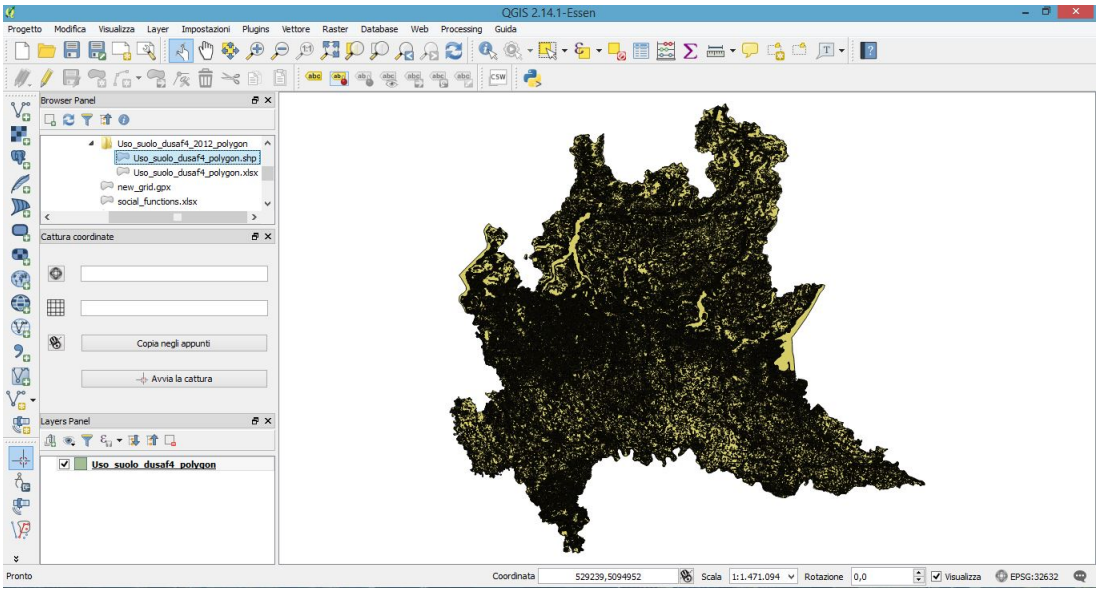


Figure 3.3: *Map of Lombardia*

Assuming that the land usage did not change in one year, I merged the Telecom Italia data-set with DUSAF in order to determine the social function of each squared cell. I assigned the social ID label minimizing the distance between the GPS coordinates of the two data-sets and associating the social ID respectively. In figure 3.4 is shown the output of this process.

Each social ID represent a different function. For example: 124 (Air-

Cell ID	Latitude	Longitude	social ID
1	45.35562833	9.012991268	2111
2	45.35562799	9.015991708	1421
3	45.35562757	9.018992148	1221
4	45.35562707	9.021992588	1221
5	45.3556265	9.024993028	134
6	45.35562584	9.027993468	1221
7	45.35562511	9.030993908	1221
8	45.35562429	9.033994347	3113
9	45.3556234	9.036994787	3113
10	45.35562243	9.039995226	2111
11	45.35562138	9.042995665	2111
12	45.35562025	9.045996104	2241
13	45.35561904	9.048996543	1421
14	45.35561775	9.051996981	1421
15	45.35561639	9.05499742	2111
16	45.35561494	9.057997858	1221
17	45.35561342	9.060998295	1221
18	45.35561182	9.063998733	1221
19	45.35561013	9.06699917	213
20	45.35560838	9.069999607	2311
21	45.35560654	9.073000044	2112
22	45.35560462	9.076000481	2311
23	45.35560262	9.079000917	2311
24	45.35560055	9.082001353	2311
25	45.35559839	9.085001788	2311

Figure 3.4: *Social functions*

LEGENDA

social IID	Social function
124	Aeroporti ed eliporti
131	cave
132	discariche
133	Cantieri
134	aree degradate non utilizzate e non vegetate
213	risaie
221	vigneti
222	frutteti e frutti minori
314	imboschimenti recenti
411	vegetazione delle aree umide interne e delle torbiere
511	Alvei fluviali e corsi d'acqua artificiali
1111	tessuto residenziale denso
1112	tessuto residenziale continuo mediamente denso
1121	Tessuto residenziale discontinuo
1122	Tessuto residenziale rado e nucleiforme
1123	Tessuto residenziale sparso
1221	Reti stradali e spazi accessori
1222	Reti ferroviarie e spazi accessori
1411	Parchi e giardini
1412	Aree verdi incolte
1421	Impianti sportivi
1422	Campeggi e strutture turistiche e ricettive
1423	Parchi divertimento
2111	seminativi semplici
2112	seminativi arborati

Figure 3.5: *Description of the social functions*

port and Eliport), 1221 (roads), 1222 (rail networks), ecc.. In total are 53 different social functions.

3.3 Base Stations data-set

For the base stations location I used an open source data-set called OpenCellID. It is a collaborative project to create a free worldwide database of Cell IDs and their corresponding location area identity. The OpenCellID project was primarily created to serve as a data source for GSM localization in areas bereft of satellite navigation coverage. It contains the following information for each BS:

- Type: GSM, UMTS, LTE;
- MCC: Mobile Country Code;
- Longitude: longitude coordinate;
- Latitude: latitude coordinate.

3.4 Data processing

Telecom Italia data-set is scaled by an unknown factor but, looking at the first internet values, it seems to be proportional to $1'000'000$ because the CDRs are integer numbers.

$$0.00002331497626 07\ CDR$$

$$0.00003999999989 90\ CDR$$

$$0.00004999999873 69\ CDR$$

$$0.00000600000021223\ CDR$$

$$0.00000700000009601\ CDR$$

Since we need to have the value of the internet field and the square ID every hour, firstly, I merged the values by the country code because, for the same moment, there are different measures depending on different users. After that, I sampled the data-set taking the highest value among the six ones for every hour. In figure 3.6 we can see the generated file after the processing, where each number is expressed in CDR as mentioned before, rows and columns represent the cells (from 1 to 10000) and the hours (from 00:00 am to 11:00 pm). In the end, we obtained 62 files one for each day from November 1 2013 to January 1 2014.

	A	B	C	D	E	F	G	H	I	J	K
1	9.26019	7.035252	6.11083	5.336269	5.506207	5.120144	5.600791	9.985428	8.902542	7.71458	7.15369
2	9.282384	7.037051	6.100188	5.345732	5.518715	5.141727	5.611087	10.04363	8.947151	7.771888	7.20301
3	9.306009	7.038966	6.08886	5.355806	5.53203	5.164702	5.622047	10.10559	8.994635	7.832889	7.25552
4	9.195904	7.030041	6.141654	5.308859	5.469976	5.057626	5.570967	9.816834	8.773332	7.548589	7.01083
5	8.312392	6.380656	5.583306	4.887456	4.974733	4.628928	5.103295	9.011268	8.135531	6.997976	6.47748
6	5.286326	5.138755	7.632317	9.723384	8.835995	8.471373	10.93813	9.554171	8.449731	9.121421	9.73961
7	5.286326	5.138755	7.632317	9.723384	8.835995	8.471373	10.93813	9.554171	8.449731	9.121421	9.73961
8	5.286326	5.138755	7.632317	9.723384	8.835995	8.471373	10.93813	9.554171	8.449731	9.121421	9.73961
9	5.286326	5.138755	7.632317	9.723384	8.835995	8.471373	10.93813	9.554171	8.449731	9.121421	9.73961
10	5.231009	3.994555	3.429867	3.408367	3.248063	3.224143	3.460088	6.518989	6.149962	5.440895	4.91754
11	2.383173	1.998932	1.611771	1.925905	1.690957	1.801188	1.949117	3.797866	4.229702	3.463698	3.03178
12	1.718418	1.664774	1.22225	1.262485	1.256943	1.181327	1.545345	2.846065	3.350316	2.817102	2.08679
13	0.969663	0.995678	0.774115	0.766872	0.761202	0.783571	0.89831	1.573837	2.113651	1.437088	1.25712
14	1.597782	1.662433	1.204149	1.31551	1.335088	1.322679	1.508528	2.648909	3.5857	2.423236	2.08833
15	1.650165	1.677827	1.202395	1.367097	1.347153	1.337498	1.560097	2.784374	3.834294	2.509023	2.17898
16	3.084855	1.977068	1.670446	1.841436	1.772371	1.803517	2.215324	3.06824	3.601984	3.00628	2.54651
17	4.808947	2.331024	2.252096	2.394558	2.254947	2.353135	2.954762	3.356614	3.356614	3.546745	2.9190

Figure 3.6: *Matrix of CDRs*

Since the data-set contains information on the location and type of base station of the entire Italian territory, I filtered by company (Telecom Italia) and mobile country code, in order to extract just the Milan area. In this way, we obtained a list of 2554 base stations mainly located at the center of the city. Then, I clustered the Telecom Italia data-set using the GPS information of the BS list, by grouping the squared cells based on the Euclidean

distance between the coordinates of the BSs in the list and the coordinates of the centers of the squared cells (figure 3.7 and 3.8), and summing the internet value respectively. In the end, we used the social functions data-set to define a vector of social ID for each BS because there are different land usages in an area covered by a single base staion.

As a result of this process, we obtained the following outputs:

- 62 matrices [2554 x 24] (one for each day), where the rows are the BSs and the columns the hours of two months, November and December;
- A matrix [2554 x 3] containing the GPS locations, Longitudes and Latitudes, and the type of BSs;
- A matrix containing the social and cell ID vectors, associated to each BS.

We computed the mean value of the Pearson's correlation among the traffic loads inside each group with the purpose to see if the transformation from 10000 to 2554 (figure 3.7 and 3.8) cells was correct. The histogram in figure 3.9 shows that most of the squared cells, inside the clusters, had very similar traffic load patterns.

3.5 Bandwidth estimation

In this work we used a network simulator which implement network resource reconfigurations, based on bandwidth demands between all source and destination node pairs. Then, bandwidth demands should be the basic input of the system. Since we had integer numbers (CDRs), we needed to estimate the bandwidth, without modifying the properties of the data-set. Assuming

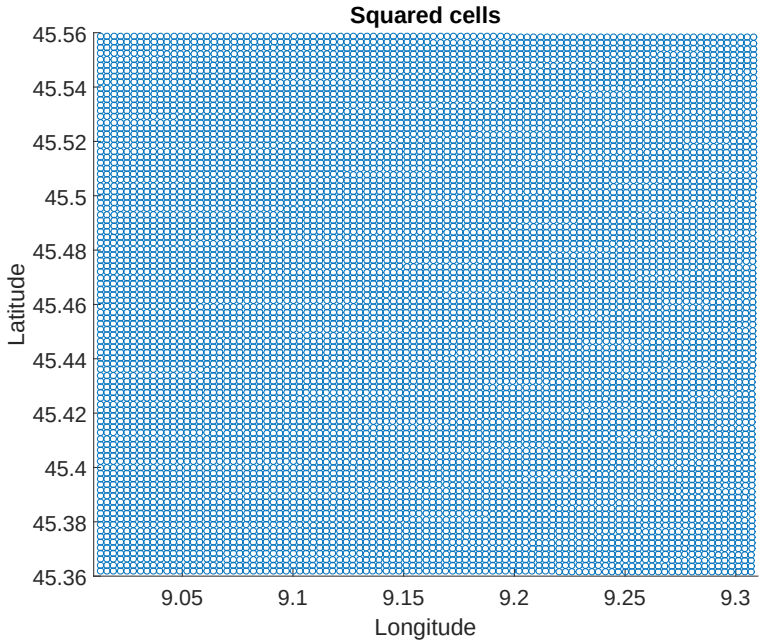


Figure 3.7: *Squared cells of the TELECOM ITALIA dataset*

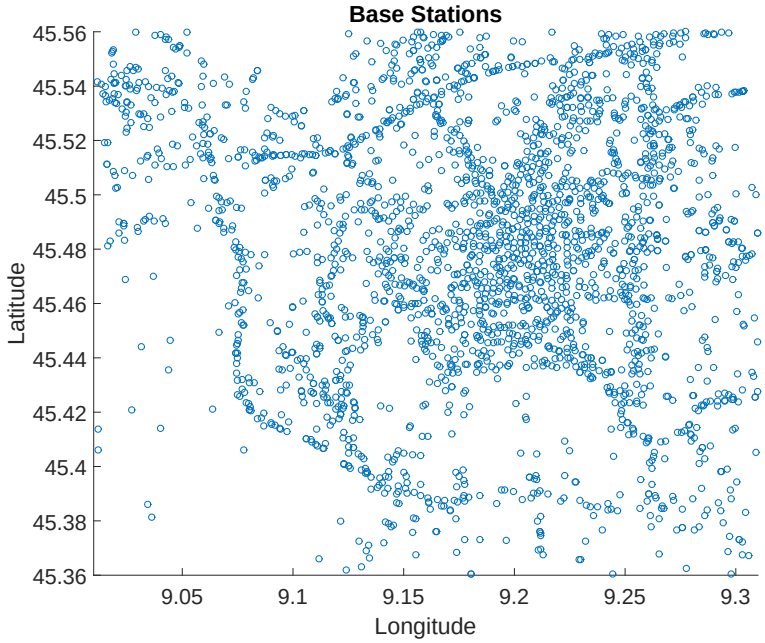


Figure 3.8: *Base Stations of the OpenCellID dataset*

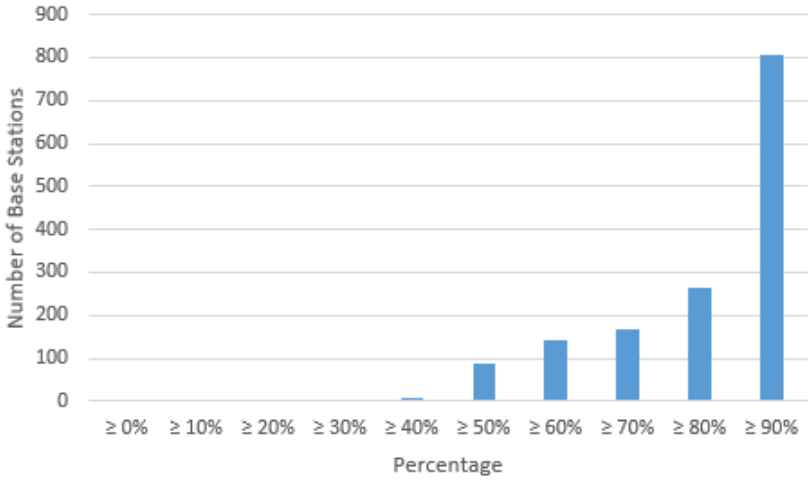


Figure 3.9: *Histogram of percentage similarity of the BSs inside the clusters*

a percentage of saturation in the network of 10 %, we can calculate the CDR threshold in which we have the saturation.

$$saturation = 0.01$$

$$Y = f(CDR_{th})$$

Y counts the number of times in which the CDR threshold is exceeded. The CDR_{th} is chosen such that the total period of saturation over the two months is less or equal to 10 %.

$$CDR_{th} \mid \quad \frac{Y \cdot 10min}{(62 \cdot 24 \cdot 60)min} \leq saturation$$

62 are the days, 24 the hours and 60 the minutes. In this way, trying different CDR threshold, the formula finds the one that makes the inequality correct. Now, assuming the maximum bandwidth for each kind of BS [34] [35] [36], the mean load for each 10 minutes [37] and the scaling factor (see 3.4), we derived a parameter, called *saturation factor*(A), for each type of

base station:

$$B_{LTE} = 300\ Mbps$$

$$B_{UMTS} = 14.4\ Mbps$$

$$B_{GSM} = 1.6\ Mbps$$

$$mean\ load = 0.044\ Mbps$$

$$scaling\ factor = 10^6$$

$$B_{LTE} = \frac{CDR_{th} \cdot A_{LTE} \cdot mean\ load \cdot scaling\ factor}{10 \cdot 60} \quad [Mbps]$$

$$B_{UMTS} = \frac{CDR_{th} \cdot A_{UMTS} \cdot mean\ load \cdot scaling\ factor}{10 \cdot 60} \quad [Mbps]$$

$$B_{GSM} = \frac{CDR_{th} \cdot A_{GSM} \cdot mean\ load \cdot scaling\ factor}{10 \cdot 60} \quad [Mbps]$$

The traffic load (L) is calculated for each hour, changing the $saturation\ factor$ depending on the type of the BS and depending on the CDR_{th}:

$$\begin{cases} L = \frac{CDR_t \cdot A_{LTE,UMTS,GSM} \cdot mean\ load \cdot scaling\ factor}{10 \cdot 60} \quad [Mbps] & CDR_t < CDR_{th} \\ L = B_{LTE,UMTS,GSM} \quad [Mbps] & CDR_t \geq CDR_{th} \end{cases}$$

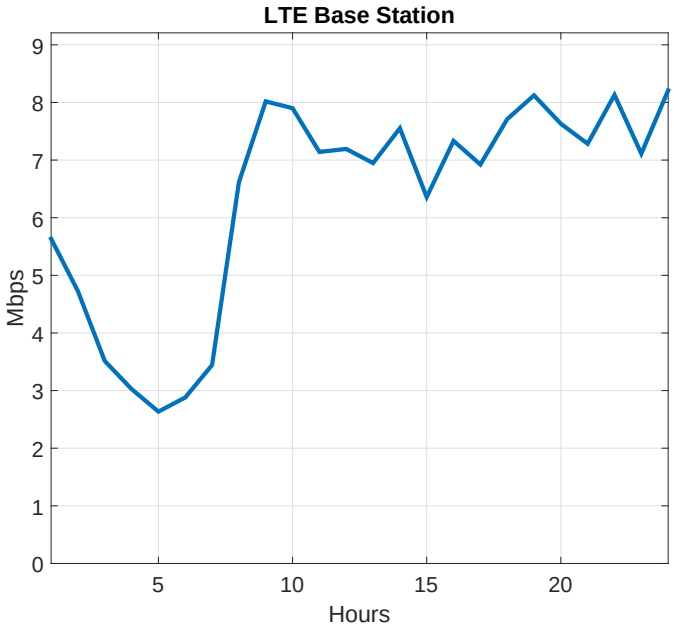


Figure 3.10: *Example of a LTE Base Station.*

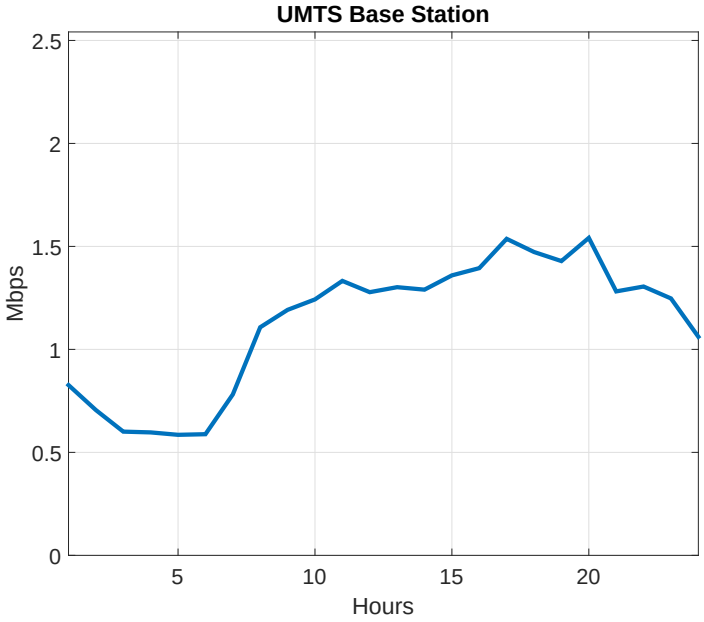


Figure 3.11: *Example of a UMTS Base Station.*

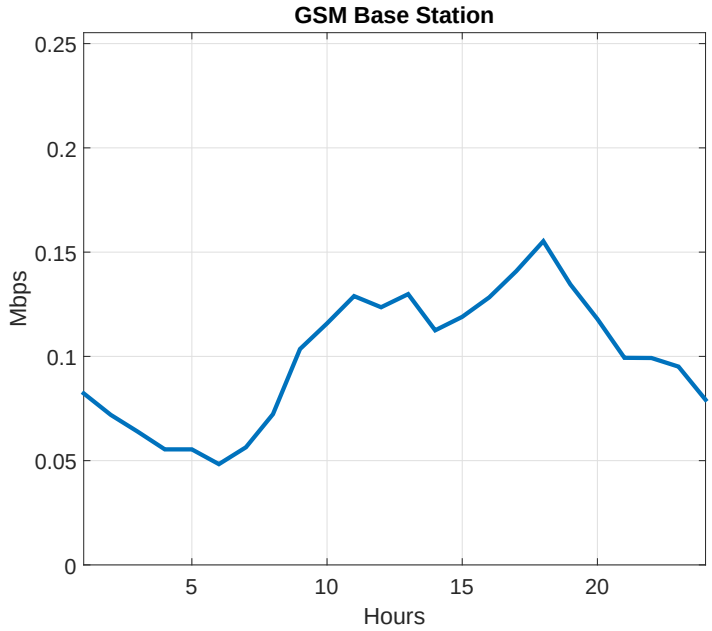


Figure 3.12: *Example of a GSM Base Station.*

Chapter 4

Prediction

In this chapter we will talk about a subject required in many situations, i.e. forecasting. Below some examples:

- Deciding whether to build another power generation plant in the next five years, it needs forecasts of future demand;
- Scheduling staff in a call centre next week requires forecasts of call volumes;
- Stocking an inventory demands forecasts of stock requirements.

Predictions can be necessary several years in advance (for the case of capital investments), or only a few minutes beforehand (for telecommunication routing). Whatever the circumstances or time horizons involved, forecasting is an important aid to effective and efficient planning. The predictability of an event or a quantity depends on several factors including:

- How well we understand the factors that contribute to it;
- How much data are available;

- Whether the forecasts can affect things we are trying to forecast.

Forecasting methods can be very simple such as using the most recent observation as a forecast (which is called the *naive method*), or highly complex like neural networks and econometric systems of simultaneous equations.

In this work we started using the autoregressive model, moving on a machine learning approch until implementing the Artificial Neural Network (ANN) to make a long range prediction. The goal is to use the forecasted traffic load in order to plan in advance the best resource allocation in the underlying optical metro network, and reduce the power consumption of the optical devices.

4.1 Autoregressive model

In an autoregression model, we forecast the variable of interest using a linear combination of past values [7]. The term autoregression indicates a regression of the variable against itself. In this model the output variable depends linearly on its own previous values and on a *stochastic term*. The notation $AR(p)$ indicates an autoregressive model of order p, and it is defined as:

$$y_t = c + \theta_1 y_{t-1} + \theta_2 y_{t-2} + ... + \theta_p y_{t-p} + e_t$$

c is the mean value of the process Y (if the process is stationary $c = 0$), e_t is the white noise.

An autoregressive model is normally restricted to stationary data, for this reason, the process represented by the random variable Y should have the mean value, variance and autocorrelation constant. Otherwise, the process could be weak-sense stationary (WSS) because only requires the 1st

moment and autocorrelation constant over time. The model parameters can be derived solving the Yule Walker equations, explained in section 4.1.1.

4.1.1 Yule-Walker equations

The Yule-Walker equations, named for Udny Yule and Gilbert Walker, help us to find the parameters θ_i. Starting from the general model $AR(p)$:

$$y_t = \theta_1 y_{t-1} + \theta_2 y_{t-2} + ... + \theta_p y_{t-p} + e_t$$

Multipling both side for y_{t-m} where m is the lag between samples:

$$y_t \cdot y_{t-m} = (\theta_1 y_{t-1} + \theta_2 y_{t-2} + ... + \theta_p y_{t-p} + e_t) \cdot y_{t-m} = \sum_{j=1}^{p} \theta_j y_t y_{t-m} + e_t \cdot y_{t-m}$$

we compute the expected value:

$$E[y_t \cdot y_{t-m}] = E\left[\sum_{j=1}^{p} \theta_j y_t y_{t-m}\right] + E[e_t \cdot y_{t-m}] = \sum_{j=1}^{p} E[\theta_j y_t y_{t-m}] + E[e_t \cdot y_{t-m}]$$

The first term $E\left[\sum_{j=1}^{p} \theta_j y_t y_{t-m}\right]$ is the autocorrelation r_m, and the second one is equal to:

$$E[e_t \cdot y_t] = E\left[e_t \cdot \left(\sum_{i=1}^{p} \theta_i y_{t-i} + e_t\right)\right] = \sum_{i=1}^{p} \theta_i E[e_t y_{t-i}] + E[e_t^2] = 0 + \sigma_{e_t}^2$$

Finally we obtain:

$$r_m = \sum_{i=1}^{p} \theta_i r_{m-i} + \sigma_{e_t}^2 \delta_m$$

Rewriting all the equations together, it yields:

$$r_1 = \theta_1 r_0 + \theta_2 r_1 + \theta_3 r_2 + ... + \theta_{p-1} r_{p-2} + \theta_p r_{p-1}$$

$$r_2 = \theta_1 r_1 + \theta_2 r_0 + \theta_3 r_1 + ... + \theta_{p-1} r_{p-3} + \theta_p r_{p-2}$$

$$...$$

$$r_{p-1} = \theta_1 r_{p-2} + \theta_2 r_{p-3} + \theta_3 r_{p-4} + ... + \theta_{p-1} r_0 + \theta_p r_1$$

$$r_p = \theta_1 r_{p-1} + \theta_2 r_{p-2} + \theta_3 r_{p-3} + ... + \theta_{p-1} r_1 + \theta_p r_0$$

The equations can also be written as:

$$\begin{bmatrix} r_1 \\ r_2 \\ ... \\ r_{p-1} \\ r_p \end{bmatrix} = \begin{bmatrix} r_0 & r_1 & r_2 & ... & r_{p-2} & r_{p-1} \\ r_1 & r_0 & r_1 & ... & r_{p-3} & r_{p-2} \\ ... & ... & ... & ... & ... & ... \\ r_{p-2} & r_{p-3} & r_{p-4} & ... & r_0 & r_1 \\ r_{p-1} & r_{p-2} & r_{p-3} & ... & r_1 & r_0 \end{bmatrix} \times \begin{bmatrix} \theta_1 \\ \theta_2 \\ ... \\ \theta_{p-1} \\ \theta_p \end{bmatrix}$$

Recalling that r_0 is equal to 1, the above equations are also written in this way:

$$\begin{bmatrix} r_1 \\ r_2 \\ ... \\ r_{p-1} \\ r_p \end{bmatrix} = \begin{bmatrix} 1 & r_1 & r_2 & ... & r_{p-2} & r_{p-1} \\ r_1 & 1 & r_1 & ... & r_{p-3} & r_{p-2} \\ ... & ... & ... & ... & ... & ... \\ r_{p-2} & r_{p-3} & r_{p-4} & ... & 1 & r_1 \\ r_{p-1} & r_{p-2} & r_{p-3} & ... & r_1 & 1 \end{bmatrix} \times \begin{bmatrix} \theta_1 \\ \theta_2 \\ ... \\ \theta_{p-1} \\ \theta_p \end{bmatrix}$$

or in a vectorial way:

$$\vec{r} = \vec{R}\vec{\Theta}$$

Note that this system has the same number of $\vec{R}$ rows as the elements θ_j of the unknown vector $\vec{\Theta}$. Further, $\vec{R}$ is a full-rank and symmetric, so that invertability is guaranteed:

$$\hat{\vec{\Theta}} = \vec{R^{-1}}\vec{r}$$

$\hat{\vec{\Theta}}$ is the estimated vector of parameters.

4.1.2 Stationary test

In order to apply the autoregressive model we need to make a *stationarity test* on our data. We derived the mean value 4.1, the variance 4.2 and the autocorrelation 4.6 day by day during the month of November.

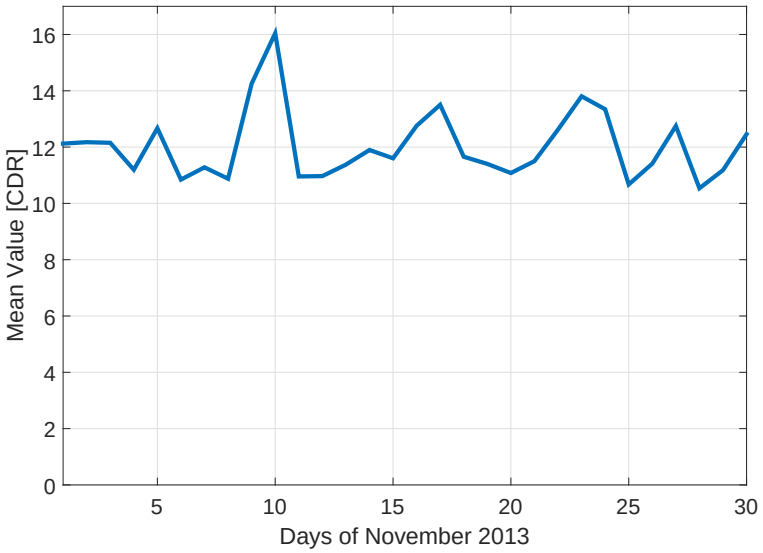


Figure 4.1: *Mean value of CDRs*

As we can see from the figures, the process is not stationary. For this reason, I tried to transform it in a stationary one by differencing the samples. If this process is stationary, or at least in weak sense, we can adapt the

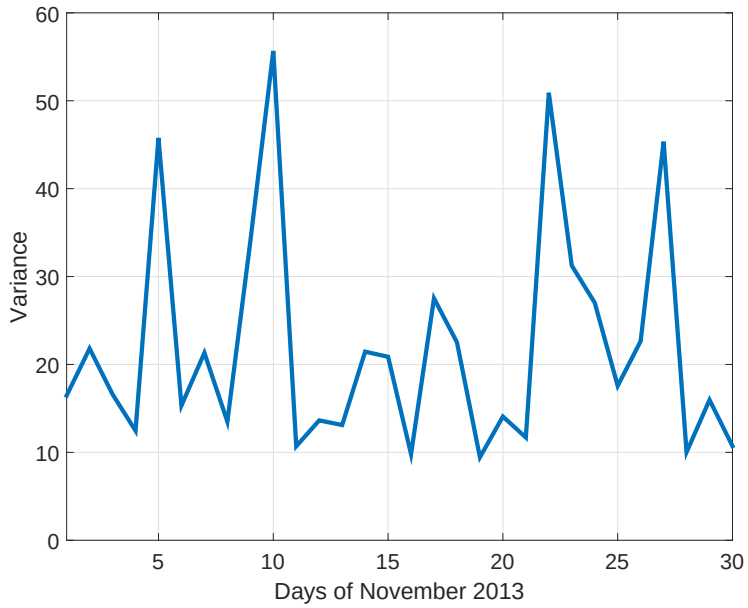


Figure 4.2: *Variance of CDRs*

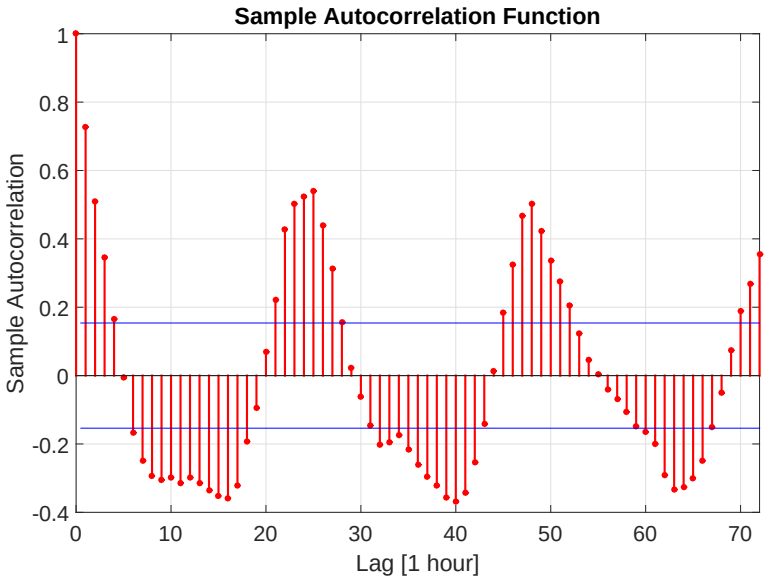


Figure 4.3: *Autocorrelation of the process*

autoregressive model to this one and forecast the differences.

$$y\prime_t = y_t - y_{t-1}$$

$y\prime_t$ is the sample at time t of the differentiated process.

Figures 4.4 and 4.5 shows respectively the mean value and the autocorrelation.

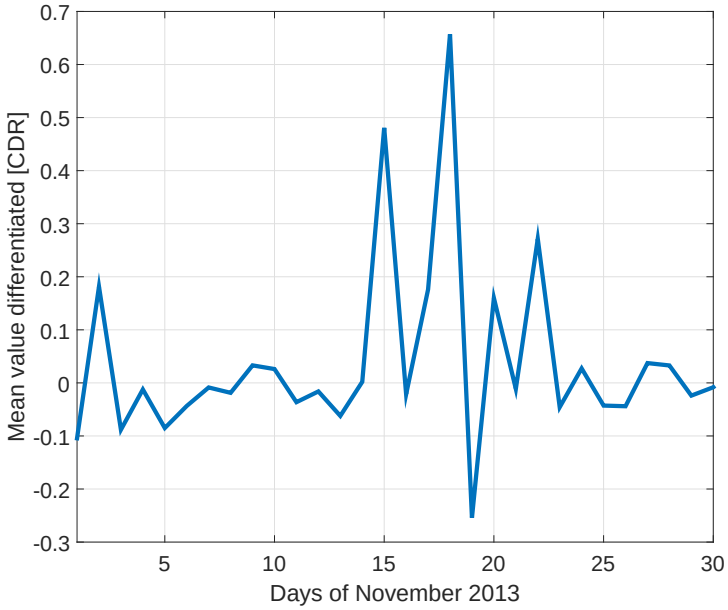


Figure 4.4: *Mean of the differentiated process*

Since the differentiated process seems to be stationary in weak-sense, we can apply the AR model to forecast the differences between consecutive samples. We can use MATLAB to run the ARIMA (Auto Regressive Integrated Moving Average) model in order to get the parameters, solving the Yule Walker equations, and taking into account even the error of previous predictions.

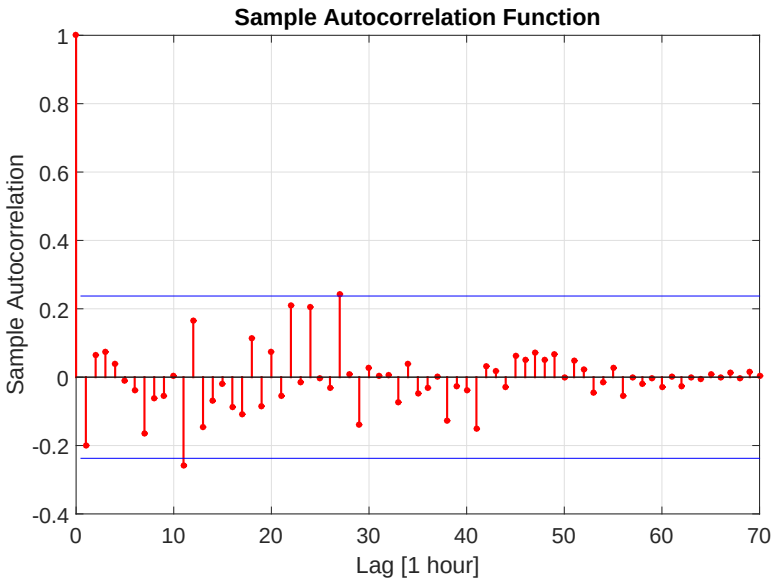


Figure 4.5: *Autocorrelation of the differentiated process*

4.1.3 ARIMA

In statistics and in particular in time series analysis, an AutoRegressive Integrated Moving Average model is an important tool to make the prediction. We saw that in an autoregression model, we forecast the variable of interest using a linear combination of past values.

$$y_t = c + \theta_1 y_{t-1} + \theta_2 y_{t-2} + ... + \theta_p y_{t-p} + e_t$$

For an $AR(1)$ model:

- When $\theta_1 = 0$, y_t is equivalent to the white noise.
- When $\theta_1 = 1$ and $c = 0$, y_t is equivalent to a random walk.
- When $\theta_1 = 1$ and $c \neq 0$, y_t is equivalent to a random walk with drift.
- When $\theta_1 < 0$, y_t tends to oscillate between positive and negative values.

We normally restrict autoregressive models to stationary data, and then some constraints on the values of the parameters are required.

- For an $AR(1)$ model:$-1 < \theta_1 < 1$.
- For an $AR(2)$ model:$-1 < \theta_2 < 1,\ \theta_1 + \theta_2 < 1,\ \theta_2 - \theta_1 < 1$.

Rather than using past values of the forecast variable in a regression, a *moving average model* uses past forecast errors in a regression-like model:

$$y_t = c + e_t + \phi_1 e_{t-1} + \phi_2 e_{t-2} + ... + \phi_q e_{t-q}$$

e_t is the white noise.

We refer to this as $MA(q)$ model. In real life we do not observe the value of e_t, but we can note that each value of y_t can be thought as a weighted moving average of the past few forecast errors. If we combine differencing with autoregression and a moving average model, we obtain ARIMA, where the full equation can be written as follow:

$$\hat{y\prime}_t = c + \theta_1 y\prime_{t-1} + \theta_2 y\prime_{t-2} + ... + \theta_p y_{t-p} + e_t + \phi_1 e_{t-1} + ... + \phi_q e_{t-q}$$

$\hat{y\prime}_t$ is the estimation of the defferentiated samples.

We call this an $ARIMA(p, d, q)$ model, where:

- p = order of the autoregressive part;
- d = degree of first differencing involved;
- q = order of the moving average part.

Following models are a special case of ARIMA(p,d,q):

White noise	ARIMA(0,0,0)
Random walk	ARIMA(0,1,0) with no costant
Random walk with drift	ARIMA(0,1,0) with costant
Autoregression	ARIMA(p,0,0)
Moving average	ARIMA(0,0,q)

4.2 Machine Learning approch

Until now we introduced all the concepts we need in order to forecast the internet traffic. In a machine learning approch we define the kind of learning algorithm based on the dataset being studied. In our case a supervised learning algorithm is more suitable than others because it is possible to get some features from the data.

A first strategy is to consider the following formula:

$$\hat{y}_t = \hat{y}\prime_t + y_{t-1}$$

where $\hat{y}\prime_t$ is the prediction of the difference $y_t - y_{t-1}$, derived by which ARIMA(3,1,0), and y_{t-1} is the previous value.

To decrease the error and improve the prediction, we can use other feautures of the data. Recalling the autocorrelation function of the original process, figure 4.6, we note the sample at a time t has a strong correlation with:

1. The previous value;
2. The same value of the prevoius day;
3. The same of the prevoius week;

After that, we introduced all these elements, weighing them to an alpha factor, and made the following prediction formula:

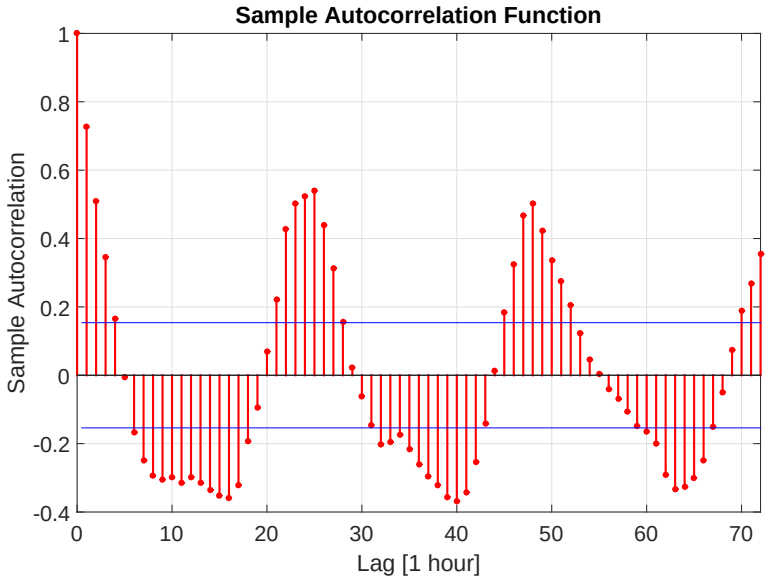


Figure 4.6: *Autocorrelation*

$$\hat{y}_t = \alpha_1 \hat{y\prime}_t + \alpha_2 y_{t-1} + \alpha_3 y_{t-h} + \alpha_4 y_{t-(h \cdot d)}$$

$h = 24$ are the hours and $d = 7$ are the days.

Section 4.2.1 showes the derivation of the α_i parameters, minimizing a *cost* function. This procedure is called *training* of the prediction model.

4.2.1 Training the model

The training part is the step where the model *learn* how to deal with different inputs. Dividing the data in training and test set, a computer calculates the α_i parameters by the minimization of the following cost function.

$$J(\alpha) = \frac{1}{2} \cdot (y_t - \hat{y}_t)^2$$

$$\min_{\alpha} J(\alpha)$$

y_t is the real value and $\hat{y}_t$ is the predicted one.

We will find a group of four parameter $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ for each entry of the training set that minimize the cost function $J(\alpha)$. We can use the *Gradient Descent* algorithm [38] to do what we said previously.

Gradient Descent

One of the most popoular algorithm to perform optimizations is the Gradient descent [39]. It is used to minimize, or maximize, an objective function parameterized by parameters α_i, and updating them iteratively until reaching the global or local minima. In this section, we are going to analyze the algorithm with one training sample; later we will see it with m training examples. Before starting, lets prepare the variables to understand easly the algorithm.

$$x_1 = \hat{y}\prime_t$$

$$x_2 = y_{t-1}$$

$$x_3 = y_{t-h}$$

$$x_4 = y_{t-(h \cdot d)}$$

$$h_\alpha(x) = \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 = \sum_{i=1}^{4} \alpha_i x_i$$

Now the cost function will be:

$$J(\alpha) = \frac{1}{2} \left(h_\alpha(x) - y\right)^2$$

The $\frac{1}{2}$ before the cost function is a trick to have a more convenient gradient for the calculations.

$$\min_{\alpha} J(\alpha)$$

We set the initial vector of α_i with random numbers, and obtain the derivatives:

$$\frac{d}{d\alpha_i} J(\alpha) = \frac{1}{2} \left[2 \cdot (h_\alpha(x) - y)\right] \cdot \frac{d}{d\alpha_i} \left[\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 - y\right]$$

The updating rule is the following:

$$\alpha_i := \alpha_i - \gamma \cdot \frac{d}{d\alpha_i} J(\alpha) = \alpha_i - \gamma \cdot (h_\alpha(x) - y) \cdot x_i$$

For example:

$$\alpha_1 := \alpha_1 - \gamma \cdot (\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 - y) \cdot x_1$$

$$\alpha_2 := \alpha_2 - \gamma \cdot (\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 - y) \cdot x_2$$

$$\alpha_3 := \alpha_3 - \gamma \cdot (\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 - y) \cdot x_3$$

$$\alpha_4 := \alpha_4 - \gamma \cdot (\alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_3 + \alpha_4 x_4 - y) \cdot x_4$$

where γ is the learning parameter and controls the step in the direction of the steepest descent. We repeat this procedure until convergence, obtaining the parameters.

Taking into account m traning samples, the above equations change in the following way:

$$x_1[1, 2, ..., m] = [\hat{y}\prime_{t_1}, \hat{y}\prime_{t_2}, ..., \hat{y}\prime_{t_m}]$$

$$x_2[1, 2, ..., m] = [y_{t_1-1}, y_{t_2-1}, ..., y_{t_m-1}]$$

$$x_3[1, 2, ..., m] = [y_{t_1-h}, y_{t_2-h}, ..., y_{t_m-h}]$$

$$x_4[1, 2, ..., m] = [y_{t_1-(h \cdot d)}, y_{t_2-(h \cdot d)}, ..., y_{t_m-(h \cdot d)}]$$

$$h_\alpha(x) = \sum_{i=1}^{m} (\alpha_1 x_1(i) + \alpha_2 x_2(i) + \alpha_3 x_3(i) + \alpha_4 x_4(i))$$

Now the cost function will be:

$$J(\alpha) = \frac{1}{2} \sum_{i=1}^{m} (\alpha_1 x_1(i) + \alpha_2 x_2(i) + \alpha_3 x_3(i) + \alpha_4 x_4(i) - y(i))^2$$

$$\min_{\alpha} J(\alpha)$$

The advantage of having more training examples is due to the fact that the algorithm can achieve a more accurate value of each alpha. The tests were made applying the algorithm with m training samples, where I derived $alpha1$, $alpha2$, $alpha3$, $alpha4$, for each entry of the dataset. I chose $m = 10$ training samples because it is a good compromise between time computation and algorithm accuracy.

4.3 Artificial Neural Network

In machine learning, an artificial neural network (ANN) is a network inspired by the central nervous systems of animals, which are used to estimate or approximate functions that can depend on a large number of inputs that are generally unknown. Neural networks are composed of simple elements operating in parallel. As in nature, the network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements.

Each time we describe a neural network algorithm we will typically specify three things.

- Architecture: the architecture specifies which variables are involved in the network and their topological relationships. For example, the variables involved in a neural network might be the weights of the connections between the neurons;
- Activity rule: most neural network models have local rules and define how the activities of the neurons change in response to each other. Typically the activity rule depends on the weights (the parameters) in the network;
- Learning rule: the learning rule specifies the way in which the neural network's weights change with time.

Activity rules and learning rules may be derived from carefully chosen objective functions. Commonly neural networks are adjusted or trained, so that a particular input leads to a specific target output. Such a situation is shown below in figure 4.7. The network is adjusted, based on a comparison of the output and the target, until the network output matches the target. This kind of neural network is based on a *supervised learning*, where typically many input/target pairs are used to train a network. Neural networks have been trained to perform complex functions in various fields of application including pattern recognition, identification, classification, speech, vision and control systems.

4.3.1 The single neuron

It is important to study how a single neuron works for two reasons [40]. First, many NN model are built out of single neurons, so that it is good

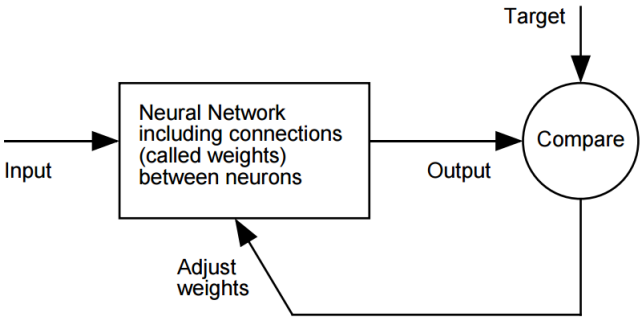


Figure 4.7: *ANN scheme*

to understand them in detail. Secondly, a single neuron is itself capable of learning, and this model will serve as a first example of a supervised neural network.

As mentioned before we start defining the architecture and the activity rule of a single neuron, and we will then derive a learning rule.

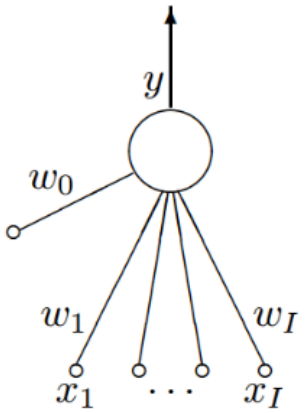


Figure 4.8: *Single neuron scheme*

- Architecture: A single neuron has got a number of inputs x_i and one output y (see figure 4.8). Associated with each input is a weight $w_i (i = 1, ..., I)$. There may be an additional parameter w_0 of the

neuron called *bias* which we may view as being the weight associated with an input x_0 permanently set to 1.

- Activity rule: The activity rule supposes two steps.
  - First, we compute the neuron activation,

$$a = \sum_{i=0}^{I} w_i x_i,$$

  - Second, the output y is set as a function $f(a)$ of the activation. There are several possible activation functions. They can be linear and non-linear.

- Learning rule: typically a learning rule is an *objective function* that measure how well the network with weights w_i solves the task. The training process is a function minimization, adjusting $\mathbf{w}$ in order to minimize the objective function. Here we used a form of *gradient descent* algorithm called Levenberg-Marquardt.

The activity and learning rules are repeated for each input/target pair (x, t) presented.

4.3.2 ANN model

In this work I used and modified an ANN model for the electricity load and price forecasting [41]. The model is calibrated to forecast hourly day-ahead loads given temperature forecasts, holiday information and historical loads. I added our case study changing the model and testing it on the training set. The three steps to building the forecaster include creating a matrix of predictors from the historical data, selecting and calibrating the

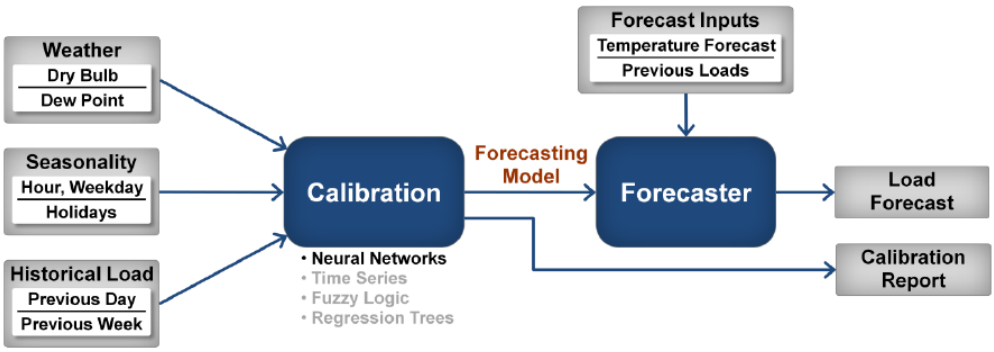


Figure 4.9: *ANN model for the electricity load and price forecasting*

chosen model and then running the artificial network live in the MATLAB interface.

In our case, since we do not have so many features of the data, I used the load of the previous day, the average and the load of the previous week to train the model. It came out that using about 50 neurons it was possible to obtain a percentage of error between 15 and 20%.

4.4 Application to metro traffic and results

To sum up, we studied three different prediction methods with the purpose of forecasting the traffic load of the metro network. The first one is the autoregressive model; the second is based on a machine learning approch, and the last is an application of an ANN model. For our purposes we used the Telecom Italia dataset before and after the grouping procedure with 2554 BS, forecasting the CDRs. The performances have been evaluated by obtaining the Mean Absolute Error (MAE) and the Mean Absolute Percentage Error (MAPE) overall the network, using the data-set before the processing ($dataset_1$ = [10000x1488]) and after ($dataset_2$ = [2554x1488]).

We split up each dataset in two part. The *training set* from 01/11/2013 to 15/12/2013, and the *test set* from 16/12/2013 to 01/01/2014.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{\overline{y_i}}\right|$$

where $\overline{y_i}$ is the mean value.

4.4.1 Autoregressive model

I applied ARIMA(12,0,0) and obtained the following mean value of each accuracy functions.

$$dataset_1 : \begin{cases} MAE = 6.44 \\ MAPE = 17.94\% \end{cases}$$

$$dataset_2 : \begin{cases} MAE = 128.58 \\ MAPE = 23.71\% \end{cases}$$

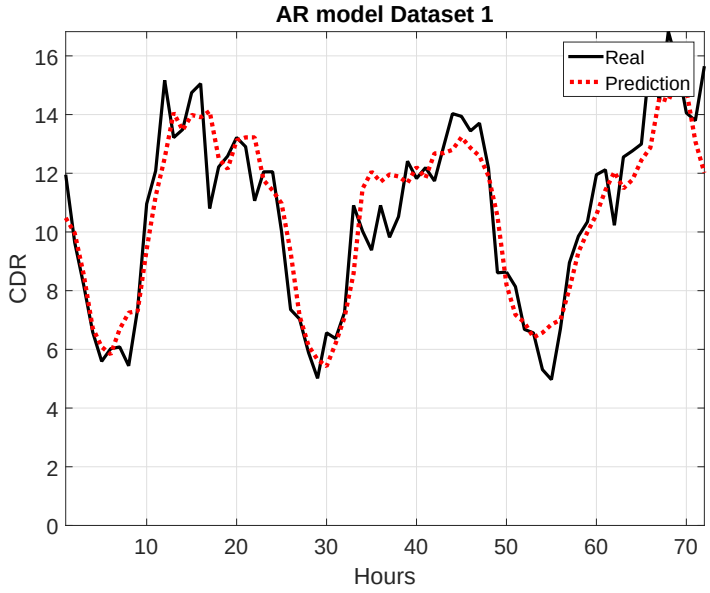


Figure 4.10: *Autoregressive model of a squared cell during 3 days of the test set.*

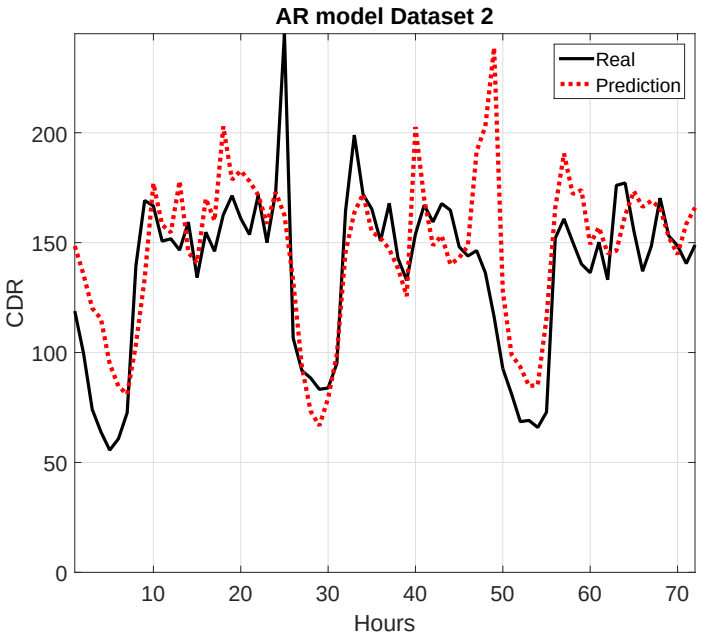


Figure 4.11: *Autoregressive model of a base station during 3 days of the test set.*

4.4.2 Machine Learning approach

This method is scalable, we have the chance to add some other feature and re-make the machine learning procedure. We note that we obtained better results with the first data-set than with the second one. This happen because the second data-set is not pure as the first one, but it has been derived from a processing of two different data-sets.

$$dataset_1 : \begin{cases} MAE = 6.53 \\ MAPE = 15.97\% \end{cases}$$

$$dataset_2 : \begin{cases} MAE = 164.44 \\ MAPE = 24.85\% \end{cases}$$

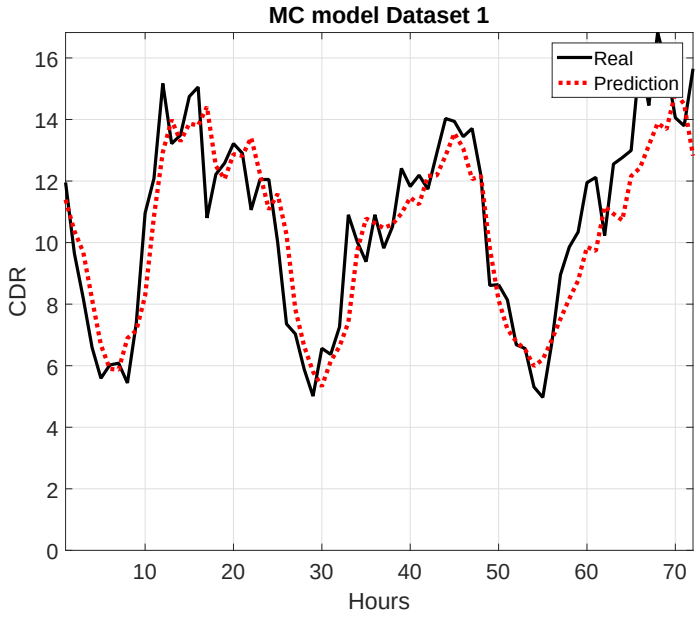


Figure 4.12: *Machine learning approach of a squared cell during 3 days of the test set.*

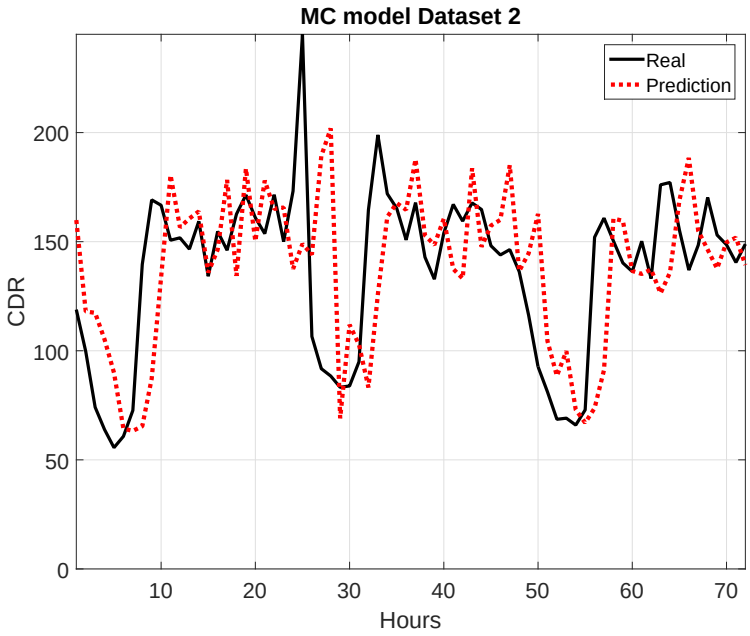


Figure 4.13: *Machine learning approach of a base station during 3 days of the test set.*

4.4.3 Artificial Neural Network

ANN performances have been obtained considering 53 neurons.

$$dataset_1 : \begin{cases} MAE = 44.79 \\ MAPE = 17.73\% \end{cases}$$

$$dataset_2 : \begin{cases} MAE = 4.71 \\ MAPE = 29.50\% \end{cases}$$

As we have seen from the state of the art and in this chapter, there are many methods that allow us to create a predictive model. In particular , the approach to machine learning is scalable, which means we can add features that make the value that we want to predict. This is the reason why we

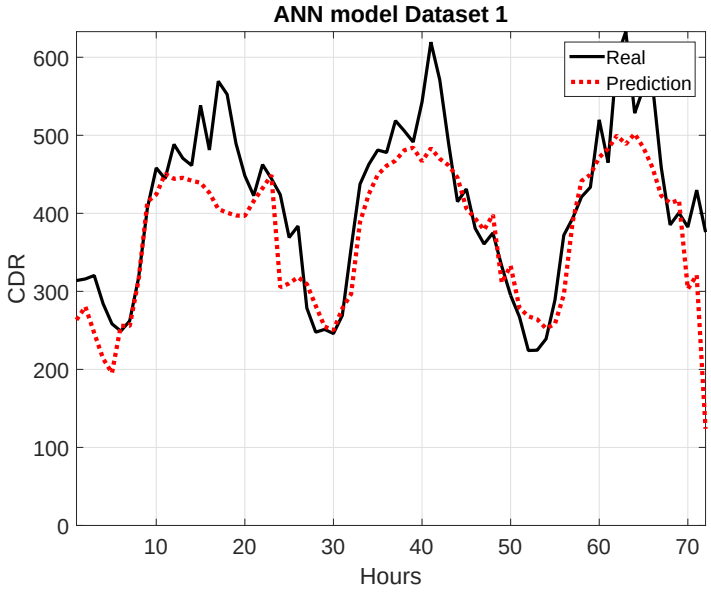


Figure 4.14: *ANN 24h prediction of a squared cell of the* $dataset_1$ *during 3 days.*

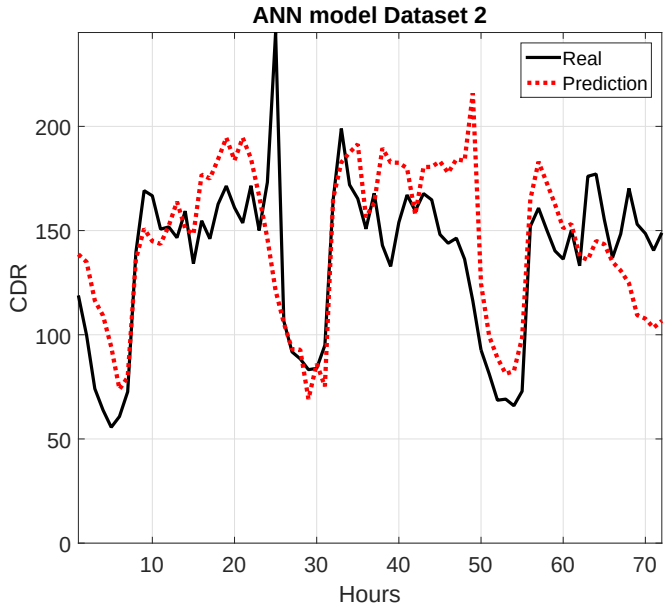


Figure 4.15: *ANN 24h prediction of a base station of the* $dataset_2$ *during 3 days.*

applied clustering and pattern recognition techniques on datasets available in this work.

Chapter 5

Clustering

Given a set of data that describe the internet traffic load of each base station, it is possible to group these data sequences based on their similarity thanks to the so called clustering methods. This could lead us to define the social function of the base stations and recognize their typical pattern during different days. In this chapter we studied two clustering methods in order to find similarities among base stations and try to define their social function. In section 5.1 we will see the presence of a similarity between cells located in different places but with the same social ID. Unfortunatly, DUSAF database is not updated and contains some errors about the type of buildings present on the territory. For this reason, in this chapter we limit ourselves only to apply two clustering algorithms, and use the output in chapter 6 as input for a method with the aim to identify typical patterns. This kind of information could be used during the planning of the telecommunication networks or even for urbanistic purposes.

5.1 Dynamic Time Warping

In order to see if there is a similarity between two different base stations, even in different part of the city, I used a tool called Dynamic Time Warping (DTW). It is a well-known algorithm which aims at comparing and aligning two sequences of data points [33], for example time series. Given two sequences a and b in the time:

$$A(t) = a_1, a_2, a_3, ..., a_n$$

$$B(t) = b_1, b_2, b_3, ..., b_m$$

DTW works by warping (hence the name) the time axis iteratively until an optimal match between the two sequences is found. We can construct a n x m distance matrix where each cell (i,j) represents the distance between the i-th element of sequence A and the j-th element of sequence B. The distance metric used depends on the application, but a common metric is the Euclidean distance, figure 5.1.

Finding the best alignment between two sequences can be seen as finding the shortest path to go from the bottom-left cell to the top-right of the distance matrix. The length of a path is simply the sum of all the cells that were visited along that path. Further away the optimal path wanders from the diagonal, more the two sequences need to be warped to match together. In our case, I chose four cells, far from each other, with the same social ID. Figures 5.2 5.3 5.4 5.5 and 5.6 5.7 5.8 5.9, shows two different social ID, 1111 (residential), 12111 (Industrial places), in a period of 24 hours. As we can see, there exist a similarity between them warping a little bit the signals.

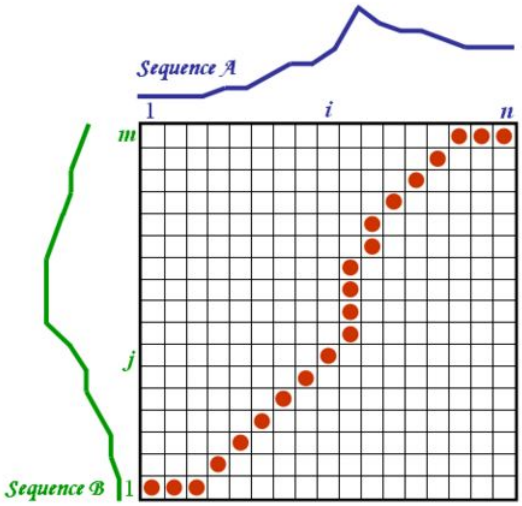


Figure 5.1: *Distance matrix*

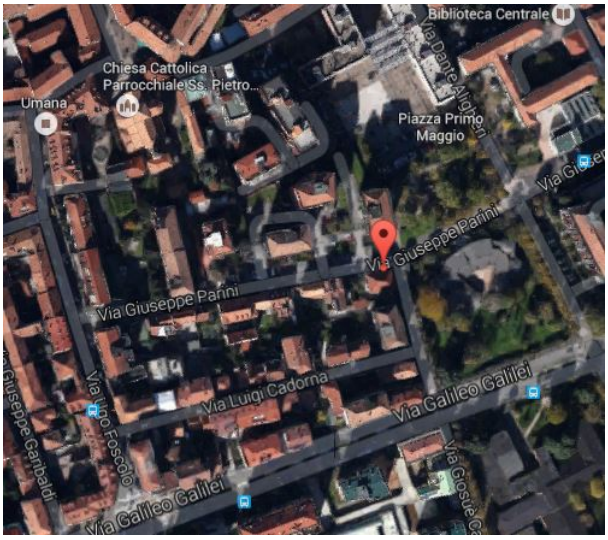


Figure 5.2: *Cell 3534 - social ID 1111 (Residential)*

5.2 K-means

K-means is one of the most popular and simplest unsupervised learning algorithm that solves the well known clustering problem [12]. The procedure classifies a given dataset through a certain number of clusters fixed a priori.

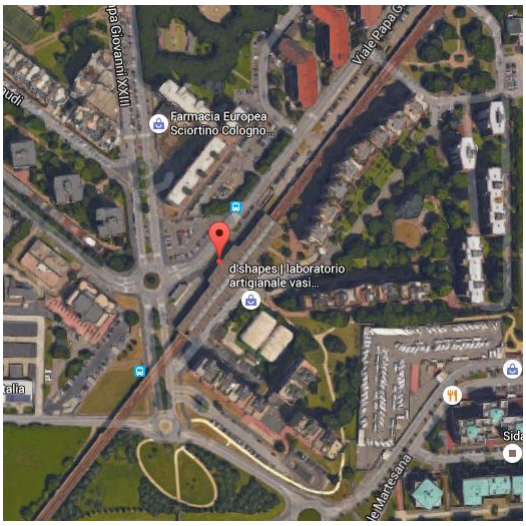


Figure 5.3: *Cell 7889 - social ID 1111 (Residential)*

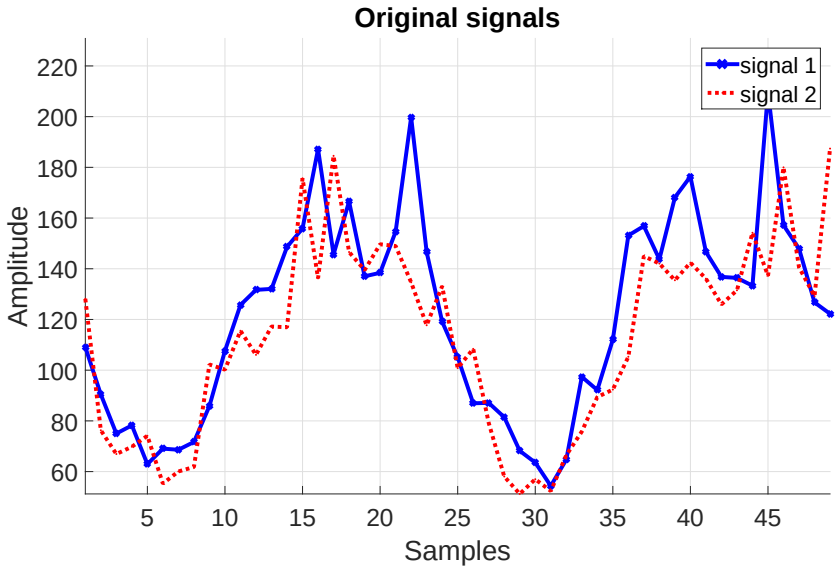


Figure 5.4: *Original signals - cells 3534, 7889*

The main idea is to define k centroids, one for each cluster, but different locations cause different results. So, the better choice is to place them as much as possible far away from each other or randomly. The next step is to take each point belonging to a given data-set and associate it to the nearest

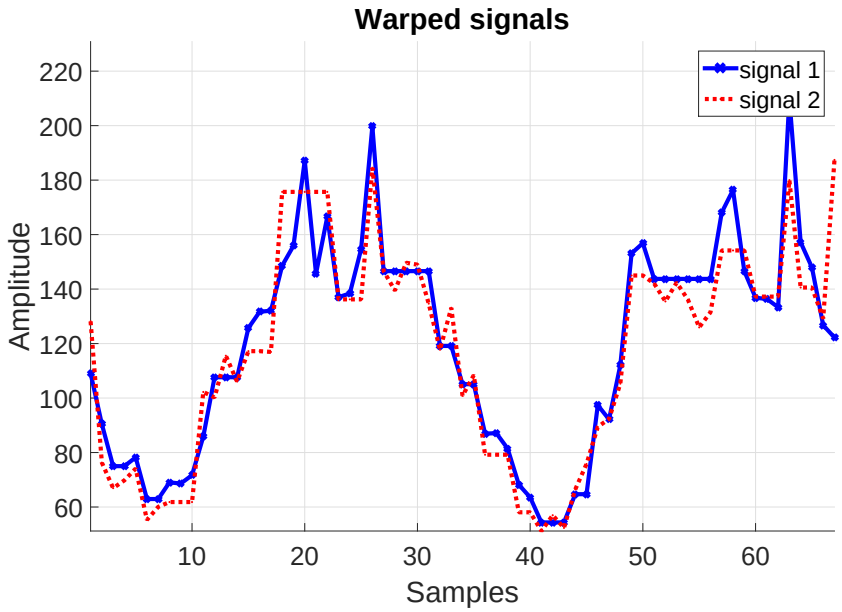


Figure 5.5: *Warped signals - cells 3534, 7889*

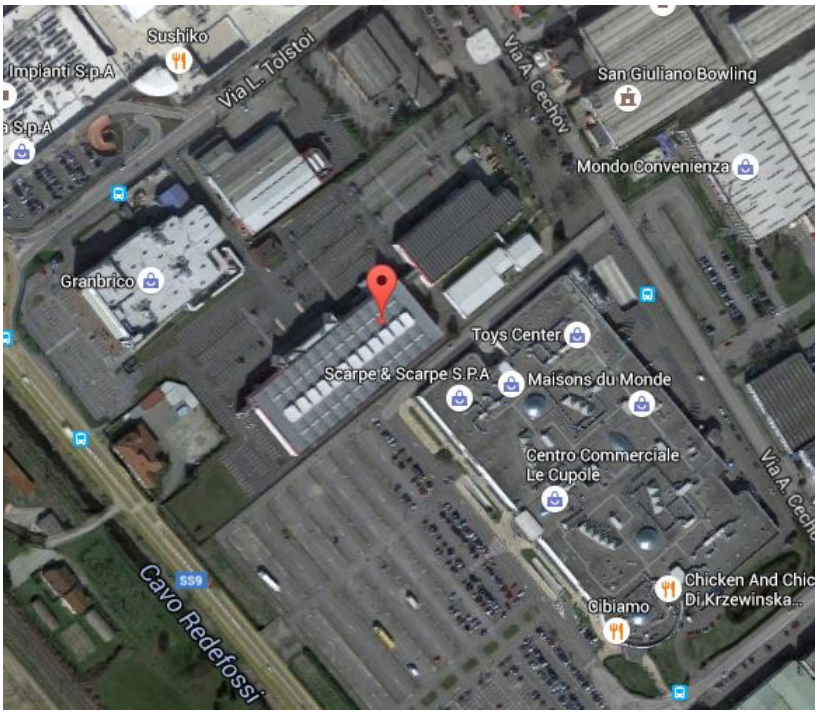


Figure 5.6: *Cell 1298 - social ID 12111 (Industrial places)*

Figure 5.7: *Cell 7943 - social ID 12111 (Industrial places)*

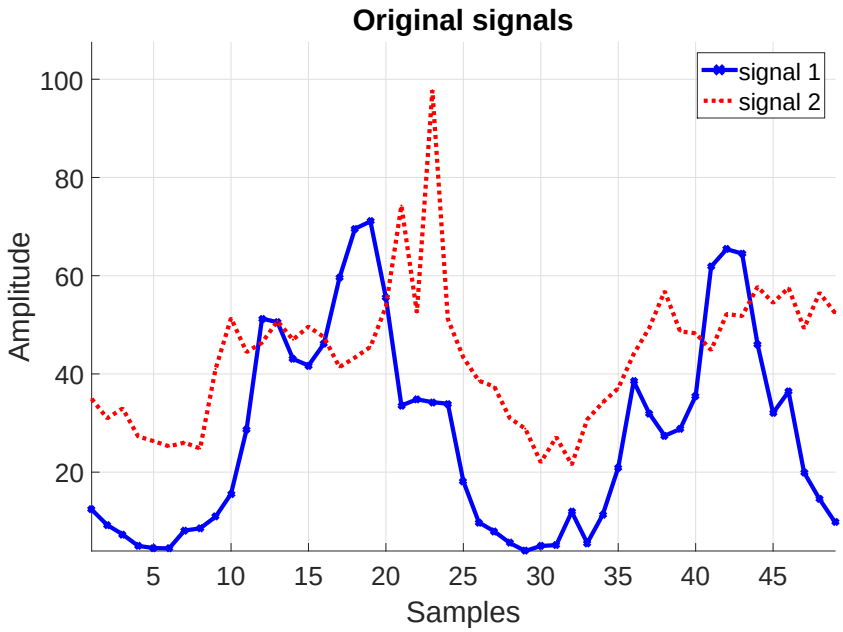


Figure 5.8: *Original signals - cells 1298, 7943*

centroid. After completing the first step, an early groupage is done. At this point we need to re-calculate k new centroids as *barycenters* of the clusters

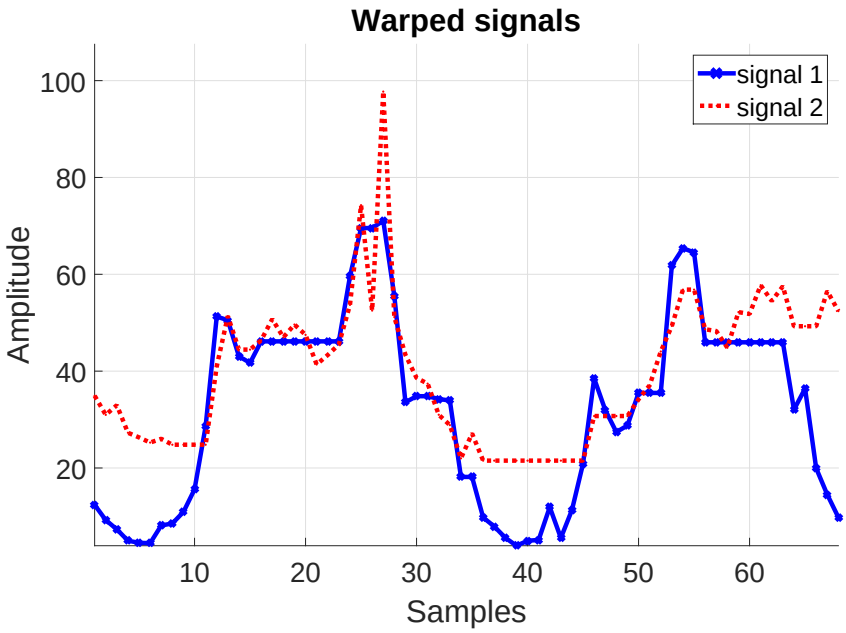


Figure 5.9: *Warped signals - cells 1298, 7943*

resulting from the previous step. Then, a new association has to be done in order to create new clusters. This procedure is repeated until k centroids do not change their location any more. This algorithm plans at minimizing an objective function, in this case a squared error function showed below:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} |x_i^{(j)} - c_j|^2$$

where $|x_i^{(j)} - c_j|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and cluster centre c_j, and it is an indicator of the distance of the n data points from their respective cluster centers.

The algorithm can be consolidated by the following steps:

- Dispose K points into the space represented by the objects being clustered. These points represent initial group of centroids.

- Assign each object to the closest centroid, generating a first clustering.

- When all objects have been assigned, recalculate the positions of the k centroids as the barycenter of each cluster.
- Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups called clusters.

This procedure will always terminate, but the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global minimum of the objective function. As we said before, the algorithm is also significantly sensitive to the initial randomly centroids distribution. The k-means algorithm can be run multiple times to reduce this effect.

5.3 Spectral clustering

Spectral clustering is one of the most popular clustering algorithm that uses the eigenvectors of some matrices to find a partition of the data with the aim to obtain groups with similar points. This simple and powerful method requires less assamptions on the form of clusters and outperforms the traditional approches, such as k-means clustering [15]. Before moving on the algorithm we will introduce the mathematical background.

5.3.1 Similarity graphs

Given a set of points $x_1, x_2, ..., x_n$ and some notion of similarity $s_{ij} \geq 0$ between all pairs of data points x_i and x_j, the goal of clustering is to divide the data into several groups where points in different sets are distinct to each other. However, we can represent the data by the similarity graph $G = (V, E)$ where each vertex v_i represents a data point x_i. Two vertices are connected if the similarity s_{ij} between the corresponding data points x_i

and x_j is positive or larger than a threshold, and the edge is weighted by s_{ij}.

Clustering procedure can be reformulated using the similarity graph, partitioning the graph such that the edges between different groups have very low weights, which means that vertices (data points) within the same cluster are similar to each other.

Lets introduce some basic graph notation:

- $G = (V, E)$ is an undirected graph with vertex set $V = v_1, v_2, ..., v_n$;
- $W = (w_{ij})_{i,j=1,2,3,...,n}$ is the *adiacency matrix* of the graph G, the matrix of the weights between each vertex, if $w_{ij} = 0$ there is no edge between v_i and v_j;
- $d_i = \sum_{j=1}^{n} w_{ij}$ is the degree of a vertex $v_i \in V$;
- The *degree matrix* D is defined as the diagonal matrix with the degrees $d_1, d_2, ..., d_n$ on the diagonal;
- Given a subset of vertices $A \subset V$, we denote its complement by $\overline{A}$;
- We define the *indicator vector* $\nVdash = (f_1, f_2, ..., f_n)' \in \mathbb{R}^n$ as the vector with entries $f_i = 1$ if $v_i \in A$ and $f_i = 0$ otherwise;
- For convenience we define a shorthand notation $i \in A$ for the set of indeces $i | v_i \in A$, for example: $\sum_{i \in A} w_{ij}$;
- For two not necessarly disjoint sets $A, B \subset V$ we define: $W(A, B) = \sum_{i \in A, i \in B} w_{ij}$;
- For the mesaure of the size of a subset, we consider two method:
  - $|A| := the\ number\ of\ vertices\ in\ A$;

  - $vol(A) := \sum_{i \in A} d_i$;

- A subset A of a graph is *connected* if any two vertices in A can be joined by a path such that all intermediate points also lie in A;

- A subset A is called a *connected component* if it is connected and if there are no connections between vertices in A and $\overline{A}$;

There are several popular way to transform a given set $x_1, x_2, ..., x_n$ of data points with similarities s_{ij} or pairwise distances d_{ij} into a graph. When constructing similarity graphs the goal is to model the local neighborhood relationships between the data points. We will show three methods:

- **The ϵ-neighborhood graph**: Here we connect all points whose pairwise distances are smaller than ϵ. Weighting the edges would not incorporate more information about the data to the graph. Hence, the ϵ-neighborhood is usually considered as an unweighted graph.

- **k-nearest neighbor graphs**: Here the goal is to connect vertex v_i with vertex v_j if this last is among the k-nearest neighbors of v_i. However, this definition leads to a directed graph, so the neighborhood relationship is not symmetric. There are two ways of making this graph undirected. The first one is to simply ignore the directions of the edges, connecting v_i and v_j with an undirected edge if v_i is among the k-nearest neighbors of v_j or if v_j is among the k-nearest neighbors of v_i. The resulting graph is what is usually called the k-nearest neighbor graph. The second way is to connect vertices v_i and v_j if v_i is among the k-nearest neighbors of v_j and v_j is among the k-nearest neighbors of v_i. The resulting graph is called the *mutual k-nearest neighbor graph.*

- **The fully connected graph**: Here we simply connect all points with positive similarity with each other, and we weight all edges by s_{ij}. As the graph should represent the local neighborhood relationships, this construction is only useful if the similarity function itself models local neighborhoods. An example for such a similarity function is the Gaussian similarity:

$$s(x_i, x_j) = exp(-||x_i - x_j||^2/(2\sigma^2))$$

  where the parameter σ controls the width of the neighborhoods. This parameter plays a similar role as the parameter ϵ in case of the ϵ-neighborhood graph.

All graphs mentioned above are regularly used in spectral clustering, and the choice of the similarity graph influences the algorithm. Below an example:

We use the Gaussian similarity to represent local neighborhood relationships:

$$W_{ij} = exp(-||x_i - x_j||^2/(2\sigma^2))(i \neq j)$$

where W is the adjacency matrix of similarity graph. D is the degree matrix:

$$D_{ij} = \begin{cases} \sum_k W_{ik} & i = j \\ 0 & i \neq j \end{cases}$$

For the following data points:

$$x_1 = (6, 8);\ x_2 = (3, 1);\ x_3 = (7, 9);\ x_4 = (3, 2);\ x_5 = (9, 8);\ x_6 = (2, 4);$$

We have the adiacency matrix:

$$W = \begin{pmatrix} 0 & 0.04 & 0.89 & 0.08 & 0.61 & 0.17 \\ 0.04 & 0 & 0.01 & 0.95 & 0.01 & 0.57 \\ 0.89 & 0.01 & 0 & 0.03 & 0.76 & 0.06 \\ 0.08 & 0.95 & 0.03 & 0 & 0.02 & 0.76 \\ 0.61 & 0.01 & 0.76 & 0.02 & 0 & 0.03 \\ 0.17 & 0.57 & 0.06 & 0.76 & 0.03 & 0 \end{pmatrix}$$

$$\sigma = 3$$

5.3.2 Graph Laplacians and their basic properties

Once we obtain the similarity graph, we use the *spectral graph theory* [42] as a tool to cluster, deriving the graph Laplacian matrix. It represents an affinity matrix where the eigenvectors describe different group of data. Finding the eigenvectors means discovering the clusters in the network. Different methods use different Laplacian matrices. Following we assume G as an undirected and weighted graph with matrix W, where $w_{ij} = w_{ji} \geq 0$.

The unnormalized graph Laplacian

The *unnormalized graph Laplacian* is defined as:

$$L = D - W$$

where D is the *degree matrix* and W is the *adiacency matrix* of the graph G. The matrix L satisfies the following properties [43] (that we will need for the clustering):

- For every vector $f \in \mathbb{R}^n$ we have:

$$f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

- L is symmetric and positive semi-definite;

- The smallest eigenvalue of L is 0, the corresponding eigenvector is the constant one vector $\mathbb{1}$;

- L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$;

The unnormalized graph Laplacian and its eigenvalues and eigenvectors can be used to describe many graph properties. One of these properties is described as follow:

Proposition (Number of connected components and the spectrum of L)-*Let G be an undirected graph with non-negative weights. Then the multiplicity k of the eigenvalue 0 of L equals the number of connected components $A_1, ..., A_k$ in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, .., \mathbb{1}_{A_k}$ of those components.*

Proof. We start with the case $k = 1$, that is the graph is connected. Assume that f is an eigenvector with eigenvalue 0. Then we know that:

$$0 = f'Lf = \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

As the weights w_{ij} are non-negative, this sum can only vanish if all terms $w_{ij}(f_i - f_j)^2$ vanish. Thus, if two vertices v_i and v_j are connected, i.e. $w_{ij} > 0$, then f_i needs to be equal f_j. With this argument we can see that f neeeds to be constant for all vertices which can be connecetd by a path in the graph. Moreover, as all vertices of a connected component in an undirected graph can be connected by a path, f needs to be constant on the

whole connected component. In a graph consisting of only one connected component we thus only have the constant one vector $\mathbb{1}$ as eigenvector with eigenvalue 0, which obviously is the indicator vector of the connected component.

The normalized graph Laplacian

In addition to unormalized graphs there are two matrices called normalized graph Laplacians. Both are closely related to each other and are defined as:

$$L_{sym} := D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$

$$L_{rw} := D^{-1} L = I - D^{-1} W$$

We denote the first by L_{sym} as *symmetric matrix*, and the seconde by L_{rw} because it is closely related to a random walk.

5.3.3 Spectral Clustering Algorithms

Now, we will define three spectral clustering algorithms assuming that our data consists of n points $x_1, x_2, ..., x_n$ and we measure their pairwise similarities $s_{ij} = s(x_i, x_j)$ by a similarity function which is symmetric and non-negative. We denote the corresponding similarity matrix by $S = (s_{ij})_{i,j=1...n}$.

Unnormalized spectral clustering

INPUT: Similarity matrix $S \in \mathbb{R}^{nxn}$, number of k of clusters to construct.

- Construct the similarity graph by one of the ways described in section 5.3.1. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.

- **Compute the first k eigenvectors $u_1, ..., u_k$ of L.**
- Let $U \in \mathbb{R}^{nxk}$ be the matrix containing the vectors $u_1, ..., u_k$ as columns.
- For $i = 1, ..., n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,...,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, ..., C_k$.

OUTPUT: Clusters $A_1, ..., A_k$ with $A_i = j | y_j \in C_i$.

Normalized spectral clustering according to Shi and Malik (2000)

INPUT: Similarity matrix $S \in \mathbb{R}^{nxn}$, number of k of clusters to construct.

- Construct the similarity graph by one of the ways described in section 5.3.1. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- **Compute the first k generalized eigenvectors $u_1, ..., u_k$ of the generalized eigenproblem $Lu = \lambda Du$.**
- Let $U \in \mathbb{R}^{nxk}$ be the matrix containing the vectors $u_1, ..., u_k$ as columns.
- For $i = 1, ..., n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,...,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, ..., C_k$.

OUTPUT: Clusters $A_1, ..., A_k$ with $A_i = j | y_j \in C_i$.

Normalized spectral clustering according to Ng, Jordan, and Weiss (2002)

INPUT: Similarity matrix $S \in \mathbb{R}^{nxn}$, number of k of clusters to construct.

- Construct the similarity graph by one of the ways described in section 5.3.1. Let W be its weighted adjacency matrix.
- Compute the normalized Laplacian L_{sym}.
- **Compute the first k eigenvectors $u_1, ..., u_k$ of L_{sym}.**
- Let $U \in \mathbb{R}^{nxk}$ be the matrix containing the vectors $u_1, ..., u_k$ as columns.
- **Form the matrix $T \in \mathbb{R}^{nxk}$ from U by normalizing the rows to norm 1, that is set $t_{ij} = u_{ij} / \left(\sum_k u_{ik}^2\right)^{1/2}$.**
- For $i = 1, ..., n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of T.
- Cluster the points $(y_i)_{i=1,...,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, ..., C_k$.

OUTPUT: Clusters $A_1, ..., A_k$ with $A_i = j | y_j \in C_i$.

In all three algorithms, the main trick is to change the representation of the abstract data points x_i to points $y_i \in \mathbb{R}^k$. This is due to the properties of the graph Laplacians that make this change of representation useful. In particular, the simple k-means clustering algorithm has no difficulties to detect clusters in this new representation. In this work, I derived the similarty matrix by using the k-nearest method with $k = 5$ and $\sigma = 1$. Then, I computed the Laplacian and performed the eigendecomposition in order to apply the *Jordan and Weiss* algorithm.

5.4 Clustering indeces

In this section, I will present three indeces to evauate the performances to measure the quality of the clustering [19]: Calinski Harabasz, Davies Bouldin and Dunn.

5.4.1 Calinski Harabasz (CH)

Calinski Harabasz (CH) called sometimes variance ratio criterion (VRC), evaluates the cluster validity based on the average between and within cluster sum of squares. Well-defined clusters have a large between-cluster variance (SS_B) and a small within-cluster variance (SS_W):

$$SS_W = \sum_{i=1}^{k} n_i ||m_i - m||^2$$

$$SS_B = \sum_{i=1}^{k} \sum_{x \in c_i} ||x - m_i||^2$$

$$VRC = \frac{SS_B}{SS_W} \cdot \frac{(N-k)}{(k-1)}$$

where SS_B is the overall between-cluster variance (n_i= number of elements inside the cluster I, m_i=centroid of the cluster I, m= the overall mean of the sample data), SS_W is the overall within-cluster variance (x=data point), k is the number of clusters, and N is the number of observations.

5.4.2 Davies Bouldin (DB)

The Davies-Bouldin criterion is based on a ratio of within-cluster and between-cluster distances. For each cluster C, the similarities between C and all the other clusters are computed, and the highest value is assigned to C as its

cluster similarity. The index is obtained by averaging all clusters similarities. So, we are looking for the smallest index.

$$DB = \frac{1}{k}\sum_{i=1}^{k} max_{i \neq j} D_{i,j}$$

where $D_{i,j}$ is equal to:

$$D_{i,j} = \frac{(\overline{d_i} + \overline{d_j})}{d_{i,j}}$$

$\overline{d_i}$ is the average distance between each point in the i-th cluster and the centroid of the i-th cluster. $\overline{d_j}$ is the average distance between each point in the j-th cluster and the centroid of the j-th cluster. $d_{i,j}$ is the Euclidean distance between the centroids of the i-th and j-th clusters. The maximum value of $D_{i,j}$ represents the worst-case within-to-between cluster ratio for cluster i. The optimal clustering solution has the smallest Davies-Bouldin index value.

5.4.3 Dunn (DU)

It is an internal evaluation scheme, where the result is based on the clustered data itself. The aim is to identify sets of clusters that are compact, with a small variance between members of the cluster, and well separated. We are looking for the maximum value :

$$\Delta_i = max_{x,y \in C_i} d(x, y)$$

Calculates the maximum distance.

$$\frac{1}{|C_i||C_i - 1|} \sum_{x,y \in C_i, x \neq y} d(x, y)$$

Calculates the mean distance between all pairs.

$$\Delta_i = \frac{\sum_{x \in C_i} d(x, \mu)}{|C_i|}, \mu = \frac{\sum_{x \in C_i} x}{|C_i|}$$

Calculates distance of all the points from the mean.

Let:

$$\delta(C_i, C_j)$$

be the intercluster distance metric, between clusters C_i and C_j. With the above notation, if there are m clusters, then the Dunn Index for the set is defined as:

$$DI_m = \frac{min_{1 \leq i < j \leq m} \delta(C_i, C_j)}{max_{1 \leq k \leq m} \Delta_k}$$

5.5 Application to metro traffic and results

I used two different dataset to show the difference in choosing a correct input. The first one is the $dataset_2$ (defined in 4.4) as input for both of the methods. The clustering will be based even on the spatial information, in this way we are able to detect similar pattern of near BSs. We tried k number of clusters from 200 to 1500 with a step of 50. Following the results table:

Method with $dataset_2$	max(**CH**)	min(**DB**)	max(**DU**)	**K**
Kmeans	$6.83 \cdot 10^3$	0.43	0.021	[1500; 1500; 1500]
Spectral Clustering	1.1269	1.43	$4.31 \cdot 10^{-9}$	[450; 550; 1200]

As we can see in figures 5.10 5.11 5.12, the indeces does not converge, while in spectral clustering, figures 5.13 5.14 5.15 ,the first two indeces, such as CH and DB, converge respectively in K=450 and K=550.

The second dataset is composed by the value of $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ generated from the training of the prediction in section 4.2.1. The results in table and in figures 5.16 5.17 5.18 5.19 5.20 5.21, showed different values of convergence of the two methods.

Method with dataset of α_i	max(**CH**)	min(**DB**)	max(**DU**)	**K**
Kmeans	$2.76 \cdot 10^3$	0.32	0.07	[1500; 1500; 1500]
Spectral Clustering	236.18	1.08	0.0027	[200; 200; 350]

Clustering methods depend on the parameters of the model and on the type of the dataset used as input. For example, a dataset whose values depend on different features, reacts better to a grouping compared to the case where a similar situation is not present. Therefore, identify the characteristics that make the values you want to clustering is a fundamental step.

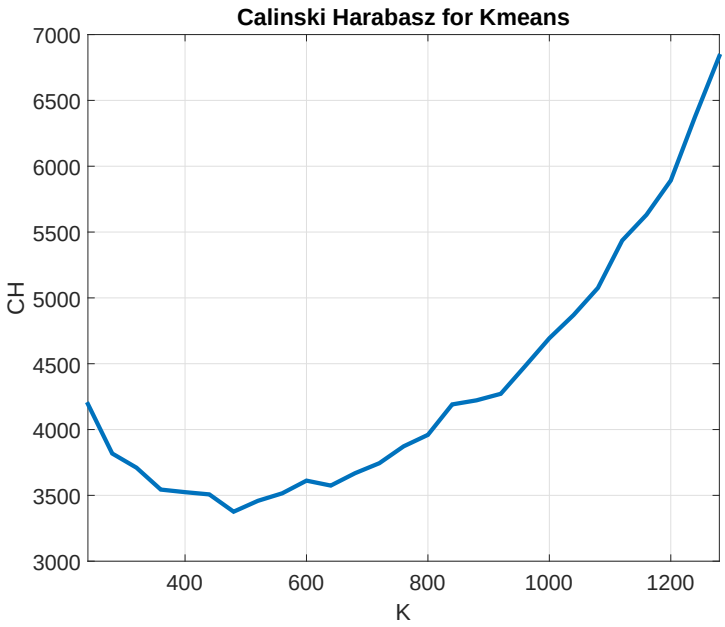


Figure 5.10: *Calinski Harabasz for Kmeans*

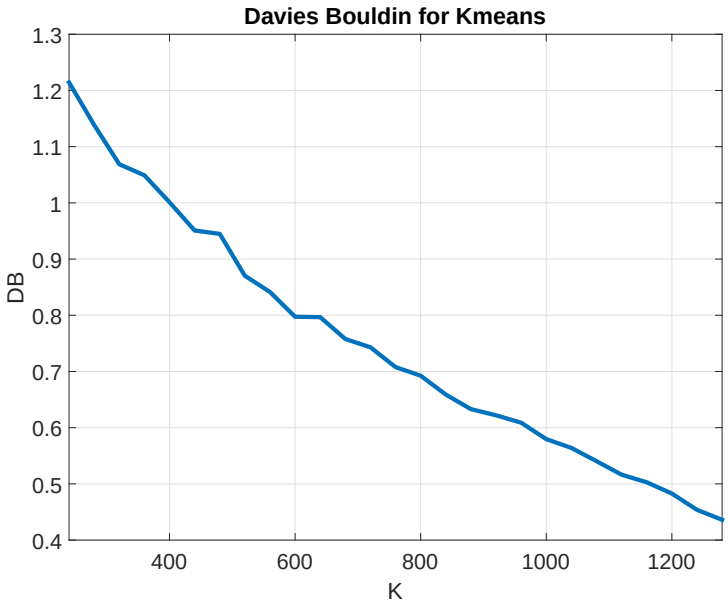


Figure 5.11: *Davies Bouldin for Kmeans*

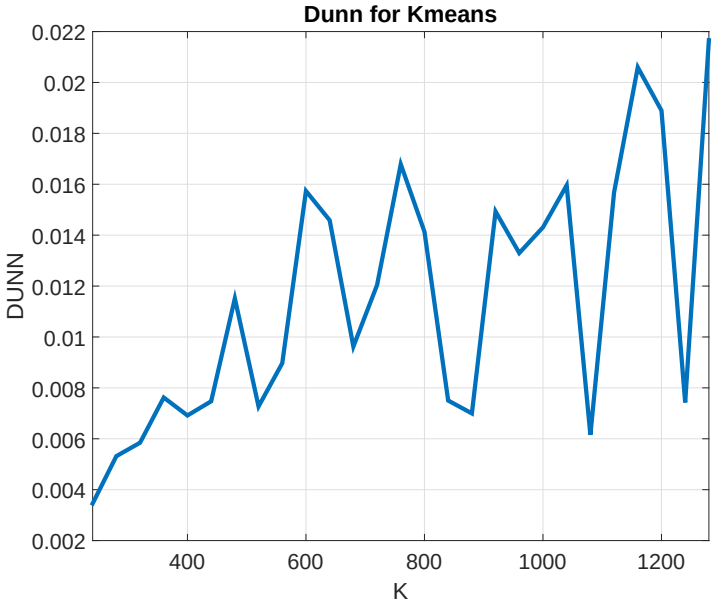


Figure 5.12: *Dunn for Kmeans*

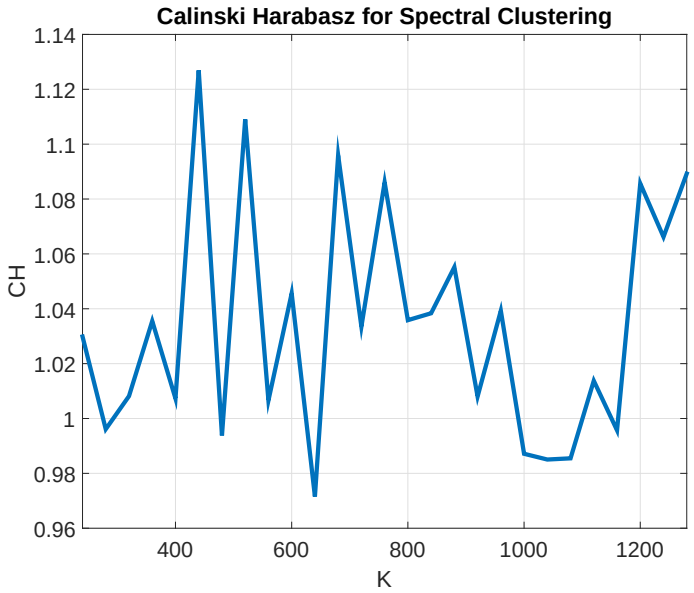


Figure 5.13: *Calinski Harabasz for Spectral Clustering*

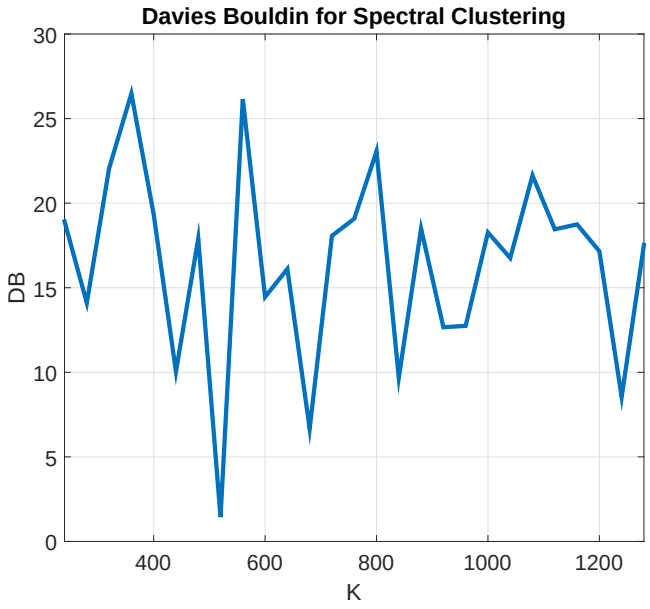


Figure 5.14: *Davies Bouldin for Spectral Clustering*

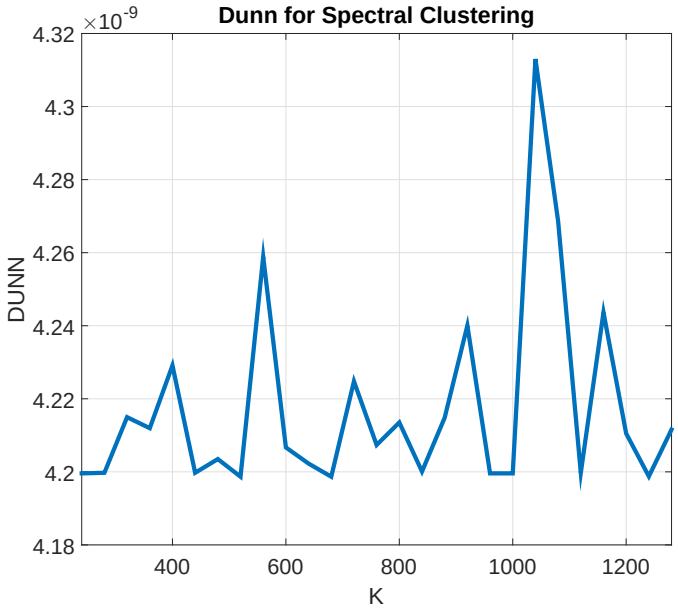


Figure 5.15: *Dunn for Spectral Clustering*

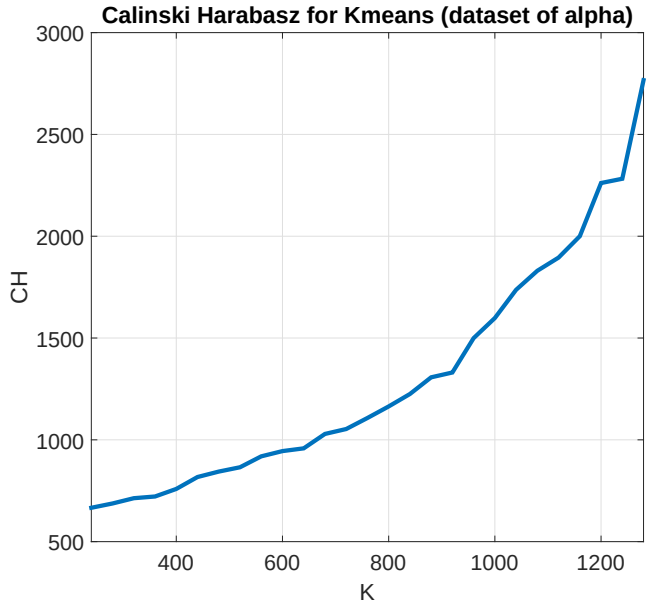


Figure 5.16: *Calinski Harabasz for Kmeans (dataset of α_i)*

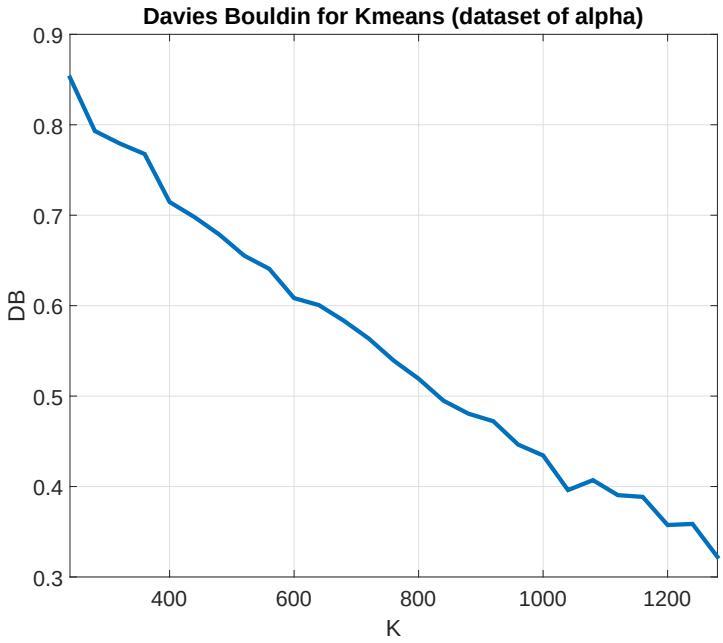


Figure 5.17: *Davies Bouldin for Kmeans (dataset of α_i)*

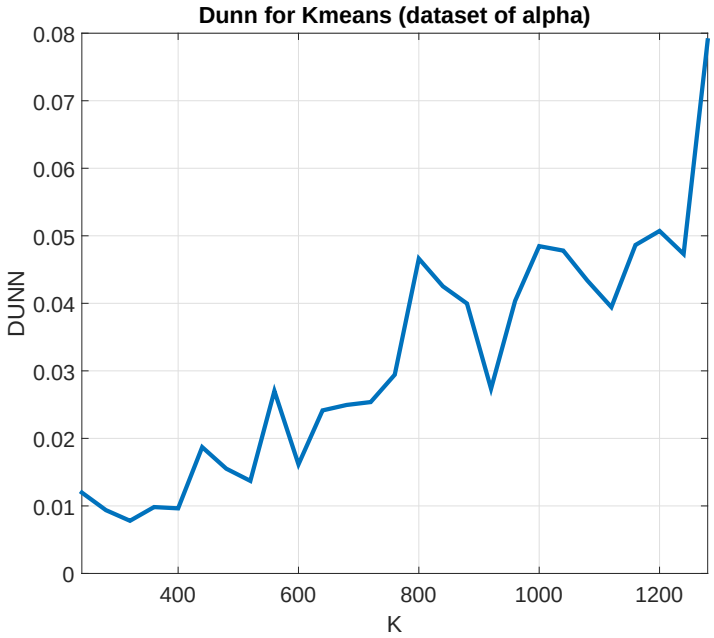


Figure 5.18: *Dunn for Kmeans (dataset of α_i)*

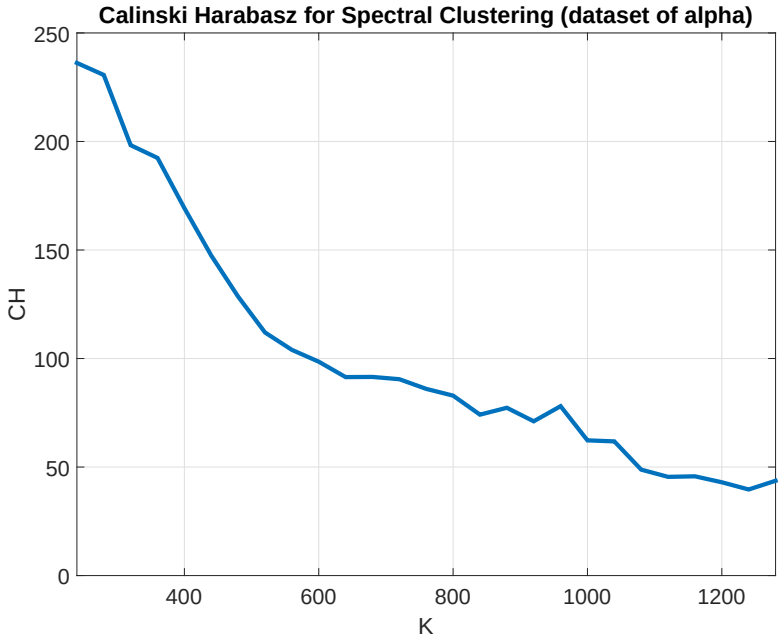


Figure 5.19: *Calinski Harabasz for Spectral Clustering (dataset of α_i)*

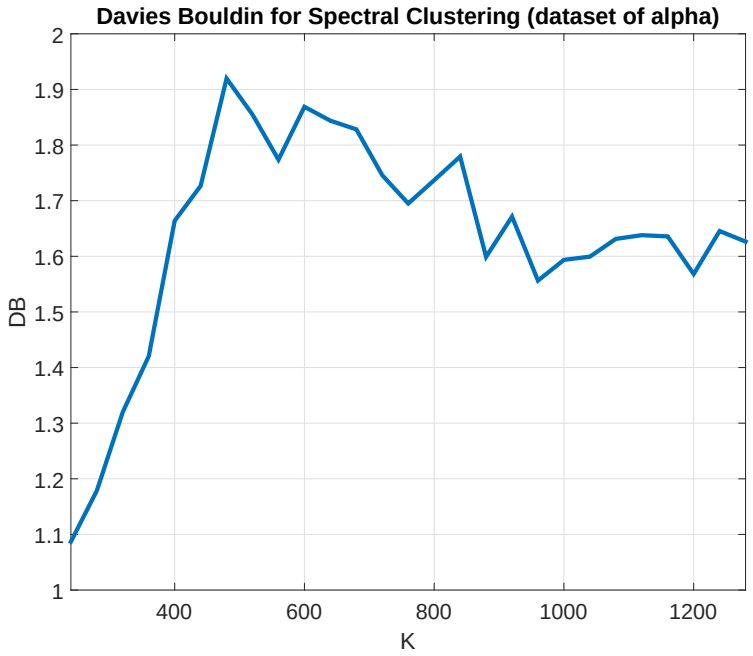


Figure 5.20: *Davies Bouldin for Spectral Clustering (dataset of α_i)*

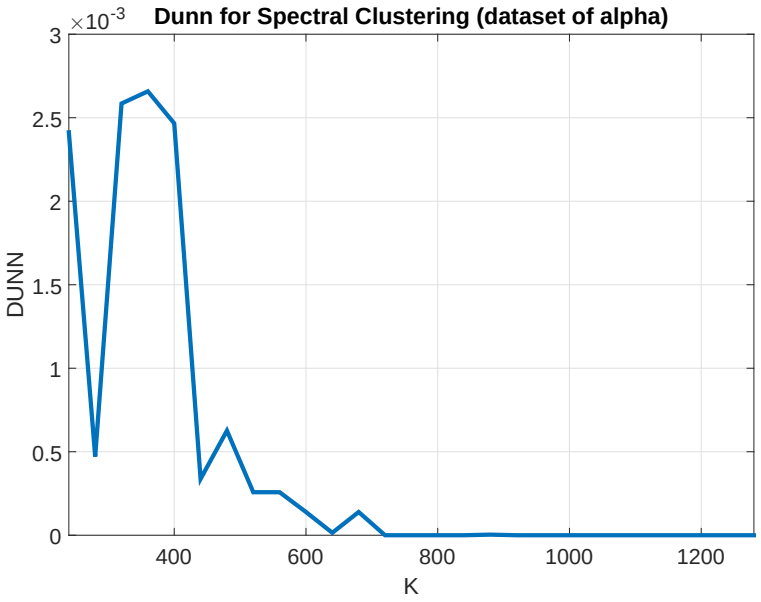


Figure 5.21: *Dunn for Spectral Clustering (dataset of α_i)*

Chapter 6

Pattern extraction

Due to the highly predictable daily movements of citizens in large urban areas [44], mobile traffic exhibits repetitive patterns with spatio-temporal variations. This behavior has been recently compared to the rise and fall of the sea levels, known as tides. Thus it was called the tidal-traffic scenario [45].

Recognizing and defining traffic patterns is a relevant information that can lead us finding an important stone in building a smart network. Along with the traffic prediction, it is possible to give the SDN controller the ability of optimization, allocating resources in response to the tidal traffic of the metro-core network.

Tidal traffic may create hot spots in a network moving in the spatio-temporal space, and following a regular pattern given by the human commutation from residential areas to working areas (academic, business, industrial, medical, governmental among others). Special events may change the tidal traffic, as for instance, maintenance, disasters, and social events. It has been shown that the daily variation of the number of users in a specific area of the city is very periodic, so that the regular traffic daily pattern can

be even used to identify the social function of the urban zone [46][47].

The goal of this chapter is to exploit a method in order to find hidden behavior, and try to answer to the following three questions: why, where and when a typical pattern is detected.

6.1 Non-Negative Matrix Factorization

Non-Negative Matrix Factorization (NNMF) is a set of algorithms in multivariate analysis and linear algebra where a matrix V is factorized into two matrices W and H, with non negative elements as hypothesis [21]. The non-negativity is a useful constraint for matrix factorization that can learn a partial representation of the data [48] [20]. The basic idea is to devide the matrix of observations V in a product of two matricies:

$$V = W \cdot H$$

The factorization of a given nxm matrix V, where m is the number of examples and n the dimension of the variable, is based on a parameter r, chosen to be smaller than n or m, so that $W[n\text{x}r]$ and $H[r\text{x}m]$ are smaller than the original matrix V. Each data vector v is approximated by a linear combination of the columns of W, weighted by the components of h. Therefore, W can be considered as a *basis* matrix where each vector is used to represent many original data vector. Following an easy example of NNMF:

$$V = \begin{bmatrix} 1 & 2 & 3 & 5 \\ 2 & 4 & 8 & 12 \\ 3 & 6 & 7 & 13 \end{bmatrix}$$

The rank of the matrix is 2, $r = 2$:

$$W = \begin{bmatrix} 1 & 3 \\ 2 & 8 \\ 3 & 7 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 2 & 0 & 2 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

As we said before, there is a set of algorithms able to generate a good approximation of the two matrices. Some of them are based on iterative updates of W and H. In section 6.1.1 we studied one of these algorithm.

6.1.1 NNMF algorithm

To find an approximate factorization $V \approx WH$, we first need to define a cost function that quantifies the quality of the approximation. Such a cost function can be constructed using a measure of distance between two non-negative matrices A and B. We can use the square of the Euclidean distance:

$$||A - B||^2 = \sum_{ij} (A_{ij} - B_{ij})^2$$

Another usuful measure is:

$$D(A||B) = \sum_{ij} \left(A_{ij} log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij} \right)$$

This last measure cannot be called "distance", because it is not symmetric in A and B, so we will refer to it as the "divergence" of A from B. It reduces to the Kullback-Leibler divergence, or relative entropy, when $\sum_{ij} A_{ij} = \sum_{ij} B_{ij} = 1$, so that A and B can be regarded as normalized probability distributions.

Now we will consider two alternative formulations of NNMF as optimization problems:

Problem 1 *Minimize* $||V - WH||^2$ *with respect to* W *and* H*, subject to the constraints* $W, H \geq 0$.

Problem 2 *Minimize* $D(V||WH)$ *with respect to* W *and* H*, subject to the constraints* $W, H \geq 0$.

There are plenty of techniques from numerical optimization that can be applied to find local minima. One of these is the gradient descent, but convergence can be slow. Other methods such as conjugate gradient have faster convergence, at least in the vicinity of local minima, but are more complicated to implement rather than gradient descent. The convergence of gradient based methods also have the disadvantage of being very sensitive to the choice of step size, which can be very inconvenient for large applications.

6.2 Application and results

I applied this method using two different input matrices. The first one is the $dataset_2$, and the second is the output of the clustering processing. Since NNMF needs to know the parameter r in advance, we tried $r \in [2, 30]$. Increasing the number of r we obtained more basic flows, but some of them are not useful because they belong just to few base stations. I wrote a code in MATLAB that extract the basic flows in an automatic way. The steps are the following:

1. Given the input dataset, apply the NNMF with a given number of r;
2. For each r I made the comparison of each basic flow with the original input by using the Pearson's correlation;

3. When the correlation was greater or equal to 90%, I stored the ID of the BS that matched the basic pattern;

4. In the end, I filtered the number of basic flows by the number of BS that matched in the previous step, and I set up this number to 5;

6.2.1 First input matrix

Figure 6.2 shows 8 basic flows with different percentage of presence in the network. The histogram in 6.1 shows the percentage of presence of each pattern.

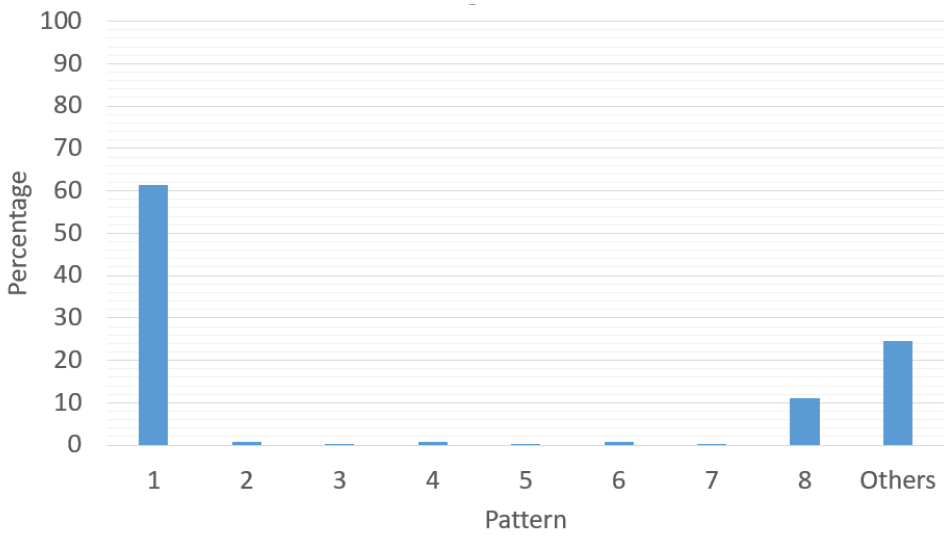
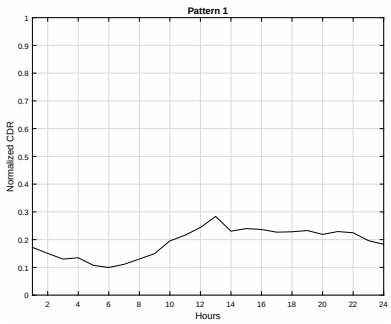
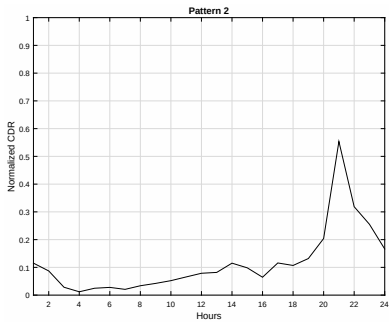


Figure 6.1: *Histogram representing the percentage of presence of the patterns generated from the $dataset_2$ in the network*

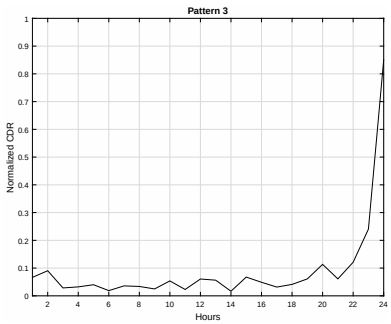
As we can see, pattern 1 and 8 are more present in the network than the others. This result is rather common, because most of the base station show a trend below the saturation point, and a bell-shaped trend during the hours of greater activity in the Internet network.



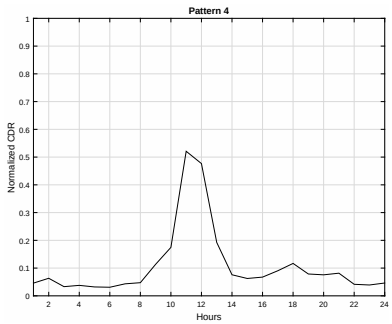
(a) `Pattern1`



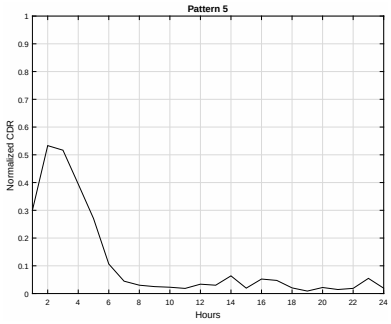
(b) `Pattern2`



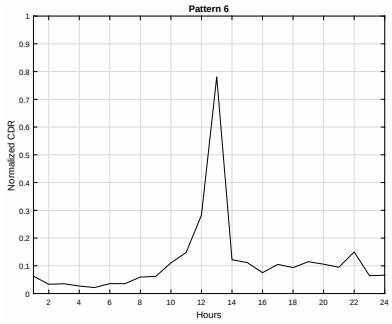
(c) `Pattern3`



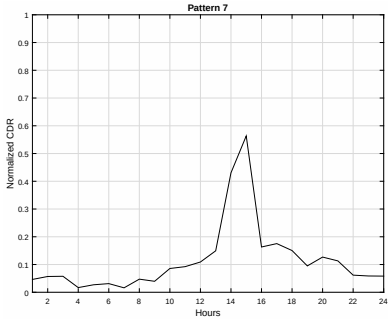
(d) `Pattern4`



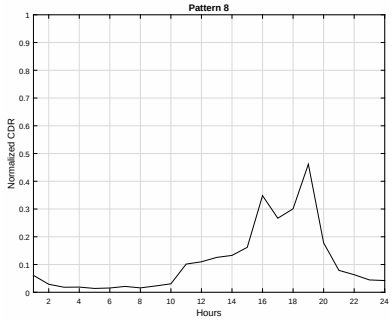
(e) `Pattern5`



(f) `Pattern6`



(g) `Pattern7`



(h) `Pattern8`

Figure 6.2: *Basic flows from* $dataset_2$

A direct example is the BS with ID = 1982 and ID = 429, they have the typical pattern in figure 6.2(e) and 6.2(c). Both figures present a peak detected during the night because of the nightlife that takes place in these particular areas of Milan.

Figure 6.3: *Area covered by the BS with a typical pattern showed in 6.2(e)*

Figure 6.4: *Area covered by the BS with a typical pattern showed in 6.2(c)*

6.2.2 Second input matrix

Figure 6.6 shows 8 basic flows with different percentage of presence in the network, almost similar to the previous case. The histogram in 6.5 shows the percentage of presence of each pattern.

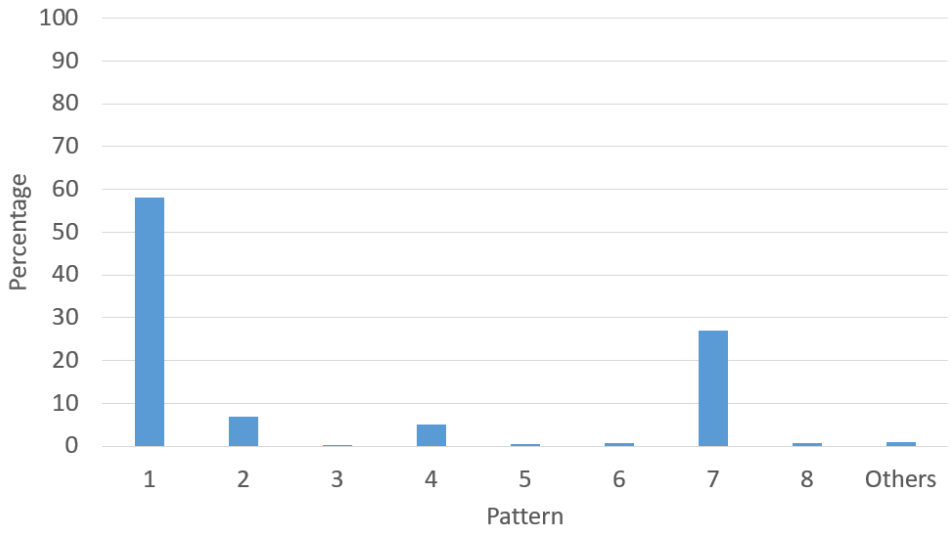
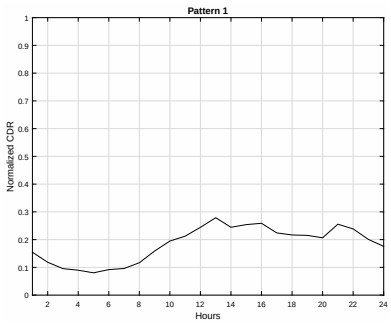
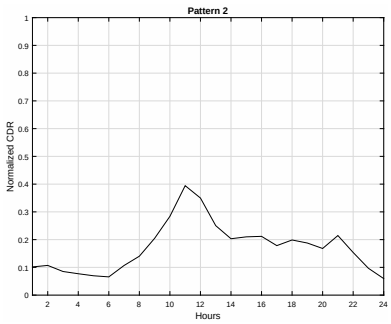


Figure 6.5: *Histogram representing the percentage of presence of the patterns generated from the clustering matrix in the network*

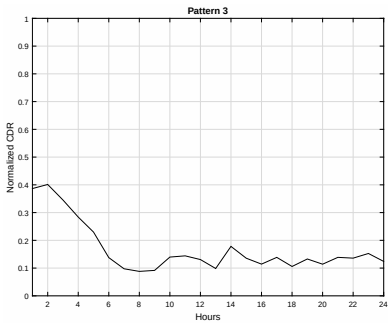
In this case, patter 1 and 7 are the most popular in the network. The two series of basic flows just found are very similar. What we did with the matrix obtained by clustering is a pattern research on the mean traffic load vector representative of each cluster. If the clustering is accurate we get clear patterns. With both methods we obtain the most famous pattern, with the difference that with the first one we can also see some particular case as in figure 6.2(b), 6.2(d) and 6.2(e).



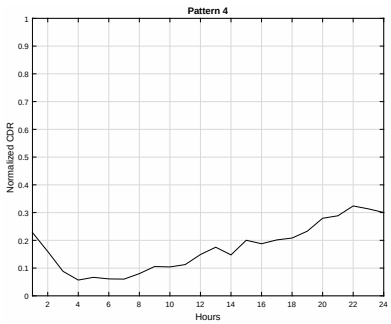
(a) Pattern1



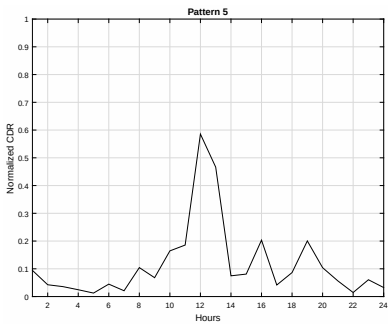
(b) Pattern2



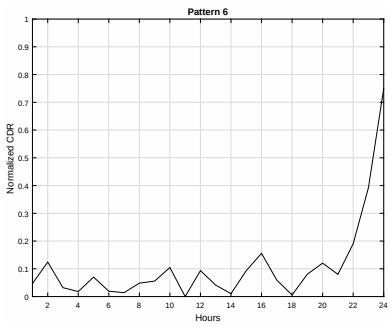
(c) Pattern3



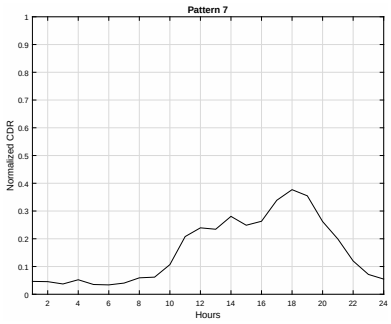
(d) Pattern4



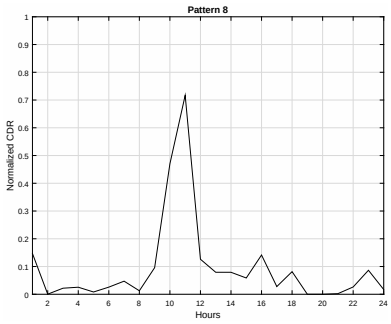
(e) Pattern5



(f) Pattern6



(g) Pattern7



(h) Pattern8

Figure 6.6: *Basic flows from the clustering matrix*

Chapter 7

Mobile metro-core network design and simulation

The high predictability in human mobility patterns is a valuable information to optimize resources allocation and increase energy efficiency by effectively adapting to expected traffic load variations [49]. In this chapter, we address the impact of the prediction in a metro area over the aggregated tidal traffic offered to the mobile metro-core networks. Tidal traffic affects todays networks, but predictable trends can be recognized even at the aggregated traffic level. We will see what dows off-line optimization means and online matheuristic to reduce the energy consumption in the optical layer of metrocore networks.

7.1 Overview

Detection of tidal trends in the traffic load of the mobile-network cells is a valuable knowledge that allows to improve the network resources management. The traffic prediction for every cell was used to activate and deacti-

vate or wake-up and sleep the network's cells to increase energy efficiency [50] [51]. Ref.[49] proposes two optimization methods to reduce the energy consumption in the optical layer of the metro-core network, by exploiting the predictable trends in aggregated traffic. Ref.[49] presents an on-line matheuristic that reuse the off-line results of regular tidal traffic patterns to improve optimality. The results show that energy savings of up to 47.5% can be achieved when comparing to a network statically dimensioned by considering traffic demand at peak hours. The role of the core network is to connect CNs to the SGW (see figure 7.1). Therefore, each aggregation ring set up a list of (bidirectional) connections with the SGW. The number of connections must be sufficient to transport at the SWG all the traffic received by the CN from aggregation ring (as well as all traffic from the SWG destined to the ring). Thus, the mobile metro-core network has to satisfy all up-link (UL) and down-link (DL) demands between aggregation rings and SGW. We assumed all these connections are provisioned with 1+1 protection scheme.

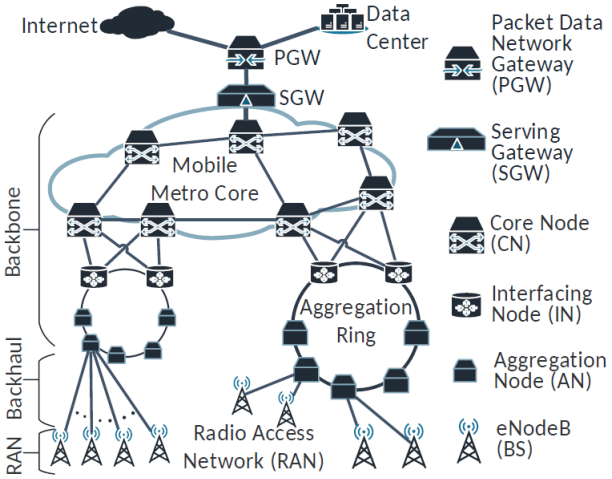


Figure 7.1: *Reference mobile carrier network (MCN) architecture.*

The common practice in MCNs is to perform a static resource allocation to meet the peak-hour demand. This method leads to poor energy efficiency, as outside the peak hour resources will be over-provisioned. By exploiting tidal traffic predictability, an operator can use optimization procedures to dynamically adapt the used resources to the actual hourly need, thus reducing energy consumption. Two mixed integer linear programming (MILP) models are proposed to minimize the energy consumption of the mobile metro core network (as depicted in figure 7.1) by activating and deactivating resources every hour based on the predicted component of the traffic demand.

Thanks to the traffic predictability, all optimization problems can be solved offline, and then the solutions found can be downloaded in the network at reconfiguration time points. The off-line planning problem consists in finding the set of paths that satisfy the spatio-temporal-dependent demand matrix of a specific time period t using 1+1 protection, with the objective of minimizing the energy consumption of the optical layer of the mobile metro-core network. In online dynamic bandwidth allocation, demand is represented by number of wavelengths. Each one of them is a request which should be assigned with a route and wavelengths along the path. Online operations are based on real time data which are not predictable. Each hour routing and wavelength assignement solution should be calculated for all requests. Solving ILP problems to get optimal solution is very time consuming. In fact, there is a tradeoff between optimality and time consumption by using heuristic algorithm to give suboptimal solution; Althought it gives suboptimal solutions but reduces the time consumption. Dealing with online issues using results of offline ILP model to guide real time RWA is the basic idea of this part. The simulator uses RWA results of

offline ILP model to calculate weights for network links of each request.

In this thesis we will show the results of the offline procedure applying the output of the forecasting. In [49] was assumed a perfect prediction and the off-line optimization was made by using the average traffic data over 24 hours. Here we will use respectively the prediction algorithms and the topology generated in chapter 4 and 3.

7.1.1 Offline bandwidth allocation

Offline bandwidth allocation can be designed beforehand because the input data are estimated 24 hour in advance. Solutions can be pre-calculated and just 24 times of calculations are needed. Optimal RWA solution for reconfiguration in each hour can be calculated by CPLEX with acceptable consumption of computation resource and time. In this way, ILP model should be built 24 times, one for each hour.

We applied the model using two path formulations for optimal resource allocation. A set of k candidate path pairs with 1+1 protection is pre-calculated for every demand in the network. Firstly, we computed the *virtual wavelength path* (VWP) model that uses full wavelength conversion at every node; Secondly, we assessed the *wavelength path* (WP) model which takes advantage of the small distances, because regeneration is normally needed after 1500 km for non-coherent wavelength channels at 10 Gbit/s [52]. We assume fully transparent lightpaths with optical bypass to avoid optical-to-electrical conversions in transit nodes (without wavelength conversion). To cope with unforeseen traffic increase (e.g., due to flash-crowd events), both formulations reserve 10% of the wavelength channels ($\epsilon = 0.1$) in each active fiber to allow flexibility upon unexpected traffic demand increases.

7.1.2 Mobile network topology

The overview of network topology is introduced by the figure 7.1. The MCN is composed by RAN (Radio Access Network), back-haul and backbone networks, in addition the backbone network consists of aggregation network and core network. RAN is a set of BSs connecting the mobile users to backbone network with back-haul links. Neighboring BSs connect to the same Aggregation Node (AN), and similarly neighboring ANs are connected with each other with ring topology. In aggregation ring, two ANs are selected as INs connecting to CNs. The core network is modeled by mesh topology containing CNs and SGW.

In the project, 2554 cells are considered, 10-12 neighboring cells connect to same AN, in order to have 8 rings. In core network, 17 nodes are placed with 16 CNs connecting to 8 aggreagtion rings and 1 SGW. We considered the gateway as the MIX in Milan to have a more realistic network. The topologies of back-haul network and aggregation network are generated under the cooperation with PRIS lab (BUPT), they provided the aggregation algorithm for grouping neighboring nodes. In figure 7.2 is showed the plot of the topology.

7.2 Results

We applied the offline procedure with the prediction obtained in 4.4.3, on the network topology in figure 7.2. The results are compared with the original $dataset_2$. Dynamic bandwidth allocation scheme minimizes the amount of active resources for network operations. Thus it optmizes power consumption of the network with respect to static scheme. To calculate the power consumption of network operations, models for optical component

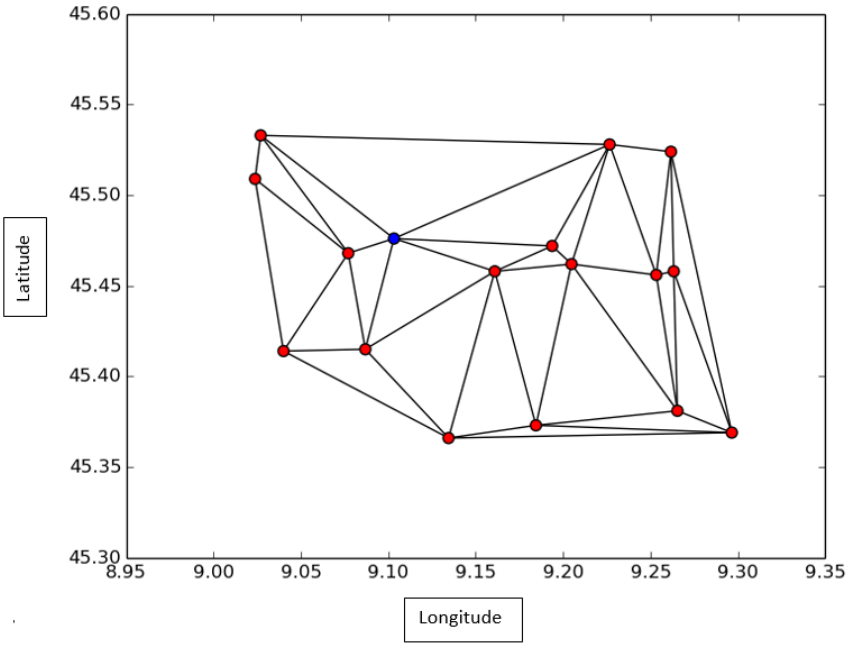


Figure 7.2: *Metro-core network topology*

power data are required. In [53], pratical models are given and in Table 7.1 are showed the power consumption information for bidirectional optical network facilities from a WDM network.

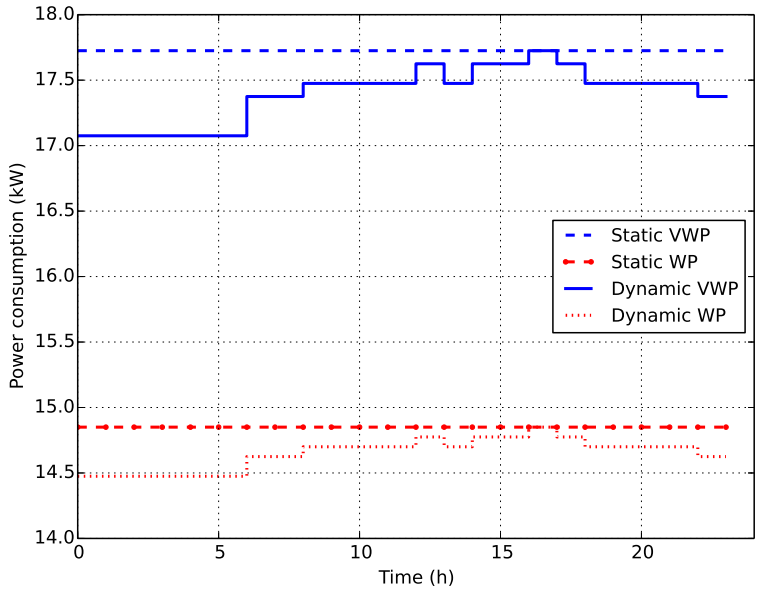
7.2.1 Results of offline procedure

In this section we show the offline operation with two set of data in order to see the defferences during the network planning. We will analize three days for each dataset, in particular 16-17-18 december 2013. The first dataset is the real one, figure 7.3, i.e. the real values of internet traffic derived by the bandwidth estimation in section 3.5. The second one is the predicted traffic loads generated in chapter 4, figure 7.5.

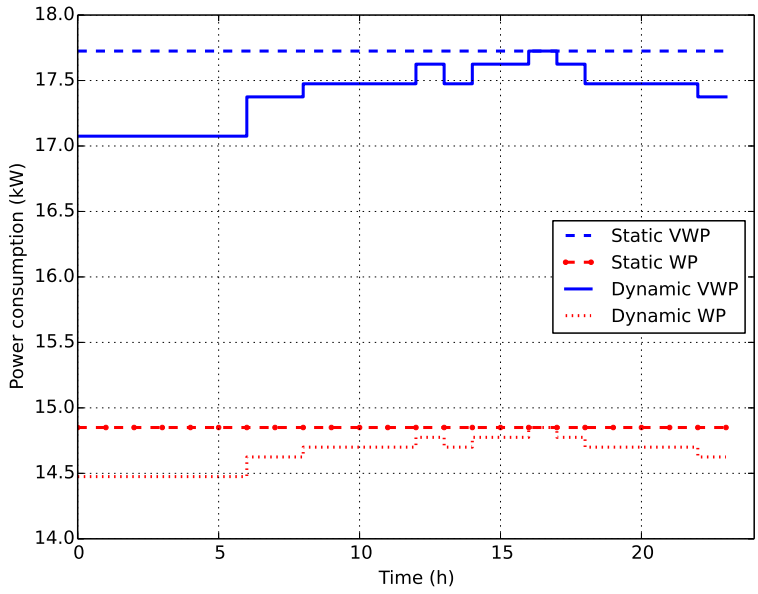
The two flat lines are static power consumption, and the other two step functions are dynamic power consumption curves. As we can see from the

Table 7.1: *Power consumption model*

Component	Power Consumption (WATT)
Transponder/muxponder, 2.5 G	25 W
Transponder/muxponder, 10 G	50 W
Transponder/muxponder, 40 G	100 W
OLA, short span 2 km	65 W
OLA, Medium span 40 km	65 W
OLA, Medium span 80 km	110 W
WDM terminal, 40 channels	230 W
WDM terminal, 80 channels	240 W
ROADM, 40channels, 100%	450 W
ROADM, 80 channels, 50%	550 W
ROADM, 80 channels, 100%	600 W
OXC, 40 channels	f*85 W + a*50 W + 150 W
OXC, 80 channels	f*85W + a*50 W + 150 W where f and a are the network side and add/drop bidirectional fiber ports respectively



(a) `RealData`



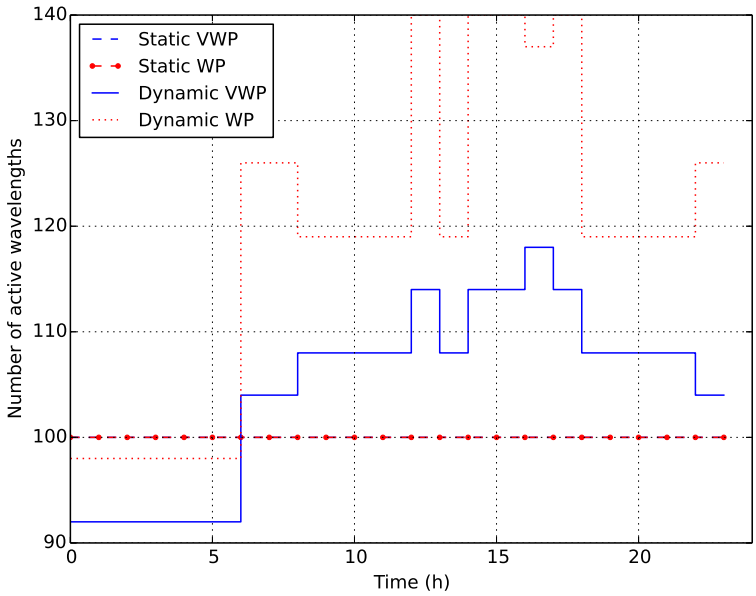
(b) `Predicted`

Figure 7.3: *Power consumption results of 16/12/2013*

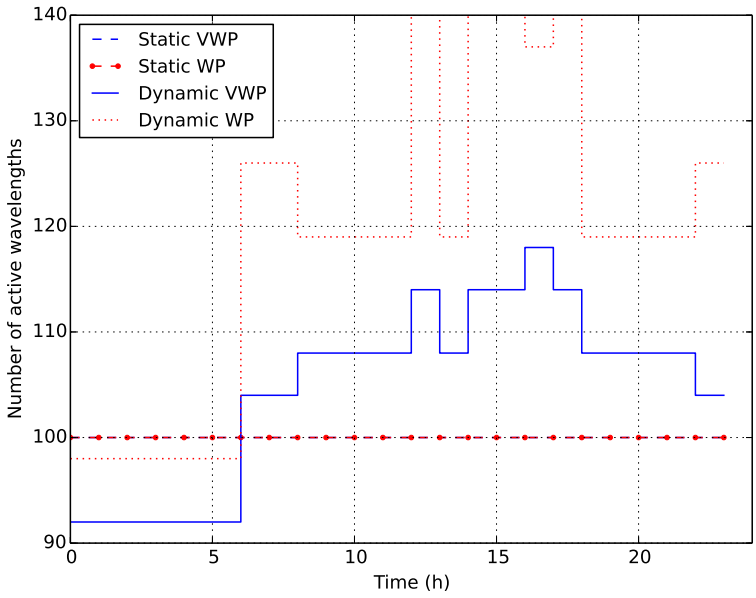
figures 7.3, there is no difference between the power consumption generated from the real data and from the predicted one. This means that the prediction error does not affect the planning of the power consumed.

In figures 7.4, numbers of active wavelengths in different conditions are presented. The difference between dynamic VWP and WP is considerable respect to the static case.

These results allow us to understand how the planning of network resources may be useful in terms of energy saving. In addition we can also get the reconfiguration points where it is possible upload the resource allocation rules.

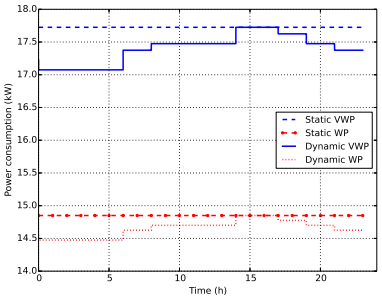


(a) `RealData`

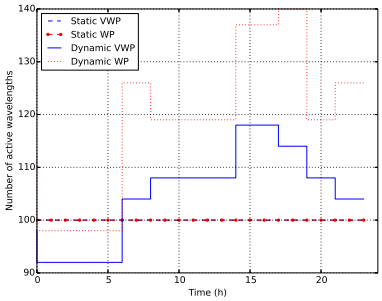


(b) `Predicted`

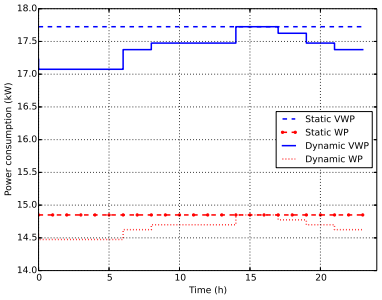
Figure 7.4: *Wavelength assignement results of 16/12/2013*



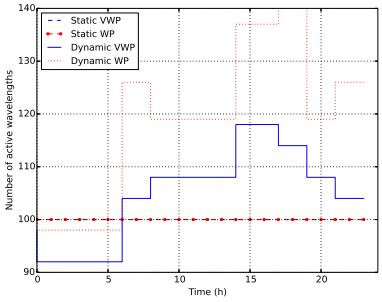
(a) RealData-17/12/2013



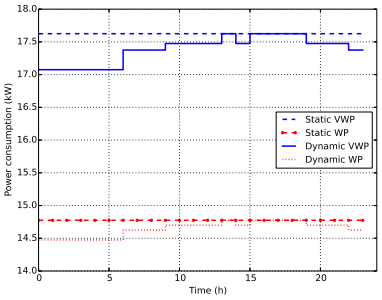
(b) RealData-17/12/2013



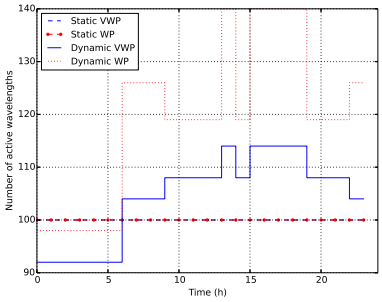
(c) Predicted-17/12/2013



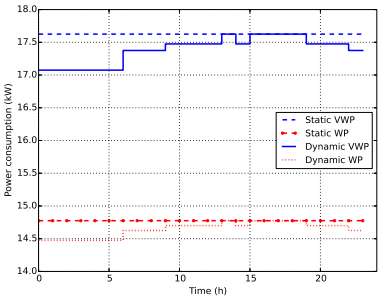
(d) Predicted-17/12/2013



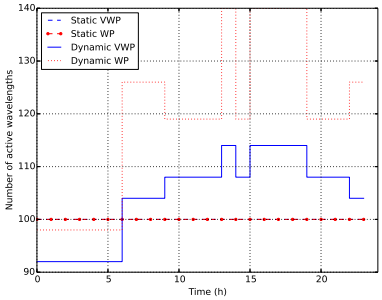
(e) RealData-18/12/2013



(f) RealData-18/12/2013



(g) Predicted-18/12/2013



(h) Predicted-18/12/2013

Figure 7.5: *Power consumption and wavelength provisioning results of 17-18/12/2013*

Chapter 8

Conclusions

In this work we focused on developing intelligent algorithms capable of predicting the traffic and be able to recognize typical patterns for each base station. The idea is to implement these algorithms in an SDN controller, giving the ability to the network to configure itself and to propose intelligent solutions based on future traffic demands.

In Chapter 4 we saw how the autoregressive model for the prediction can work even for processes that are not entirely stationary. The average error among cells varies between 18 and 24%, and it could be a good result. It is necessary to carefully choose the parameters of the model because of some matrices that have to be inverted, must therefore comply with the criterion of non singularity. Then, we explored an approach based on machine learning techniques, identifying four characteristics which determine the traffic load and weighing them by four parameters respectively. Finally, through an algorithm that iteratively generates the correct parameters, we obtained the prediction hour by hour, obtaining a lower error than the first method of 3-5%. As the last model we turned our attention to the neural networks. They are complex methods based on neuronal connections that carry out

learning operations of the process being studied. The latter method has proved to be very competitive because during the training phase establishes dynamic connections weights able to cover non-linear processes. Results led to good performances during the planning of the power consumption in the metro-core network. As we said at the end of Chapter 4, machine learning makes scalable methods and are presented as a good starting point for future developments. The search for patterns in Chapter 6, and clustering in Chapter 5 is an example of how we might reformulate the prediction problem based on typical trend of the traffic loads in the network. In addition, if the objective is the planning of resources, we can apply methods of classification rather than regression methods, making the prediction of a class of values rather than the value itself. A future work could be to apply Hidden Markov Models to the labeled dataset of the traffic loads, and discover the probability distribution of the network states. These methods could model the tidal effect of the traffic and bring us to a greater awareness of it.

In Chapter 5 we argued about if the clustering can define what are the social function of the base stations. In particular, we applied two methods: the first one is the Kmeans, one of the most famous and used in many different circumstances. The second one is the Spectral Clustering, a simple and powerful method that requires less assumptions about clusters. In general, the clustering results were not very satisfactory in terms of the social function definition. However, it was possible to see a connection between the cluster and the type of the traffic patterns in the network. In fact, we used the output of the clustering method as input for the algorithm of the basic traffic flows extraction. This work is still at an initial stage and that needs reliable database to define the tag of a base station. Being able to understand the behavior of a particular station, it could be an information

that can be used not only in the field of internet networks but also in urban planning and security. If we extend the research also to voice traffic and text messages we may find other information that with only traffic data can not be found.

In Chapter 6 we exploited an interesting method used in pattern recognition called Non Negative Matrix Factorization. It aims to decompose a matrix into a product of two smaller ones. The result of this operation defines the basis of the original matrix and then identify carriers who frequently make up the rows of the original matrix. In our case, finding the basis of the traffic matrix means identifying those that are the typical pattern in the network. This method greatly depends on the input dataset, if the size of the matrix is too large then approximation does not give a good decomposition. We applied this method to our dataset getting interesting results, getting 8 traffic flows including 3 most common and some not, but very significant.

As last work we used a simulator that emulate the resource assignment in the optical network. We have seen how the prediction error does not mine the results of network planning. In this work it was possible to do the simulation in offline mode. In the future, we can apply the prediction hour by hour in the online mode and schedule resources in subsequent network reconfiguration points, where the routing information and wavelength assignment are downloaded into the nodes of the network.

We believe that the contributions given on this work can be used to implement network analytics applications that use big data SDN programmability to create an artificial brain able to learn and make decisions.

Acronyms list

BS Base Station

CDR Call Detail Record

GSM Global System for Mobile Communications

UMTS Universal Mobile Telecommunications System

LTE Long Term Evolution

MCC Mobile Cloud Computing

MCN Mobile Carrier Network

IN Interface Node

AN Aggregation Node

RAN Radio Access Network

CN Core Node

SGW Service Gateway

DL DownLink

UL UpLink

MIX Milan Internet eXchange

WDM Wavelength Division Multiplexing

ILP Integer Linear Programming

RWA Routing Wavelength Assignment

MILP Mixed-Integer Linear Programming

VWP Virtual Wavelength Path

WP Wavelength Path

SP Spectral Clustering

DUSAF Destinazione d'Uso dei Suoli Agricoli e forestali

QGIS Quantum GIS

DTW Dynamic Time Warping

AR Auto Regression

MA Moving Average

ARIMA Auto Regressive Integrated Moving Average

NN Neural Network

ANN Artificial Neural Network

CH Calinski Harabasz

DB Davies Bouldin

NNMF Non Negative Matrix Factorization

Bibliography

[1] Mobile and wireless communications Enablers for the Twenty twenty Information Society (METIS). Channel models. ICT-317669-METIS/D1.4, 2015.

[2] Ericson. Ericson mobiliy report, 2015.

[3] Saeid Abolfazli, Zohreh Sanaei, Ejaz Ahmed, Abdullah Gani, and Rajkumar Buyya. Cloud based augmentation for mobile devices: motivation, taxonomies, and open challenges. *IEEE Communications Surveys & Tutorials*, 16(1):337–368, 2014.

[4] Fangming Liu, Peng Shu, Hai Jin, Linjie Ding, Jie Yu, Di Niu, and Bo Li. Gearing resource poor mobile devices with powerful clouds: architectures, challenges, and applications. *IEEE Wireless communications*, 20(3):14–22, 2013.

[5] Jaime G Carbonell, Ryszard S Michalski, and Tom M Mitchell. An overview of machine learning. In *Machine learning*, pages 3–23. Springer, 1983.

[6] Peter Russell, Stuart; Norvig. Artificial intelligence: A modern approach. 2003.

[7] R.J. Hyndman and Melbourne Australia. `http://otexts.org/fpp/`. Athanasopoulos, G. 2013.

[8] J. Mack Robinson College of Business Georgia State University University Plaza Atlanta GA 30303 USA G.Peter Zhang, Department of Management. Time series forecasting using a hybrid arima and neural network model. In *Neurocomputing*, volume 50, pages 159–175 vol.50, January 2002.

[9] Yantai Shu, Zhigang Jin, Lianfang Zhang, Lei Wang, and O. W. W. Yang. Traffic prediction using farima models. In *Communications, 1999. ICC '99. 1999 IEEE International Conference on*, volume 2, pages 891–895 vol.2, 1999.

[10] D. C. Park, M. A. El-Sharkawi, R. J. Marks, L. E. Atlas, and M. J. Damborg. Electric load forecasting using an artificial neural network. *IEEE Transactions on Power Systems*, 6(2):442–449, May 1991.

[11] Huifang Feng and Yantai Shu. Study on network traffic prediction techniques. In *Proceedings. 2005 International Conference on Wireless Communications, Networking and Mobile Computing, 2005.*, volume 2, pages 1041–1044, Sept 2005.

[12] Mingjin Yan. *Methods of determining the number of clusters in a data set and a new clustering criterion.* PhD thesis, Virginia Polytechnic Institute and State University, 2005.

[13] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584, 2001.

[14] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[15] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[16] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. 2005.

[17] FRBMI Jordan and F Bach. Learning spectral clustering. *Adv. Neural Inf. Process. Syst*, 16:305–312, 2004.

[18] Jacob Goldberger and Sam T Roweis. Hierarchical clustering of a mixture model. In *Advances in Neural Information Processing Systems*, pages 505–512, 2004.

[19] Evgenia Dimitriadou and Dolničar, Sara and Weingessel, Andreas. An examination of indexes for determining the number of clusters in binary data sets. *Psychometrika*, 67(1):137–159, 2002.

[20] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[21] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[22] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004.

[23] Chris HQ Ding, Xiaofeng He, and Horst D Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *SDM*, volume 5, pages 606–610. SIAM, 2005.

[24] Howard Michael. Using carrier ethernet to backhaul lte. Infonetics White Paper, 2011.

[25] Lange Christoph and Gladisch Andreas. Energy efficiency limits of load adaptive networks. In *Optical Fiber Communication Conference (OFC)*, page OWY2, 2010.

[26] F. Idzikowski, E. Bonetto, L. Chiaraviglio, A. Cianfrani, A. Coiro, R. Duque, F. Jimènez, E. Le Rouzic, F. Musumeci, W. Van Heddeghem, J. Lòpez Vizcaino, and Yabin Ye. Trend in energy aware adaptive routing solutions. *IEEE Commun. Mag.*, 51(11):94–104, November 2013.

[27] Xu Chao Long Qingliang, Chen Yufeng. Transmission bandwidth requirements calculations of lte access network. *Designing Techniques of Posts and Telecommunications*, 11:61–65, 2012. Document in Chinese.

[28] Deng Zerong. Study on the analysis and selection of lte bearer technology for guangdong unicom. 11:20–23, 2013. Document in Chinese.

[29] Ron Nadiv and Naveh Tzvika. Wireless backhaul topologies: Analyzing backhaul topology strategies. Ceragon White Paper, 2010.

[30] GreenTouch consortium. Greentouch green meter research study: Reducing the net energy consumption in communications networks by up to 90 percent by 2020. A GreenTouch White Paper, 2013.

[31] Zhichao Zhu, Guohong Cao, R. Keralapura, and A. Nucci. Characterizing data services in a 3g network: Usage, mobility and access issues. In *IEEE ICC*, pages 1–6, June 2011.

[32] `http://www.territorio.regione.lombardia.it/`.

[33] Meinard Muller. Dynamic time warping. In *Information Retrieval for Music and Motion*, pages 69–84 vol.50, January 2007.

[34] "LTE". 3GPP web site.

[35] IS-95 (CDMA) and GSM(TDMA). Retrived 2011-02-03.

[36] UMTS/HSPA (3G) Mobile Broadband Wireless from AT&T.

[37] `https://dandelion.eu/datamine/open-big-data/`.

[38] `https://www.coursera.org/learn/machine-learning`.

[39] WH Press et al. Book review: Numerical recipes in fortran: the art of scientific computing/cambridge u press. *The Observatory*, 113:214, 1993.

[40] J.C. MacKay, David. Information theory, inference, and learning algorithms. pages 467–492, 2003.

[41] `http://it.mathworks.com/discovery/load-forecasting.html`.

[42] Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

[43] Bojan Mohar. Some applications of laplace eigenvalues of graphs. In *Graph symmetry*, pages 225–275. Springer, 1997.

[44] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert Làszlò Barabàsi. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[45] Z. Niu. Tango: traffic aware network planning and green operation. *IEEE Wireless Commun.*, 18(5):25–29, October 2011.

[46] Jing Yuan, Yu Zheng, and Xing Xie. Discovering regions of different functions in a city using human mobility and pois. In *ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pages 186–194, 2012.

[47] Yajing Xu, Gongfu Li, Chao Xue, Angen Luo, and Yizhe Song. Affinity-based human mobility pattern for improved region function discovering. *The Journal of China Universities of Posts and Telecommunications*, 23(1):60–67, 2015.

[48] Daniel D Lee and H Sebastian Seung. Unsupervised learning by convex and conic coding. *Advances in neural information processing systems*, pages 515–521, 1997.

[49] G. Maier Y. Xu R. Alvizu, X. Zhao and A. Pattavina. Energy efficient dynamic optical routing for mobile metro-core networks under tidal traffic patterns. *submitted for publication to IEEE/OSA Journal of Lightwave Technology.*

[50] Chunyi Peng, Suk-Bok Lee, Songwu Lu, Haiyun Luo, and Hewu Li. Traffic-driven power saving in operational 3g cellular networks. In *Int. Conf. Mobile Computing and Networking*, pages 121–132, 2011.

[51] L. Budzisz, F. Ganji, G. Rizzo, M.A. Marsan, M. Meo, Yi Zhang, G. Koutitas, L. Tassiulas, S. Lambert, B. Lannoo, M. Pickavet,

A. Conte, I. Haratcherev, and A. Wolisz. Dynamic resource provisioning for energy efficiency in wireless access networks: A survey and an outlook. *IEEE Commun. Surveys Tuts.*, 16(4):2259–2285, Oct-Dec 2014.

[52] Ward Van Heddeghem, Filip Idzikowski, Willem Vereecken, Didier Colle, Mario Pickavet, and Piet Demeester. Power consumption modeling in optical multilayer networks. *Photonic Network Commun.*, 24(2):86–102, 2012.

[53] Ward Van Heddeghem, Filip Idzikowski, Willem Vereecken, Didier Colle, Mario Pickavet, and Piet Demeester. Power consumption modeling in optical multilayer networks. *Photonic Network Communications*, 24(2):86–102, 2012.