



UNIVERSIDAD PONTIFICIA COMILLAS  
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

OFFICIAL MASTER'S DEGREE IN THE  
ELECTRIC POWER INDUSTRY

Master's Thesis

**HYBRID MODEL BASED IN DEEP MACHINE  
LEARNING WITH DATA MINING  
APPLIED TO  
TIME-SERIES AND  
DECISION SUPPORT**

**Author:**

**Roberto Martín González**

**Supervisor:**

**Valentin Garcia Rodriguez**

**Madrid, July, 2015**

## Master's Thesis Presentation Authorization

THE STUDENT

Roberto Martín González



THE SUPERVISOR

Valentín García Rodríguez



01/07/2015

Authorization of the Master's Thesis Coordinator

Dr. Javier García González

Signed:

Date: / /



A todos los que han hecho esto posible, gracias



WE'RE NOT THAT MUCH SMARTER THAN WE  
USED TO BE, EVEN THOUGH WE HAVE MUCH  
MORE INFORMATION

AND THAT MEANS THE REAL SKILL NOW IS  
LEARNING HOW TO PICK OUT THE USEFUL  
INFORMATION FROM ALL THIS NOISE.

Nate Silver



Empathy

Reciprocity

Progress and improvement

The tool offered to us by Machine Learning along with the Big Data revolution, will provide us medium and long-term support for a countless number of applications.

Similar to when the light was only used to illuminate or internet to send text messages. For a drastic evolution of ML is necessary that the support, hardware and software improvements as well as more, better and different data continue to evolve on the already started paths.

Until recently the vast amount of unstructured data hindered its analysis. This has been reduced due to the improved AI approaches and the great reduction of computing and memory cost. This has led to develop specialized optimization techniques and demonstrate their application in hundreds of fields, from hard and soft-science, to lines of existing business or ventures to explore.

Being able to teach a machine to be much better than yourself in many tasks, not only will give us plenty of options but also challenges and opportunities. Even more so if the ability of these systems is expected to grow at a geometric rate over the coming decades.

The risks, ethical and social dilemmas that the new generation of machines will make us face, will ultimately define part of the progress of our society.

In this document we demonstrate the great advantages of these tools applied to various problems related to energy and water. Starting with the basic theory, we'll continue developing and comparing models of varying complexity. Specifically prediction systems, support decision and risk-minimization under an automatic and deep analysis.

## Table of Contents

1. Introduction .....	9
1.1 Motivation.....	9
1.2 Objectives.....	11
1.3 Thesis Organization.....	12
2. Machine Learning.....	18
2.1 Introduction .....	18
2.1.1 Typologies .....	18
2.2 Theory .....	20
2.2.1 Bias-variance tradeoff .....	20
2.2.2 Computational learning .....	21
2.2.3 Risk minimization .....	21
2.2.3.1 Empirical.....	23
2.2.3.2 Structural .....	23
2.2.4 Probably approximately correct learning .....	24
2.2.5 Statical learning.....	24
2.2.6 Vapnik-Chervonenkis (VC theory and dimensions).....	25
2.3 Limitations and Strategies.....	26
2.3.1 Statistical classification .....	26
2.3.2 Cluster analysis.....	27
2.3.3 Regression analysis .....	28
2.3.4 Anomaly detection.....	29
2.3.5 Association rules .....	29
2.3.6 Behavior psychology .....	30
2.3.7 Structured prediction.....	32
3. Variable Selection (IVS).....	33
3.1 Introduction .....	33
3.2 Function complexity.....	33
3.3 Dimensionality .....	34
3.4 Noise .....	34
3.5 Data kinds and Linearities .....	34

4. Unsupervised Learning.....	36
4.1 Introduction .....	36
4.2 Bayes theorem and interpretation .....	37
4.3 Competitive learning.....	39
5. Supervised Learning .....	40
5.1 Introduction .....	40
5.2 Artificial Neural Networks.....	40
5.2.1 Introduction .....	40
5.2.2 Differences .....	41
5.2.3 Evolution .....	42
5.2.4 Characteristics.....	43
5.2.5 Learning.....	43
5.2.6 Properties.....	44
5.2.7 Criticisms and counterexamples .....	45
5.2.8 Types .....	46
5.2.8.1 Radial basis function .....	46
5.2.8.2 Self-Organizing Map (SOM).....	47
5.2.8.2.1 k-Means Clustering .....	48
5.2.8.3 Recurrent Networks .....	49
5.2.8.4 Boltzmann machine .....	49
5.2.8.5 Reservoir Computing .....	50
5.3 Online Machine Learning .....	52
5.4 Hidden Markov Models and Bayesian Networks.....	53
5.4.1 Baum-Welch algorithm and Markov property.....	55
5.5 Bagged Trees.....	58
5.6 Multiple Linear Regression .....	59
5.6.1 Stepwise regression .....	62
6. Neuronal Network Development.....	63
6.1 Introduction .....	63
6.1.1 How a Neuron Works.....	64
6.1.1.1 Neuron .....	64
6.1.1.2 Transfer Functions .....	64

6.1.1.3 Vectors as Inputs.....	65
6.1.2 Network Architectures.....	66
6.1.2.1 One layer.....	66
6.1.2.2 Multiple layers.....	67
6.1.3 Concurrent and Sequential examples.....	67
6.1.3.1 Static Networks.....	68
6.1.3.2 Dynamic Networks.....	68
6.2 Multilayer Neuronal Networks.....	69
6.2.1 The importance of Inputs.....	69
6.2.1.1 Saturation of the Input Channels.....	69
6.2.1.2 Overfitting and Subsets.....	70
6.2.2 Gradient and Jacobian development and performances.....	70
6.2.2.1 Development.....	71
6.2.3 Training Algorithms.....	74
6.2.3.1 Incremental Training.....	74
6.2.3.2 Batch Training.....	75
6.2.3.3 Functions.....	75
6.2.4 Analyzing and Improving the Network.....	77
6.2.4.1 Performance Function.....	79
6.2.4.2 Bayesian Regularization.....	80
6.3 Dynamic Neuronal Networks.....	80
6.3.1 Introduction.....	80
6.3.2 Time-Delay Structures and Training.....	83
6.3.3 NARX Time Series.....	84
6.3.4 Multistep Prediction.....	86
6.3.5 Radial Basis Networks.....	88
6.4 Improvements.....	89
6.4.1 Computing.....	89
6.4.1.1 Parallel Computing.....	90
6.4.1.2 Distributed Computing.....	90
6.4.1.3 GPU Computing.....	91
6.4.1.4 Hardware Machine Learning & FPGA vs Cloud computing.....	92



6.4.2 Bootstrap aggregating.....	93
6.4.3 Pruning functions and Evolutionary Algorithms .....	94
7. Comparisons .....	96
7.1 Input selection and preprocessing.....	96
7.1.1 Factors and data selection .....	96
7.1.2 Initial preprocessing.....	101
7.1.3 Clustering and dataset division.....	103
7.1.4 Feeding the data and deeper inputs analysis focus in NN.....	107
7.2 Results.....	110
7.2.1 ARIMA .....	110
7.2.2 Multiple linear regression .....	112
7.2.3 Bagged Decision Trees .....	115
7.2.4 Neuronal Networks.....	116
7.2.4.1 Introduction .....	116
7.2.4.2 Number of neurons.....	118
7.2.4.3 Delays and feedbacks.....	119
7.2.4.4 Preprocessing, variables and simple divisions .....	121
7.2.4.5 Common Learning algorithms.....	123
7.2.4.6 Simple supervised clustering .....	124
7.2.4.7 Unsupervised clustering and SOM.....	125
7.2.4.8 Radial Basis and k-means Classification.....	128
7.2.4.9 Shallow vs Deep Architecture and customized connections .....	129
7.2.4.10 More configured learning algorithms.....	131
7.2.4.11 Integration in decision support.....	133
7.2.4.12 Results.....	135
8. Conclusions .....	137
8.1 Achievements and limitations .....	137
8.2 Further developments and research .....	138
8.2.1 Energy Industry Applications .....	138
8.2.2 Business Intelligence and Learning Intelligent Optimization (LION & RSO) .....	138
8.3 Open tools and libraries.....	140
Annexes.....	142

List of Figures .....	148
List of Tables .....	150
References .....	150

# 1. Introduction

## 1.1 Motivation

The main incentive to spend more than a year self-learning this topic has been the clear conviction about its future, and the desire to understand and apply its known benefits to current issues, either professionally or for personal interest.

Another important aspect has been the curiosity to grasp the problems in the development, limitations and real aspirations of these models. In order to apply them in various projects, either software or hardware made in the past, where these models would have meant a qualitative progress in many aspects, or could have significantly expanded the range of applications, or in the future. It is worth to say that 90% of the scientists in this field believe that before 60 years, intelligence of machines will be superior to that of humans, with all the risks, dilemmas, problems and consequences of all kinds that it entails much before reaching these capabilities. (FutureOfLife, 2014) (Bostrom, 2014)

The integration of these systems either directly in hardware, software or via internal cloud services for applications, devices or robotics will suppose an increment in the quality of findings and decisions made in all fields of the "hard and soft science" and business. (Howard, 2014)

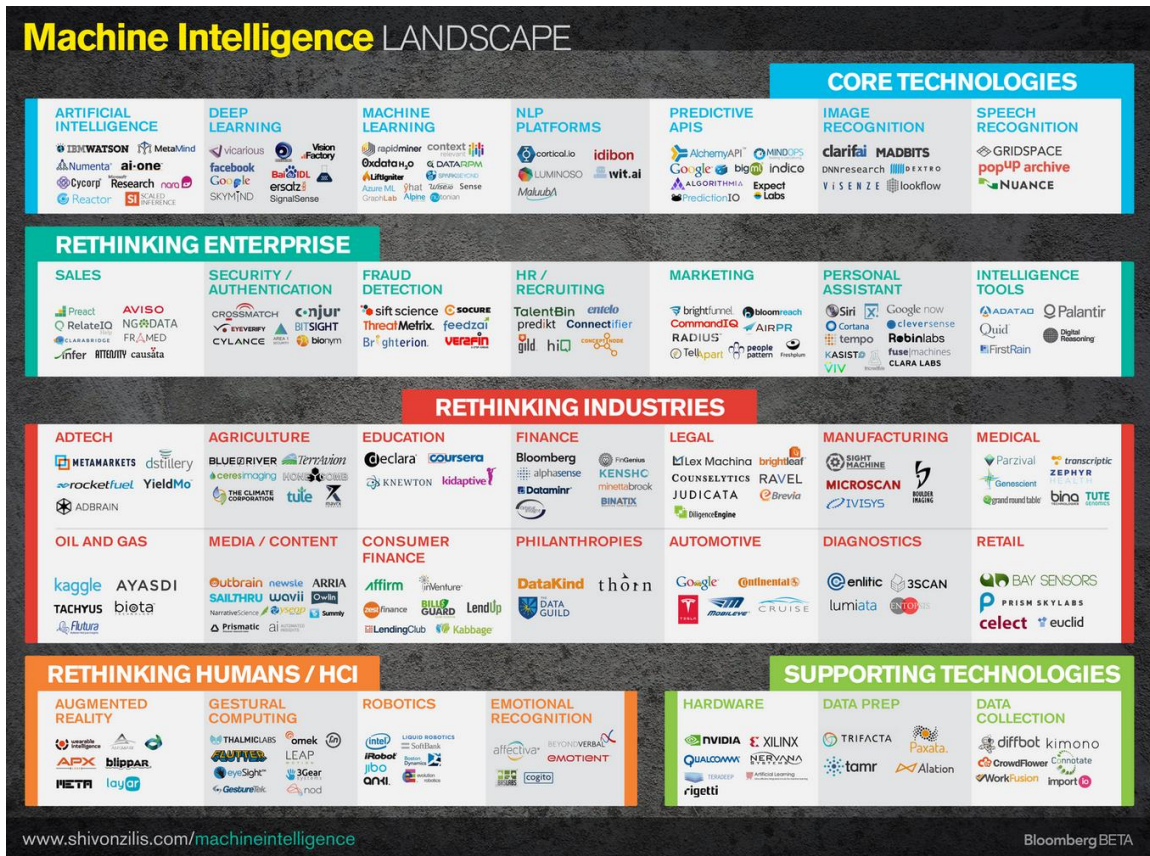
With the foundations granted by the vast amount of data that is generated unconsciously and gets collected and analyzed throughout the years. This new scope, or tools available to the new society, if applied to its progress, can help us evolve while providing us with aspects that were unthinkable years ago. (Norvig, 2012) (Markram, 2009) (Kurzweil, 2005)

Making more complex, deeper and better grounded strategies is a direct application of the union of gigantic databases with software expected to reach super intelligence and even virtues of a Turing machine in the next decade. (Bloom, 2014) (Brynjolfsson, 2013)

Therefore, the final decision-making is up to us, with ranges of confidence about forecast, assumptions or even future scenarios outside historical data. This is the reason why the scope of this document has been oriented towards decisions, either automatically or manually taken, with information that has been ultimately predicted.

For such purposes and to learn a little of everything, both types of machine learning have been used, without discussion whether it is considered a statistical model or not. Unsupervised learning will be responsible for the definition of scenarios and patterns and supervised learning will guide the final objective. Other approaches and algorithms, much better in many aspects and considerably more optimized, have been studied. However, their scope and development falls outside the time available for this document.

Applying all the theory, the expected progression and joining personal tastes with real problems of the industry that feeds and will feed such progress or not, it is surprising not to see utilities or companies directly related to the electricity sector and water, either in regulation, supervision, generation, supply, transmission, distribution or marketing being more firmly involved in these developments. We are on the threshold of a new generation, as was the light or the Internet. One change is the large amount of data available and the most effective tool to take advantage of these changes, to which must be added the hardware and software is machine learning. (Siemens, 2015) (SAS, 2013)



Given the numerous benefits and advantages they can offer at all stages affecting the electric power industry and water they should be exploited. While many of the approaches studied have existed for decades, only in recent years when have they begun to bloom in a practical manner. (SEO, 2015) Which turns these technologies into useful tools to increase profitability, as Philip Evans defends in the foreseeable future will shift from vertically integrated to horizontal business lines. “The plummeting of transaction costs weakens the glue that holds value chains together, and allows them to separate. The polarization of scale economies towards the very small allows for scalable communities to substitute for conventional corporate production.” (Evans, 2013)

This document tries, attempts to demonstrate that it is possible to take advantage of all the information available to these companies and that different machine learning approaches

have shown great results in almost any discipline and goal, compared with traditional algorithms and methodologies.

## 1.2 Objectives

*The main objective of this document, and the non-written work that has been needed, will be to learn about machine learning and the necessary about data mining and intelligent optimization in order to improve the model. Identify and understand with which methodologies and algorithms are being achieved the best results mainly oriented to the short-term and midterm load forecasting; without forgetting that recognition of scenarios, decision support or risk management are key aspects in Electric Power Industry.*

With the intention of no limiting this study only to forecasting, and taking advantage of the time spent in gaining a good foundation that could be exploited beyond the realization of this document, a significant amount of the work would be dedicated to understand the theoretical basis and the current limitations of machine learning. This veteran discipline whose foundation lies in statistics, mathematics and programming with certain aspects of optimization and artificial intelligence certainly will be one of the pillars of the next revolution that mankind will live, therefore it becomes a necessity for a sector as essential as electricity or water, to accompany, prove and apply its advantages.

Given the stage of this topic and its continuous improvement it is necessary to find, understand and carefully analyze the new and different approaches that the scientific community is proposing. They have surpassed previously existing limitations as well as increasing the success rate, thanks to new models that combine algorithms with different properties to deal more efficiently with specific problems or simply to obtain more accurate results. These methodologies, as well as the latest developments that are taking place, will be applied to various real problems directly related to the Electric Power Industry.

Defining a problem, predicting load curves in the short term and with the intention of demonstrating the improvements of these developments regarding the methodologies previously used by the industry in at least one practical case, several models will be developed and compared. The ARIMA model will be taken as a reference since it is the most widely used by various public utilities and generally achieves very good results in the estimation of load curves.

The other models developed and compared at this stage must meet various requirements such as being applicable to aspects of the electricity sector, like forecasting, decision support and risk management. They have to demonstrate near state-of-arts results in their respective applications in recent years. Initially, they should cover different approaches with several fundamental properties and limitations if possible. They should be able to be developed in a practical way with free software and/or completely open source. Greater

incentive will enable the implementation of successful improvements in a way that does not require too much time. They must have room for enhancement both in accuracy and optimization as in the range of fields in which they can be applied. Finally the results should provide measurable, comparable improvements and information that could be applied in real cases in the industry.

After this comparison, the model that offers more capacity for improvement will be studied in depth and will be in this one where we'll try with greater emphasis to apply the recent advances, in configuration as well as the essential factor of integration with other methodologies, both prior and subsequent treatment of data and its efficiency, analysis and application.

The numerical goal which will validate the developed model will be to get a lower rmse than the one achieved with a properly defined ARIMA model or a  $rmse < 2.9\%$ , in case lower values are obtained.

Once the main objectives have been checked and fulfilled, it is desirable to be able to integrate this model with other tools foreseeing its automatic exploitation and generalization. The model must be able to adapt to small variations in the requirements without any or with the minimum necessary variations.

If needed, the proposed procedure can be written in other languages that use it or connect it with different modules or applications in order to obtain greater adaptability and better efficiency and integration.

A great reward would be the ability to adapt the model in question to different aspects of the Electric Power Industry and obtain results that encourage them to continue working in directly related topics. Given the foundation that will be obtained during the first stages, it is expected to achieve interesting tools with a considerable range of utilities and improvements.

Specific modules and libraries for Matlab, Python and R will be used. Other services have been used to try to reduce the hardware limitation, mainly cloud services and specified programs like Weka-Hadoop to manage big amounts of data. Despite these possibilities and tests, economic and data availability constraints have limited practical use to Oracle-SQL and VBA.

### 1.3 Thesis Organization

The structure of the present document follows the process that has been created for the final development of the model. In order to enable the reader to understand from the most basic concepts up to some of the more concrete knowledge obtained as well as the various

stages of development, the information has been organized according to the learning flow used in research, development and improvements.

The first point of this document introduces the motivation that has led to its implementation, the objectives and requirements of the model that will be developed and the structure of the document.

The second point concentrates a large part of the work performed, focusing on showing the basics of Machine Learning with the following structure. First, there is an introduction of what is considered Machine Learning in a broad and generic form as well as their types. Subsequently, the foundations of which are based are described, the depth in which each paragraph of ML theory will be treated depends on our goals as well as its importance and the advantages of its properties. We will finish this point with a review of the main limitations as well as the different solutions that have been proposed and developed to overcome them. Most of the models studied use part of these strategies and their fields of improvement are defined by their limitations. Therefore, understanding this section along with the theory that defends its progress is mandatory to proceed with any development, especially when it comes to implementing non-trivial improvements.

The third section exposes part of the theory, therefore the selection of data which best represents our case would pose a problem. However, its analysis and treatment is essential to progressively achieve better results. It also adds more foundations to what would later be seen as Data Mining and Unsupervised Learning, a field that has shown great progress and better results in various aspects directly related to Machine Learning, which is a fundamental requirement and without a doubt will accompany the ML in its evolution.

Going into more detail, the fourth point will show one of the most recognized theorems in unsupervised learning, Bayes theorems and their interpretation theorem. The reason why we have chosen this theorem from the range offered by Unsupervised Learning, it's due all the applications that used its approach in both models of learning. Unsupervised Learning has strong ties with Data Mining and today there is no doubt that they will be, possibly already are, the methodologies with the higher expectations of growth and application. However, as far as our interest is concerned, for the time being, it is more practical to clearly define the results we want to obtain or more precisely our function objective.

For this reason we have focused on Supervised Learning in the fifth section. Firstly, neural networks will be exposed in a generic way since they will be used the most and will be fundamentally treated in the final model. After an introduction to define what is considered an artificial neural network, we'll see their differences as well as the evolution that they have had and are expected to have in the next few years. Going into greater detail, we analyze the main characteristics that will define our typology of network as well as their respective properties. Due to the great expectation that its foundations might create and all problems, both mathematical as well as human and computational cost involved in its development, we expose



part of the criticism against it, which is somewhat answered with real-life examples or foreseeable future improvements.


Continuing with artificial networks, we treat with several depths the main types that define a network approach, their results and their applications and limitations. Since they all vary, not all of them will be considered when predicting load curves. However, all exposed networks have proven to be close to state-of-art examples that can be applied to requirements of the electricity sector. Radial Basis Networks have modifications according to its end use that with a simple implementation achieves good results efficiently, Self-Organizing Map with support Vector Machines Algorithms is widely used in classification problems but with direct application to regression problems or just scenario selection. Boltzmann is a version of a recurrent network theoretically capable of solving complex problems. Reservoir Computing for its part, is a dynamic network methodology that has given a lot of information on the input data, a large mapping problem, is able to dynamically adapt very effectively, reason why it has been taken as a reference for the development of various approaches, as we will see.

For these reasons, this diverse selection of algorithms represents the types of neural networks that not only offer the best results but are also substantiating part of its evolution. Not only that, all of them have direct approaches presented and demonstrated on the field that deals with this document's application.

But Artificial Neural Networks are only one of many approaches to Machine Learning. Although some of the methodologies described below can also be applied to Unsupervised Learning, the explanation and descriptions are oriented to Supervised Learning. Online Machine Learning is a process where each new value updates the model in question, as all the others, but the fact that the amount of memory remains constant offers various properties. Bayes Networks build upon the theorem with the same name and given the widespread use of its formulation by Markov's approach, whose applications are used in lots of fields, makes it an opponent of equal height to the latest developments in Artificial Neural Network. Not only that, Online Machine Learning with Bayesian Networks and Hidden Markov Models would have been the models to use in the development of the final model if it were not for the advances in Deep Learning that has been made in recent years. Yet these three methods, along with their properties and limitations are, after all current studies and opinion of the author, the three with higher growth prospects, especially if we divide the range of applications of each.

Continuing with different approaches, there are two that have been the most used by different companies in the sector over the last decade. While the ARIMA model will be the reference because of its wider understanding, Bagged Trees or their modifications are used in a wide range of business and its results in regression functions have always been at the height of the best. In this case, these two models together with typical regression models, will be used since they are the most commonly used when dealing with problems of adjustment and prediction.





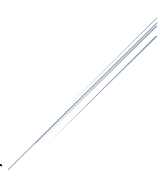
In the sixth section, we begin with the development of the model upon which the improvements will be made and which is expected to continue working in the future, probably accompanied at various stages by some of the improvements previously exposed. Neural Network Development seeks to explain from the basic operation of a neural network, we start with an overview of how the algorithms and approaches in the theory are implemented. We will continue with how it is, what features it has and how a neuron Works as an independent unit. We will expand our network with a finite number of neurons and see the problems that add the configuration of these layers. Subsequently, we will analyze, while still complicating our model, the differences of having a static or dynamic network. It is worth mentioning at this point that some of the models previously exposed, especially recurrent or dynamic neural network, only differ from the exposed one in detail, in this configuration of neurons and their interconnections.

Since our model will be multilayer in the end, we will continue the development, focusing on the characteristics of this type. Turning to see the importance of the input data and how it defines the behavior of the model as well as the results obtained with applied examples. And we will continue with the bases of the training functions of our network mainly based on gradients and their characteristics. Taking as reference the previously exposed learning methodology, we will see other types of learning and different functions because adapting our model to other problems entail a fundamental part of the optimization process. Continuing the development phase we will see how various functions can allow us to improve and analyze our network from this stage onwards.

In the same way that our final model will be multi-layered, it will be dynamic. For this reason, we continue the explanation of the development with a brief introduction to the characteristics of a dynamic model. It continues to add various configurations aimed directly at our problem at hand, first preparing our model to analyze and learn time series, subsequently setting up the model so that the outputs depend on multiple entries and finally adapting the model to produce multistep forecasting. Because of the use to be given later to the Radial Basis Networks we will introduce their behavior, structure and main applications.

In order to present possible improvements that could be applied to our model, in the chapter dedicated to Improvements we will describe important points in the development of this type of models. These improvements in development and its potential applications are more oriented to more professional models or with higher requirements than the ones we have. Initially the great virtues of most of these neural networks are discussed. Subsequently, briefly theoretical and hardware aspects as well as the logical evolution of these systems, in both development and implementation, are introduced. Finally, we analyze two widely used techniques to improve the performance of a forecast and model optimization oriented network, either base on continuous or discrete learning.

Later in the seventh chapter, we will analyze the results obtained with different models and the progressive improvement achieved in the case of neural networks with explanatory



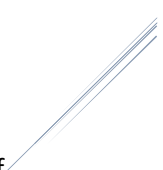
variables. At the beginning, we will see how the selection and processing of variables has affected the results. Then we'll analyze different types of error which, in certain circumstances, we can reduce or even eliminate, this way we came to the conclusion to add another dimension to the input parameters. Finally, before dealing with the results, we further elaborate the consequences of the way values are loaded and some improvements aimed at improving the behavior of the model in general will be proposed and implemented.

Then, we present the process undertaken to achieve the values obtained with ARIMA and its best results. Following the same mechanics, we present the results achieved with different linear regression adjustment algorithms, trees decision and a simple generic neural network.

Entering the first stage in the adjustment and optimization of neural network NARX we will make a brief introduction. On this simple network, we will study the behavior it has when increasing the number of neurons, although given the importance in the last stage we will be searching for the optimum point of neurons. In the next phase, we will compare the difficulties in the implementation and consequences of changing inputs and feedback delays in our network, in such way we'll define objectives in our first layer of parameter. We will continue by comparing the consequences of changing the processing, number, form of data loading and in which part of the model they are considered. Following the strategy of optimization, we compare how the results and requirements are affected by modifying generic parameters of the model as well as convergence requirements.

In the last stage of our configuration, we will see the consequences of dividing the problem manually; we will analyze how this automatic pattern selection process should be conducted. The first point of unsupervised shows the results obtained, as well as its meaning and explains the values of the weights of the network and the conclusions reached by the machine without more information than the input data on the one hand or the output on the other. Considering the results obtained in this last point we will see two ways to apply the results in our model. In a way that help us distinguish between conclusions in order to improve the recognition of patterns and behaviors. We also analyze how the network internalized exogenous values in their intermediate layers and the final result.

Before we conclude, we will see in a simple way how in our regression case the tests do not seem to demonstrate a substantial improvement when we increase the complexity of the network architecture. What comes to corroborate the behavior described in the first point of the theory, which describes the reason why a single network with sufficient number of neurons can replicate the behavior of others with greater number of layers, backed or with different structures. Before reaching the results, we show a table with the results and their evolution up to the point of convergence. While some algorithms are not focused on regression and its use in forecasting are not correct, we will see the reason for the final choice. Understanding their behavior to different parameter settings and everything to do with the data that serves as the



base, we will detail the final choice of Bayesian regulation with part of the foundations of Levenberg-Marquardt.

We will finally see the application of the same model in a relatively discrete problem that will help us to make decisions according to the forecast of requirements by the system operator in auxiliary services. In this case, in order to show the simplicity and the virtues of this model to various problems in the real world without big variations, we will not modify the network configuration nor the parameters.

The conclusions chapter is divided into four points. The first focuses on the objectives achieved and the limitations that have been taken during the process and those that affect the result. Then summarizes the conclusions obtained during the process of learning, its applications, the benefits and complications that exists to develop, implement, and integrate these models in an intelligent system. Finally a brief mention apart from the tools used and available for learning and development of this type of models is made.

## 2. Machine Learning

### 2.1 Introduction

The main objective of this field of computer science and statistics is the search and study of algorithms which can conclude and learn from the data which are presented to them. The way of working with these models is based on the inputs, decisions are made according to the final objective of the model, but not based only on structured pre-programmed instructions, i.e. the model will be capable of adapting to the inputs it receives.

It's worth mentioning at the beginning of this point the differences between Machine Learning and Knowledge Discovery in Databases, since while using a very similar theory they aim for different although closely related. However, we can even consider them as equals if we extend the perspective of machine learning, so it is somewhat debatable.


Knowledge Discovery, thanks to the tools provided by Data mining, focuses on finding properties, structures or relationships in data that were previously unknown. As we shall see in more detail, this objective is similar enough or is the implementation of unsupervised learning with a particular purpose and manner.

On the other hand, the main objective of machine learning is the prediction and helping in the decision making, but with reasonably clear information provided by learning input values. However, in addition to the many similar techniques used in both cases, usually data mining is used as pre-processed data before loading it on the model of machine learning.

The main methodologies when it comes to the use of these algorithms will be presented in the section where they correspond according to their typology of supervised or unsupervised. Later we will see in more detail the approaches and algorithms whose latest results do indicate they may help to get better results in the part of the Electric Power Industry problem that we are dealing with. Finally, we will progressively come into neural networks from the basics up to some complexities and other algorithms will be defended with certain length for their high expectations when it comes to applying them in the industry that interests us.

#### 2.1.1 Typologies

Much of the theory comes from the fields of computational complexity theory and statistics along with artificial intelligence and optimization. Machine learning is often divided into two types depending on the method of learning and the desire of their behavior. Although it is a very broad field normally used and mixes about ten approaches as we will explain and analyze.



In the first case is usually divided into three categories depending on whether the way to learn is by "sign" or by a "feedback". Of course, there are problems where a mixture of two or even all three models are used in different stages of learning in order to improve the results.

It is interesting how as a data preprocessing method, we use unsupervised learning, then pass the results to a model where we are directly marking a desired goal and then use several of these models with reinforcement learning to go beyond the capacity of predicting these values.

As we will see in more detail, on the feedback side, we have supervised learning, where the model is given the input data and the output values it should get with more or less confidence. This modeling occupies most of our cases, applied to the energy industry. Without abandoning the models that allow to teach to the machine according to its own interactions with the environment that surrounds it.

Unsupervised learning for its part, is independent and, is solely responsible for finding patterns, relationships, similarities and therefore differences that allows the auto-creation of a solution tailored to the structure of these input values. The series of algorithms have been given the freedom to find patterns that are not displayed directly.

Last but not less interesting, we have reinforcement learning, with this type of model is allowed to interact with one or more models able to adapt to the behavior of the model we are trying to train but without saying which goals should be sought. Thus, our player will be learning the rules governing the models that interact with itself and therefore the consequences and generalizations that may occur.

The other way to categorize the type of machine learning, according to the target behavior we are looking for, would be classification, regression, clustering, density estimation and dimensionality reduction. In our case we will focus on the first three as they will be the most attractive for our purposes.

Classification looks to spread data across various sub-groups in which other data have previously been distributed, so it is generally a way of supervised training.

Clustering resembles to classification in its goal, but with the difference that the division of groups is not known beforehand. The model therefore must be able to create these groups and decide which group data should be assigned. Whether by the biggest difference with other groups or their relative similarity with their peers in the candidate group.

Meanwhile, regression, in which this paper focuses, is generally also a methodology of supervised data. Looking to find the functions that achieve the desired outputs depending on the number of inputs that are offered. Unlike classification where a discrete set of groups is generated or pre-fixed, these values are continuously learning.

## 2.2 Theory

It is an essential briefly comment on which theoretical foundation machine learning is based, then we will see the main problems or limitations. Although, we will follow a logical order for his understanding from the most basic, there are sections of the document that could be considered a theory not found at this point. This is because these explanations or developments are not generic to machine learning, but approaches or algorithms that we will see more closely as we focus on our goal, for this reason they will be found in the corresponding sections.

### 2.2.1 Bias-variance tradeoff

This dilemma is of considerable importance in this field of study, when much of our goal is to minimize two types of errors, which will be in charge of deciding whether we have been able to learn beyond our initial data.

We have a clear definition of bias and variance. Bias refers to the error as false assumptions in the learning algorithm. Variance at the same time, measures the error produced by small variations in the training data.

Breaking down these errors together with the inherent error of noise in the signal or data set to analyze, we can generalize the error that the algorithm will have.

The tradeoff is that ideally we look for a model capable of generalizing sufficiently well the problem, to be able to predict outside of the data provided in the learning stage, but at the same time is desired that the algorithm in question takes into account the generalization of the data used. As expected, it is arguably impossible to conform perfectly to the loaded examples while attempting to correctly predict the behavior variations of these.

Algorithms with a large variance will be able to represent data well, but fail in predicting further space defined by our values. On the other hand a very high bias model tends to generate simpler models, but they fail to capture the relative behavior of the problem. If our algorithm has lower bias, it will be able to better capture the complexities and therefore it will reproduce the problem proposed with better accuracy, but also considering the noise in the values used as valid, which reduces at the same time this precision. We would therefore have a complex model capable of learning the values of input, but not effective to generalize. (Snijder, et al., 1998) (Stuart, 1992)

### 2.2.2 Computational learning

This branch is responsible for studying the calculation complexity of these algorithms. As we will see many of these models require exponential growth depending on the complexity or size of the problem in question, in addition to that, many operations can be complex; the time required for learning can render a model useless.

Very generically and applying the example of supervised learning, the algorithm in question, calculates the difference between the generated output, and the output that should have been obtained producing a classifier. This in turn affects the model in question and the values that have not yet been processed. It is in this part where we try to optimize in such a way that it reduces as much as possible, errors or unnecessary decisions that will be applied to the next dataset.

Time complexity (measures the time required to find a solution regarding the length of the input), feasibility of learning and performance like the ones introduced before are also analyzed by this branch. We can divide part of this time complexity depending on whether the functions may or may not be learned by a polynomial algorithm.

Many of the assumptions are based on being able to generalize principles taken as valid, although they have not been tested, and thus being able to analyze from a faster point of view part of the theory behind the computational learning.

For example, measuring the probability of something occurring can be seen as the probability with regard to the frequency that the event occurs in a finite number of trials or as for example the interpretation that makes Bayesian probability, where in order to evaluate the probability of a hypothesis new cases provided are updated and taken into account.

Citing some of the approaches, since its analysis is not the subject of this document, stand out performances that have led to practical examples applied today in machine learning as we shall see in more detail by its remarkable results are: Support vector machines/regression thanks to VC theory, belief networks based on Bayesian inference, online machine learning driven by Nick Littlestone, algorithmic learning theory or boosting in learning thanks to probably approximately correct learning theory. (Enzo Busseti, 2012) (Xiao, 2009)

### 2.2.3 Risk minimization

This principle is used in order to have some theoretical limits on the performance of learning algorithms, so that part of the theory of statistical learning is integrated and applied, for example, in simple problems studied in computational complexity. Let's see what Empirical Risk

minimization and Structural Risk minimization is based on step by step under a single generic example of supervised learning.

To calculate the risk we need to know certain distribution which is unknown. We can then approach it by calculating the mean of the objective function on the data we use as training. Therefore, if we optimize a hypothesis that minimizes the empirical risk we'll obtain the scenario where this objective function is smaller.

Formulating the problem, since it is an important base for our type of problem. We have an issue where we want to know which assumptions have to be followed to get Y from X data. We know the probability distribution of X and Y, however, we don't know their relationship known as conditional probability.

Similarly, we have our loss function, which is responsible for measuring the difference between our hypotheses (unknown) and true (provided by and belonging to Y known). Therefore the risk associated with a hypothesis is the expected value of the objective function, or the integral of the objective function with the result hypotheses applied to x, and with y result, respective to x and y. In order to find the correct hypothesis where the risk is lower.

In supervised learning we have a data set x, and a set of targets y to which they should direct those data, our problem is to find the function or hypothesis h such that  $h: X \rightarrow Y$  where X and Y are the respective spaces of x and y.

If we understand the problem as a joint probability distribution such that  $P(x,y)$  we can generalize and say that it does and does not depend on x, we have a certain conditional distribution for each x,  $P(y/x)$ . See in the pictures below the graphic meaning of joint probability and conditional and joint relationship in the second.

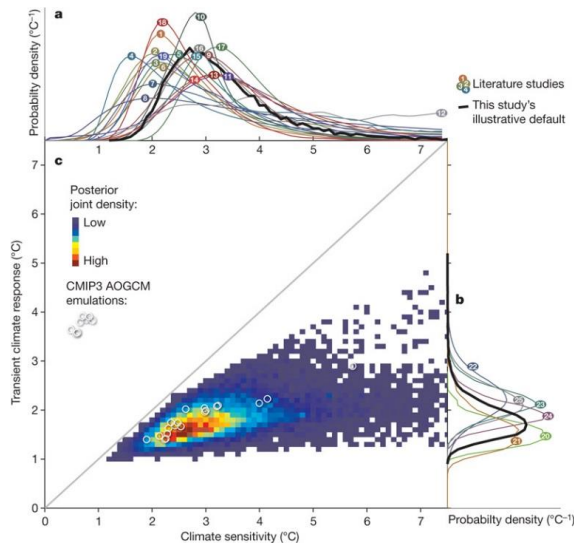


Figure 2.1 - Probability density example

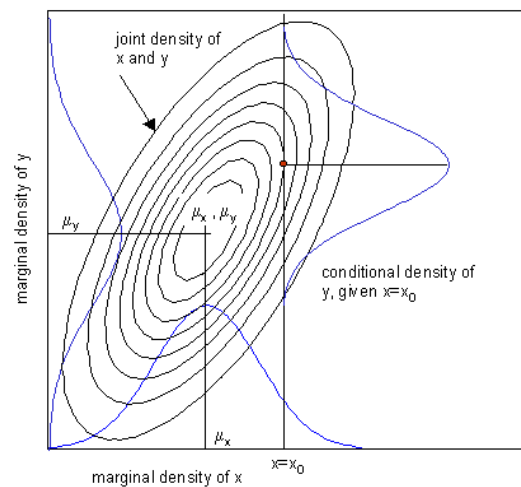


Figure 2.2 - Different probability densities



Before seeing the differences we must also define the loss function, which represents the difference between the value obtained by certain assumptions and the actual value  $\hat{y}$ ,  $L(\hat{y}, y)$ . Therefore the risk of each hypothesis  $x$ , is defined as the expected value of the loss function of some hypothesis  $h$  for some value  $x$ . The following formula represents the risk of hypotheses respect to the conditional probability.

$$\mathbf{R}(\mathbf{h}) = \mathbf{E}[L(\mathbf{h}(x), y)] = \int L(\mathbf{h}(x), y) d\mathbf{P}(x, y) \quad (2.1)$$

The goal of our algorithm will obviously be to find the hypothesis that minimizes this risk to certain hypotheses  $h \in H$ . As we are considering that we are able to differentiate between the results obtained so in many cases the models based in part on this approach are known as discriminative training. (Ke-Lin Du, 2014) (Suvrit Sra, 2012) (Jaakkola, 2010)

### 2.2.3.1 Empirical

In the case of Empirical Risk Minimization, we are trying to find the hypothesis  $h$  that best fits our input data, which best represents these probabilities.

As the joint probability  $P(x, y)$  is unknown, the risk associated with each scenario cannot be calculated, but if an approach which we call empirical risk, which will be the average of the function with respect to the input data loss (from which we know the  $y$ , with respect to each  $x$ ).

$$\mathbf{R}_{emp}(\mathbf{h}) = \frac{1}{m} \sum_{i=1}^m L(\mathbf{h}(x_i), y_i) \quad (2.2)$$

Similarly as in the previous approach, our algorithm should seek the hypothesis that minimize this error, converting the problem into an optimization problem, which will be raised from the principles of Enterprise Risk Management.

### 2.2.3.2 Structural

Furthermore Structural Risk Minimization is based on the above but adds a function that controls the bias-variance tradeoff (penalty function), so that the overfitting is avoided and tries to find the simplest solutions representing the issue as significantly as possible.

Because there are different definitions of complexity, multiple formulations have been proposed, without going into detail, we'll see their expression and a graph where the part on the right of the hypothesis  $h^*$  mean that we selected a hypothesis with underfitting and the opposite side overfitting, as described in the bias-variance tradeoff point.

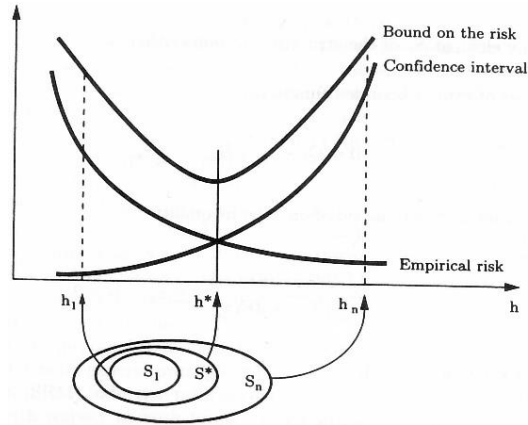


Figure 2.3 - Risk minimization trade-off

Being  $C(g)$  the penalty function, if  $\lambda=0$  we have a low bias and ERM with large variance and backward with lambda close to one, such that lambda is calculated by cross validation in most cases. This adds great complexity when applied to models, but we will not go into detail since this will not affect our model.

$$J(g) = R_{emp}(g) + \lambda C(g) \quad (2.3)$$

#### 2.2.4 Probably approximately correct learning

Defines the framework for mathematical analysis of machine learning. At first, not very different from the previous approach to ERM.

Under this theory, the model with certain input data must find a function (usually called hypothesis), from a repertoire of functions. Therefore, we seek the most probable hypothesis that is able to internalize data, or seen otherwise, return a small error. Given its generalization the model should be able to learn about any probability or data distribution.

There have been important advances in this framework thanks to the concepts shown previously, with which the objectives to find procedures that get functions efficiently, with time and size limits provided that can be solved by a polynomial function is introduced.

#### 2.2.5 Statical learning

It's a related, but not the same approximation to a similar field. Usually machine learning focus on: network, graphs, weights, learning, generalization, supervised and unsupervised learning. However, statistics are related to: models, parameters, fitting, test set

performance, regression, classification, density estimation and clustering. So we can consider this work part statistics with some of the machine learning theory.

Statistically, in whose theory the model of support vector machines is focused and on which we'll delve into its corresponding point. Focuses on finding the prediction algorithm regarding the provided data. Even though it enters all categories of learning we can explain it as mapping functions that lead from an input to an output in such a way that the function is able to predict future outputs.

In our problems, the values will be continuous so statical networks will be a regression problem. With Ohm's simplified law example, our algorithm given voltage as input and current output returns that the function that converges will be when the current is equal to  $1/R$  multiplied by voltage.

Let's understand it better with the mathematical formulation, similar to the demonstration of Empirical Risk Minimization performed previously, but with a slightly different approach,  $x \in X$  where  $X$  is the space of possible inputs of  $y \in Y$  outputs. The way to analyze the problem of statical learning consists of (same as ERM), there is a probability of a distribution (unknown at the beginning), such that  $p(z) = p(\vec{x}, y)$ , the product such as  $Z = X \otimes Y$ . Having  $n$  couples of training in such a way.

$$S = \{(\vec{x}_1, y_1), \dots, (\vec{x}_n, y_n)\} = \{\vec{z}_1, \dots, \vec{z}_n\} \quad (2.4)$$

Therefore, we need to find the function such that  $f(\vec{x}) \sim y$ . If  $H$  is the space of hypotheses where there are all  $f()$  which generalizes  $X$  to  $Y$ . If we have the objective function  $V$ , we can calculate the risk implied that exists between the difference of the outputs and inputs, with the same approach and formulations as (2.1) y (2.2).

The previous function concludes that the best function  $f()$  is the one where the value tends to infinity (the area of the optimal objective function is the largest), respecting the boundaries. When calculating the integral we find the value  $p(x, y)$  which is unknown, so we need to approach it. As we have seen this is the ER, and recalling the previous development the ERM is the function that minimizes the risk and therefore our objective. (Andrew Ng., 2014) (Trevor Hastie, 2009) (Breiman, 2001) (White, 1989)

### 2.2.6 Vapnik-Chervonenkis (VC theory and dimensions)

The goal of this theory is the understanding of learning processes from a statistical point of view, with an empirical basis. It is defined as the number of elements in a (cardinality) subset which is divided and have parts in common. Explaining it over the following image, we choose that points, and the algorithm "name" those points and seeks the hypothesis that meet a correct division or classification of those subsets, may occur that some subsets do not capture

these points but the dimensions or points which manage to classify the information are the concept of VC dimension.

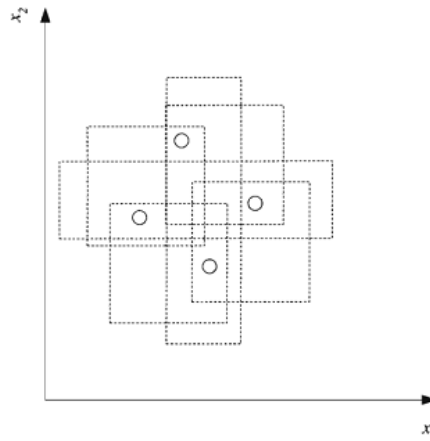


Figure 2.4 - Vapnik-Chervonenkis dimension

Starting with the consistency in learning processes (finding the laws to categorize the subsets) it defines which are the necessary and sufficient conditions to meet the principle of ERM. It continues with the speed of convergence in a possible solution and the ability of the algorithm to generalize the problem. Lastly, we study how to create algorithms that are able to carry out the third step, in order to improve the ability of the algorithm to learn the hidden complexity of the problem, in this case the divisions of input data. (Bousquet, 2003)

In this way VC is able to find the function that stabilizes the subsets in the given dimensions, provides reference conditions to generalize the learning algorithms.

## 2.3 Limitations and Strategies

### 2.3.1 Statistical classification

Statistical classification refers to the problem of giving data divided into groups to identify which sub-group it is from. When referred in unsupervised learning it is known as cluster analysis as we will see later.

The classifier or the algorithm responsible for analyzing instances due to certain characteristics of a variables sub-set, known as features, will assign a certain value to a class or subgroup.

There are several types of statistical classification based on frequency, Bayesian, binary and multiclass, which focus on the feature vectors or linear models. One of the most widespread examples are those based on probabilistic classification, which using statistical inference (where

several hypotheses are considered to then apply them to larger groups of data) finds the best recognition or subgroup that most likely belongs.

With respect to not probabilistic models (like autoencoders (Anon., 2015) (Baldi, 2012)) it has several advantages, in the first place and continuing with the idea exposed in the previous paragraph, we have a value for each data that measures the probability that this data corresponds to a specific subset, therefore we might decide to create a sub-group of values that just does not fit into any group with any level of trust required. This way when a new data is presented, we are not littering the feature vector.

### 2.3.2 Cluster analysis

Unlike statistical classification, which is in charge of identifying to which class certain observation is referring to, cluster analysis automatically generates multi-dimensional groups called in this case clusters. It is an important part of the data mining, the main difference in algorithms is the definition of what is considered a cluster and what is the most efficient way of achieving it.

Some of these methods are, assigning observations to clusters in such a way that there is less distance between them, so we would have to be defined what it is considered as distance. Another way to deal with this problem is to find the densest zones from the given data characteristics spaces. Next, we can give greater importance to the intervals between observations and consider them as statistical distributions that can be later combined.

Although considered part of unsupervised learning, some parameters (referred to in the above paragraph) should, however, be raised in the development of the algorithm. Being an iterative process in the search for patterns in the data, it can be considered a multi-object optimization problem and therefore this opens up other roads with their tools from where we can analyze it.

One of the main differences with knowledge discovery is that, in our case, we're not giving importance or interest to the cluster, since our goal is the initial classification of these values.

This branch of the analysis is very interesting and promising, both by its approach as the achievements that different techniques which are based on it are making. (Kumar, 2012)

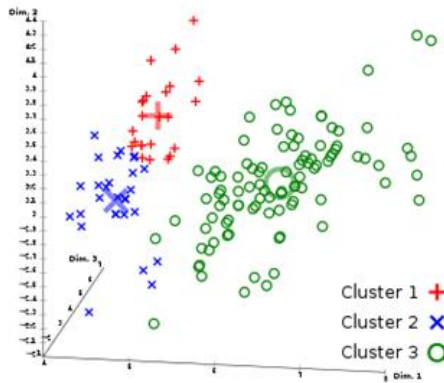


Figure 2.5 - Clustering example

### 2.3.3 Regression analysis

As its name suggests, regression analysis is a broad field that is in charge of finding function relationship, or more broadly, relations between variables. Keep in mind that at the stage results analysis, correlation does not imply causation. Therefore, it is necessary to analyze and model the system in such a way that those relationships between the dependent variable; and that one or more independent variables are represented in the most efficient possible way, trying to respect possible limitations of the mathematical analysis or from the own problem in question.

The result of this branch of the analysis is the function known as regression function, although this result is accompanied by various probability distributions which are reflected as a result, in variations in the dependent variable; it is the relationship with the independent variables which are recorded in the function.

In this way we can find relations, values, shapes or interrelationships and draw conclusions that will help us to predict our values. In cases of problems that you can generalize to parametric, where a limited number of data is proposed, and there are no big complications if they exceed specific problems limits, we have simple examples and others with significant assumptions and simplifications such as linear regression and ordinary least squares.

Nonparametric regression has the freedom to search these relationships using not an unique function in a multidimensional spectrum. Therefore the model is not pre-limited in the number of dimensions and will be itself the one that, with a large amount of data, will conclude what are the interrelationships between its inputs. (Rob Hyndman, 2010)

#### 2.3.4 Anomaly detection

As we have mentioned in statical analysis, we have data that does not come within any class. These values do not have to be only outliers; also we have novelties (new values that the model had not been previously trained with), noise, detours, or simply exceptions.

If we are dealing with data in an unsupervised manner, we can find these small groupings and decide post-analysis they are values that were not expected. However, in certain cases of supervised learning these decisions must be taken and we come to the choice of whether these values should or should not be considered since they represent at the same time part of the problem behavior.

#### 2.3.5 Association rules

It is a method that seeks to find the rules that govern the data set, seen another way, the strong relations that we can conclude by measuring different levels or relationships that may be of interest.

Discovering these rules or behavior that does not take in count the time or order in which events happen or presented data, follows a series of widely studied processes due to their direct application to real cases.

We can start with a binary representation with a specific range of trust and in order to discover the relationship, we consider the number of times that some value has been generated and what has caused that to occur. Considering the frequency of one of these paths a candidate is generated, who is compared against another to the reduce cases as desired.

Is at this stage where the beginning of A priori becomes important, as we see in the image if certain scenario has been caused by a certain range of trust and frequented by a certain sub-scenarios, we can conclude that in order for the general scenario to occur those prior scenarios should exist.

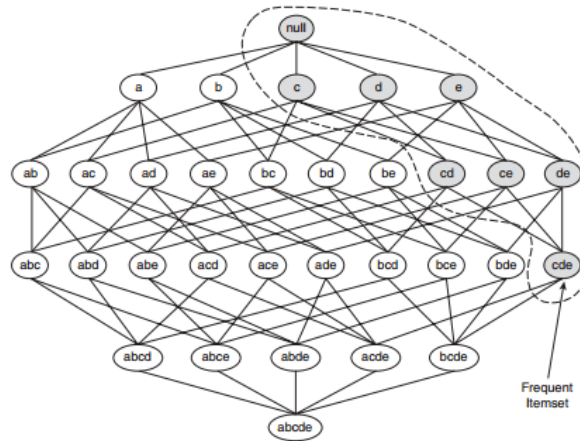


Figure 2.6 - Association rule example

Another important property is that of monotonicity, which comes to the conclusion that if certain events is infrequent or unlikely, all events that depend in any way on it will be unlikely. Viewing the case in the image above, suppose the event  $ab$  is rare, we can decide that  $abx$  and  $abxx$  will be rare because they are strongly related with  $ab$ , this can be extended up to  $abxxx$  and that branch of scenarios may well be disqualified or will be strongly reduced with some level of security. (Petr Berka, 2010)

### 2.3.6 Behavior psychology

Behavior psychology is the basis for reinforcement learning, which analyzes how must an algorithm or model behave such as the accumulation of its actions under the particular model generate or accumulate the maximum benefit.

In economic theory and game theory, this would be seen in how two agents or participants reach balance when neither of them have the necessary information to make the decision, i.e. under bounded rationality. There are models who seek these balances among actors who predict individual cases from the econometric point of view with considerable success.

From the point of view that concerns us is usually oriented through the Markov decision process (MDP). This dynamic model provides a series of possible states, possible actions, the real values of "reward" and a description of how every decision affects each state. Note that the effects of an action that is taken at each stage are only affected by that decision, not the decisions which have been taken up that point.

Games can be given with a finite limit or a theoretical infinity, in such a way that policies that can be learned create different expectations of reward. Since the same state can be reached from different paths, we should try to limit it.



Therefore, it uses a dynamic programming where the global problem is divided into smaller problems and solutions are combined in each iteration. In this way, we try to find rules that enable greater reward along the path or in the most efficient way (short in this case).

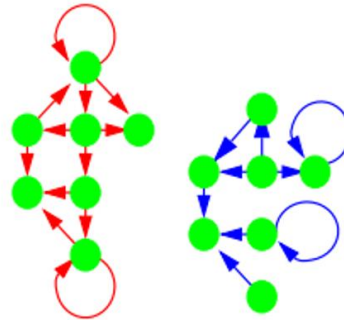


Figure 2.7 - Reinforcement learning example

If we consider the actions as deterministic, for every decision at every stage we define a new state. If it is stochastic, we will define the probability distribution of the following statements. The path will be called a policy (a policy that has been taken) to reach from a certain point to the next and go iterating. At one point, we will see what is the accumulated (deterministic) value or the total expected reward we get for following this policy; in both cases we might obtain an infinite value.

Therefore, we need to resort to the objective function, which will be in charge of mapping all these policies and transform them into a value. For this, we need to define a finite horizon and therefore a maximum benefit, or as it is more usual to decide whether we prefer higher profits in initial step (seeing step as time).

Because of the need to find the optimum from these dynamic programming models we use the Bellman equations, or dynamic programming equations, which divide the initial optimization problem into simpler problems. Taking the value of each decision at some point (time) as an accumulation of the initial decisions as well as those yet to be taken but that are limited by the initial decision.

One of the reasons we used this methodology is the ability to deal with major problems which is where many of the previous methods are not viable.

Another advantage of reinforcement learning is that since couples of correct data are never presented, its performance is to find the balance between the unexplored and explored paths, i.e. being able to deal with the trade-off of exploration vs exploitation, which of itself represents a useful strategy to focus machine learning on. (Matthew M. Botvinick, 2008) (Krichmar, 2002) (Boden, 1988)

### 2.3.7 Structured prediction

This study focuses on finding structures that shape some problem or data set. The most popular models of this methodology are random fields and Bayesian networks. In both cases there is a large supervised problem, with the added complexity of understanding the relationships arising. Training in itself is generally unfeasible therefore the use of approximate inferences, such as the Markov Chain Monte Carlos method, are usually considered. Which deals with the tradeoff of reducing the processing time in exchange for accuracy.

We have a wide repertoire of methods to choose from, but given the purpose of this document, we use Bayesian Networks which will be detailed at greater length later. In general the results of these models can be represented by acyclic graph and conditional probability tables, so that it shows the conditional probability that a certain event occurs when an event or prior hypothesis has occurred.

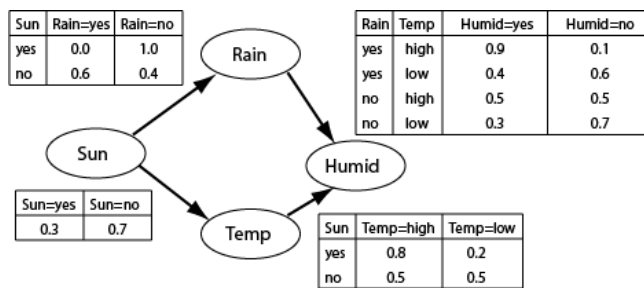


Figure 2.8 - Inputs examples

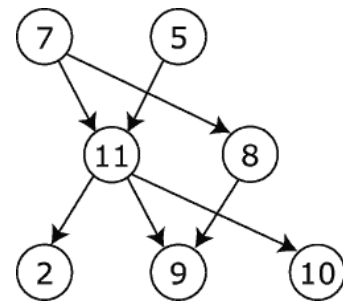


Figure 2.9 - Graphical dependencies

In figure 2.8 above, considering a day where there is sun, ie ( $Sun = Yes$ ) = 1,0 , the odds of that same day being humid are as follows.

$$P(Humid = Yes) = (0,5 \cdot 0,8 \cdot 1,0) + (0,3 \cdot 0,2 \cdot 1,0) = 0,46$$

$$P(Humid = No) = (0,5 \cdot 0,8 \cdot 1,0) + (0,7 \cdot 0,2 \cdot 1,0) = 0,54$$

## 3. Variable Selection (IVS)

### 3.1 Introduction

The main problem in implementing this type of models is what, how much, and how to provide the data. Just as in our example this is largest constraint, its optimization and analysis is a real problem in the application of more advanced models to real-life problems. We have done considerable testing by trial and error to select these three factors, but currently there are some more advanced methodologies as the modified versions of the Standard Step Method, to name one.

As defined by some researchers, once we achieve a correct algorithm capable of efficiently representing the problem to be treated, it is preferable to focus the efforts on achieving better input data. Small variations in the approach to the problem, or just slightly different configurations can achieve in the best scenario, given the prior exposed case, a very slight improvement due to the limitations of the theory of this type of problems.

The previous paragraph comes to defend a greater dedication on the time invested in the selection, collection, analysis and processing of data, possible improvement with respect to a deeper re-optimization of a previously optimized model. As we will see throughout the document the understanding of the errors, the analysis and meaning of the parameters learned by the model as well as the results, will help us directing the search and definition of the data used in order to optimize the behavior of the model. In this way we will have greater freedom to direct the effectiveness of our algorithms. (Eirola, 2014) (CheGuan, 2013)

### 3.2 Function complexity

An evolution of the problem of bias variance tradeoff and focusing on supervised problems we have to deal with the complexity of the problem to represent regarding the data provided. If these are correct, we can generalize correctly while reducing the error, however if the function or model you need to represent is complex this complexity must decode from the data loaded.

If the problem is complex and we have considerable limitations on the input data our model can only generate results with low bias and high variance. Different models are capable of self-adapting this trade-off according to the complexity observed on each stage of processing and the utility of the input data.

### 3.3 Dimensionality

The importance of the dimensionality of the data set to be loaded will influence both, the solution reached and the route the algorithm will have to take to find sufficient conditions to be considered as valid.

In practice a problem with a relatively simple representation may become a complex problem for the model since these extra dimensions are forcing the system to take them into account, which affects the outcome provoking a large variance. So if this problem is known, we must pre-process the input data to try to reduce the unnecessary dimensionality or try to find the initial relationships between the inputs prior to loading them into the model. If part of it persists, then we configure the model to obtain a small variance and high bias.

### 3.4 Noise

As we have previously described if we want our result to fit the actual values extremely well, with supervised training, we risk overfitting. The noise in the results can be there even having this noise in the input data, if the model we want to represent is more complex for our approach. This is because the complexity that cannot be properly represented in the model is corrupting the performance of the algorithm, so that adds to what is known as deterministic noise.

As there are ways to deal with the noise of the input data and focusing on the unreduced data sets, as in our case that they get to reduce the error, once we have used this data pre-treatment we must look a configuration which allows us to obtain a lower variance and higher bias.

### 3.5 Data kinds and Linearities

Although as a rule the data is processed so that whatever their rank or meaning they are on the same scale, to avoid saturating the intermediate functions, there are problems in which we'll have types of continuous and/or discrete data in various scales, in most analysis, except decision trees it is necessary to transform the meanings into numerical values to be treated by the algorithm later.

If you do not correctly use regularization in the input data, due to the treatment of data and numerical instabilities in the process many models get results that are far from optimal.

On the other hand, if the problem we deal with having strong nonlinear relationships, different models as neural networks or decision trees are undoubtedly the best to predict the behavior because they represent and detail these relationships more easily. However models for representing linear relationships, such as SPV, or models based on Bayes regression won't behave efficiently.

## 4. Unsupervised Learning

### 4.1 Introduction

This category of machine learning deals with the problem of finding unpaired relationships with a correct interpretation. So at this learning stage there isn't any error or signal to evaluate the process, therefore the task of the algorithm itself is to find information or patterns hidden behind the data.

Though much of the concepts discussed above can be applied to both supervised learning and unsupervised learning, at this point we will briefly introduce more characteristic concepts of unsupervised learning and we won't go into much detail since most of the applications that we use will be with supervised learning. The models that are getting more support from researchers as we have previously defended are clustering and Markov models, although there are neural network models as self-organizing map (SOM), which seeks the closest elements represented dimensionally to find similarities, which have proved a correct performance.

The vast majority of the models can be seen as finding the probability of a certain event on the loaded data. That is, the probability of a certain new input considering the values previously analyzed, taking or not its time representation into account. Thus, these models are useful for predicting rare or rather unexpected behavior or to monitor the expected progress of certain parameters because some values may not fall within that probability previously calculated.

They are also useful for classification as we have seen with the example of clustering, looking at the probability that some new input  $x$  has of falling into two distributions of different probability; we can decide which one corresponds with varying success (many other conjectures can be taken to decide what forms a cluster).

Although not the objective of this document, these methods have also proved their worth for communications and data compression. Not only that, these machine learning systems, possibly accompanied by other algorithms to improve efficiency, are likely the future since they enable to learn what is really important and draw on their own what is most useful for a certain purpose, without the need to specify what matters. Similar to a higher level of supervised learning as how humans learn, create and conclude ideas. (Langkvist, 2014) (Kim, 2012)

## 4.2 Bayes theorem and interpretation

Today there is no doubt of the importance that the application of the Bayes rule, known as Bayesian inference, has in a large number of fields and applications. A basic understanding on the part of the mathematical statistics as well as Bayesian probability is required in order to understand the functioning of many of these algorithms.

In essence, it corresponds in part to a formula with considerable importance in the work of conditional probabilities (measures the probability that a certain event will occur if another has previously occurred). It is important to say that when considering hypothesis, there is some discussion on its supports, however, the most successful developments are based on objective methods and can be derived directly from the probability axioms and different considerations on probability as we will see later. Next we have the formula and a partial demonstration.

$$P(A | B) = \left( \frac{P(B|A)P(A)}{P(B)} \right) \quad (4.1)$$

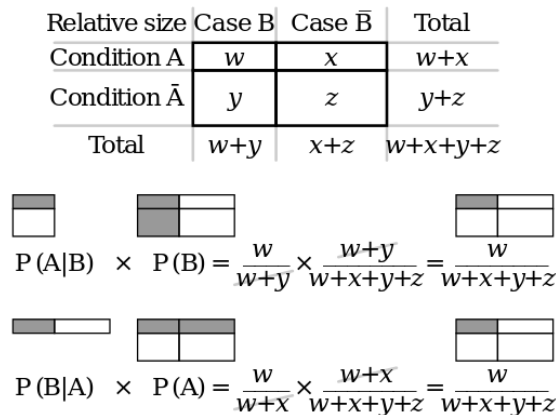


Figure 4.10 - Probability equality example

Being  $P(A)$  the prior probability,  $P(A | B)$  the conditional probability of A if previously B occurs. We can also extend this dependence as  $P = (X = x)$  and then it can be interpreted as the frequency with which X is x (frequentist statistics) or as a confidence level representation in which X is x. Simply extending this vision we can generalize to  $P = (X = x | Y = y)$ .

Generalization and depth of Bayes rule together with other aspects of statistics as Cox's axioms, which have been used to defend the Bayes approach, have led to this branch to become an interpretation with lots of applications and very good results once implemented. Also, thanks to this theorem we can update our values as we receive new information. However, the factor of considering "subjective" probabilities creates some discussion, but is part of the basis of their applications and multiple extensions. So we can have the probability of something that has not been confirmed, but we know could happen.

Suppose we have a space generated by two unknown variables, which we consider random. So if we apply Bayes' theorem it results in zero when any of the two has a finite probability density. To continue we need to modify the initial approach from which this theorem is concluded and instead of using the conditional probability previously exposed, useful in the use of events. We will get to the Bayes theorem from the definition of conditional density and so can apply this theorem equally to continuous variables. To simplify the formulation typically the denominator is eliminated by using the law of total probability so the denominator becomes an integral. See the formula and obtained space if both variables are continuous.

$$f_x(x|Y = y) = \frac{f_Y(y|X=x)f_x(x)}{f_Y(y)} \quad (4.2)$$

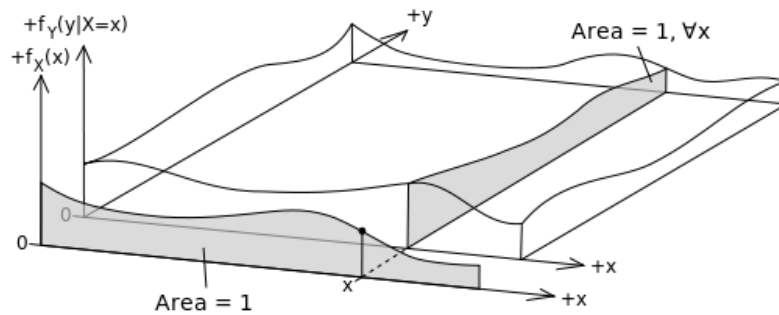


Figure 4.11 - Event space example with two continuous variables

The Bayes rule numerically defines the relationship that exists between the probability of two events, thus came to the conclusion that according to this approach to probability, the probability that there is a future for something is the number of times occurred previously by the Bayes factor. Bayes factor is a way of considering hypothesis that are given by the following application of the own theorem.

$$K = \frac{\Pr(D|M_1)}{\Pr(D|M_2)} = \frac{\int \Pr(\theta_1|M_1)\Pr(D|\theta_1,M_1)d\theta_1}{\int \Pr(\theta_2|M_2)\Pr(D|\theta_2,M_2)d\theta_2} \quad (4.3)$$

As we have seen the implementation of these principles can help us predict what we hope will be in the future. Not knowing the exact odds and having a subjective part we can have a greater ability at the time to interpret the results. It is here where we see part of its application in Bayesian Networks; first it will allow us to find relationships between all variables even if we are missing values. Given their nature also they will give us some information about casual relationships with what we can better define the domain of certain variables or events. In short, thanks to all this theory, we can see graphically and efficiently the joint distribution of a large set of data. This can be improved depending on the problem with which we deal based on various of the inference tasks, unobserved variables, parameter learning and recently more popular structure learning. (Anon., 2015) (Triola, 2006) (Olshausen, 2004) (Joyce, 2003)



### 4.3 Competitive learning

Another approach that is useful for our purposes is based on the winner-take-all strategy, with this form of training the neuron that was to receive the major upgrade of its weight receives the total variation of weights for that scenario, or rather pattern. The rest of neurons are not updated at this stage.

A less radical version of this model is to make interconnections to this neuron in such a way that certain variations with winning neurons, others will be proportionately affected. Following the way of operation, in the first place the updating of weights would be calculated, we will normalize all these modifications, would choose and would update the neuron with the major change, will normalize the weight once that neuron has been updated and we'll see whether we would modify other neurons connected or we would start the next iteration.

In this way a neuron or a group of them specialize in a scenario or pattern, which, if applied would be a fundamental part of our model. Different types of outputs will activate different neurons, specialized solely in its particular problem. The generated vectors will be able to either represent or adapt, minimizing the errors of their sub group or single neuron.

This approach comes in relationship and has similarities to algorithms that we will see later such as self-organizing Maps with K-means clustering or specific versions of Online Learning. In both cases a neighborhood of winning neuron is updated at each iteration with its ensuing consequences, the definition of neighborhood, as well as the rule for updating the weight among other features, define the behavior of each model.

The following image represents how as a subset of neurons and their possible connections with the next layer will be modified. (Peter Sussner, 2011) (Stanley C. Ahalt, 1990) (David E. Rumelhart, 1985)

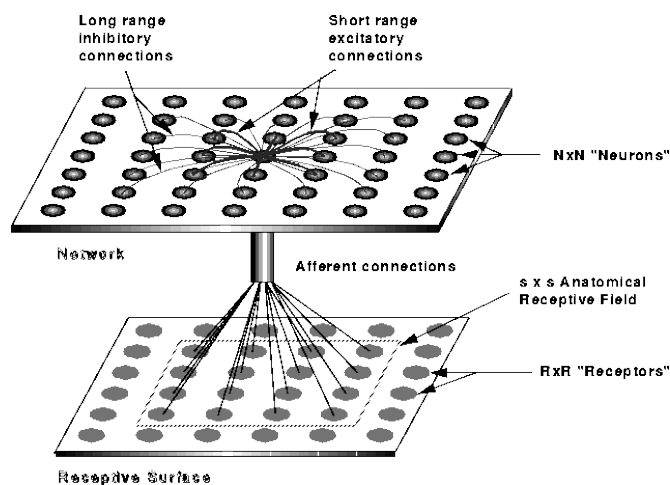


Figure 4.12 - Example of relationships between neurons

## 5. Supervised Learning

### 5.1 Introduction

In this methodology, the data are displayed in pairs or at least with an indication of what results should be obtained given the number of entries so that the difference or inferred-function serves in a guided or supervised way to understand the behavior of the data and information well enough to generalize the steps to be taken and thus to correctly transform values, scenarios or units not previously analyzed.

Generally the overall process of this problem begins with the selection, cleaning and preprocessing of the input data. The next issue to deal with is the selection of the learning function, this will depend on numerous factors such as the type of problem, the type of input data as well as its size, level of generalization and detail we want to achieve, among others.

Setting the methodology or architecture that will be used to address the problem in question. Considering the pros and cons of each model and the characteristics of our approach, there's no such thing as a free lunch. Determine the specific parameters of each resolution system if needed, whether related to the treatment of data, processing or sub-system processes that are created.

To assess and validate the model, adjust it and try as far as possible to optimize it in order to apply or draw conclusions from the results. (Bontempi, 2013) (Mishra, 2008) (Kumar, 2008)

### 5.2 Artificial Neural Networks

#### 5.2.1 Introduction

It will be the base of our model and upon which we will implement improvements, both in data processing and the analysis and configuration of an architecture that allows us to find and give a numerical value to the interrelationships that are discovered in the dataset, simplifying and approaching depending on the information's configuration that it contains. We will therefore extend with greater length in its introduction.

Based on the performance of the biological brain whose neurons are represented in the artificial model by nodes called neurons, and which connections or synapses have values or weights. These weights will, along with the bias affecting the intermediate value processed by a neuron as a separate unit, adjust their values during the learning process in such a way that will let us store the knowledge just as a real animal brain. As is the case with our brain, according to

the interconnections of our neurons we will have some memories or knowledge recorded in the form of synapses in our memory, in our model, these synapses are represented by the weight which a neuron has over another (in a simplified model without recurrence or other architectures or added complexity). (Schmidhuber, 2014) (Amjady, 2011) (Ong, 2011) (Guoqiang Zhang, 2007) (D.C. Park, 2001)

### 5.2.2 Differences

The differences are enormous when we enter into details or scales, while the vast majority of artificial models does not exceed the hundred neurons configured in one or more layer, as we will discuss later, and a few tens in one or three layers are more than enough as a general rule to represent patterns of considerable complexity, even with these small units the computational requirements easily become unacceptable.

The human brain has more than 30.000 million neurons (depending on gender, age, condition...) and each one is connected with up to 10.000 other neurons through different synapses, interconnected by electrical signals, all this is immersed in a complex system filled with different cells and proteins that interact at the same time in a different way.

However, the theory of artificial networks tells us that with a single layer with a finite number of neurons, any problem can be represented without added complications. The problem is once again time and the calculations and memory requirements necessary to carry out such training.

One of the main advantages and at the same time its similarity to the brain of living beings is that data processing does not occur in a linear manner, i.e., as in a real brain performing millions of operations at the same time, in our case we will be able to perform operations in parallel. At the same time we can delimit these operations in sub-functions or spread them in distributed systems, to reduce the memory limitations and increase the calculation power.

In reality the differences are the vast majority, since when it comes to represent this model with algorithms, the theory that cements it come from statistics or from some aspects of signal processing, part of its parameters are adaptable, but others - a vast majority of utmost importance - are not, since as the configurations, interconnections, layers, features among many others that in the end define their capabilities and behaviors, aren't either. In turn, these dynamic systems have had the need for essential support from other branches for its evolution, such as artificial intelligence, machine learning, and expert systems among others.

### 5.2.3 Evolution

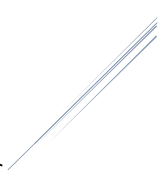
Because we rely on their work and contributions, we will make a small, mandatory and more than deserved mention of major events that have allowed the advance of this type of modeling. In 1943, Warren McCulloch and Walter Pitts developed a model known as threshold logic that will be the first system in addressing a neural network with mathematical algorithms in a very simplified way. As we know one of the greatest virtues of a brain is its plasticity and adaptability, which we seek to exploit with our network of artificial neurons, it was Donald Hebb, who around 1950 with his hypothesis of Hebbian learning, laid the groundwork for a mechanism of neuronal learning.

In 1954, these advances are applied for the first time with computers, at then calculates. Until in 1958, thanks to Frank Rosenblatt, they received a major boost with the development of the perceptron algorithm, in a simple model of two layers using addition and subtraction. The perceptron although it is detailed in this document was and is a fundamental part of the development and application of artificial neural networks, enabling the learning of linear type. Subsequently, using backpropagation techniques to train the network allowed the use of multilayer perceptron, thanks to which nowadays we can differentiate between data that are not linearly separable. Also Rosenblatt describes a circuit only with OR doors in a mathematical way, so that when it was possible to be perform in 1975 by Paul Werbos it led to the well-known backpropagation, which is still widely used. Until then, this algorithm of Rosseblatt was limited by the problem of processing OR doors in a single layer and the computational power required, which were solved in 1969 by Marvin Minsky and Seymour Papert. (Dataconomy, 2014)

Around 2010 with the advances in recurrent and deep feedforward networks significant progress was made in the recognition of patterns, at the same time developments in the back propagation algorithm and deep training have shown significant improvements in many scenarios and casuistry. Meanwhile the use of these improvements with various modifications, creating convolutional NN launched in advanced GPUs systems have won numerous awards as the models with better results in pattern recognition in recent years.

Although for many years the interest of the scientific community for this methodology was losing incentives, the advent of deep learning and increasing power while reducing costs has allowed these algorithms to be the architects of many state-of-arts in various applications, especially in broader issues such as classification (pattern recognition) and on a few occasions in terms of regression. Looking into the future, as will be discussed in extension in its corresponding section, the evolution of FPGA's and even prepared nano-devices for convolution can open a new stage in the use of neural networks, not only for its computing power but because it will mark the change from digital to analog. (Lloyd, 2013)

In addition, the use of big data along with the next generation of neural networks is allowing to reach the cerebral cortex-like virtues. For example, if we have a network of millions



of neurons, which in turn has several network structures in its interior, these sub-networks interact among them. If these structures are organized in layers, the data processing is divided more efficiently, which impact that plasticity is dealt in a more optimal way. This is just one example, but there are many others that show how progressively the artificial brain resembles, slowly but with a progressive evolution, the crucial parts of our brain without having to be dramatically simplified. (Cukier, 2014)

#### 5.2.4 Characteristics

Most systems are defined by the number and interconnections between different neurons in different layers, the learning methodology in which the values of the weights and the function that converts the internal value of the neuron update its corresponding output.

Analyzing it from the unique viewpoint of functions, a network is a function that is represented by a series of functions which in turn can be represented by a number of other functions. This architecture is responsible for finding the dependencies between the data loaded.

In the context of optimization, we can see the behavior of this series of functions like an input  $x$ , from which a number of vectors  $v$  are obtained, which in turn are transformed into other vectors, or a vector with a number of dimensions to finally become a one-dimensional function. From the probabilistic point of view, the output value depends on a probability of  $a$ , which in turn depends on a certain probability of  $b$ , which comes from some probability  $x$ , where  $x$  is the variable initially charged.

If our network only allows one sense of direction, as the cases described in the previous paragraph, is called feedforward and is able to represent problems of considerable complexity given the ability to learn the weights if they were not limited by the computing power. If there are loops or the flow of the output is redirected to certain layers or previous neurons then it will be called recurrent.

#### 5.2.5 Learning

The ability to learn as we have seen is what makes these systems interesting. This implies the need for an objective function, preferably convex, which seeks to minimize or maximize its opposite, during the learning process, iteration after iteration; we will be looking for the space of possible solutions for which our objective function is the smallest. Whereas our space depends on the variables that are teaching the system and the possible risk of being

trapped in a local minimum generates certain aspects to take into consideration to properly configure our network.

A typical feature is the mean-squared error (that can be viewed from a broader perspective as a way of seeing the variance being obtained), which measures the difference between the value obtained and the value that should have been delivered (supervised learning). If we minimize this function by the method of gradient descent (which we will see in detail, the reason why it is desirable that the objective function is also easily derivable) we have the known and aforementioned backpropagation algorithms. This algorithm updates the values of the weights, using the error obtained by mse which is calculated on each iteration, either using the complete set of data or part of it. First is derived with respect to the parameters, and the gradient with respect to the weights are updated in successive layers, opposite the processing direction input values. Thus backpropagation is responsible for efficiently calculate the gradient of a nested function, generated by the deep network.

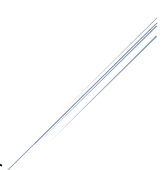
Although there are many learning algorithms, most of them are based on the previous example and they are applications of optimization theory and statistical estimation. (Hugo Larochelle, 2009)

### 5.2.6 Properties

In a study conducted by Hava Siegelmaan and Eduardo D. Sontag, it is demonstrated that with a specific recurrent structure and rational values (previously it was thought real values were necessary) in the weights is sufficient to create a universal Turing machine (it can emulate any other machine with any variable). (DeepMind, 2014) They have even shown that the use of irrational values in the weights gives the power of a super-Turing machine.

Turning to practical cases the parallel processing capabilities and adaptation are the most practical while fundamental. Other features are the capacity of information that certain network can store, once extracted from the data set. Convergence to a global minimum which will in turn depend on the definition of the objective function. Not forgetting that many methodologies become unviable when the data or parameters of the network are very high, because depending on the data treatment functions it may require exponential growth.

In the previously cited text we must add all the limitations and problems drawn from the machine learning theory exposed as an introduction at the top of this document. Some of these limitations as the bias-variance tradeoff can be delimited effectively in a confidence interval assuming a normal distribution if we use the mse objective function. Others may be reduced or redirected using other functions and the needed data pre-treatment or network configuration and settings according to the type of input data and the purpose of the network.



We briefly expose part of their main problems when applying all this theory to real cases in a practical and efficient way. Like any machine learning system is necessary to provide the model with sufficient data, containing sufficient information so that the algorithm can generalize correctly when it is exposed to new variables.

### 5.2.7 Criticisms and counterexamples

One of the main problems we do have to recognize is relative to the amount of processing consumption, or in other words time, although as a rule very good results are obtained the amount of human time and complexity required for its development and the cost of computing are often far from being the most efficient. On the other hand, thanks to the evolution of GPUs it has enabled the scientific community to expand its uses while developing better adaptations according to each field, rather than profound advances directed or targeted to specific purposes.

For example, in many works of this type of models, it is argued that once you have an effective and capable model, it is best to focus on the collection and preprocessing of data rather than in the optimization of the model.

Another point to recognize is that the use of neural networks has spread and has proven effective for a wide range of sectors and real-world problems. We can really understand the process and internal findings of the model or not, but we must not forget that if something works, there is no reason to push it away because it is too complex or expensive to be internally analyzed in detail.

Advances in deep learning in unsupervised learning are progressive, with some limitations, able to overcome many of the algorithms that in recent years had reached the top positions in various problems they deal with a wide range of researchers now, including traditional neuronal networks, Support Vector Machines or Support Vector Regression and Hidden Markov Models. However, as we will see the optimization process and the goal-requirements are a key aspect to choose the optimal model.

Finally, if processors finally give the big leap and allow analog operations that would cause a terrifying growth of these algorithms that loaded on specially designed machines for processing will be able to have unimaginable capabilities.

## 5.2.8 Types

There are dozens of types of neural networks based on the software; in turn each type has multiple configurations that can use some of the ideas of others so we have a huge collection of algorithms. Advances in various aspects of machine learning accompanied by the growing popularity of artificial neural network in recent years and the drive that it has received from the "deep learning" has increased the range of models and they are expected to continue to grow as they are optimized and adapted to specific tasks.

Starting with the feedforward networks which we could be considered the simplest since they don't have any kind of feedback, ie information during the learning process only goes in one direction, later adapting the weights with some methodology as seen in previous sections.

Later we do a brief description of the most important features of some of the networks that are achieving better results in the fields that interests us, in order to not overextend the document length we will not enter into details or specific models. (Yann LeCun, 1998)

### 5.2.8.1 Radial basis function

Without any doubt, RBF-based models are one of the most interesting nowadays. Their outputs are a combination of input values treated with the model parameters along with radial basis function as function of activation. In a simplified way, this function measures the distance of a point with respect to an origin, hence its name with the following formula generalized.

$$f_x(\vec{x}) = \sum_{i=1}^k c_i h_i(\|\vec{x} - \vec{z}_i\|) + p(\vec{x}) \quad (5.1)$$

Where  $x$  is the vector with input data,  $z$  centers,  $c$  coefficients,  $h$  the scalar function and  $p$  a polynomial function. It Works the following way, first the distance of the vector of entry to their respective centers is calculated. These values are transformed by the function  $h()$  to be subsequently added to the input values modified by a constant and  $p$ . Generally the functions that are often used to define  $h(x)$  are multiquadratic (i.e  $\sqrt{x + y^2}$ ) or gaussians (i.e  $e^{-x/y^2}$ ).

Since the values are real results, we can normalize the output values if we consider they do not conform entirely to our problem or provide meaning independently.

You can also apply this methodology to represent nonlinear data or multi-layer networks, although its more widespread use is the monolayer. Depending on the characteristics of our problem we can set the network one way or another. As it will be done in the development of this model in order to compare it and it is introduced with greater depth in the corresponding section of Radial Basis Networks. (Kon, 2006) (Schilling, 2002) (Leung H, 2001)



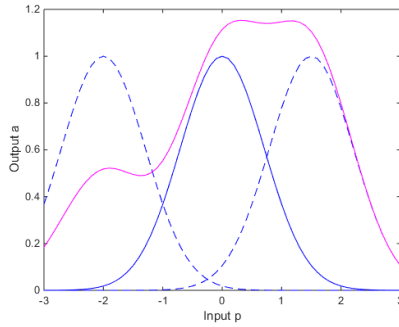


Figure 5.13 - Weighted sum of Radial Basis Function

### 5.2.8.2 Self-Organizing Map (SOM)

Based on the known Learning Vector Quantization (LVQ), it represents an algorithm generally used for dimensional maps, the main difference with the majority of neural networks is the use of neighborhood function (in extremely simplified manner the neighborhood will be the space in which the point can vary without losing its characteristics) in a way that does not lose the topological features (characteristics of space that makes up the neighborhood mentioned above) in the training phase. It is the expansion of LVQ for supervised learning applied to unsupervised learning.

In the image we see the learning process of the algorithm, having the dataset defined by the blue region, and being the white point the first value in reading, the structure of the map is able to adapt to the values of a set of inputs.

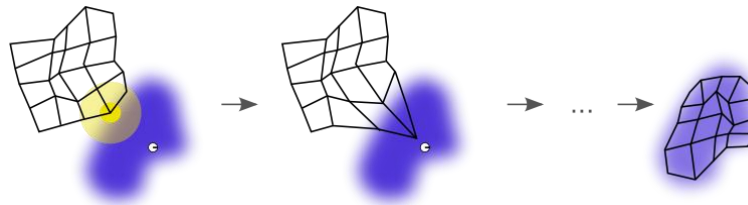


Figure 5.14 - Process of SOM learning

In practice, we are learning to classify data vectors in different groups of subspaces according to their differences, the definition of "difference" is where part of their differences with other models reside. In our case we use k-means as we shall see in the next item.

Another characteristic feature of this approach is that each neuron is able to affect the neighboring group in the space, similar to an RBN with multiple centers and high radios. This can be a positive factor because it allows us to learn the data space distribution and the topology of the training.

When choosing which topology function we will use, or how the neurons will form the network to map the subgroups, we can choose from a variety of functions as long as our processing limits and/or time will allow us. Similarly, the distance between neurons can be calculated by a distance function quite freely in the choice.

We have mentioned previously in the second approach to competitive learning that there are similarities with SOM. In the example, if a winning neuron affects a number of neighbors proportionally - while other rules can be applied -, these will focus on its type of scenario, very similar to the result that we get with SOM. With each iteration the neurons groupings will be representing better characteristic vectors while neurons near the borders will tend to move away if the topology and the distance function to facilitate it. (Ceperic, 2013) (Bullinaria, 2004)

#### 5.2.8.2.1 k-Means Clustering

It is responsible for dividing the data into different clusters, in our case we will implement it in order to divide the set of data by clustering to help us better define the scenarios, patterns, and decisions. Although it is a problem that requires a lot of processing, heuristic algorithms that help to converge more quickly in optimal locations have been defined.

The definition of difference as well as the formulation that defines the optimal, either by minimization of the objective function, Gaussian distributions or a mixture of both, has substantially improved the results that are expected from these models. These approaches generate groups with significant differences and therefore the behavior and specialization of each subgroup of related neurons is modified.

The advantages of hierarchical clustering (either by an ascending or descending process) reside in the greater efficiency when dealing with large data sets. This way each value is represented in the space defined by a certain number of dimensions. Because of this, we can search the distances in this space between each value and grouped them according to the number of desired clusters. This approach, which also can be defined according to the distance to the "next" point in the space, creates a lot of variations in defining the final algorithm.

The centroid is defined as the point where the subset of points is smaller, similar to the RBN Center. This way of defining the cluster can vary in such a way that we could achieve a space defined by cluster as compact as possible and where their respective points differ significantly from the rest of the cluster.

The most typical way to iterate, is a heuristic optimization problem which seeks to minimize the distance from each point to its centroid. The points may vary from centroid until

the sum of the distances between each point and its centroid, may not be reduced any more. (Pham, 2004) (Tapas Kanungo, 2002)

### 5.2.8.3 Recurrent Networks

There are dozens of algorithms that would fit within the definition of recurrent networks; many of the ones analyzed in some detail in this document are of this type mainly due to their memory property. Another of their properties is due to some dynamic capacity given the fact of being in states of change or adaptation according to the learning methodology. Other recurrent networks are called in relaxation when they are changing their parameters (weights and bias) and temporary processing while only applying the stored values.

Parts of its drawbacks are a possible chaotic behavior, for such cases the problem should be treated with a model more focused on dynamic algorithms. On the other hand, most of these models tend to require processing time, which grows inefficiently when the number of inputs is very high which makes them less practical models in these cases. This problem can be reduced if we convert the problem from continuous to discrete, with this we'll lose some accuracy but not necessarily its usefulness. However, depending on the characteristics of the problem to treat various configurations of these recurrent models have proved to be at the highest level when they are used in conjunction with other algorithms for pre-processing or post-processing of data.

In the following sections, and more especially when we develop recurrent network models, we will carefully review some stages, configurations, and some of the types used when we want to focus the results to the forecasting.

### 5.2.8.4 Boltzmann machine

Networks such as the Boltzmann machine that could be considered similar to a Hopfield NET with some variations (Monte Carlo), in such a way that it forms a recurrent and stochastic network, although they are based on mathematical foundations and very interesting ideas they have too many problems when it comes to put them into practice, for this reason they won't be described. For example, with this approach is necessary to deal with more important the risk of falling into a local minimum. So random values or functions, methods, proper Stochastic Neuronal Network, are used to reduce this drawback up to some extent.

They have had lots of study, due to the interesting approach of connections and part of its findings are applied to Markov Models as we will see later. Moreover, improvements in the learning methodology have been developed to deal with their efficiency as Restricted

Boltzmann Machines, which has proved to be useful for various purposes such as recognition. However the majority of studies that we are interested in, in case of being related, is using derivations of this model and not the model itself. (Roweis, 2010)

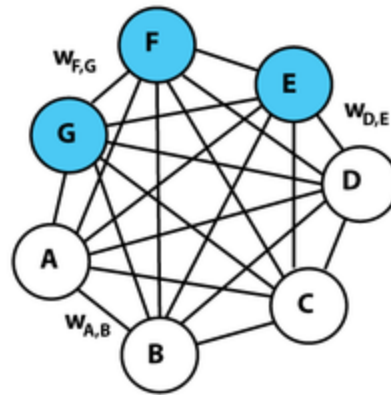


Figure 5.15 - Boltzmann nodes interconnections

#### 5.2.8.5 Reservoir Computing

It uses a dynamic system, known as a reservoir, which is loaded with input values, and which is capable of independently training to achieve the desired output. The biggest advantage lies in the possibility of having a great capacity for adaptation for this reserve yield better performance than other models in similar problems.

Most of the configurations of artificial networks could be considered to be recurrent and therefore tend to some instability. Echo State Network, based in part on Reservoir Computing, focuses on this problem in such a way that it tends to be used to represent simple problems of time series with certain characteristics, it differs from the rest in the limited interconnection between its neurons. Which, summed up to the fact it can only vary the weights, allows that while still having the ability to approach to non-linear models, the error function is easily derivable (quadratic, since it can only vary weights) and therefore transformed to a linear system. Without forgetting its greater dependency on the initial values unless a specific training.

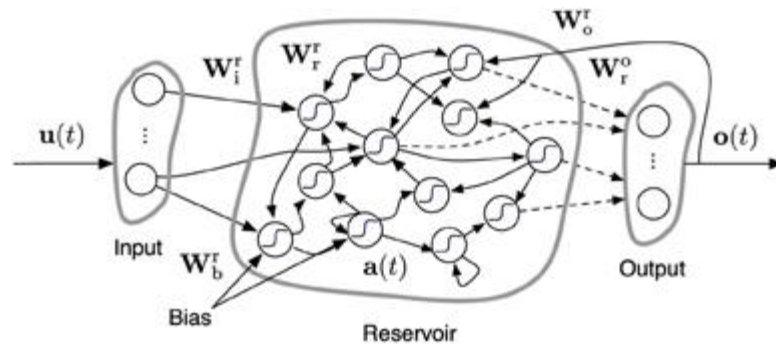


Figure 5.16 - Reservoir Network architecture example

The trend in recent years has been to combine a number of approaches and exploit their properties to improve efficiency and/or accuracy. The following picture shows a typical example, which aims to extract the information stored in a data set with some of the types of neural networks and subsequently applied to other models to give greater value to the conclusions that the first model has reached. Deciding which analyzes should be performed on the input data set before processing it is complicated, because tools such as Data Mining can clean noise or information that initially didn't add value to our model but may at the same time be delimiting subsequent information that the network or model could extract from the dataset.

Whether the first stage is under an unsupervised model or the network is supervised, deleting information or grouping datasets in information clusters will be subtracting generalization ability and we will be back to the bias-variance tradeoff problem. For this reason and without forgetting that the quality and quantity of data that we give to the model will be significant in obtaining a model better suited to our requirements, the scientific community is focusing on the process of that information taken directly from the dataset, similar in principle but with refinements.

As we know one of the main problems of neural networks is learning and its computational cost. So this methodology, which main advantage deals with this problem, is becoming one of the paths chosen to skirt these current limitations. The following picture shows the application of a Neuronal Reservoir model limited in memory usage with a slight modification of Online Algorithms. As we will see in the next section, it is applied to extract information from the iterative changes that are produced in the reserve while new observations are added. The use of the formulation based on Bayes, as we will see later, is used to get probabilistic values rather than specific predictions.

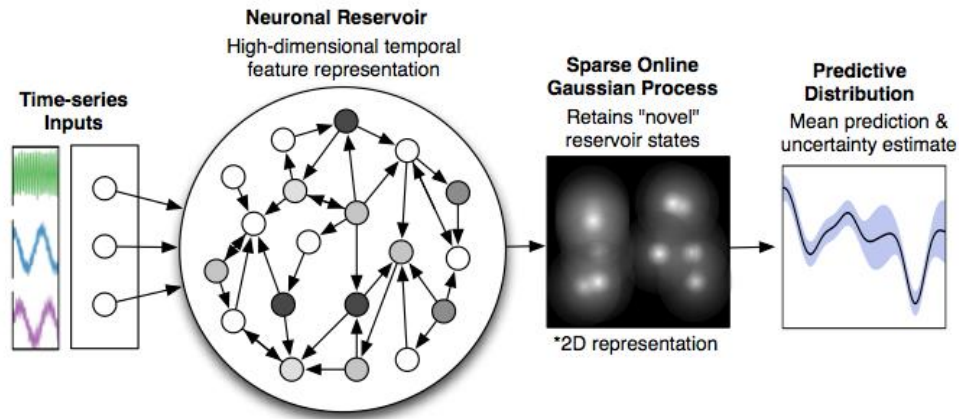


Figure 5.17 - RN with Sparse Online postprocessing

In this way a single model gets probabilistic results (thanks to Bayes) and continuous and dynamic learning (thanks to Online Algorithms). All this is possible thanks to the advantages in terms of memory usage and simplification of the order of complexity that offers Neuron Reservoir, besides and as an even greater incentive this mixed model can be applied close to real time. (Harold Soh, 2014) (Benjamin Schrauwen, 2007)

### 5.3 Online Machine Learning

Sometimes the data set follows a sequential evolution, with some input data that point to output values, as it is part of our main problem. In these cases the model is being updated as new data is coming, we have other ways in addition to the already known feedback in neural networks. This time we'll talk about Online Machine Learning, whose memory is adapted to the input data and targets received without changing its size which carries its consequences. As in all previous cases, these approaches can be paired with others to adapt the algorithm or make it more flexible, in this case reducing processing time.

The initial approach is exactly the same as in most of the machine learning algorithms, as we have already detailed, we seek to minimize certain objective function, which as a general rule, will be the error or risk, but the interpretation is different as we will explain later. What differentiates a big part of the model are the functions and limitations that we will include in our algorithm.

In a simplified but not very far from the reality case, our model in the  $t-1$  stage, we will only store one function, in the next iteration  $t$ , this function is updated with the new values of input and output targets. In other versions, the function in  $t$ , will depend on the function in  $t-1$

and also all the subsets of values previously analyzed, this directly affects the resources needed to find a solution while it is impossible to adapt the function of  $t$ , in a single iteration.

We have the values of input  $X \in R^d$ , the target values  $Y \in R$ ,  $H$  is the set of functions  $w \in R^d$  and  $V$  is the objective function, that as in all other occasions we will assume to be convex or more directly differentiable. As we have described we will have the function in  $w(t + 1)$  which will depend on  $w(t)$  minus the gradient of the objective function of  $\{x(t), y(t)\}$  applied in  $w(t)$ .

$$w_{t+1} \leftarrow w_t - \gamma_t \Delta V(\langle w_t, x_t \rangle, y_t) \quad (5.2)$$

If we use recursive least squares (RLS) algorithm, which seeks to minimize the average mse objective function value, as the majority of the cases of regression, looking for values iteratively as in a filter. The mathematical proof of this algorithm is long and complex and does not fall within the scope of the document.

Its conclusions applied to our interpretation give us the formula with which we can update our function. Not only that, the complexity of the algorithm is an order of magnitude less than the problems were all the subsets are considered at each stage and the memory required does not depend on the number of stages since it is constant, thus we can get more efficiency if we apply the Bayesian statistical inference for example. (Jonathan Eastep, 2006) (StatLearn, 2003)

$$\Gamma_i = \Gamma_{i-1} - \frac{\Gamma_{i-1} x_i x_i^T \Gamma_{i-1}}{1 - x_i^T \Gamma_{i-1} x_i} \quad (5.3)$$

$$w_i = w_{i-1} - \Gamma_i x_i (x_i^T w_{i-1} - y_i) \quad (5.4)$$

## 5.4 Hidden Markov Models and Bayesian Networks

To understand how this methodology works, it is necessary to understand the theorem on which is founded, the theorem of Bayes, which is useful mainly when it comes to decision support rather than forecasting problems. The bases of this theorem have been explained in the theory section of this document.

This section continues with the explanation, focusing on the prediction of time series based on the theory developed by Andrey Markov. One of the main underlying properties is that the result does not depend on previous processes, but only in the process currently addressed. The applications found for this property is immense and cover practically all the fields in which match can be applied in one way or another. The mathematical properties of this stochastic model where only the previous stage is considered, include reducibility, periodicity, transience (recurrence, expectations of values, states), ergodicity and states of balance.

Here's an example of a chef known as Chip and a regular friend named Eric. Eric always chooses his dinner depending on what he has eaten that day. Meanwhile, cook Chip does not know exactly what his friend has eaten, but does have some ideas, so depending on the choice of Eric for dinner (in Markov's Models called observations), our cook is known with more or less certainty what Eric has eaten. For Chip, the food eaten by Eric is unknown, but he knows that it should be between some "historical choices", ie a discrete Markov chain. Chip knows some probability of Eric behavior, for example, what he usually eats on a Sunday or Friday, or that he never wants fish on Monday... so, then Chip can guess or have some "hypothesis" of what is what Eric is going to order that day.

If we write this approach in code, we will start with a medium probability of what Eric has eaten based on comments received by Chip. The balance will come when we take into account the probability of change, in which the likelihood of change is represented regarding only the previous election. That is, if Eric in  $t$  has eaten chicken, there is less chance than in  $t+1$  he will eat pizza and therefore Chip will have delimited with greater certainty what Eric might choose for dinner.

There are a lot of methods to learn from observations along with the uses of the Markov Model properties with other algorithms or recognition or forecast techniques. In this case, and taking the basis of Bayes' theorem and Markov chains, we will see a dynamic version an algorithm based on these two models.

Thanks to the basis in which, to predict a future event we only consider the current values, and the parameters learned, we can greatly simplify our forecasting model. So if we rewrite Bayes' theorem it becomes.

$$P(Y_{1:T}) = P(Y_1)P(Y_2|Y_1) \dots P(Y_T|Y_{T-1}) \quad (5.5)$$

If we want to give more freedom to our models, we can allow "temporary" jumps, i.e. we can accept that it jumps from  $Y_{t-n}$  to  $Y_t$ , this adds dimensionality, but allows us to encompass more complex nonlinear problems. We can also set up the model so that each observation depends on a hidden variable (in our previous case, directly associating a meal with a distribution of possible dinner) which we will call a state, being the sequence of states a Markov chain.

Regarding the learning methodology and operation, let's make an example that allows us to build on the models. We have a new observation and we want to incorporate this information into our model, as in the following image, from  $n$  we will update the values of the two dependent subspaces of  $n$ . In this example  $e^-$  will receive the probability of a certain  $n$ , given certain configuration  $e^+$ , if values are real, the probability will be given by the probability density between the values that can take  $n$ . The values that  $n$  will send to the subspace  $e^+$  will be the probability of observations received by  $e^-$ . In such a way that the probability of  $n$  will be proportional to the observations of  $e^+$  weighted by the conditional probability of  $n$  given  $e^+$  and observations of  $e^-$ .



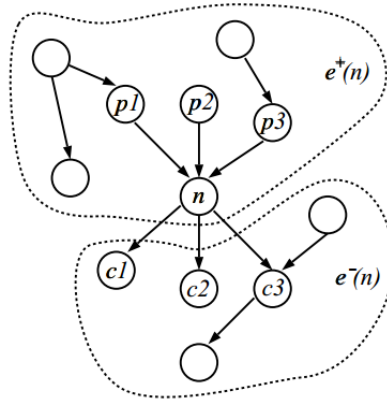


Figure 5.18 - Bayes Network dependencies

In the above example, we have a very simplified graphical representation of dependencies between variables, and the vectors represent the conditional probability that exists among variables. In most of these dependencies, when we use the definition of dependency provided by Markov, they are calculated by the Markov Blanket (image and top paragraph), where the subspace  $e^+$  are considered parents and the  $e^-$  children, the operation are exactly the same as described previously. In such a way that it is only that subspace of parents and children the one necessary to predict the behavior of the node  $n$ .

Updating the stored information is typically done through what is known as Bayesian inference. It's based on the application of Bayes' theorem and various axioms, as we know in this approach a degree of expectation is assigned for the belief that with some probability a hypothesis will be fulfilled. In theory, with the update of successive observations, arguably in a finite number of iterations the accumulated information will reach the same state. (Ghahramani, 2001) (Heckerman, 1996) (Geiger, 1990) (Sharon-Use Normanda, 1992)

#### 5.4.1 Baum-Welch algorithm and Markov property

Let's look at the implementation of the Baum-Welch algorithm which generally is used to efficiently find parameters of Hidden Markov Models. We will first explain its operation and will end up continuing the example exposed according to the understanding of Bayesian probability in Hidden Markov Models. While in the application we use Expectation-maximization algorithm, in the example we will observe a simplified way to use forward-backward algorithm (in this case also called smoothing), to calculate the parameters with the estimated observations, although the Viterbi algorithm teaches a more optimal way to converge.

Previously we will mark some limits of use, if our problem is discrete then we will use Markov Chain. If it is a continuous problem, for instance the treatment of a signal which provides information to the model, then we use Markov Process, generalized in Wiener Process.

In general, both cases are stochastic events with the Markov property where the conditional probability distribution  $P(A|B)$  of the future value of a variable depends on its present value, but not on the history of that variable.

We continue our search for a model that is able to understand and predict a time series. Given a set of observations  $O_{1:t} = (O_1, \dots, O_t)$ , iteratively we will seek the model parameters  $X_k \in \{X_1, \dots, X_t\}$  that maximize the likelihood  $P(X_k|O_{1:t})$ . To do this we need to calculate the expected number of transitions leading from O to a state i, and again the expected number of transitions from state i to a j one. To this end we define the probability of being in state  $i(t)$  and  $j(t + 1)$ , efficiently calculating with Forward-backward algorithm. Subsequently, we iterate updating the values, being the most optimal sequences between states i have given some observations O, the one obtained by the Viterbi algorithm, which exceeds the scope of the present document.

With our example, we want to predict Eric breakfast by this methodology, but the choice of Eric depends on what he had the previous night. Let's suppose, to simplify, that there are only two variables; sleeping S or awake W. A priori, we ignore both, the likelihood of if has not slept a day that will he do the next, and vice versa, and what breakfast will he choose depending on what has happened to him that night.

First, we try to guess the initial point, as well as the transition and the emission matrices. Suppose that the probability of if he has slept a night that he sleep the following is 0.5. Besides supposing that if he has slept that night he will ask toasts with tomato with a probability of 0.7 and eggs with bacon with 0.3. The rest of the initial odds for cases of the night awake are the values that normalize these probabilities. In the tables below, we can see the early estimates of values and the observations available.

Transition		
	S	W
S	0.5	0.5
W	0.3	0.7

Emission		
	E	T
S	0.3	0.7
W	0.8	0.2

Initial	
S	0.9
W	0.1

$$O = (TT, TT, TT, TT, TE, EE, ET, TT, TT)$$

If we calculate the probabilities of each transition if S->W and the sequence where there is more probability of finding such a sequence, we get the following table.

Observed sequence	Probability of observed sequence and state is S->W	Highest Probability of observing that
TT	0.024	0.358 WW
TT	0.024	0.358 WW
TT	0.024	0.358 WW
TT	0.024	0.358 WW
TE	0.006	0.134 WS
EE	0.014	0.049 SS
ET	0.056	0.089 WW
TT	0.024	0.358 WW
TT	0.024	0.358 WW
Total	0.220	2.423

If we perform these calculations for all observations we get the "Pseudo probabilities" of going from S->W which is  $0.22/2.423 = 0.0908$

The way to continue iterating would be to recalculate the transition matrix, normalizing the values and estimating the new array of emissions, with new observations.

Transition'

	State 1	State 2
State 1	0.3973	0.6027
State 2	0.1833	0.8167

In the following table only W->E appears, but we need to update all the pseudo probabilities, we should do so for, W->T, S->E and S->T.

Observed Sequence	Highest probability of observing that sequence if E is assumed to come from W	Highest Probability of observing that
ET	0.1344 WS	0.1344 WS
TT	0.0490 SS	0.0490 SS
TE	0.0560 SW	0.0896 WW
Total	0.2394	0.2730

The new estimate we get for "E" if we come from "W" is  $0.2394/0.2730 = 0.8769$

If in Eric's case we converge sufficiently in the first iteration, we could conclude that the probability that not sleeping if the previous night he slept  $P(W|S)$  is 9% and in these days the percentage of he having eggs with bacon for breakfast is 88%. (Chang, 2007) (Agosta, 2004) (Baggenstoss, 2001) (Pitt, 1971)

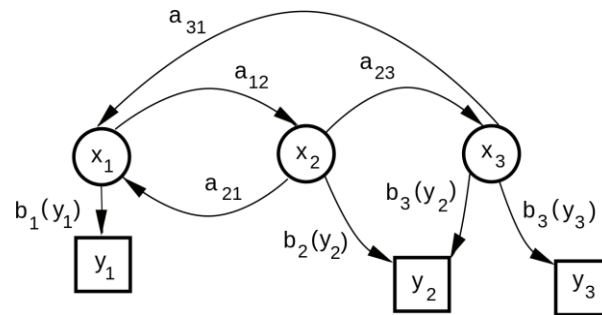


Figure 5.19 - Example of discrete-time Markov process

## 5.5 Bagged Trees

There are currently several versions of decision trees, these models have grown from the need to take out valuable information from the immense databases which nowadays keep increasing their pace of growth. In this case the analysis focuses on classification instead of forecast, but due to uses that are being found and the great achievements that are being reached in recent years in data mining is worth understanding the approach as well as the generalization and different versions of these models.

Most of the algorithms for predicting discrete or continuous values, either oriented to classification or with a statistical basis, have some memory. In this case we create rules that allow us to analyze data on an individual basis. Using the greedy algorithm approach which allows us to find the quasi-optimal solution in each iteration. Subsequently, several algorithms have been developed which have turned these models into very practical solutions depending on the entropy that exist in the database to analyze.

Iterative Dichotomiser 3, which evolved more into a statistical classifier with the C4.5 algorithm, included several improvements such as allowing continuous and discrete values by dividing the data in a threshold. Another substantial improvement is to allow certain values to lack features or to read the tree in order to find the branches that don't create value and replacing them with potential candidates. These were the main problems when developing tools, so this supposed a considerable advance, improving some of the features with the version C5.0 of the algorithm, mainly aimed at the computational cost. In addition to those mentioned previously there are various techniques that allow us to efficiently find non-linear relationships, which is useful in the field of prediction.

The model is presented with nodes as class definitions and the branches as the features. Its graphical representation shows the data and no conclusions, but it does not prevent that this output can be used in decision models. (P. Rivas-Perea, 2011)

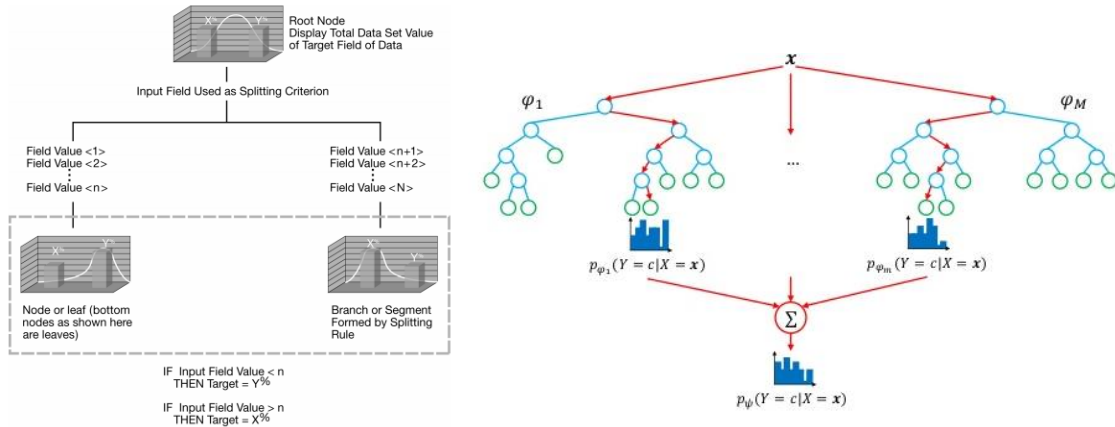


Figure 5.20 - Decision Tree and Random Forest

Summarizing their characteristics we can conclude that it is a Non-parametric model but consistent. It can be adapted to the problem in question so it is able to handle heterogeneous data. In our examples it achieves good results with continuous and discrete values. It is very quick and easy to train, to predict and interpret. While a single tree gets very small bias and high variance, this can change if we use forests, in such a way that we reduce the variance to increase the very low bias. The best way to adapt the decision tree to the problem is changing the functions that generate the leaves and which pruned it.

If we use random forest probabilistic can get value at the same time that we better adapt the model. This algorithm has a number of advantageous features compared to neural networks. Virtually no need for tuning since fitting the same, control of the bias-variance is simple using randomization to get the best performance. Continues to be fast in training and forecast, but losing performance against a single tree. There are other models based on trees such as Gradient Boosted Regression Trees that adds a function to better adjust the time series. Generally, when GBRT is recommended to use it, it produces better results, is more flexible in terms of the objective function, allows regularization but adjustment is necessary and significantly increases the time required in the training.

## 5.6 Multiple Linear Regression

Without a doubt one of the methodologies most frequently used in prediction of load curves are the ARIMA models. These models are mainly useful when there is a relatively constant level over time, i.e. they are "stationary" or don't have seasonality in which case we should use SARIMA models. Although there are several methods for the analysis of time series, we can divide them between methods of frequency and time, which are related by the Fourier transform. In our case, we will focus on the domain of time and verifying that the relationships exist and how they affect the final values.

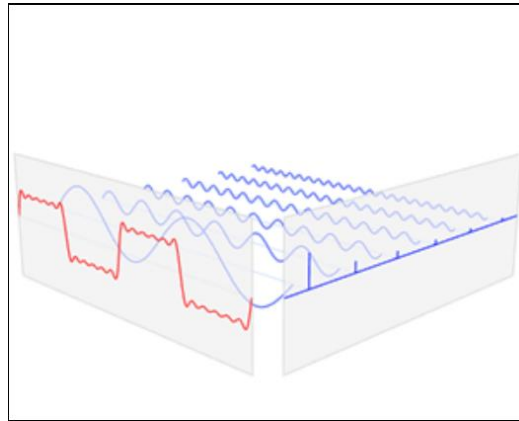


Figure 5.21 - Time and Frequency domains

Time domain is shown in red and in blue is the conversion by Fourier transform to the frequency domain, or to put it in another way, the composition of the red function given the amplitudes of the blue functions.

Other ways of dividing the time series analysis can be between linear and nonlinear or between parametric and nonparametric. In the case of parametric, we understand that the process can be described by a number of parameters and therefore the aim is to find which ones best fit the loaded values, although as we have seen in *chapter 2.3* there are several limitations. Non-parametric processes seek covariance (dependency between variables) without necessarily having data structure.

Returning to the ARIMA model, and without going into depth on mathematical aspects, let briefly see how it is constructed and what it is based on. An auto regressive model or AR attempts to explain certain values set by a linear dependency of the previous values plus an error. The nomenclature is AR (B), where *B* is the order of the model, or the number of previous observations that are considered for the value *t*.

$$Y_t = \phi_0 + \phi_1 Y_{t-1} + \dots + \phi_p Y_{t-p} + a_t \quad (5.6)$$

Generalization for AR(p)

$$\phi_p(L)Y_t = \phi_0 + a_t \quad (5.7)$$

In theory the mean of the error must be zero, its constant variance and the covariance between errors of different observations should also be zero. If we abbreviate the upper expression and define our delay as *L*. Our delay slows the values considered so in general low values are used or values that match the periodicity of the series to be represented.

$$\phi_p(L) = 1 - \phi_1 L - \phi_2 L^2 - \dots - \phi_p L^p \quad (5.8)$$

Another model that is known as a moving average model or AM, seeks to give a value in a time series regarding an independent valuation and an adjusted weighting of previous errors. The formulation and meaning is the same as for AR models.

$$Y_t = \theta_p(L)a_t + \mu \quad (5.9)$$

Now we can go into the ARIMA models, before applying our model we have to define the order of its components (p, d, q). That defines the auto regressive function, the integrated and the average mobile respectively. The resulting function is.

$$Y_t = -(\Delta^d Y_t - Y_t) + \phi_0 + \sum_{i=1}^p \phi_i \Delta^d Y_{t-i} - \sum_{i=1}^q \theta_i \varepsilon_{t-i} + \varepsilon_t \quad (5.10)$$

Where  $\phi$  is a constant,  $\varepsilon_t$  is the error,  $\theta_i$  are the values of AR, MA and  $\Delta^d Y_t = (Y_t - Y_{t-1})$  used to convert the series into stationary. To use this model properly we must be able to identify the functions of full and partial autocorrelation in our series and compare them to the models we obtain independently of AR, MA and ARMA. Another simple way is to observe the error obtained by our model and see if it meets the theoretical requirements of the error.

How many parameters of each model must we incorporate will be the main problem when setting our model. For this, besides the two ways previously exposed, we can use the Box-Jenkins method.

This method consists of three steps. First, to make sure that the variables are stationary, identifying the seasonality in case of having it. The first stage is to determine if is stationary which can be seen in the autocorrelation graph, if it is not stationary then the graph should decay slowly but progressively. In the same graph, we can see if the series has seasonally and if so derive the function, even though there are other methodologies to reduce this seasonality. Subsequently, it should identify the value of order for p and q, for this purpose there are multiple methods, in our case will do so based on the autocorrelation and partial autocorrelation plots.

The verification of the model's results, including that the residues or errors fulfill their theoretical properties. As it will be seen in practice it is not so simple, although there are generic rules based mainly on the role of autocorrelation, choosing which model and which parameters (order) to use as stated in this methodology is part art and part science. Moreover, the choice gets complicated when we gather several models, so lacking experience the method of trial and error will be the one to use to find the configuration that best fits our series. (Hongzhan Nie, 2012) (L. Suganthia, 2012) (Mehdi Khashei, 2010)

### 5.6.1 Stepwise regression

It refers to a typology of models in which the selection of the variables is performed automatically. There are several approaches as well as different selection criteria, being the most used: t-test, F-test, R-square, Akaike information criterion or Bayesian information.

Bayesian information (BIC) and Akaike information criterion (AIC) are used to measure the quality of the model, and is normally added a penalty term to avoid overfitting. In the case of BIC models they are compared from the perspective of decision theory, as measured by expected loss and in AIC models are comparable from the perspective of information entropy.

Because of these differences in the final processing of the model varies substantially, especially because we are using the same technique to measure the efficacy of the model as we use to make the adjustment. This causes that the models are worse than they appeared initially, the same occurs with the degrees of freedom which must be previously defined and we should not only consider them as the resulting variables chosen in the setting.

Although these models have considerable criticism for oversimplifying the data set, in certain circumstances once they have been verified by another method and dataset not previously used in the initial setting, they can be a practical way to adjust the regression to the required level.

In our case we will be eliminating or adding data to our model according to their statistical significance in the regression. On each iteration the p-value of the model with and without the element in question, if the result indicates that the new hypothesis is more then we continue with the procedure and we increase the value of the required p-value to consider that information is being given to the model.

It is deduced from this method that depending on the order in which the variables are provided one or another model will be produced.



## 6. Neuronal Network Development

### 6.1 Introduction

At this point, it is explained how to create, configure, and train neural network along some possible improvements. We will begin with an introduction to the most fundamental aspects, the configuration of a neuron, different types of provision of these neurons that form the structure of the network and how different concepts about the training of these networks are implemented.

We will continue to complicate our network, adding several layers of neurons, commenting on the dramatic importance of input data, different types of functions that are applied to input data and that define, together with the architecture of the network, the global performance and capacity to learn the information hidden in the data provided. We will see the best-known learning algorithms and will give a few brief comments on the meaning of the values that our trained network returns and its results.

As we move forward, we will focus on the type of network that interests us for our purpose of predicting time series in the short term. After an introduction to dynamic neural networks, we will see what the structures are and algorithms of training that are achieving better results nowadays. We will explain a fundamental point of these networks, the delay, and some of the different architectures with which these networks can be built. Such as NARX (Nonlinear AutoRegressive with eXogenous inputs) which will allow us to predict values that depend on more than one input value in an efficient way for not too big databases. After some initial testing together with a few adaptations by trial and error, we will better parameterize the network's settings once trained, closing the cycle, i.e. giving the value calculated in the previous step as an extra input value and we will this network to predict values more distant in time.

We will continue focusing on our goal of prediction, but these may also be applied to the varied challenges of the power sector.

Finally, it is necessary to mention the problems and different solutions or improvements involved in this type of development. Similarly the hardware where they are processed as well as the necessary infrastructure to interacting with other models and applications is discussed.

Before going into details, part of the studied documentation proving their great benefits in predicting load curves will be listed. (WahHe, 2014) (Dalto, 2014) (Awan, 2012)

## 6.1.1 How a Neuron Works

### 6.1.1.1 Neuron

We will begin explaining how the simpler model of neuron works, in the case of the image below, we have a single input  $p$  to an output  $a$ .

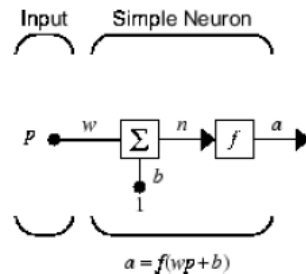


Figure 6.22 - Simple Neuron

Now the procedure that is described in the block diagram of the neuron is the following: first input  $p$  value is multiplied by the weight  $w$  forming the weight function, in such a way that in the block of the sum it reaches  $wp$ , where it joins the bias  $b$ . This produces the net input function  $n = (wp + b)$  which is given to the transfer function  $f()$ , once this function has been applied it will define the outcome  $a = f(wp + b)$ .

In this simple example, we have applied to the sum function to the weight function and the bias, but it could have been any other, according to the behavior that we want our network to have. We should mention that both  $w$  and  $b$  are scalar parameters that define the operation of our network according to their values, these are the values that we will be changing during the training phase, and these together with the network architecture will define how our network behaves with a certain dataset.

### 6.1.1.2 Transfer Functions

Although you can use practically any function that is derivable (for reasons of optimization in the stage of training) because of the advantages of the Log-Sigmoid Transfer function is the most widespread in the hidden layers, and a simple linear transfer function is the one used in the last layer of our network. Besides the advantage of being easily differentiable the Transfer Log-Sigmoid function limits the output values between (1,0) which is highly recommended especially if the values to be processed may have considerable differences.

Imagine that entry values or the values intermediate between layers (if you do not use this type of function) are of the order of  $10^6$  but the characteristics of the problem to consider are of the order of  $10^2$  when training the network, the weights and biases are changed, the

values won't be taken with the same consideration, since the difference between the target value (supervised training) or problem definition (unsupervised learning) as we will see later, could differ considerably.

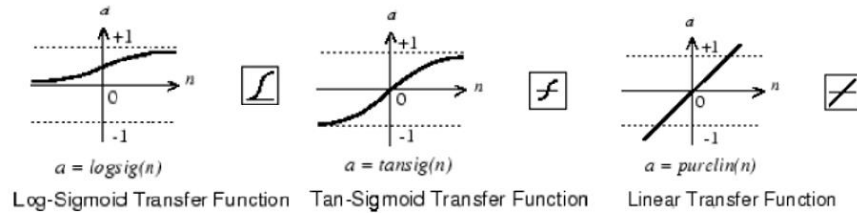


Figure 6.23 - Common Transfer Functions

### 6.1.1.3 Vectors as Inputs

In the majority of real world problems, we will not use a single scalar as input value, but vectors with several of these values. This directly forces us to define as many numbers of weights as values in the vector and then - now it makes more sense - we will apply the bias. Let's see the block diagram to better understand the procedure.

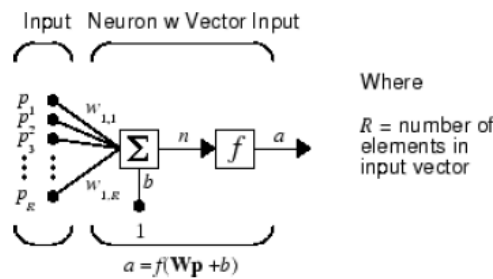


Figure 6.24 - Vector Inputs

As we conclude from the image, the vector  $p(p_1, p_2, \dots, p_R)$  is multiplied by its corresponding weights  $w_1, w_2, \dots, w_R$ , subsequently the bias and the net input function  $n$  is added, and the function  $f()$  that we have defined is applied.

$$a = f(w_1p_1 + w_2p_2 + \dots + w_Rp_R + b) \tag{6.1}$$

Now that we understand the basic operation of a neuron and the basis on which a neural network is built, let's take a moment to generalize the representation to expand our network in a way that is directly understandable with only a view its structure diagram.

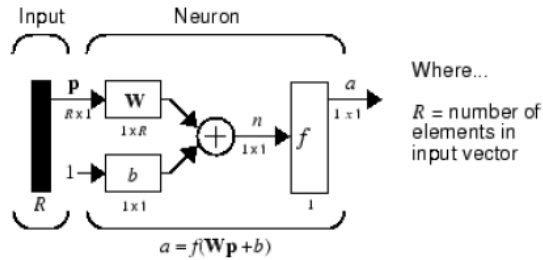


Figure 6.25 - Neuron

In this case, we have a vector  $W(1 \times R)$  as we have seen in the previous example, which is multiplied by a vector  $p$  of size  $R \times 1$  and then the  $1 \times 1$  bias is applied. After applying, in this case the sum, it returns a net input function  $1 \times 1$  that after being applied the transfer function will return the output of  $1 \times 1$ .

## 6.1.2 Network Architectures

### 6.1.2.1 One layer

In all the above examples we have seen how a neuron works. If we have  $S$  neurons in parallel, we have a layer of neurons. Let's take a look at the graphic example and its consequences.

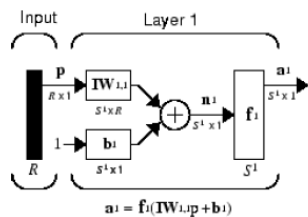


Figure 6.26 - One layer

Where...

$R$  = number of elements in input vector

$S$  = number of neurons in Layer 1

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,R} \\ w_{2,1} & w_{2,2} & \dots & w_{2,R} \\ \dots & \dots & \dots & \dots \\ w_{S,1} & w_{S,2} & \dots & w_{S,R} \end{bmatrix}$$

Figure 6.27 - Weight Matrix

Directly, we see that if several neurons are connected to input values, we need to define the weight of these connections for each neuron. Therefore the matrix of weights  $IW$  (input weights) will define the importance with which each neuron should treat each input value vector  $\vec{p}$ . Two considerations are necessary before proceeding, first as we can see from the annotation,  $R$  should not be equal to  $S$ , we may have an array with five values that will be processed by seven neurons in the first layer; on the other hand and referring to the system that will be taken into account in all work, the superindex of the values of the weights indicate, in the first place which neuron is being referred, and in second place the position of input vector, in this way  $w_{2,3}$  will be the weight which will take the third value of the input vector in the second neuron.

### 6.1.2.2 Multiple layers

Along with freedom of connecting neurons or layers in the way desired, the factor of having several layers of neurons is one of the main advantages of the neural networks, since is this depth which will allow the network to adapt to complex nonlinear problems effectively. Note that while in theory, one layer with a finite number of neurons could adapt to solve every linear problem; this is not a competitive advantage over other models. While again theoretically a network with only two layers, being sigmoid the transfer function of the first one of them and the second a linear function, can be approximated to any function.

Before proceeding to explain the operation of a network with several layers we know that just as there is an array of weights that connect with each of neurons input values, there are some weights that connect the output of one layer with the desired connections, in the following cases all the outputs of the second layer will be the inputs of layer three, once applied the matrix of weights that we will define as  $LW$  (layer weights) whose superindex as well as of the one in  $IW$  will define the target and source layer.

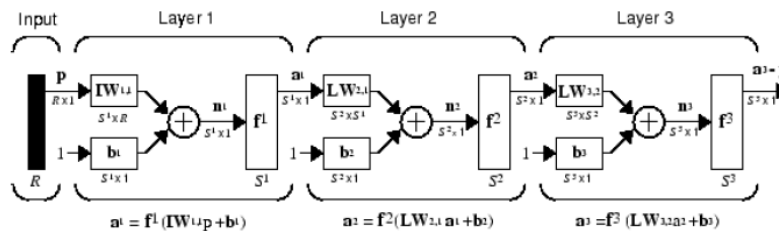


Figure 6.28 - Simple Multiple Layers

We see in the picture that the way of analyzing each layer does not differ from the simple examples above. Before continuing, let's define the last layer before the output values as output layer and the other layers as hidden layers, in this way the output will be  $a^3$  values, if required might be treated later by one or several functions , in the example of the image.

$$a_3 = f_3(LW_{3,2}f_2(LW_{2,1}f_1(IW_{1,1}p_R + b_1) + b_2) + b_3) \quad (6.2)$$

### 6.1.3 Concurrent and Sequential examples.

After treating the input data as we will see in greater detail later, we need to create our network, configure it, give some initial values to the weights and bias, train it, validate it and improve it. Before continuing with some simple examples, let's define as concurrent input values those who do not have a temporary unit or follow a sequence, those who comply with this will be denoted as sequential.

### 6.1.3.1 Static Networks

All of the examples we have seen previously, have been static networks, hence this example should not cause major problems, therefore it will help us to better understand the following example which introduces the concept of dynamic networks with very simplified.

Following the order described at the beginning of this topic, let see step by step the stages up to the output values. Having four vectors  $p$  as shown in the following example, in this case the treatment of datasets is direct (image), however, is in the more complex problems where the power of these networks resides.

$$p_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, p_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, p_3 = \begin{pmatrix} 2 \\ 3 \end{pmatrix}, p_4 = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$$
$$P = [0 \ 1 \ 2 \ 3; 1 \ 2 \ 3 \ 0]$$

Let's create and configure the easiest possible network, a static, i.e. without feedback or delays and with a linear transfer function. We will continue configuring the values of the weights and the bias as  $w = [1 \ 2]$  and  $b = 0$ , respectively.

The internal procedure that will make the network  $a = f(pw + b)$  will be:

$$a = [p_{1,1}w_{1,2} + p_{1,2}w_{2,2} + \dots + p_{1,R}w_{1,R} + p_{2,R}w_{2,R}] \quad (6.3)$$
$$a = [0 \cdot 1 + 1 \cdot 2 \quad 1 \cdot 2 + 2 \cdot 2 \quad 2 \cdot 1 + 3 \cdot 2 \quad 3 \cdot 1 + 0 \cdot 2]$$
$$a = [2 \ 6 \ 8 \ 3]$$

### 6.1.3.2 Dynamic Networks

Because of the temporary nature of our problem, as a general rule, we will use dynamic networks, we go more in depth on the following points, however let's see a simple example to understand its operation.

We have a dynamic network with a delay equal to one, i.e. the input values follow a sequence of value one. To further simplify this first example we omitted the bias and we will set the weights with the same values than in the example above  $w = [1 \ 2]$ .

$$p_1 = [1], p_2 = [2], p_3 = [3], p_4 = [4]$$
$$P = \{1 \ 2 \ 3 \ 4\}$$
$$a(t) = w_{1,1}p(t) + w_{1,2}p(t - 1) \quad (6.4)$$

$$a(*) = [1 \cdot 1 + 2 \cdot 0] \ [2 \cdot 1 + 1 \cdot 2] \ [3 \cdot 2 + 2 \cdot 1] \ [4 \cdot 1 + 3 \cdot 2]$$

$$a(*) = [1] \ [4] \ [7] \ [10]$$

## 6.2 Multilayer Neuronal Networks

### 6.2.1 The importance of Inputs

As we have been insisting, one of the most important factors to optimize the results of a network, is what and how we will introduce the input values. Depending on the level and depth of data preprocessing, we can provide better information that will influence significantly the results of our network, especially for complex problems like the ones in real scenarios.

Although these factors will be explained in greater detail in the following sections, with the preliminary analysis of the input data, we can conclude relations, groups, clean out the data that we consider is not useful or reduce the dimensionality and size of a large dataset without losing information.

#### 6.2.1.1 Saturation of the Input Channels

Some of these steps, mainly if we limit them to functions, also can be applied to the output values. Then we will see how these improvements will help to create a model that is both more efficient, and more accurate.

Commenting on the simplified case of the use of functions in the hidden layers generally sigmoid function, this function has its advantages but is also limited, as discussed in previous sections, specifically when the values it transforms are of the order of  $10^3$  it starts to become saturated. If this occurs at the beginning of the training, the gradients which change our weights and bias will be very small so that the network will learn very slowly.

Similarly, if the input values are very high, the weights must be very small to avoid saturating the transfer function. Therefore, we can generalize that, except in specific cases, it is convenient normalize at least the input values before loading it on our network.

Obviously, if we have normalized input values, we must do the same with the objective values and of course the values returned by our network will equally be in the range that we have defined in our initial values, we must therefore do the inverse function to be able to make sense of the values. As a small comment, it is worth mentioning that in the majority of the tools, both paid or free, that are used as a the basis to develop neural models, these functions are applied to the values of input and output even if the user does not notice it.

These normalization functions can be configured; however the standard use of some of them tends to facilitate the process. Normalize input and objective values between  $[-1,1]$ , delete values that do not add information (constants), extract the main components of the vectors, adjust them so that they have a mean equals to 0 and variance equal to the unit or a grouping of extreme values if required are all examples of these functions.

#### 6.2.1.2 Overfitting and Subsets

One of the most widespread practices when processing data before passing them to a neural network is to divide it into subgroups. This step, which is practically used in 100% of cases, except where the amount of data that is available is very small to emulate the complexity of the underlying problem, consist in separating the data into three groups. Generally, the first and largest will be the "training set" which will be used to train the network, i.e., calculate the appropriate values of weights and bias. The second subset will be used to validate that our precision is enough to finish the training; on each iteration of the training, the error of the model will be calculated and compared with the validation subset.

However, when the network begins to learn the problem and not the underlying complexity, this error starts to increase, since our network has learned to represent the problem in question, but it will not be able to predict future values logically, this is what is known as overfitting and will be greater when the data you give to the problem in question is not enough. Since the number of neurons and layers are able to represent the given problem, but it will not learn how to solve them, this problem is accentuated if you add more layers than really necessary, since it is increasing the capacity of the system to reproduce without understanding the intrinsic complexity of the problem.

The third subset will be the testing one, if the minimum error we get is not on par with the subset of validation error; it meant that the data is not properly divided.

Some of the methods most widely used at this time to divide the data into these three subsets are: to choose data randomly, continuous blocks or defining specific indexes.

#### 6.2.2 Gradient and Jacobian development and performances

The training process is responsible for adjusting the values of weights and bias for the network to be able to reproduce the problem in the most effective and optimal way. One of the most used functions to measure the performance of the training is the mean square error, mean-square error between the outputs and target values.



$$F = mse = \frac{1}{N} \sum_{i=1}^N (e_i)^2 = \frac{1}{N} \sum_{i=1}^N (t_i - a_i)^2 \quad (6.5)$$

As we have previously said many other functions can be used, if we focus of multilayer networks, other functions can be used to measure the performance of our stage, however, most of the outstanding ones are using the gradient of the network performance with respect to the network weights (from the previous stage) or the Jacobian of the errors of the network with respect to the weights.

In both cases, these are calculated by the widely studied backpropagation algorithm which supposes an undoubted boost to the development of such networks.

### 6.2.2.1 Development

Given the importance and the widespread use of the Backpropagation algorithm let see what it is based upon and its development in order to better understand how the network is learning in each iteration.

Let's start with the basics; the weights will be updated according to the following formula.

$$W(t + 1) = W(t) + \Delta W(t) \quad (6.6)$$

But we want to vary proportionally to the gradient of the error function that we have defined, which has been described four paragraphs above, *mse* (6.5).

$$W(t + 1) = W(t) - \alpha \nabla E[W(t)] \quad (6.7)$$

Let's leave weights aside for a moment, to remember how we had formulated our problem initially. We had one output, which we called from a signal to which a function was applied, see *simple formulation* (6.1). If we simplify this function to the maximum, our preprocessed signal (before transfer function) will only be input data by weights.

$$a_i(t) = f_i(h_i(t)) \quad (6.8)$$

$$h_i(t) = \sum_j w_{ij} x_j(t) \quad (6.9)$$

By comparing the output we get  $Y_p$  with the value target or expected  $D_p$ , again using the function described above. Where  $k$  is the number of neurons in the last layer and  $M$  the number of neurons of the mentioned layer.

$$e_p = \frac{1}{2} \sum_{k=1}^M (d_{pk} - y_{pk})^2 \quad (6.10)$$

So far we have only applied knowledge that we had previously described in detail. Let's move forward a little more, if we extend the previous function we get the total error of the network, being  $p$  the entered value and  $P$  index the total number of values used.

$$e = \frac{\sum_{p=1}^P e_p}{P} \quad (6.11)$$

Recovering the function weights change and recalling that we want to be proportional to the gradient of the error function, we get that by applying the chain rule it becomes.

$$\Delta w_{ji} = -\alpha \frac{\partial e_p}{\partial w_{ji}}; \quad \frac{\partial e_p}{\partial w_{ji}} = \frac{\partial e_p}{\partial h_j} \frac{\partial h_j}{\partial w_{ji}} \quad (6.12)$$

In the above equation, we have the derivative of the error depending on other two derivatives. The first indicates how the  $h_j$  result varies when the error varies by changing the input neuron  $j$ . The other derivative represents how it varies regarding the weight of the neuron  $w_j$  when you change the input value of the neuron  $j$ . This second derivative can be expressed with the following terms.

$$\frac{\partial h_j}{\partial w_{ji}} = \frac{\partial \sum_i w_{ji} y_{pi}}{\partial w_{ji}} = y_{pi} \quad (6.13)$$

Thus, if we simplify the first term with the previous equation, it is.

$$\frac{\partial e_p}{\partial h_j} = -\delta_{oj}; \quad \frac{\partial e_p}{\partial w_{ji}} = -\delta_{pj} y_{pi} \quad (6.14)$$

Therefore, our equation becomes as.

$$\Delta w_{ji} = -\alpha \delta_{pj} y_{pj} \quad (6.15)$$

If we again apply the chain rule, the equation expands.

$$\delta_{pj} = -\frac{\partial e_p}{\partial h_j} = -\left( \frac{\partial e_p}{\partial y_{pj}} \frac{\partial y_{pj}}{\partial h_j} \right) \quad (6.16)$$

And we get thanks to the fourth and fifth expression, the following formulation.

$$\frac{\partial y_{pj}}{\partial h_j} = \frac{\partial f_j(h_j)}{\partial h_j} = f'_j(h_j) \quad (6.17)$$

Here we need to pause and differentiate in two cases, if we are in the output, i.e.  $j$  neuron layer, it will return an output or if the neuron  $j$  is in a hidden layer and their signal will be processed later.

In the first case,  $j = k$  and we can express the function as.

$$\frac{\partial e_p}{\partial y_{pj}} = \frac{\partial \frac{1}{2} \sum_{j=1}^M (d_{pj} - y_{pj})^2}{\partial y_{pj}} = -(d_{pj} - y_{pj}) \quad (6.18)$$

Therefore the variation of the weights of a connection that is passed to the outer layer of output is calculated by.

$$\Delta w_{ji} = \alpha (d_{pj} - y_{pj}) f'_j(\mathbf{h}_j) y_{pi} \quad (6.19)$$

On the other hand, if the neuron  $j$  is on a hidden layer, we must apply again the chain rule, being  $k$ , the index of the next layer of neurons.

$$\frac{\partial e_p}{\partial y_{pj}} = \sum_k \left( \frac{\partial e_p}{\partial h_k} \frac{\partial h_k}{\partial y_{pj}} \right) \quad (6.20)$$

This equation can be rewritten as by applying the equation obtained in (6.13) we get.

$$\frac{\partial e_p}{\partial y_{pj}} = \sum_k \left( \frac{\partial e_p}{\partial h_k} \frac{\partial (\sum_j w_{kj} y_{pj})}{\partial y_{pj}} \right) = \sum_k \left( \frac{\partial e_p}{\partial h_k} w_{kj} \right) \quad (6.21)$$

$$\frac{\partial e_p}{\partial y_{pj}} = \sum_k -\delta_{pk} w_{kj} = -\sum_k \delta_{pj} w_{kj} \quad (6.22)$$

Therefore weights variation of connection in a hidden or intermediate layer, will be.

$$\Delta w_{ji} = \alpha \sum_k (\delta_{pk} w_{kj}) f'_j(\mathbf{h}_j) y_{pi} \quad (6.23)$$

If we remember the learning rate, in our example proposed  $\alpha = 0,1$ . We can see directly into the equation that the greater the value the faster the process will be, with the risk of oscillating near a local minimum as we will explain in more detail in the future. To reduce this impact, we can use a moment which defines as  $\beta$ , hence and as we can see in the following formula, this time determines the effect in step  $t+1$  of the change that we have performed in step  $t$ .

$$\Delta w_{ji}(t+1) = \alpha \delta_{pj} y_{pi} + \beta \Delta w_{ji}(t) \quad (6.24)$$

This addition has more advantages than those mentioned in the preceding paragraph. Suppose that the modification of weights in  $t$  and  $t+1$  is going in the same direction (i.e. is increased or reduced in both cases), thanks to our  $\beta$ , the surface impact in  $t+1$  (imagine it as the area where we are looking for the solution as either a local or global minimum) will be higher.

However, if the direction of the changes is the opposite (in one case the value of the weights is increased and in the other is reduced), the change in  $t+1$  is lower. This is the point, because this means that we have passed close to that minimum and therefore we want to adjust the search even more and progressively reduce our changes, since if we imagine the surface of the problem once more, we are approaching the point with maximum gradient, and therefore we have to give greater importance to the gradient and try to reduce the search area not to jump without letting ourselves fall into the solution.

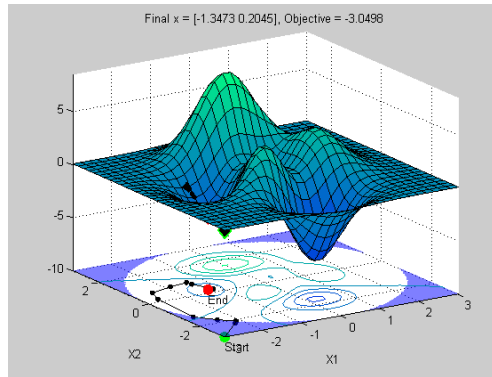


Figure 6.29 - Jacobian Learning with adaptive  $\beta$

Let's see briefly what the previous image; the area represented is the area where we are looking for the solution, as we can see we have fallen into a local minimum, since it is computationally too high if not impossible for the majority of problems to find out the global minimum. In the plane  $x$  the variation of gradients is represented and dotted is iterations where it has "taken measures", as we have explained previously due to  $\beta$ , the more we approach a point with maximum gradient the smaller the distance between point and we can therefore find that minimum more effectively. (Chien-Cheng, 2002) (J. Leonard, 1990) (Riedmiller, 1994)

### 6.2.3 Training Algorithms

During the implementation of the backpropagation algorithm, learning occurs through the subsequent presentation of a training set. Each full presentation of the training set to the multilayer perceptron is called an epoch. Thus, the learning process is repeated epoch after epoch until the synaptic weights are stabilized and the performance of the network converges to an acceptable value.

The way in which synaptic weights are updated results in two different training modes, each one with its advantages and disadvantages.

#### 6.2.3.1 Incremental Training

In this training mode the weight update is performed after the presentation of each example of training, for this it is also known as pattern mode. If a training set has  $n$  examples, the sequential mode of training has as a result  $n$  corrections of synaptic weights for each epoch.

### 6.2.3.2 Batch Training

With this mode of training, weight update is performed only once after the processing of all the values in the training subset. On each epoch the mean quadratic error is calculated as we have seen above (6.5) in our network. And changes in our weights defined by (6.19) and (6.23), which will have a derivative as we have seen in the previous development.

$$\Delta w_{ji} = -\alpha \frac{\partial e}{\partial w_{ji}} = -\frac{\alpha}{N} \sum_{n=1}^N e_j \frac{\partial e_j}{\partial w_{ji}} \quad (6.25)$$

If training sets present themselves to the network randomly, the sequential training mode will turn the updating of the weights space in stochastic by nature, and decreases the likelihood that the backpropagation algorithm gets trapped in a local minimum. However, the stochastic nature of the sequential training mode hinders the establishment of the theoretical conditions for the convergence of the algorithm.

Furthermore, the use of the batch mode training provides an accurate estimate of the vector gradient, thereby guaranteeing the convergence towards a local minimum.

### 6.2.3.3 Functions

It is difficult to anticipate in advance which function will be the one to converge more quickly in a given problem. The complexity of the problem, is one of many factors; the number of neurons, layers, recurrence or if we are looking to approximate a function (regression – in our example) or patterns (discriminant - possible scenarios) next to the desired error will make the time spent at this stage vary substantially.

We will see three examples of functions with better results when it comes to approaching functions. We will see that generally speaking Levenberg-Marquardt is usually the one that returns the best result.

*Simple function (sin):* a network with a hidden layer of five neurons and an output, and error measure with mse. As we have argued, we will use tansig transfer function and a basic pre-processing. We see in the two following graphs how LM is up to four times faster, but this type of problems is where LM stands out since it requires a simple approach with a few hundred weights.

We see in the first graph, than the accuracy directly affects the performance of the algorithms, comparing the averages of 30 executions, together with the mse and the time required. The algorithms in comparison that matters the most to our model are: LM (Levenberg-Marquardt), SCG (Scaled Conjugate Gradient), OSS (One Step Secant) and GDX (Variable

Learning Rate Backpropagation), although in some graphs others are shown, we will in the examples that the most useful ones for our purpose will be those previously mentioned.

In the second graph we see the time that it takes to reach a valid solution or the number of maximum epochs versus the mse. LM stands out since it behaves better when the objective error is less.

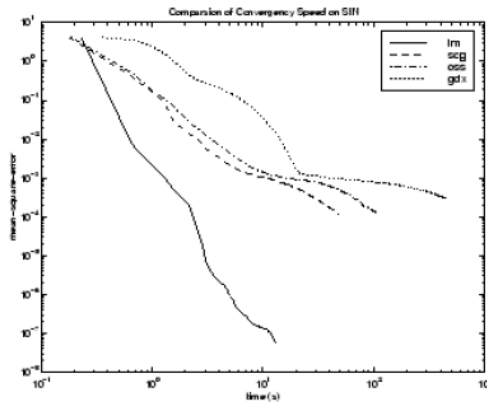


Figure 6.30 - LF comparison #1, convergence vs speed

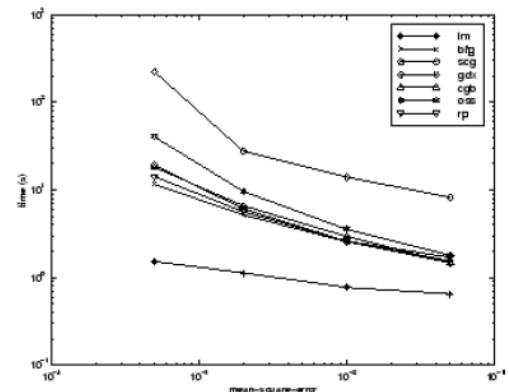


Figure 6.31 - LF comparison #1, mse evolution

*Real function #1 (nonlinear)*: in this case, networks are configured with two inputs and two outputs and 30 neurons in the hidden layer, again the tansig function will be responsible for providing values to the output layer and it will process them in turn with a linear function. The convergence is obtained when the mse with validation data is less than 0.005. We will again see how LM together with SCG outperforms; this is because one of the advantages of LM, not very extensive networks, has been reduced by a factor of 10. Let's see again the results with a more complex problem, drawing the same conclusions as in the example above, while the network size begins to be important.

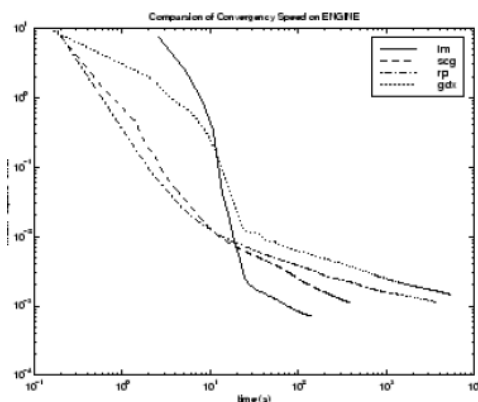


Figure 6.32 - LF comparison #2, convergence vs speed

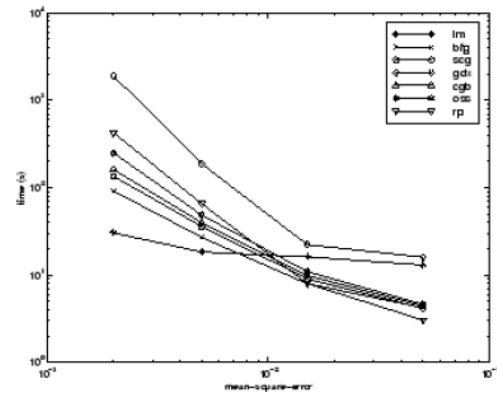


Figure 6.33 - LF comparison #2, mse evolution

*Real function #2 (nonlinear)*: following the increase of the complexity of the network, and therefore its added complexity of network design, we will see how the advantages of the use of LM cease to be so remarkable. In this case, we will have 21 entries, with 15 neurons in

the hidden layer and 3 outputs, weights and bias started randomly in each of the 20 examples, from which the mean is taken and an objective to converge by less than 0,027 mse.

Following the progression of results from previous examples, LM has reduced its performance mainly by increasing the size of the network, 100% with respect to case two (*Real function #1*), and SCG is the best choice since it does not require a geometric increase of the calculations regarding the network parameters as is the case with LM.

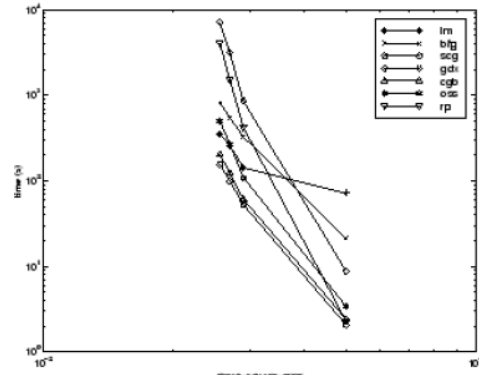
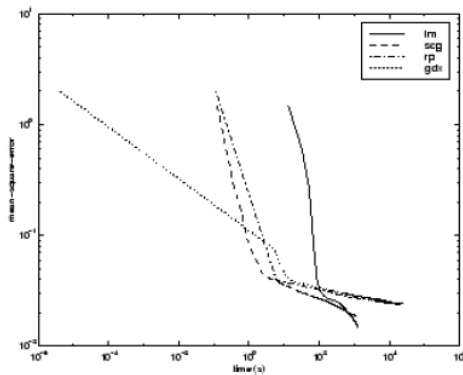


Figure 6.34 - LF comparison #3, convergence vs speed    Figure 6.35 - LF comparison #3, mse evolution

As a conclusion, we can say that as shown in example one, with a network containing a few hundred weights and bias LM is considerably better as we reduce the level of objective error, while memory and processing capacity requirements soar as we increase the complexity and size of the network. For its part, Scaled Conjugate Gradient behaves better both in time and reducing errors when the size of the network increases. (MathWorks, 2015)

#### 6.2.4 Analyzing and Improving the Network

In this section we will not analyze in depth, nor will we detail important historical aspects in uni-layer or linear networks, neither in the known and studied perceptron, decision boundary, outliers in multi-layer networks, nor Hopfield NN since they will not be used in our model. All these studies and development have been part of the process of the development of neural networks and many of its limitations have been greatly reduced with the use of multi-layer networks, more advanced algorithms and the increase of the computational capacity.

However, some important factors when it comes to validate the training of our network should be mentioned. Remember that even though we have explained in detail a learning algorithm many others can be used and some are more efficient than others for the management of different types of problems.

Let's make a small stop with the overfitting problem, imagine that the neuron ratio is high in comparison to the amount of data that you enter, or this data does not contain much

information or is not enough to reflect the problem's complexity. In this case, while training takes place, the error will progressively be very small (since it is not a problem for the network to assimilate those relations or information) but when we introduce new data, an error will peak very far from the errors previously obtained. In other words, the network has learned the examples shown but has not learned to generalize the problem for different inputs.

Graphically, with the passing of each epoch the error with our validation subset increases, as shown in the right picture below; in this case if we are limited by the number of layers or neurons or is not possible to vary the analysis of input data (expand the amount of information that is loaded into the model), then we can stop before our learning is done well enough, this is known as early-stopping.

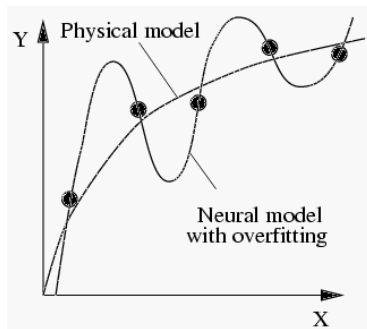


Figure 6.36 - Overfitting example

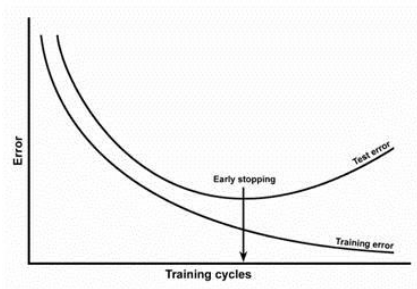


Figure 6.37 - Selection of early stopping

Let keep in mind that when increasing the number of neurons in the hidden layer(s) we are giving the model greater flexibility since we have more parameters to learn.

Although reviews of the style have been done already, is worth to mention in this section a few concepts with important consequences if they are not taken into account. The point of the space of solutions where the problem converges will vary according to the initial values of weights and inputs, as well as how we divide the input data that we have (learning, testing and validation). Therefore, it is essential to train our model completely several times to stay with the model that best fits our needs.

Continuing with the concept previously described, we can use all of these trained models to, whenever possible, ask each one for values, and then make an average among all of them. If we see that as a general rule, we get more appropriate values, this simple generalization could increase our accuracy a few tenths or hundredths.

If once trained we represent our output vs targets values as a regression, we can see with a good confidence how the values we get are in accordance with our objective values. If the numerical value that we get is far from 1, we will have to perform the above steps or change our learning function as we will see in other chapters. It is expected that with sufficient, correct and cleverly pre-treated input data and settings that gives enough freedom to model the problem, a neural network with good convergence should return values close to one; as long as



randomly provided or not correctly loaded factors affect the results in a remarkable way. (Adam P. Piotrowski, 2013) (Kakade, 2012) (Prechelt, 1998)

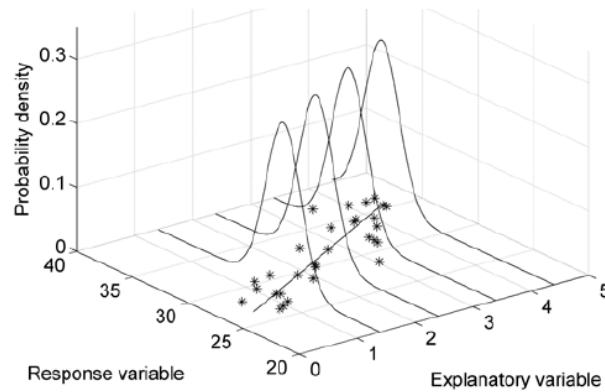


Figure 6.38 - Output analysis example

#### 6.2.4.1 Performance Function

Another way to adjust our network is regularization, by automatically changing the performance function, we can better generalize the limited data that we have. Remember that as a general rule, we always talk about the use of *mse* when calculating the performance during the training.

If we recall when we calculated the value that should be considered when calculating the gradient (6.24) we had been able to give greater importance to the closest *t* values, in our case in an exponential manner.

Suppose, however, that we want our function to have a more gentle behavior to avoid the effect that occurs with the overfitting. This can be done in an efficient manner with a factor and or performance ratio, which will be the average sum of weights and bias values to the square, leaving our new function as follows.

$$msereg = \gamma mse + (1 - \gamma)msw; \quad msw = \frac{1}{n} \sum_{j=1}^n w_j^2 \quad (6.26)$$

Having lower weights and bias, the network will have a softer behavior in the face of input variations and therefore such problems will be reduced in part. This change of course brings its contraindications, so we will try that the model itself would look for the most suitable values.

One of the most extended ways to automatically calculate this factor is the Bayesian method. Thus, weights and bias are initiated randomly (as normal) but with some specific distributions. Regularization parameters relate to the variance in these distributions. It is important to note that this ceases to be a small change and affects the performance of the algorithms described in this document; therefore it must be used carefully.

Following this development, thanks to this system, we can see what weights and bias are being used (or being modified), in theory once our network is trained and with a convergence that we consider to be acceptable, i.e. without overfitting or high error, we can see what parameters are being consulted by the network when calculating the outputs, hence reducing our number of parameters. If the network is well trained, as much as we increase this number it shouldn't be affected. This is theoretical, will largely depend on the degree of convergence achieved and on the function and our network configurations. Another way of seeing that this convergence has been reached is to see that the values the network really uses are not being modified.

## 6.3 Dynamic Neuronal Networks

### 6.3.1 Introduction

In such networks, the output does not exclusively depend on the input values, but the previous values for both input and output are also considered.

Within a dynamic neural networks, we have two categories, those that only have feedforward connections and those that have feedback or recurrent connections. Let's see the difference with three simple and very representative examples. In the first case we have the signal shown in the picture, if we make this signal go through a static network (use the example as a reference) with a single neuron, a weight of two and a bias of zero, we get the output signal shown in the picture on the right. As we see the treatment of the pulse only lasts while the signal is processed.

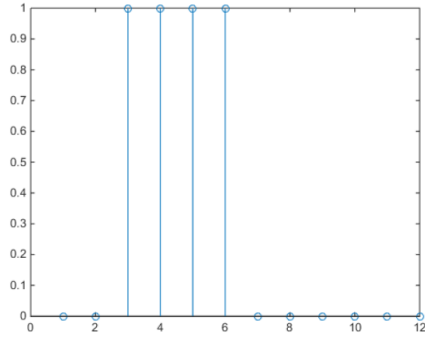


Figure 6.39 - Input example

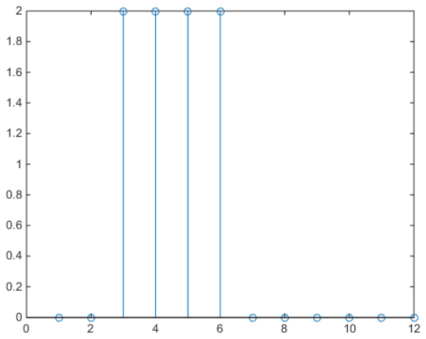


Figure 6.40 - Output example with non-dynamic network

Let's now apply the same impulse to a dynamic network but without recurrent connection, the values will be [1 -1] for weight, zero for the bias a delay of one, a neuron and a single layer. We'll see how the response to our pulse lasted more than the own input signal, because as we have shown our network has the ability to remember the values previously processed. Therefore the output returned by our network will be conditioned by a number of factors.

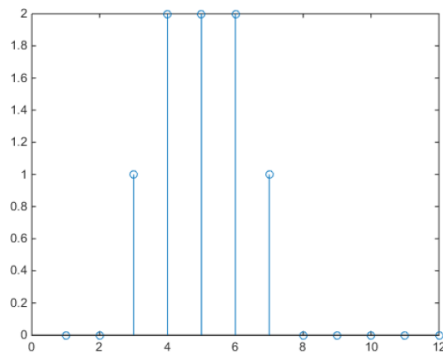


Figure 6.41 - Output example with dynamic network

Another conclusion that we can draw from the processing shown in the previous picture is that not having recurrent connections, only a finite capacity of data will be taken into account, our memory may not be extend. Our delay was one, we see that a unit is as far as a previous value affects the value  $t+1$ .

Now, let's build a dynamic recurrent network, e.g. a NARX network with closed loop. In this case and following the policy of maximum simplicity our network configuration will be, IW = 1 (input weights), LW = 0.5 (weights between layers), a delay equal to the unit, a neuron and a layer.

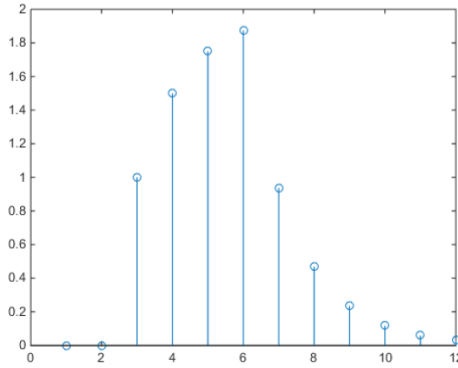


Figure 6.42 - Output example with a dynamic and recurrent network

Let's focus on how this time the memory effect or previous values that affect our signal processing are extended in time, and theoretically never reach zero. It is important to note of this, we have been insisting on the importance of the data with which we teach our network, it will be what defines its accuracy and efficiency to represent the problem.

Returning to the block diagrams and formulation we represent the simplest case of a recurrent dynamic network as we've seen before with examples of pulses by way of signal processing.

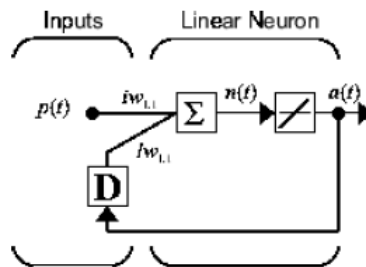


Figure 6.43 - Simple dynamic network

$$\mathbf{a}(t) = \mathbf{I}W_{1,1}\mathbf{p}(t) + \mathbf{L}W\mathbf{a}(t - 1) \quad (6.27)$$

Note that the weights, including the initial parameters, have two effects on output. At each stage they affect the resulting value of  $a(t)$  but also affects the results of  $(t-1)$ . Turning these last values into a function of the parameter that the neuron previously had, i.e. its weight in the previous stage. This second influence has several consequences, the first one is that the surface where we search for the minimum is complicated considerably and there are more possibilities of being trapped in a local minimum. On the other hand the computational requirements also increase and in order to have enough confidence of finding a good result, or a reliable network configuration, we have to train our model several times without forgetting to change the initial values of the weights and the bias.

### 6.3.2 Time-Delay Structures and Training

Let's focus on our objective, time series forecasting. It is no longer necessary to justify that we begin using dynamic networks (open that will be later closed) with backpropagation to calculate the gradients and as we go in depth we can use other systems. It is worth mentioning at this point that the most common way to predict more than one value in a time series is to load that value back in the model, so be processed thus generating the value of  $(t+1)$ .

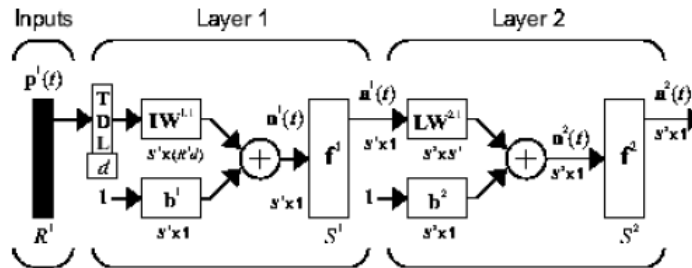


Figure 6.44 - Network with Time-Delay

The image above is a simple case of a network with time delay, TDL, the network is powered by inputs according to the set delay. In the above image we see an example that expose what we have previously explained, the output of the first layer serves as feedback for that same layer, which gives part of the advantage of recurrent networks. Finally the output is used as the value in the first layer and the second, the consequences and values of  $LW^{2,3}$  will be more difficult to understand.

This time we complicate our previous signal of one byte, once treated, normalized and converted in such a way that our model understands it as a sequence of data, we can load it and train our model. Let's configure the maximum delay of our model for better understanding with a value of eight, in the first iteration will try to predict the value of the ninth, as the minimum value of the delay will be one.

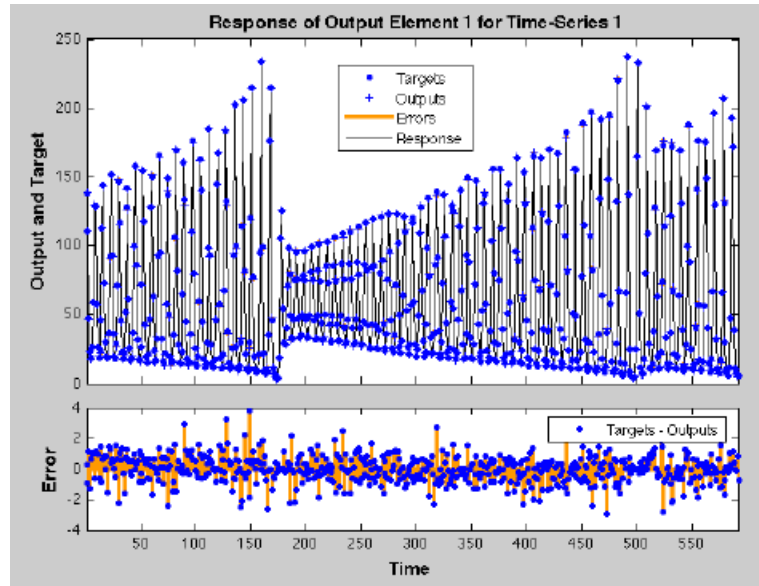


Figure 6.45 - Example of a simple Time-Delay Network response and error

As we can see, the error we get is small, not only that, since there's no need for a dynamic workout to calculate the gradient that changes our weight, training has been significantly faster than in the opposite case. This advantage shown here is due to the fact that the value calculated for  $t$ , is not entered in an intermediate layer and doesn't change an intermediate parameter, so learning would be just like on a simple network.

### 6.3.3 NARX Time Series

This type of network will be the base that we'll use for part of our models, so we will introduce them with special care, keeping in mind the basic concepts previously exposed. Since we know what each one of these concepts involves, we can define a nonlinear autoregressive network with exogenous inputs (NARX) as a neuronal dynamics and recurrent network, with feedback connections enclosing several layers. See the simplest example and the simplified equation that governs it.

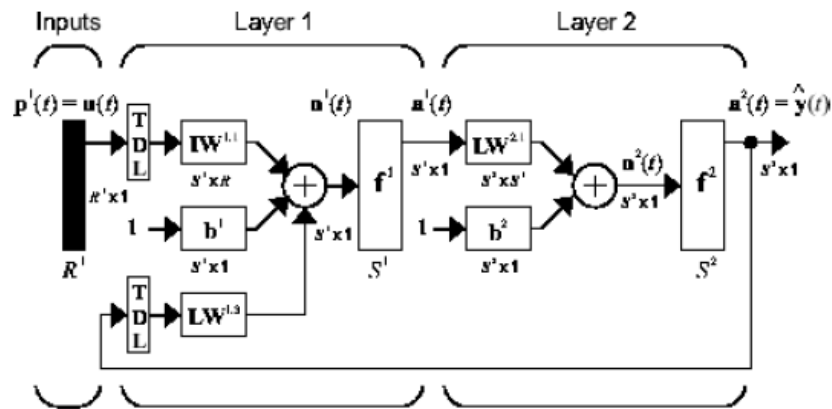


Figure 6.46 - NARX Network with Time-Delay

$$y(t) = f(y(t-1), y(t-2), \dots, y(t-ny), u(t-1), u(t-2), \dots, u(t-nu)) \quad (6.28)$$

The equation means that the value we get in  $y(t)$  depends on the previous values that has been returned by the model  $y(t - ny)$ , as we can see by its feedback branch with TDL and the values of our matrix  $LW^{1,3}$ . But also the  $u(t)$  values of input that we are loading as independent (exogenous) signals. In the example exposed above, a two-layer model with feedforward from exit 2 to 1 is shown.

This type of model has considerable advantages since the value we get depends on other values calculated by a non-linear dynamic network which tries to represent the complexity of the problem treated. An improvement to this model is the possibility of using the value return by the model in  $t$  as another exogenous value, in cases where a real value is available; we can provide this value directly. This obviously entails a series of advantages, the first and most direct is that we are providing real value as input, so that the data with which our network shall be adjusted will be more accurate. Another advantage, as we have mentioned with the previous model, is that being a network with a feedforward architecture (exactly the same as in the example previously presented and therefore with the same consequences) we can use a static backpropagation algorithm thus the training is considerably faster, and in some cases very accurate.

As shown in the image above, we got that thanks to the feedback by the branch the process becomes static, going from parallel to a series-parallel architecture. In this way we can feed this dimension of neurons with values that can go from the initial ones, others with some level of processing from any other stage or layer or the expected output from the network. In this last way we are forcing the network to analyze once again their results and trying to understand the consequences of what will have happened in that environment with this previous expected output. In such a way that it is able to learn from their actions and above all as they influence their decisions in order to minimize the loss function. Once the network is trained, we can close the loop and we can use our new network, with feedback from the outputs, to predict one or more values at the time.

The logical procedure that we conclude with our development would be to train and validate our network with a configuration in open loop, to obtain the benefits of this type of network. And then, close and the feed the input back (one of our exogenous values) to be able to predict more than one distant value over time. The importance of training, which will be done with a series-parallel structure, will be a key factor so that once we close the network with a parallel structure the error is less. A common practice already exposed is to train the model several time with different weights and initial bias in such a way that the model will find different minimums.

Remember that by introducing a series of values, which do not have to be on the same scale, since their meaning can vary widely, the standardization of all of them before using them in our network is necessary. As we have explained previously, when calculating the gradient, with a typical sigmoid function, we begin to saturate it with values that tend to  $10^4$  and therefore the information contained in these values not only is being lost; its noise that is inserted directly in our network, when changing our weights with values close to their maximum.

It remains to explain how we can learn as we introduce real values, even with our closed loop, as we are going to see in the next chapter. (Xiaofeng, 2014)

#### 6.3.4 Multistep Prediction

As we have seen with NARX models in open or closed mode, we are able to make predictions that are extended in time because we use as inputs to  $(t+1)$  the value calculated in  $t$ .

NARX network configured with the closed loop allows us to, predict  $d$  values, and where  $d$  is the delay that we have set, without having to depend on any previous calculated value.

This methodology also carries negative consequences, if it is predicting values far away in time and our problem is not stable over time, as we go away from the real values, we will be progressively depending more on the no-real values calculated in previous stages, or we will be making predictions far in time with values progressively less updated.

To reduce part of this impact, we can configure the calculation of our mean squared error, so that it gives weights or different importance according to the position or value that we are dealing with. In our case it would be -large amount of data with no specific problems in its collection- convenient to give greater importance to the latest objective values. In this way, the error obtained from the latest values of the time series influences more the weights due to that the gradient will be multiplied by some factor. If our problem meets these conditions, it is more efficient to increase the global mean error to significantly reduce the error in the short term past values of the series. For this we can use an exponential function which is also easily



derivable, and as we have seen is a very considerable advantage when it comes to consumed resources and therefore training time.

Slightly complicating the architecture, these internal relationship coefficients, either pre-defined or dynamic, give us enough freedom if are applied in intermediate layers. Since we can limit regions of the network in a premeditated way, getting into deeper networks we can force certain groups of multidimensional neuron to focus on generalizing certain patterns of scenarios. Thanks to that these sub-networks will be focused in how understating the evolution of certain patterns leads to certain scenarios. Therefore we can predict values in a multi-dimensional way and so minimize the objective function, optimizing the overall process.

For this we can use an exponential function which is also easily derivable, and as we have seen is a very considerable advantage when it comes to consumed resources and therefore training time.

$$F = mse = \frac{1}{N} \sum_{i=1}^N w_i^e (e_i)^2 = \frac{1}{N} \sum_{i=1}^N w_i^e (t_i - a_i)^2 \quad (6.29)$$

Let's recall the previous result of the *Figure 6.45*, with this single change of value over the data closer to  $t$  when  $\lim_{t \rightarrow n} y(t)$ , being  $n$  the number of input up to  $t-1$  values. Let's see the result of applying this exponential to the weight of the errors.

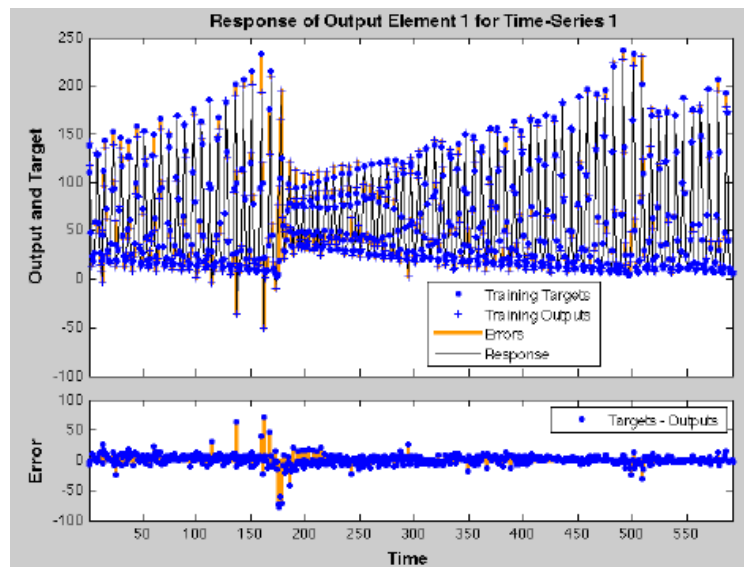


Figure 6.47 - Network response with exponential weight of errors over time

It stands out that the errors in the last part of the forecast are minor, at the expense of increasing them for values closer to  $t = 0$ ; Note that in *Figure 6.45* the values were between  $(2,-2)$  and now the graph shows  $(75,-75)$ . As expected, this decision has its counterpart; however, in the majority of cases for ST predictions with large amount of input data, in the short term, it can be very useful when making adjustments for predictions not very far from  $t$ .

### 6.3.5 Radial Basis Networks

As we have previously described radial neural networks have certain advantages over the traditional models of NN, such as backpropagation or feedforward. They have a greater number of neurons, they work best with lots of values (but not good enough with raw huge databases) and can be trained in considerably less time. In our typology of radial networks that we will always use the radial transfer function in the hidden layer, and as it has already been detailed in the section for the description of these models, that the values that are inserted into the neuron will be the distance to the "center" multiplied by the bias.

$$\mathit{radbas}(n) = e^{-n^2} \quad (6.30)$$

Following the representations we see the block diagram of our new network.

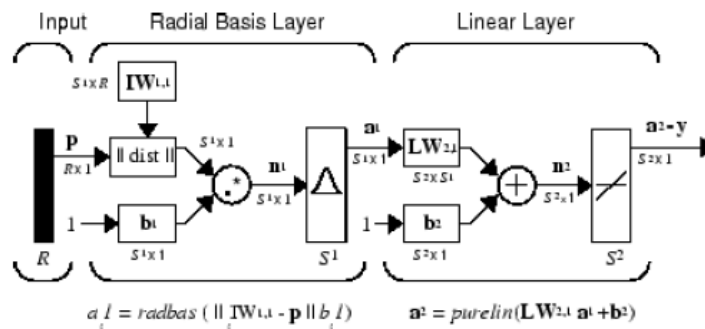


Figure 6.48 - Basic Radial Basis Network

The values that are obtained from subtracting inputs to the center of each neuron will be the distance, which after being transformed by the transfer function; if they are similar it will have values close to zero and therefore very low importance. More directly, if the value is close to the center of that neuron the value received by the layer with linear function will be high, i.e. it will be taken more into account and therefore will affect the output value to a greater extent.

If in our training we want our model to approximate more to the values initially provided and therefore generalize for the cases not shown, we can use this approach. Let's have a network with as many radial neurons as input parameters, in such a way that adjusting the input with the output values; we have exactly the same values. Only have to define the area (of the space formed by the problem's parameters) that we want to cover with each neuron. This area should be large enough so that several neurons are activated when asked by the result in a certain point in the space which in turn will give some level of generalization to the problem, but allowing differences in the responses.

This approach creates a problem with  $n$  constraints (data pairs) and each radial neuronal has  $n+1$  unknowns values ( $n$  weights and its bias), which have theoretically infinite solutions. As you would expect this methodology is only practical when there are few parameters or more

specifically, when the problem needs to present a lot of data for its representation, actually it is not so easy.

Another even more interesting practice that will be used in the comparison between different models will be to iteratively add a radial neuron to the model, until a previously defined goal is reached. Similarly to the previous case, the radio or area covering each of the neurons must be broad enough to encompass areas of other neurons, giving certain dynamism, but not to the point where it might remove value to the characteristics of that region of space.

In the next two figures we can compare how a different number of centers affect the decision boundaries.

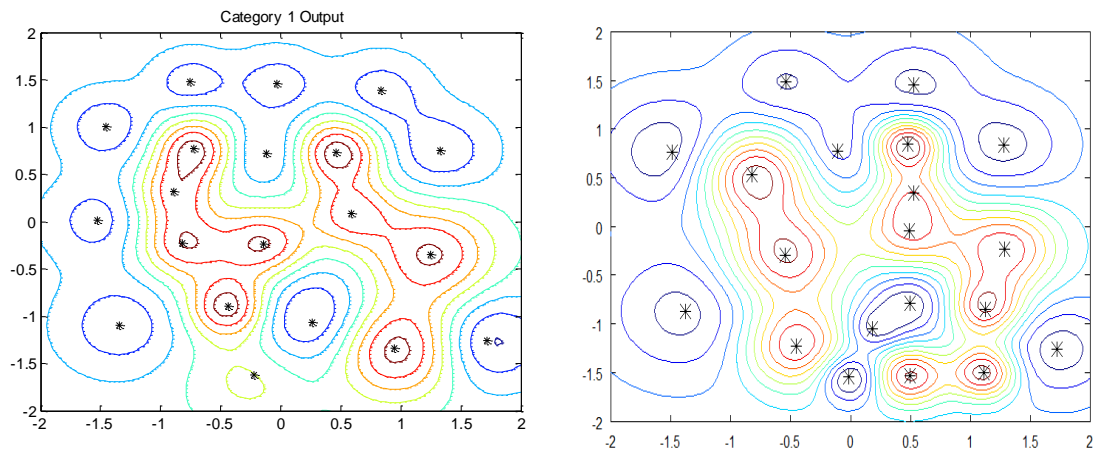


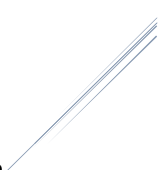
Figure 6.49 - Graphical representation of RBN centers

Directly, we understand that to have a good radial model we need a considerably larger number of neurons (memory) but given that they are considerably easier to train than traditional models using sigmoid function as transfer functions, we will have a significant advantage. We must add to this that if we use the methodology explained in the previous paragraph, we will have a network with only the number of neurons that are really useful to us, given this approach and configured so that in certain cases, we can have models with even less neurons than recurring or non-traditional models. (Hutchinson, 1994)

## 6.4 Improvements

### 6.4.1 Computing

It's worth mentioning that the three milestones in neural networks that have enabled its development up to today have been the backpropagation (optimization of the method of learning), the increase in computational capacity in recent years and the deep learning that we will study later.



It is therefore required, to detail some of the most important factors relating to the computational ability of these models, since it will not only enable us to understand and reduce training time, but also to optimize the models to be launched with higher requirements or add more complex architectures or casuistry.

Due to the objectives of this document, we will not give special consideration to the characteristics of the hardware, but we'll try to develop from basic knowledge to the application of the problem we are dealing with.

Neural networks are a series of algorithms in parallel. We are trying to perform thousands of operations at a time to be able to process functions, calculation of gradients, operations, update values and the re-feeding in a certain way and following certain steps. Everything could be done in parallel.

Therefore, first we will try to focus on exploiting this feature of this series of algorithms, and then we'll try to find other approaches that allow us to take advantage of the structure and static part of an already defined and optimized network.

#### *6.4.1.1 Parallel Computing*

The main advantage of a GPU, along with its low cost in relation to its power, its great capacity for parallel processing and the advantage of the efficient use of the floating point make this architecture designed to carry most of the primitive operations in the most direct way to increase processing capacity. Other examples of multi-core or multi-processor combine advantages of a single machine with parallel processing, since the current limitation on the frequency can partially limit other types of architectures.

Internally this leads to considerable complications, errors in software such as race conditions (the result of the operation depends on the order of execution) or theoretical limitations such as Amdahl's law which study part of these problems.

Within the GPU many instructions were carried out simultaneously, but at the same time the tasks, data, instructions or bits can be processed in parallel so small adaptations will allow us to make these calculations in a distributed manner.

#### *6.4.1.2 Distributed Computing*

Processor needs can be divided between multiple processors or computers, as long as they have enough RAM to gain access to the data in an efficient manner. The way of processing

would be to split the input values, once the values are calculated; they return to the matrix or initial vector and so on.

If the amount of data to be treated is very high and you can take decisions as the data is divided, these sequential partitions allow dealing with more data, for example with equipment without enough RAM, or to better spread the work load. Also, if certain specifications for the equipment vary the load to each unit, sizes can be also varied so that the loads between units are balanced.

Let's imagine that we have organized a lot of data in a sequential manner, each sub-set is treated by different groups, after then its reload.

Regarding the use of used memory while you run the learning and the gradient is calculated limitations may arise, the required size can be reduced  $n$  times, if we divide the process in  $n$  threads with  $n$  subsets of data, whenever possible. (Dean, 2012)

#### 6.4.1.3 GPU Computing

Due to the characteristics of the GPU is not recommended to use it with Jacobian, but yes, they have proven to be useful with training using gradients. Other considerations when using the GPU for processing of neural networks in an efficient way, matrices need to be loaded transposed and padded so that the first item of each column is loaded on the first block of memory at the time of launch to the GPU, remember that we are optimizing the processing cost.

As well as the inputs, the outputs should be processed. Another highlight is the use of exponential functions, since they are not implemented in hardware, software libraries are normally used. Because this slows down the process considerably, if using exponential functions is required an approximation like Elliot-sigmoid may be used and you will not need to call higher order functions. On the contrary, using this function instead of tan-sigmoid (whose use is more widespread and continues to have progress) can, theoretically, reduce the speed of training. Again, your choice brings with it their pros and cons. In the following graph this function can be seen in purple.

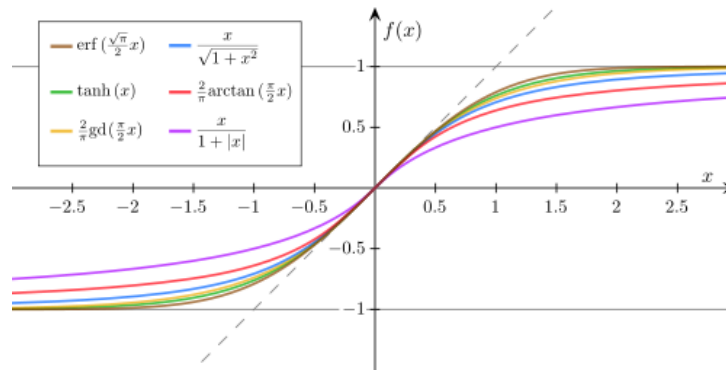


Figure 6.50 - Alternatives to sigmoid function

Following the defense of the processing of neural networks in this type of hardware architecture, remember that parallelism (any level) occurs between different sub-sets or in our case time-series, between different series. However if the network only has inputs delays (not layer delays) as in our case, inputs with delay can be pre-calculated so that when calculating each stage data can be separated and therefore be processed in parallel.

Our training, NARX in open loop, NARNET or time delay net are examples of cases where applying what we described in the previous paragraph we can use this parallel computing. However, if we want to directly train our closed loop NARX network when there are delays between layers, this cannot be flattened and therefore processing cannot be performed in parallel; on the other hand if our problem consists of multiple data streams, the same can be parameterized for parallel processing.

#### 6.4.1.4 Hardware Machine Learning & FPGA vs Cloud computing

Realizing that these models have reached a point in which is really economically feasible to invest there are various industries that are trying to take advantage and push this technology forward.

In this case the main bottleneck that we find is the hardware, there are now dozens of companies focused on the development of chips with different types of machine learning, although most of them use different neural network to approach according to the purpose of their products.

As we have commented in the paragraphs above, the fact that it is a problem that can be performed in parallel and distributed facilitates the evolution of these businesses since in the majority of cases, their processors, modules or devices take advantage of these benefits among many others.

The big difference of these companies lies in the optimization of brute force, while other companies are more practical to use large servers and data processing centers, even though in some cases they have the same approach to analysis and information processing.

The future of nanotechnology and its ability to deal with analog data will be the one true breakthrough that allows us to really get the dreaded Super Machine Intelligence. In the meantime and with the expectations held today in the evolution of systems and algorithms, we are at a stage where root limitations are arithmetic but its field for application will grow exponentially.

Is for this reason that the development of this hardware equipment designed and oriented for such purposes sooner rather than later will have a direct application and a market expansion, either for their more efficient management of energy or for its speed in massive databases. Integration with various tools, applications and a growing amount of fields make its usage in the medium and long term something certain. In annex one we can view two images that summarize part of the above explained in a very graphic way. (Alex Graves, 2014) (Narayanan Sundaram, 2010) (Janardan Misraa, 2010) (Markos Papadonikolakis, 2009) (LeCun, 2007)

As an example that less than five years in NN tournaments began using GPUs instead of CPUs. Today on many occasions already used specific processors for neural networks as cuDNN and they are starting to sell more complete equipment focused to these algorithms, like Neuronal Processing Units (NPU) or TrueNorth. All this exploded with the generalization of cloud services, but above all with the not distant aspirations of this decade like equipment based on memristor, silicon photonics, crossbar switch and even efficient focused quantum processors to ML. The optimization of the algorithms to take actual advantage of all these improves will open other branches to study. (Anon., 2015) (MIT\_Press, 2015) (Intel, 2014) (IBM, 2014) (HP, 2014) (Qualcomm, 2013)

#### 6.4.2 Bootstrap aggregating

Is a meta-heuristic technique aimed to improve the stability and accuracy of the machine learning, it also reduces variance and overfitting. Widely used in regression and statistical classification. Not being possible to directly find a global optimum is a practical way to improve the overall efficiency of the model using a kind of averaging models.

Commonly it is also known as bagging; we will explain how the technique works to obtain these advantages, due to the needs of our model we will apply in k-nearest neighbors. Assume it is true that the risk of k-nearest neighbors is at most twice than the one based on Bayes classifier; however we cannot verify that the classification is consistent.

Thanks to bagging if we choose  $n'$  with a large number of elements the classifier will be consistent. Although  $n' \rightarrow \infty$  diverges,  $n'/n = 0$  in a set with large number of observations. In reality, where the number of elements is finite, when practical application is more similar to k-nearest neighbors with weights. In this way the risk is defined by the following formula, where  $R_R(C_{n,n'}^{bn})$  reflects the risk of K-nearest neighbors with bagging and  $R_R(C^{Bayes})$  the Bayes one,  $B_1$  and  $B_2$  are constants and seek to choose the size  $n'$  which balance the two terms of the expression.

$$R_R(C_{n,n'}^{bn}) - R_R(C^{Bayes}) = \left( B_1 \frac{n'}{n} + B_2 \frac{1}{(n')^{4/d}} \right) \{1 + o(1)\} \quad (6.31)$$

It is for this reason that it is called packaging, since we are looking for the optimal number of elements that when grouped, will reduce the risk of diverging.

Seeing the results in the following example, we have a number of observations and instead of performing a regression for all elements, since as we know in this case it would not give good results. What we do is to divide the inputs into smaller packets, but large enough to have a distribution and variability similar to the data set. We repeat this process and then calculate a simple average of all the models obtained, as we see in the picture we get a result with an average much more adapted, stable and considerable less overfit. (Zheng, 2006) (Zhang, 2002) (Richard Maclin, 1997)

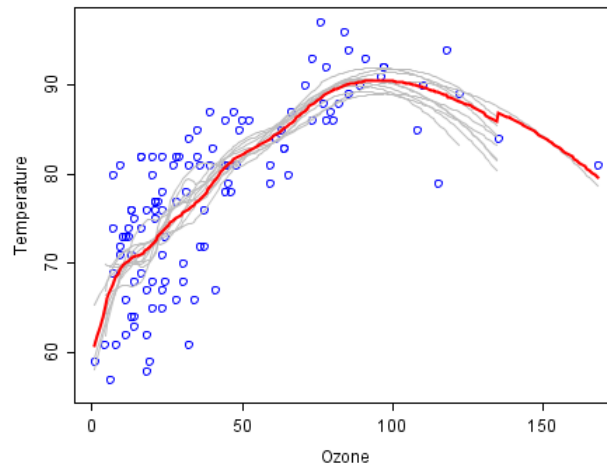


Figure 6.51 - Bootstrap aggregating example

### 6.4.3 Pruning functions and Evolutionary Algorithms

With this modification in the algorithms we seek to trim the network size in order to improve the performance of the network. The way to achieve it is by identifying the nodes that



if deleted won't affect the outcome of the network significantly. There are several ways to make this optimization, one of the easiest ways would be to see the value of the weights once the network is trained, if the value is very close to zero compared to the rest, we can deduce that it hardly brings added value to the network.

The problem of doing this during training is the risk of over trim, and we are possibly changing part of the network architecture. A correct pruning algorithm should approach the network to its optimal configuration in a similar manner as a Genetic Algorithm would do it.

However, this has drawbacks in cases very similar to the ones they are trying to reduce, value setting before or after the tuning, complexity of the calculations... which as a rule is often not applicable to real problems. A place where an interesting application of the GA in NN has been shown is in the initial approximation of weights or helping other more traditional algorithms to escape from local minimal.

There are a number of studies on the uses of GA to optimize neural networks abandoning the old use of the technique of pruning mentioned at the beginning of this point, training and gradually eliminate. In a simplified way there are two traditional types of pruning incremental and selective. In the first case the training begins with a basic configuration and is progressively increasing its complexity.

The picture below comes to reflect the previously mentioned use of "Neuroevolution" or GA in NN, although it should be noted first that there is a wide variety of these algorithms and secondly, is a field that is still in continuous study and the joint use of traditional Back-Propagation algorithms with some advantages of the GA can improve both training and general model optimization. (Dimitrios Mantzaris, 2011) (Huang, 2005) (Mao K., 2004)

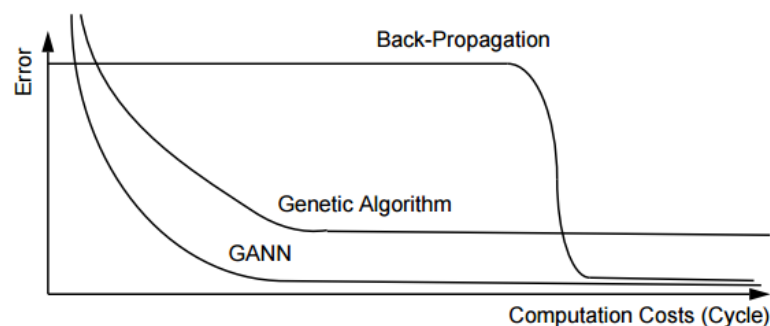


Figure 6.52 - GANN, GA and NN w/ Back-Propagation learning rate

## 7. Comparisons

We begin this chapter by defining the data set with which we'll work and the numerical results we will use for further selection. We'll apply the selected models to two different load curves: last forecast supply in d-1 offered by load curve REE (Spanish TSO) and the actual demand. Another load curves with a wide range of customers spanning the three economic sectors and finally five groups of different industries: farms, dairy, confectionery, metallurgy, workshops, transport and supermarkets have been studied. The results do not differ substantially from the actual demand despite having completely different patterns and behaviors, so that we can generalize the results of the real demand, having a respectable amount of white noise.

For comparison as we have defined and defended, with different studies referenced throughout the document, we will use the MAPE obtained for the load curve that will be used for learning and validation data collected from 2009-2011 (the first tranche of relative economic recovery) to a total of 609 days, see graphs in Annex 2. And one second MAPE obtained from the prediction that our model makes with a month in advance. Although our main goal is to predict the load curve for the next 24 hours, we proved how the MAPE varies for different days and under different conditions from values very close to one - even lower - to value of seven, with means by models between three five as detailed below. It is for this reason it was decided to do the average for a month with these models, that not being recurrent, and given all the information necessary to not depend on calculated values to predict subsequent iterations, does not change the average MAPE that would be obtained in the 24 hour prediction.

### 7.1 Input selection and preprocessing

#### 7.1.1 Factors and data selection

Because of the peculiar difficulties of the prediction of load curves more even if we deal with a wide range of consumers, from industrial sized to small consumers with different rates, it is rather limited the number of factors that we can include. So a key element to our correct prediction will be the use of relatively high groups of customers with similar characteristics of consumption and cost.

For the selection of which factors will be included in our models first we have to keep in mind how the models and their algorithms work internally, so it isn't the same loading data to a neural network that uses NARX or to an ARIMA model. Generally, we try to give information that provides the ability to predict values both within the range of scenarios - or values - previously analyzed and slightly outside them, see bias-variance tradeoff.

Since we only have the load curve and the daily average temperature. It will not interest us to force the model to analyze values carefully from the day before, since dynamically it considers them due to being a network trained by total stages. Feeding with so many values without actual information which deviates significantly we are losing considerable efficiency in the process.

One of the factors that will be used to decide which factors are valuable to our set will be the coefficient of Pearson, without forgetting that, as we have cited previously, correlation does not imply causality or normally distributed and uncorrelated does not imply independent. This factor calculated according to the following formula measures the force and direction in dependence of two variables, i.e. the correlation, in the linear case.

$$r_{xy} = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \sqrt{n \sum y_i^2 - (\sum y_i)^2}} \quad (7.1)$$

Note that there may be a wide range of relationships for the same value of Pearson as in the following image.

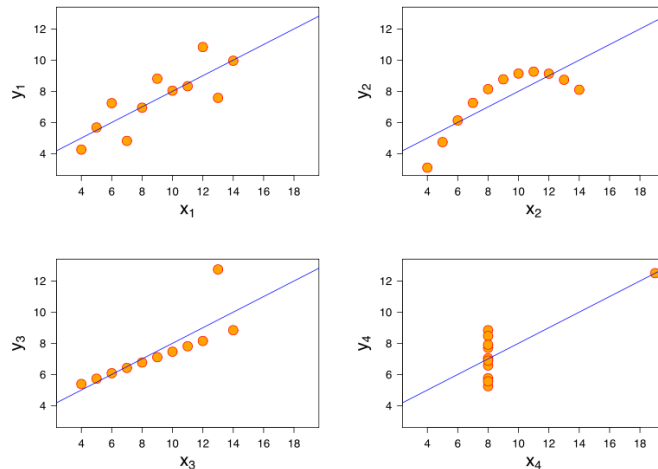


Figure 7.53 - Examples of same correlation coefficient

The correlation between the load curves and the values that we introduce will therefore be important to decide a priori which factors influence, in our case we present data with examples for the curve of demand in Spain.

First, let's look at a few graphs that represent the load curve according to months, days and hours.

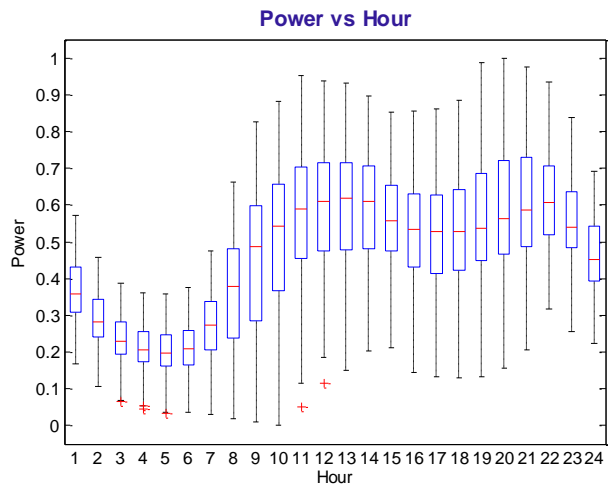
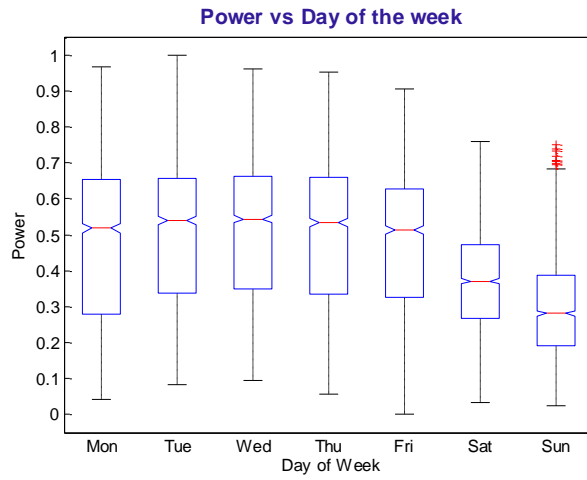
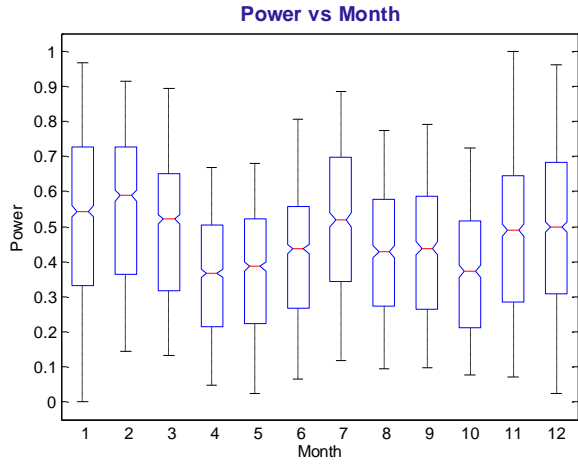



Figure 7.54 - Load time series divisions



Knowing our purpose and seeing the graphs we don't need greater defense of their inclusion as values of the time series. Even though they are progressive values, our algorithm should be able to replicate the evolution, i.e. knowing that after summer comes fall that there are weekends or that most people doesn't consume a lot at night. More graphs of this type are not shown since they have a relative similarity, even with the respective variations daily, weekly and monthly to the different areas of business generally.

On the other hand the holidays are a factor of relevance, in our case, we have used a percentage of the population affected and the long weekend effect, but the fact that many companies have work calendars slightly different to the officials or peaking when certain coincidences of dates are met. In summary, there are many factors that we can include both if we're trying to model the behavior of a company or a group of them. We'll see that these factors that are not included affect our results significantly and we'll see several methodologies to analyze the data and errors that we get to be able to better choose the data we load in our model.

Temperature is another known factor that affect consumption, mainly in low-voltage customers or companies with a moderate power consumption. In our case the maximum and medium daily temperature will be used, since we do not possess hourly, weighted by inhabitants of the ten most important cities of Spain. Other factors such as relative humidity, solar radiation, time and sunset would add more information to our system, but the conclusions, or rather internal algorithms parameters values, won't change significantly.

In certain industries or services the price of a commodity on which they depend could be added, or factors of evolution of the sector provided by the Government, expectations, or even manually adding predictable growth factors or temporary depending on expected developments and their impact on power consumption. Seeing some examples, for instance, if we are analyzing the curves of hospitals and it is expected that flu season will have its onset and peak a week earlier than usual, we can add this information if this factor affects consumption. If a new law or direct or indirect tax affects an important aspect of production for better or worse, we can include the expected change, whether progressive or not, on our load values. If we know events with a measurable impact on demand that are not public holidays and for example occur every few years, sometimes on different dates we can get conclusions of their relative impact in previous circumstances and apply them to our forecasts...

Therefore the only thing that we have is the historic load curve, which we try to find factors that provide more information without looking only at its correlation. It seems logical to include consumption at the same time in the previous day and the average of the previous day. To continue providing information after several tests it was decided to also make the average of the last four hours of the previous day. In the latter case we could choose to contribute more hours individually, however to simplify calculations and because the results in our tests do not vary significantly, the decision above was chosen.

As mentioned, values that are more significant a priori, along with the time and month, are loaded at the same time the previous day and the same hour a week earlier, but the final choice of variables has been chosen in such a way to provide information without being "unnecessary" data. Finally, we have performed multiple tests to check whether certain values are getting better results without excessively increasing the processing time. Two decisions resulting from these studies have been to include the average of the previous three days and the average of eight hours earlier in the d day, factors that initially we would think would not improve the results. See a table with the MAPEs initially obtained, we see how the neural networks stand out since is able to store more information.

	NarxN65	BD Trees	Simple RC
Both included	2.2866	3.3265	13.7201
w/o mean(d-1(h:h-8))	2.6492	3.3257	13.7201
w/o mean(d-1:d-3)	2.4768	3.3257	13.7201

Table 1 - MAPEs comparison between different inputs

The variables that after several trials and errors have been selected to be used, on occasion depending on the model and its Pearson correlation factor and variance, are shown to be the best idea of the relationship between the variable described and load in hour h. This table does not include other variables that are loaded, but are more specific, such as type of behavior of the day, level difference with preceding and following days, etc. Variables such as the day have been eliminated as they don't provide information to the model, but others as the year even though it has a very small correlation, if it influences our model then will be considered.

<i>vs load(h)</i>	Correlation	Variance
year	0.0266	0.2899
month	0.0770	14.9271
hour	0.5541	40.9423
weekday	0.0803	3.4385
holidays	0.2402	0.0326
maxDailyTemp	0.1893	0.0536
minDailyTemp	0.1909	0.0545
load(h-24)	0.8033	0.0378
load(h-7*24)	0.8569	0.0355
mean(load(d-1))	0.6102	0.0133
mean(d-1:d-3)	0.3437	0.0068
mean(d-1(h:h-4))	0.7107	0.0086
mean(d-1(h:h-8))	0.4779	0.0308
slope(d-1(h:h-4))	0.3845	0.0030
slope(d-1(h:h+6))	0.5659	0.0022

Table 2 - Pearson correlation and variance for inputs

The main problem we have here is the redundancy, since many of these values have a very high correlation and as we shall see in the next section the dimension of the loaded matrix can be easily reduced as to reduce this dimensionality. So after checking its benevolence for each model, principal component analysis will be used regardless to reduce the dimensionality of the data set if considered appropriate.

### 7.1.2 Initial preprocessing

As we have argued throughout the document, a key part in obtaining good results, regardless of the model that we use, is the data we load and how we load it. In our case and for reasons of length and time we will not extend in advanced techniques, we'll consider unsupervised learning oriented to this aspect, but we will use the most basic and necessary techniques with the ones we will obtain the presented results.

Most of the data has been obtained through searches in databases. The first step is to make sure that the dates for which values are obtained exactly match all data we load. In our case we have had to deal with the 23 and 25 hour days. Another important point is to analyze at a glance that the results have coherence, for reasons such as simple errors in the readings or measures, we have had to clean some curves since we obtained unreal values or behaviors.

Once we have properly sorted the data, we have calculated the values we want to add as information, mean of the previous day, the value of the same time in the week before... And subsequently they have been normalized to values between 0 and 1 according to the following formula.

$$x' = \frac{x - \min(\vec{x})}{\max(\vec{x}) - \min(\vec{x})} \quad (7.2)$$

Regarding the dependence between years, even though the  $\Delta$ GDP is a decent indicator in the case of this example, in practice, we will take out the trend of our objective values. We do this with the fast Fourier transform, obtaining a mean value equal to zero. In addition, we will make sure our vectors have a standard deviation equal to the unit.

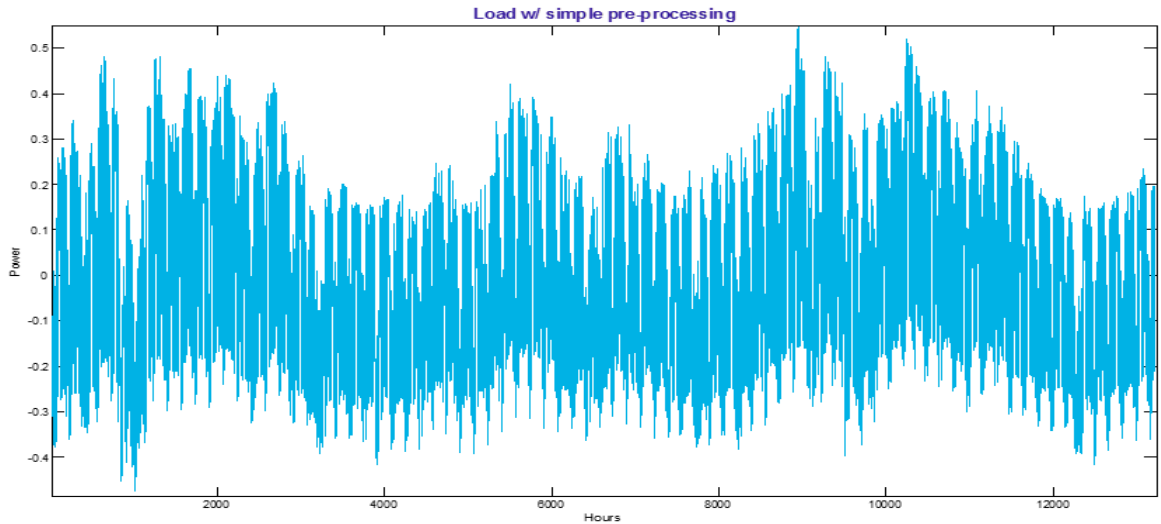


Figure 7.55 - Load with simple pre-processing

As an addition, and because during the first  $168 + 1$ , seven days in advance to get the time on the day  $d-7$  h and one by the delay that we initially chose. It is convenient to use two columns in such a way that if the algorithm is not capable of handling "empty values" or Not a Number, it knows how to behave in such a situation. For this, it is common to replace values unknown by the average value of the vector and add another column with 1 where the value is correct and 0 where the value should not be taken into account, i.e., where value were not available.

In our case, the last step before we load data to the models will be a principal component analysis or PCA, whose main characteristic components as detailed in the last paragraph of the previous point is to reduce the dimensionality. It also allows us to find the pattern of that variability in the data set and sort them by importance. This representation of the data set, considering square minimum will be used one time to obtain data and as an attempt to improve, since there are several methods according to the correlations, covariance and limitations that forces us to divide the set of data that we previously analyzed. For example, it is assumed that all data are linear combination to a unknown reference and if we use the covariance method means that the data follow a Gaussian distribution and only focuses on the sense of the space generated by the set of data, which in our case is not met.

The three main effects of this technique appreciably affect to the data set, since they orthogonalize the vectors so they are uncorrelated between them. Subsequently ordered the main component of these in such a way that those who have a greater variation reach first the loaded model and finally removes values that provide less variation in the set of data. It is important to mention that this technique can be useful for some algorithms, but sometimes counterproductive for others.

Another methodology that has been tested is Partial Least Squares regressions (PLS) which are based on projecting the predictors and objective values in a plane then get a linear



regression. In this way the relationship between two vectors or matrices are reflected. If we extend this methodology we can find in a space of multi-dimensional predictors the direction of maximum variance in the multidimensional space of objective values. However the results have not risen to the level of those shown.

Continuing with PCA and keeping in mind the table of correlations when applying this technique the size of our matrix is has been reduced from fifteen to eight, in particular and as you might expect, the vectors that were trying to provide more information on the evolution and behavior of the load have been reduced from ten to three.

In the table below we have the MAPEs calculated with PCA and the best values obtained previously without Cap. In neural networks, mainly since the number of neurons and settings has sufficient capacity to collect and decide on the correlations between variables it is not affected significantly. In simple regression, coefficients are worsened drastically, not having so many dimensions it is given less parameters to configure and thus loses flexibility. However, it is striking the drastic improvement in Bagged Decision Trees, reducing the dimensionality 47% has improved the result by 15%, in reduced values is an enhancement that we appreciate. It's understandable, again, is by the way of working on algorithm, even if we do not limit the nodes or branches while less "noise" or rather in this case, information not optimal, we load our model, the better able it will be to divide the roads that will end up deciding the predictions.

	NarxN65	BD Trees	Simple RC
Best w/o PCA	2.4768	3.3265	13.7201
w/ PCA	2.6136	2.8431	26.8869

Table 3 - MAPE values with and without PCA

### 7.1.3 Clustering and dataset division

At this stage we must first decide how are we going to divide our initial data set. Generally, we need three data sets, even though on some models with two will be enough since they are not dynamic processes. The first subgroup will be the learning one, with which our algorithms will conclude information in case of unsupervised learning, or the data that it will need to learn to conclude the objectives in the case of supervised learning.

The other two groups will be the validation and testing ones. The first is used during the learning process as one of the methods to terminate the learning, when we reach some level of performance or success, the learning finishes. Is then when we enter the testing set with which our model tries to actually predict and check the outcome.

Given the importance in the choice of data, and with the aim of not tweaking the results, it has been decided to split data randomly whenever the learning algorithm is launched.

Similarly, all the curves for all models work with the same data, being the group of testing the same month in all of them.

Regarding the month election, we have chosen June as it contains a bit of a varied complication, there are holidays, we begin to notice the summer holidays, the temperature increase, and different patterns in different businesses... We've found that typically the month of February, March, April, May, September, October and November as a rule have achieved better results because the values on which the load curve depends are greater. June, along with December are the months when our models, as a rule, perform slightly worse.

This directly leads us to try to improve where the model performs worse. Something that is totally understandable and can give us lots of ideas on how to improve it. In this section we will focus on the data that we can improve to that end. As we see in the picture below, where we have the error obtained -in this example on a neuronal network model I- in respect to time, we try to focus on why the greatest amount of error is seen in the various periods.

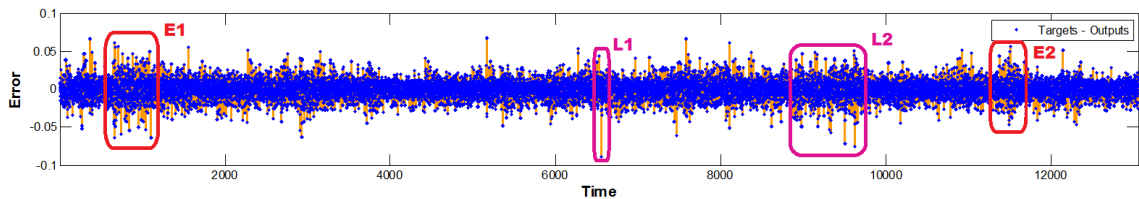


Figure 7.56 - Example of fit error

We can see that in E1 considerable errors are obtained along almost three weeks, but overall we see what information can be given to the model to learn that the values are not the ones that the example data is expected for that period. Similarly, in E2 where the error number is concentrated only in a week, maybe we can improve our values of holidays in that period, without forgetting that if we load information for that period, we must do so for the rest of the time series and check that the results don't get noisy.

Instead L1 possibly from an error in inserting data to an unpredictable factor with which we have to bear. If we focus on where we see a high L2 error for just over a month, especially for two weeks. We see how the normal pattern of the week changes considerably and although the model, probably due to the holidays, is able to predict with relative accuracy it stops being so efficient. In this case as in that of L1 is more difficult to reduce because either we manually change these events and we have to load larger data sets with tighter values and a variety of them to reduce the error in this type of circumstances. In other words, we have to increase the ability to predict outside the typical behavior, which returns us to the bias-variance tradeoff.

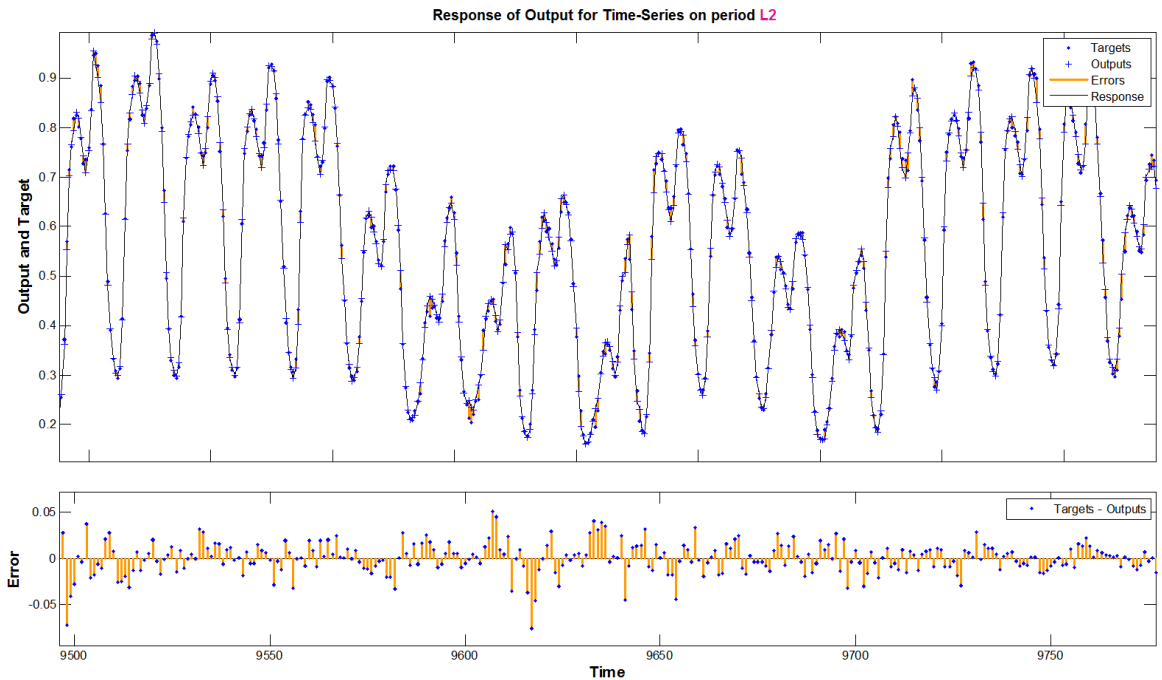


Figure 7.57 - Zoom in L2 error

On the other hand, if zoom even more on the errors, see picture below, we can see directly as it obtains very good results on the slopes and hours valleys, but fails right at the most important time, at peak times, where all services are more expensive or higher paid. In the example, sought with intention, we see very clearly a Monday that the model was not able to proceed correctly, is perhaps a long weekend, or in the case of being a festive can raise the factor that we have considered and in this way, try to force that under those conditions, the model should give greater importance to certain factors.

In keeping with what matters to us, the cost, we will have to find ways to improve the prediction at peak times even at the cost of reducing it in the high slopes or base hours. One of the ways has been mentioned previously and is as simple as applying algorithms of preprocessing more drastically. Others will be seen when we go into greater depth in customized neural networks, where you can specify in considerable detail how our network should behave or more applied to the case, which importance to give to which factors, for example, giving greater weight to the last values of the time series, being closer to the horizon that interests us to predict, or values that exceed a certain threshold, because they have more value for us.

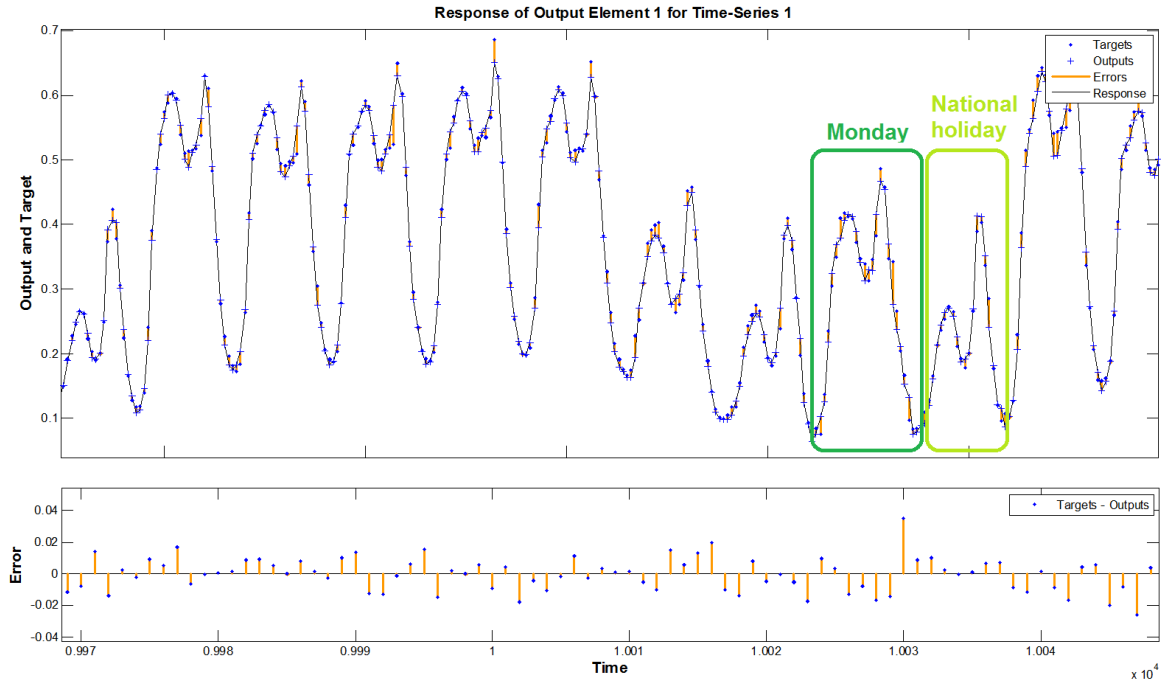


Figure 7.58 - Example of error due to long weekend

A third form, with considerable amount of benefits in the case being possible to implement is the one we'll name clustering. As we have seen even though our model behaves normally with a reduced MAPE, we can separate the holidays - including bridges - weekend, even if provide enough valley and peak hours information, in certain circumstances it could even be interesting to split the sets into summer and winter. This entails the complication of having to train several models and better adapt the input values according to the requirements as appropriate for each one of them.

But at the time of its development, it is not so difficult, without leaving aside other methods that we have seen and we will see for the classification, for simple tasks we could use an easy decision tree so depending on the date or hours, it will calculate a set of data and run a certain model. An approach having nothing to do with calculating various models and find the average value of their predictions.

The results we get in this case, will be, as far as possible due to the limitation in our input data, good enough to develop them. A model specializing in holidays, will predict much better than one focused on general regression. We are giving the model a capacity of classification which it does not possess by itself and that in these cases, where possible, is a substantial improvement, but we will have to use various models, both classification and regression, to significantly improve our results.

What has been exposed previously is one of the many reasons why in machine learning, even in aspects mainly focused on forecasting, we can see algorithms initially thought for classification and decision support. It is therefore necessary to understand the operation,

advantages and disadvantages of the models that nowadays are getting state-of-art results and the need to describe it at least superficially in this document is defended.

#### 7.1.4 Feeding the data and deeper inputs analysis focus in NN

Once we have chosen and pre-treated the data we needed to load it into our models. As discussed briefly, we'll divide the data set into three subdivisions, depending on the models. In our case we use 80-85% for training, 10-20% for validation during the learning process -if necessary- and 5-0% for testing. These divisions even if randomly, as in our case, create significant variations in the results so it is important to do multiple trials to find a relatively optimal balance point. Similarly, being random subdivisions and initializing the weights and bias in the case of NN randomly each time we teach our network will get us significantly different results.

Unfortunately, all our results tend to confirm that if they're much more value available in the time series our results would improve. Since this is not possible, we will considerably force the percentage of data to devote to each end, trying to mitigate it with a larger number of trials as explained below.

We'll add that the validation set is used to measure the generalization that holds a particular state in the training so that learning stops if you think you can generalize well enough. Another part is to define what is considered good enough. In our case, we will repeat multiple times as we go optimizing the results and we are not interested in the mean values of the configuration MAPEs but in the configuration that gets best results. We will directly find these configurations, even if they are not the best; in certain situation they have been able to internalize the problem better. In practice, we'll stay with the excellent model, even if we only have one, and not with others who are very good on average.

While our requirements to stop learning will be that the learning is reducing continuously towards success of the results, we will keep the parameters that have achieved the best result, or considerable requirements in which the increment made on the parameters will produce inappreciable changes on the outputs. Given that our goal is to have models that either do not re-learn in a systematic fashion, or they are simply updated, it doesn't matter to us if the first learning time is high.

In the case of the subset of validation, it must be accompanied by requirements under which the algorithm stops updating its parameters. It is important to recall again that our aim is to predict with lower percentage of error when its knowledge brings greater economic value. Therefore, we should use all the data that we can.

Our models are provided as part of the pre-processing of several special vectors to indicate this type of days, in such a way that if the behavior of a workday - supposedly - distorts the behavior of the time series too much, we add a value of difference as a flag to try to provide more information for the algorithms.

With this marker and because of that we accept a reduced level of error we are deceiving the network. Thanks to this, we reduce the noise and pollution on our results. Let us remember that our data are far from being perfect and are our biggest constraint. Very briefly, in the case of electricity demand, there are hundreds of factors that we do not include; in the example of power reserves there is an important human factor that we don't provide any information about. Ask for our simple model try to internalize these scenarios without more information only worsened our media results.

Given that this analysis should be able to generalize identically for all the data set, once we have the results we should look for the reason of why these behaviors occurred. In our case, and with the purpose of providing two examples, we will measure the difference between the peak and the valley from the average value of the day to define the type of day pattern on one side and the average value of the day on the other. We will compare these values with those expected depending on the type of day and date of the year obtaining a value that will identify uncommon days. In this way we will have two extra identifiers, which we will use in the cases that we consider to be necessary, or rather, in cases where the input improves the results.

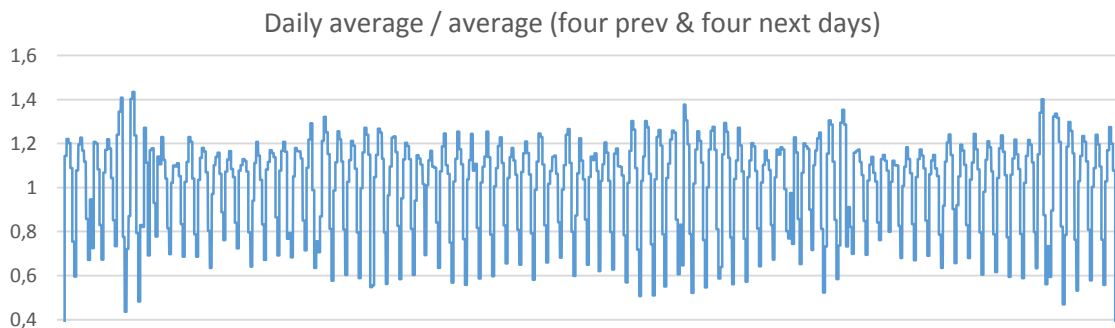


Figure 7.59 - Example of time series used to get expected errors

With this example, without any other additional pre-processing for highlighting the improvements, we have done a sampling of 20 executions with the identical NARX neural network configuration and we have improved the results 3.6%, passing a best result of 2.11 to 2.03 and an average of 2.28 to 2.25 being these the values of the obtained MAPEs. In our case, we will make dozens of small improvements gradually while we will make various optimizations of our configuration, in such a way that the actual percentage of improvement that each approach adds will be considerably more reduced, but we'll try to add them and achieve better results.

On the other hand, but without abandoning the analysis of input data and preprocessing, we'll see other techniques that improve both the speed of learning and the precision of the output data. Given that initially we won't use many ratings or labels, it will be enough to indicate the values in the corresponding vector. However, in the case of different problems, for instance decision support; we can load the information with the method known as thermometer coding where the definition of the tag is described in binary manner. Label one could be 0 0 0 0, label two would be 0 0 0 1, the third 0 0 1 0 and so on.

According to our own experience in hundreds of tests with different models and multiple configurations of optimization algorithms or the improvements discussed in point 8.1, we get to significantly reduce the MAPE between one and three points, compared to a gross insertion of a lot of raw data and no other process than the normalization.

Before proceeding with the comparisons, remember that in the case of predictors the same processing done to the target values, has to be done to the outputs of any model.

For clustering with several models according to their classification, we can automate the division and subsequent union in a trivial manner. As a general rule our models will be better interpolating than extrapolating (bias-variance trade off) so it will be interesting to get a target with a range of probability or even better a confidence interval.

In addition to the normalization and scaling, we can transform our vector in the dataset in such a way that they have a normal distribution. The trends also has been processed and eliminated in part, but if we want to analyze the relative volatility - not in the example in which we focus - we can limit this part of the preprocessing for results with different meanings according to the purpose of our problem.

Another point to consider is the seasonality, in our case 24 hours, 7 days and seasons. In none of the studies that have been read about the development of this document have they achieved better results with delays of 24 hours. In our attempts, besides seeing the RAM requirements increasing considerably - expected - the processing time and the results do not provide good expectations, MAPEs over 10. So this factor, although it is important, we understand that the models we use -focusing on all configurations of NN- are able to internally understand this seasonality without affecting the results. It also shows that it is important to know the state at earlier stages, even if they won't be compared (same time or same day or month) as well as in the case of hours and days, from previous stages.

The needs to introduce dates will create a circular discontinuity problem in our network, since we pass from December 31 to January 1. Because of our approach we cannot load it as a simple temporary series since it does have those repetitive decisive factors. To better understand how a neural network thinks, in the time step 365 a series of neurons activate but in 365 + 1 another quite different set of neurons is activated. However the difference is only a time step, i.e., we are saying to our network that the 365 value differs substantially from the value at 365 + 1, concerning time step, but we know that it is not so. If we see it with the PCA applied

example, where we have seven vectors and four of them are indicative of dates, we will have that more than half of the information that we provide is not giving information to the model in the simplest way possible. Again, we're saying that circumstances in  $t + 1$  has nothing to do, 4/7 in our example, with respect to the events in  $t$ , although we know only one hour passed.

The best and most efficient way to deal with this problem, is devoting two neurons only to date. These two neurons in the input layer will change very slightly as the time series develops. An example, which we will test, will be encoding the date 0-365, and divide this vector into two vectors, for example:  $\sin(t)$  and  $\cos(t)$ .

## 7.2 Results

### 7.2.1 ARIMA

In this section only the final values of MAPE should be considered. The accomplished process, adjustment of parameters and final results with the exception of the MAPE should not be taken as a reference or comparison. This chapter has the sole and exclusive purpose of showing which results can occur in the form of MAPE with traditional methods and not to be understood as part of the study, learning or development accomplished.

ARIMA methodology already has been explained very briefly in Chapter 5.6 Multiple Linear Regression; at this point we will take into practice the theory previously exposed. We have a stationary model, i.e. without trend, a linear trend model and a model with seasonality. Understanding that this approach to the problem of the time series is that there is a systematic behavior and the value obtained in  $t$ , depends on the systematic behavior and a random disturbance. Other methods such as ARIMAX give more parameters and adaptability to provide more external variables to return the objective result. However, these parameters are outside our comparative to take as a reference.

The ARIMA model is affected by the volatility of the time series, however, for "type" periods is hopefully a good behavior. In our example to predict demand, we see that we have a high frequency, different seasonality, media and non-constant variance, high volatility, influence of other factors such as weekends and public holidays and a considerable amount of atypical consumption, especially during periods of high demand. The procedure we will follow is presenting the series with constant mean and variance. In the case of the mean we use differentiators and logarithms to stabilize the variance. We'll make use of the autocorrelation and partial autocorrelation functions to identify initially, which model best fits our time series; also it will help us to identify seasonal behaviors.



Following the Box-Jenkins methodology, we will first remove the seasonality of the series. In this first graph we see the function of autocorrelation linearly decaying by what we can conclude that our function is not stationary and we must derive it. (Pankratz, 1983)

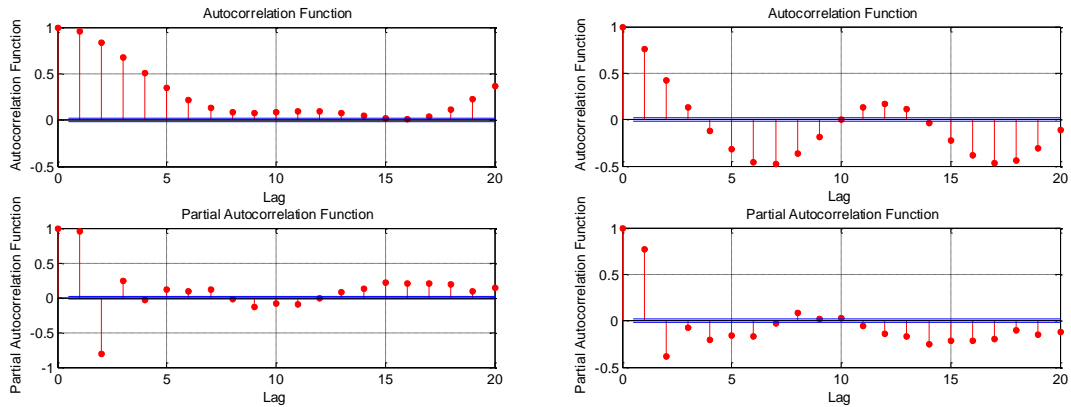


Figure 7.60 - Initial and first derivative autocorrelation and partial autocorrelation

The next step of this methodology is to estimate the parameters of the model, in our case the one with the best results is with ARIMA (3,1,5) and higher values, we can say that according to (Galván, 2011) an ARIMA (1,0,1) model should get significantly similar values. In the next image we see the residual of the resulting model; we will take the opportunity to ensure that they follow a normal distribution and are uncorrelated.

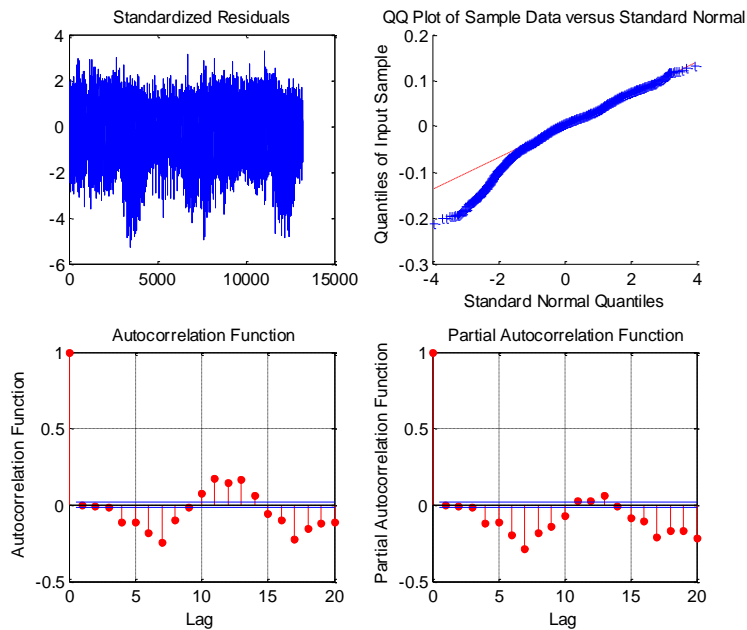


Figure 7.61 - Errors and correlations after ARIMA

The residuals are reasonably normal distribution and are uncorrelated. Despite this, the results obtained with (3,1,5) differ significantly from being acceptable outside a horizon more than eight hours. So any discussion in the configuration should be heard, since with Monte Carlo simulations the obtained MAPEs are superior to 12, with values of p, d, q close to what one would expect.

```
ARIMA(3,1,5)
Coefficients:
    ar1    ar2    ar3    ma1    ma2    ma3    ma4    ma5
    0.7188 0.7298 -0.9852 0.1603 -0.9777 0.2652 0.6014 0.1876
s.e. 0.0019 0.0011 0.0018 0.0084 0.0083 0.0101 0.0074 0.0082

sigma^2 estimated as 0.001084: log likelihood=26372.4
AIC=-52726.81 AICc=-52726.8 BIC=-52659.4
```

**Table 4 - Coefficients and significant values of ARIMA(3,1,5)**

On the other hand, if we calculate the similarity of the estimated with the objective value, we obtain much better MAPEs, primarily with AR (n, n), but to get to the MAPEs estimated with one or two orders of polynomials in our case we need to multiply these values up to 10. With even higher orders we have reached a MAPE for the time series of **2.77** in total, however the fact that it is necessary to increase considerably the orders of polynomials does indicate that it is not done correctly. Using ARMAX hasn't achieved best values of MAPE neither, nor even close to expectations which again indicates an error in the definition of its parameters.

In short, the best MAPE results that have been obtained by this method, for the global data set called Fitting, and for the next month of the time series and Forecasting, are shown in the following table.

<i>MAPE</i>	Fitting	Forecasting
<b>ARIMA</b>	2.77	2.93

### 7.2.2 Multiple linear regression

With this model, we estimate the general trends of the time series, however it can't predict nonlinearities. Basically, it generates a vector with coefficients that as they multiply different values of the input vectors they try reduce the difference with the expected targets.

The lower MAPE achieved was **13.71**, as for the time series as well as for the month we used to measure the ability of prediction. If we go more into detail, we can see the values and their meaning.

The estimate column refers to the approximate value, is the standard error of coefficients calculated. And pValue comes from t statistics under the assumption of normal

errors, we can see as there are values very close to 0, these will be the values that interest us the most, since they will be with those who can better predict the behavior of the time series. The MAPE only increases to 14.16, 7% and have reduced the amount of data in more than 30%. This analysis is useful for models that are based on this optimization, similar to the pre-processing techniques oriented to neural networks, which reduce the results obtained with this type of models.

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	-1.096	2.5335	-0.43261	0.6653
x1	0.00051159	0.0012598	0.40608	0.68469
x2	7.6937e-06	0.00019065	0.040355	0.96781
x3	0.0050201	0.00011704	42.894	0
x4	-0.0034111	0.00030831	-11.064	2.5115e-28
x5	-0.040713	0.0028163	-14.456	5.3104e-47
x6	-0.0029989	0.006944	-0.43187	0.66584
x7	-0.0051953	0.0067911	-0.76502	0.44427
x8	0.40181	0.013591	29.565	5.426e-186
x9	0.33075	0.0049527	66.781	0
x10	0.6712	0.0067805	98.99	0
x11	-0.078815	0.0093147	-8.4614	2.9215e-17
x12	0.43507	0.016402	26.526	4.7772e-151
x13	-0.70457	0.019098	-36.892	1.7726e-283
x14	-0.66556	0.035756	-18.614	2.395e-76
x15	-0.34161	0.01805	-18.926	7.9471e-79
x16	-0.010046	0.0026236	-3.829	0.00012926

Table 5 - Statistical coefficients for inputs

To identify the outliers we will select in the high-leverage points, those that have greater Cook's distance. This measure serves to estimate how it influences each point when a square regression analysis, as in our case is, done. Thus we can select two periods where values are "more in consideration" than others, so you should make sure to minimize the error and if possible provide better or more information.

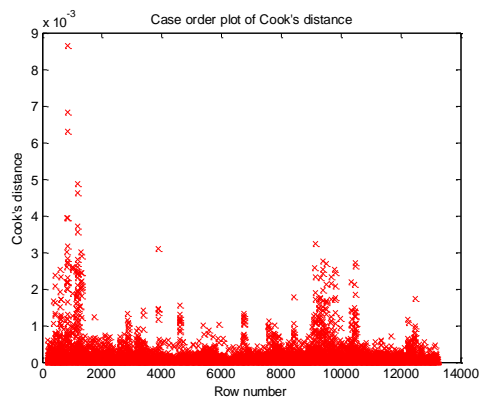


Figure 7.62 - Cook's distance

In the histogram of residuals, we see that they follow a normal distribution with zero mean by which the model has been adjusted well to the entered values. However, we see in the following graph that there is a correlation between the residuals, which shows us another field of improvement. We can also identify a cluster that should be analyzed to find out the reason for its existence and treat it if possible.

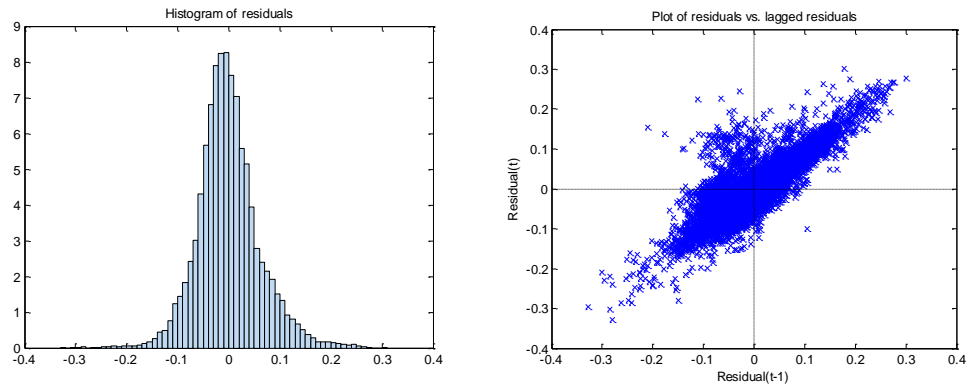


Figure 7.63 - Histogram and plot vs lagged residuals

Following the analysis of results, we can see how the final value influences the variations of each variable. We can understand the values, x1 refers to the hours of the day, and generally consumed more when these values are higher. A similar thing happens with x2 that reflects the type of day, however, in the graphic in question when calculating the average it is not appreciated significantly since 7 is Saturday and 1 refers to Sunday. We can also conclude, according to the graph the x4 maximum temperature affects more than the minimum x3, and without abandoning these parameters than the range in which the effect of low temperatures is less than the higher, so on days where this factor is relevant, we can expect one range less than the expected values. The chart continues with parameters such as load from yesterday and last week, daily and by groups of hours, mean values as well as pending.

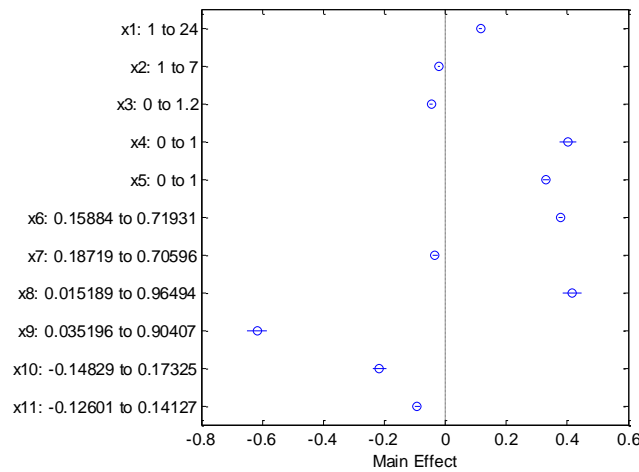


Figure 7.64 - Effect of each variable on the outcome

Other methodologies to fit the curve have achieved close, but worse results, highlighting Fourier eight parameters achieves a MAPE of 16.23, Gaussian with eight parameters a MAPE of 16.25 and a ninth grade polynomial function 16.26. Due to the high number of inputs,  $R^2$  values obtained by different interpolation methods are very close to one, but when calculating the MAPE does not prove to be as good. In the picture an example of an interpolation between the load nearest neighbor in hour h of day d-1, d-7 and the target value, the load on h.

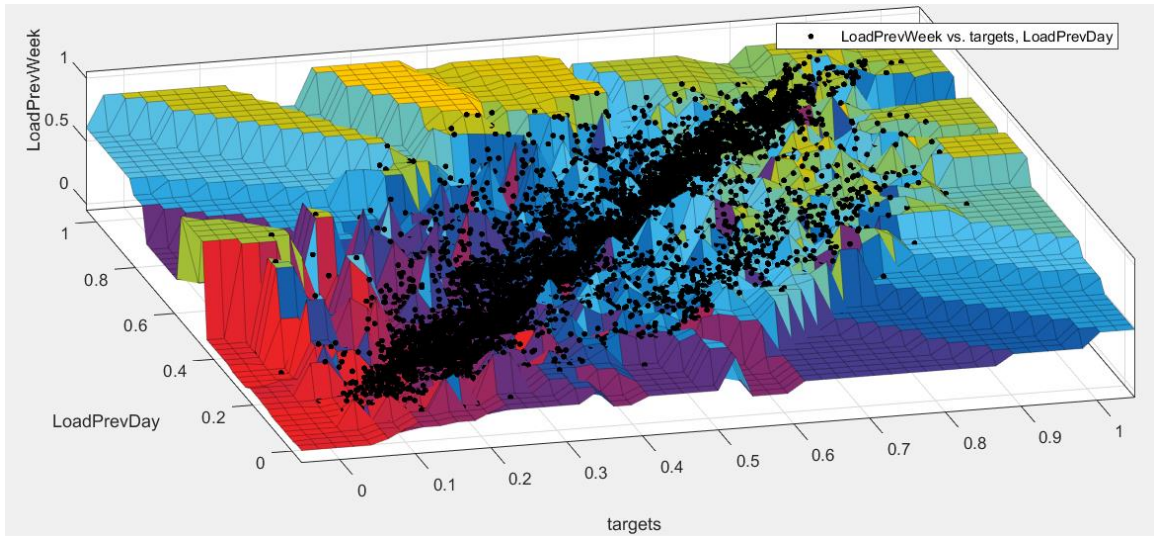


Figure 7.65 - Load in  $t$  vs  $t-24$  vs  $t-7-24$  and fit

The best results obtained by this method have obtained MAPEs as shown in the following table.

<i>MAPE</i>	Fitting	Forecasting
<b>MLR</b>	13.71	13.86

### 7.2.3 Bagged Decision Trees

With this methodology, in our case focused on fitting and forecasting, which is explained in the section 5.5 *Bagged Trees*, the best result obtained is a MAPE of 3.3 for the complete data set and **2.34** for the month forecasted. Below is a picture of a very small part of the decision tree created.

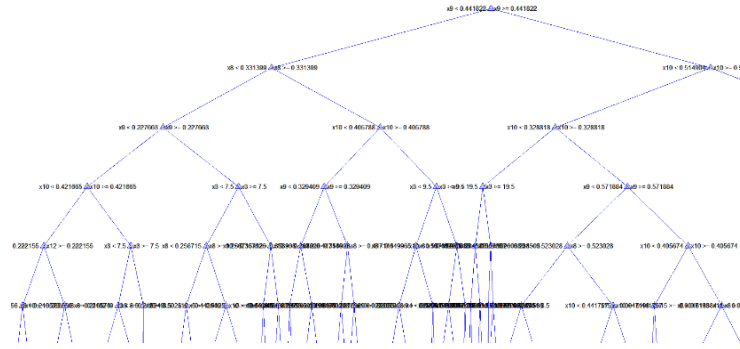


Figure 7.66 - Small section of the Bagged Tree

It has been decided not to reduce the number of nodes or branches, or increase the number of possibilities or ways since the memory requirements are not high and although slightly it affects the retrieved MAPE.

	MAPE	Fitting	Forecasting
BD-Tree	3.30		2.34

## 7.2.4 Neuronal Networks

### 7.2.4.1 Introduction

We will discuss some of the results that we have obtained in this section of the neuronal network. Firstly the results shown will be made with Levenberg-Marquardt backpropagation training algorithm; however, we will analyze other algorithms and Bayesian regularization backpropagation is used since normally it returns better results despite its longer calculation time as it will be discussed later. These other algorithms with Bayesian regularization will be analyzed in the final stages because of the good results they have in learning or better said in the case of Bayes, the cumulative update Bayesian probability that an event occurs according to the historic of events that have been empirically until such time.

To obtain reliable results in each configuration we'll take ten samples, five in the case of algorithms not LM. We will get on the one hand the mean value and the minimum value for each model, generally we'll keep the configuration that gets the lowest MAPE for the time series.

However, due to the difficult decision of some parameters in certain occasions, we will chose points or settings that even though they are close to the local minimum, our expertise makes us believe that according to the theory they should behave better in general. In the final stage of selection we'll add another indicator of performance being the average of the best model among the MAPE obtained for the entire time series and obtained as a pure predictor.

Another point to consider is the aggregation of different models are trained networks and getting its average, in this aspect we will use this idea to try to find a more optimal bias-variance's point as we will see in the tables. As already explained there is a considerable difference between the results for the different configurations, the standard deviation of the results sometimes is substantial, so in addition to our intuition we will use the results to raise questions and corroborate the theory previously studied. Again, we are still limited largely by data, the time, the computational power and memory for the learning phase.

On the other hand, we will understand if one improvement has been chosen, unless our knowledge tells us otherwise, it means that it is a good choice, and remains as the right path to follow while various improvements are accumulated. Yet different checks will be made during the process to make sure the selected configurations do not go against decisions previously taken. In this way we can choose the relationship between bias-variance that best meets our needs.

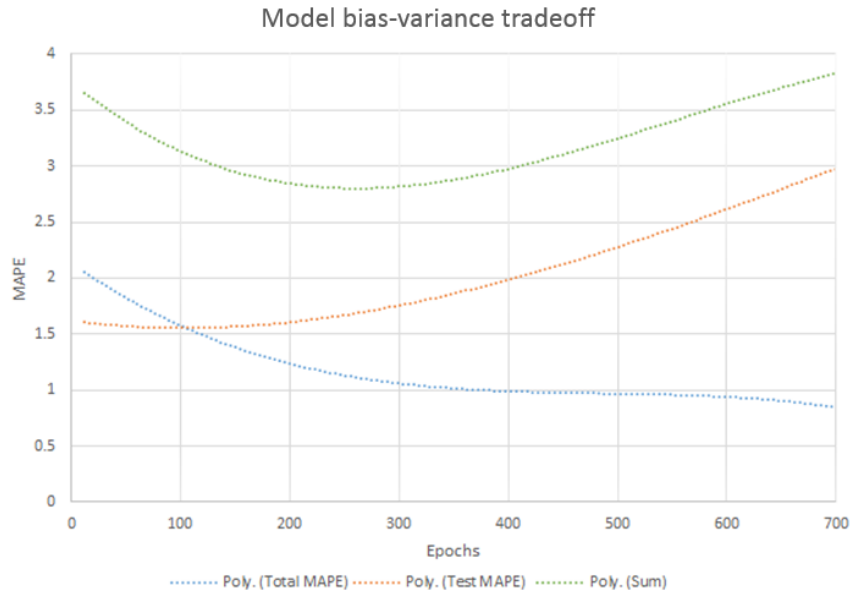


Figure 7.67 - Actual bias-variance dilemma

In the following example as well as in subsequent ones, until is mentioned otherwise, the basic configuration will be a NARX neural network, with two layers, 170 neurons in the hidden layer, without recurrence (like in open loop), sigmoid function in the hidden layer and linear in the exit. In this way, even though we know that it is different from the final result, we can compare the performance of different changes in a more precise manner. We start with a NARX model because we have the possibility of bringing more information than only the temporal series and in certain circumstances, holidays, temperature and events add more information that otherwise could be considered noise or random component depending on the features. Finally, remember once more the bias-variance trade off, that accompanies us throughout the process and will be our decision according to the demands of the objective

relationship, for our part we will try to achieve the lowest value in both within our current limitations. The actual third grade polynomial regressions of the values obtained are shown in the next graph to show the behavior of the model vs the number of epochs.

#### 7.2.4.2 Number of neurons

We will begin with the initial optimal number of neurons in order to store and understand the problem to represent. As we shall see in the following graph, as we expand the number of neurons, our bias is increased, but the ability to predict outside the set decreases rapidly. On the other hand, the results of the average value of the total MAPE, indicates that we are approaching an asymptote with the configuration where it corresponds to the factor of neurons. So the values that have been analyzed to a greater extent [160 and 180] have increased to 20 the number of repetitions. The results have not been expected, but tend to reflect the ones retrieved previously, although it would be ideal to repeat this process with a greater number of iterations once the structure of the network, it is well defined.

If we see how the improvements in other aspects evolve, we can conclude that the point of intersection between the curves have sacrificed it in exchange for getting a slight increase of the bias. The number of neurons where the reduction of the total MAPE is negligible and the MAPE of the set of testing does not skyrocket is (170-180), for our future cases, we will use the reference of 170 as the number of neurons, since we don't want to move unnecessarily from the previously mentioned intersection point. Although values around 50 or less showed better performance in addition to significantly reducing the time required as we will see later, at the moment the choice of 170 gives us more flexibility in testing without seeing us limited by this factor. The great variation in the monthly values is due in part to the configuration of the network, in our case, the amount of data is not very far from the optimal but it is not reduced. So during this stage 15% of all data have been used for terminating the network training. This is reflected in major changes since a small percentage of the value of the time series is used for stopping the learning, so we'll have iterations where the level of learning change significantly. This aspect and its consequences will be treated in greater detail later. Increasing the number of repetitions has failed to draw a more linear graph, but if it gives us an idea of its behavior. (Eric B. Baum, 1989)

Finally, we should highlight the reduced MAPE obtained with values around 50, which also significantly reduces the training time. Let us remember that we cannot abandon the value obtained in the testing month, since the economic results in a relatively high point of interval confidence given by the model could well be these values. So, subsequent analysis will return to these numbers.



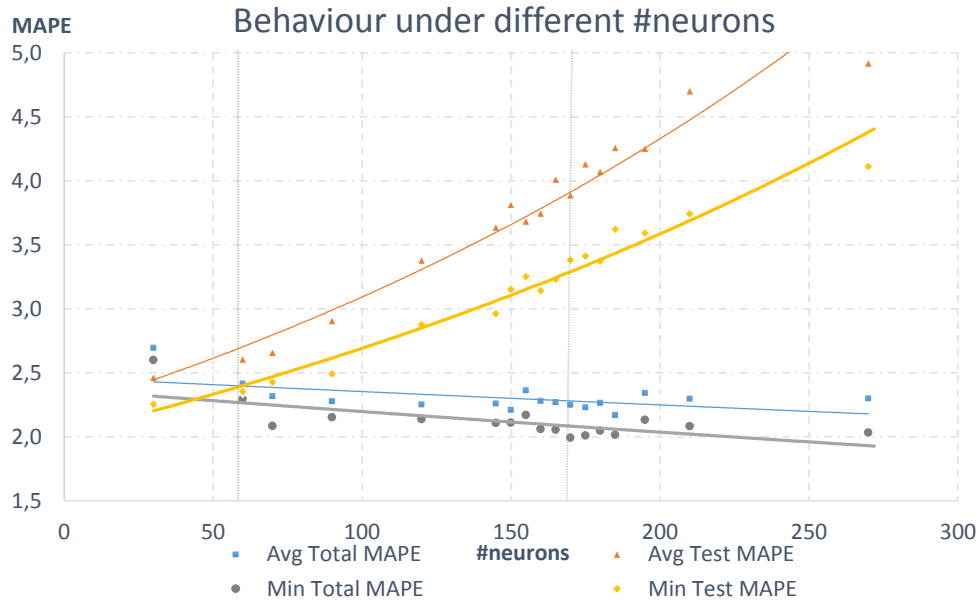


Figure 7.68 - Comparison under different number of neurons

### 7.2.4.3 Delays and feedbacks

Secondly, we will make a comparison between the input delays, which we will call  $id$ , and the feedback delays which we call  $fd$ , in the majority of the papers read a better result is obtained with the use of one delay, but this is affected by how the data is loaded, and this is not usually detailed with sufficient depth. For this reason we will do our study of our data and how we load them.

In the first test set we analyze how it changes the MAPE only varying the delay on inputs, with  $fd=1$ . As we can see in the following table, with  $id$  zero we obtain better values than with one, this is because in the input values, the descriptive time series is paired with objective values without gaps. So to describe the hour  $h$ , we must consider exogenous  $h$  parameters, since the values of  $h-1$  (which explains the result in  $h-1$ ) provide less information, and they are the ones we use in the case of  $id$  equal to the unit.

This is because of the data and how we are loading the data in each stage, according to our variables the model in each iteration has the necessary information on  $h-24$ , so if you need that information to carry out the forecast they are directly available. Similarly, the other variables that we are giving are not displaced in time, so to predict the value  $t$ , we are giving the values  $t$  Otherwise, we are teaching our network to predict the value  $t$  with values of  $t(id)$ , when the last values it has received and adjusted their weights are the  $t-1$ .

On the other hand, we already are in the first stage of the configuration with the third boundary that will prevent us to achieve results worthy of this methodology. Adding time and input data, we need to add capacity calculation and memory for that state. In all cases, these three factors can be reduced substantially in simple ways as explained throughout the document, but in our case we cannot extend our limits in a practical way.

Observe as our MAPE is less if we use all day as added information values, this requires a nearly exponential increase of memory and thus in our case it can't be realized in hundreds of trials as it is our aim. Other values have been considered, for example, if we include the  $h$  value and the previous value in the same stage of calculation of gradients to change weights and bias, we obtain an improvement in the variance, which as we have explained, has an economic impact. The following table shows the values have them as reference later for a  $fd = 1$ .

InputsDelays	Average MAPE		Minimum MAPE	
	Total	Test	Total	Test
0	1.98	3.57	1.71	2.78
1	2.32	2.52	2.16	2.23
24	2.49	4.23	2.19	3.55
0:1	2.00	2.49	1.69	2.21
0:6	1.83	1.95	2.55	2.52
0:24	<1.74	<1.84	<1.68	<1.66

Table 6 - Several inputs delays with feedback delay equal to one

This parameter will be linked, especially in the case of a NARX simple network, with the feedback delay, even more so when we close the network, the value of the feedback delay will be decisive when it comes to predicting values without more information than continuing with the time series. Unfortunately, we are very limited and we cannot configure the network in a manner radically aimed at the lower values of MAPE. However, we have obtained good values with  $id = 0$  and  $fd = 1:24$ , being reasonable learning time. The following table shows the values obtained with  $fd = 1:24$ .

FeedbackDelays	Average MAPE		Minimum MAPE	
	Total	Test	Total	Test
0	1.60	1.99	1.43	1.86
1	1.62	1.92	1.55	1.73
1:6	1.82	2.18	1.69	1.97
0:1	1.71	1.76	1.65	1.61
0:3	1.74	1.70	1.67	1.67
0:6	1.77	2.19	1.71	2.02

Table 7 - Several inputs delays with feedback delay from one to twenty four

The number of explanatory variables directly affects the considered optimal number of neurons. If we only have two explanatory variables the number of neurons that we need will be considerably less than 50 or 170. After considerable testing, we have concluded that best results are obtained with our current model and the previously selected and parameters and variables, the more information we provide the better results we'll obtain. This is one of the capabilities of this plasticity of the neural networks; it knows how to differentiate what information is important, when and under what conditions apply to it. Below is a small table, the relationship of variables has been obtained following the values of the p-value table. As we can see between the SubSet #4 and all the DataSet we have the point of intersection between bias and variance, so the last data that we are loading is too focused on understanding the data set and not the generic behavior of the load curve. The subsets are composed by:

- SubSet #1 = prevDayLoad
- SubSet #2 = SubSet1 & averL4PrevHours & averL8PrevHours
- SubSet #3 = SubSet2 & weekDay & holidays & slope4PrevHoursPrevDay & slope6PrevHoursPrevDay
- SubSet #4 = SubSet3 & hours (the NN knows the 24/7 time-series pattern anyway) & lPrevWeek & averLPrevD & averLPrevDtoDminus3 & SpecFlag

SubSet #	Average MAPE		Minimum MAPE	
	Total	Test	Total	Test
1	9.08	7.10	9.02	7.04
2	3.95	3.47	3.83	3.34
3	2.62	2.40	2.59	2.33
4	2.22	2.04	2.14	1.95
DataSet	2.10	2.28	1.94	2.11

Table 8 - Comparison between inputs subdivisions

While the Bayesian regulation algorithm does not need validation set, due to the considerable increase in time required for training, we'll see how the division of data affects the results. In this case, we'll use 20 samplings due to the small variation expected to be found. Because, as the factor of "only" providing a year and a half of data and not excessive explanatory values, it is, has been, and will be the main limiting factor of our results. We will aggressively try to devote the largest amount of data to learning and as a method to reduce the consequences a greater number of repetitions will be performed. In the case of using the Bayesian approach we will reduce the set of testing since we checked separately, this way we can dedicate all data, except those used to check the forecast, for learning skills.

Division	Average MAPE		Minimum MAPE	
	Total	Test	Total	Test
70/15/15	2.17	2.30	2.08	2.21
80/15/5	2.23	2.27	1.93	2.11
88/10/2	1.95	2.08	1.89	1.87

Table 9 - Comparison between divisions for learning, testing and validation

With the following comparison, we seek the optimum point of pre-processing. While we talked extensively about this aspect, the following table only shows the results that have contributed something to the final model. In the majority of cases the results have not improved or have been ambiguous, although as a general rule the training time was reduced; in some cases information was lost.

Returning to the table, as we can see we don't have decisive results. And although sometimes PCA seems to get good results, we remember that we considerably simplify the information we deliver to the model. This goes against the peculiarities of neuronal networks, such as memory, dynamism etc. as we expand the number of neurons and layers we will be granting greater freedom to conform to the dataset. For example, if we are providing a massive amount of data and have a limited RAM and time, a strong pre-processing would be advisable, it won't be necessary initially using PCA or other regressive methods of variable 'simplification' or variable's relationship. Otherwise, we can try to give freedom to the model to fit itself using simple and plain pre-processing methods, as it will be our case, normalization, cleaning of data that do not provide information, zero mean and deviation standard equal to the unit and fixing the unknown values.

Preprocessing	Average MAPE		Minimum MAPE	
	Total	Test	Total	Test
Reference	2.17	2.30	2.08	2.21
MapStd + MapMinMax	1.99	3.24	1.91	2.94
MapStd + FixUnknown	2.08	2.67	1.96	2.51
MapStd + MapMinMax + PCA	1.97	2.71	1.89	2.44
MapStd + FixUnknown + PCA	1.97	3.01	1.80	2.34
MapStd + FixUnknown + MapMinMax	2.24	3.19	1.98	2.80
MapStd + FixUnknown + RemoveCst + PCA	2.14	2.95	2.00	2.59
MapStd + FixUnknown + MapMinMax + PCA	2.09	2.47	1.95	2.21
MapStd + FixUnknown + MapMinMax + RemoveCst	2.11	2.98	1.81	2.67
MapStd + FixUnknown + MapMinMax + PCA + RemoveCst	2.07	2.68	2.00	2.39

Table 10 - Comparison between preprocessing

We come to the most interesting part where we see how the learning algorithm affects, due to the increase in time and the need for at least 10 iterations in order to have reference values we only compare the algorithm used so far Levenberg-Marquardt with Bayesian regulation, which does not require validation set.

Again, we come to the point of intersection of bias variance, as in all the previous examples, the chosen point is slightly displaced towards better value of the total test for the main reason of being a global data set considerably larger than the test and therefore we can deduce with greater certainty that they will have a greater economic impact. Most likely, we can define our results, i.e. the uncertainty range.

First, let us remember that we use the Levenberg-Marquardt algorithm for solving nonlinear least-squares. We'll complete the training once the number of iterations is reached; we remember that an iteration may take from seconds to days, depending on the data that has to be considered in each iteration to adjust the values of the weights. Also, we will stop once the goal is reached, or the gradient to update the values are very small. Modifying these parameters, along with others such as relationship of memory, processing, use of GPU, parallelism and states in memory retrieval... We get a clear improvement in the results as expected, although the significant increase in training time causes that it is not practical for cases in which a full re-training is required.

	Settings	Average MAPE		Minimum MAPE	
		Total	Test	Total	Test
Levenberg-Marquardt	Common	2.17	2.30	2.08	2.21
	Advance	1.93	2.11	1.89	1.87
	Advance+	1.85	2.00	1.78	1.92
Bayesian regulation	Common	1.69	2.37	1.65	2.15
	Advance	1.56	2.80	1.46	2.34

Table 11 - Comparison between learning requirements

These results are significantly limited by the training time we deem practical, the relationship between memory and processing capacity has been chosen in the limit of the technical characteristics of the equipment used, both on CPU, GPU, RAM and virtual memory. Numerous examples indicate that greatly increasing processing time only achieved a decimals improvement in the outcomes, for example, if we get a MAPE of 2 with a time of 5 minute training and just 1 GB of memory RAM requirements. Increasing the time to four hours and the memory to almost 40GB we can obtain values close to 1.5 MAPE. While these differences will be smaller as we go optimizing the model. This has its positive side since in theory, we only train the network only once and thereafter we can simply update the values as new data are obtained, however, since generally speaking, it is necessary to perform several iterations to

obtain a good model with the chosen settings, it can take several days only to train a model in which small changes lead to changes in outcomes. As we have seen this limitation can be reduced considerably if there are multiple computers, renting power and memory online or by dividing the problem, due to its capacity of parallelism.

#### 7.2.4.6 Simple supervised clustering

Let's see how results are affected by an approach that first, via unsupervised learning, decides the type of day - not necessarily exclusively to the number of days - and then applies a model to predict with greater certainty every day type, based on the cumulative behavior of a set of days as if they were clusters of behavior. The following table shows a simplified way, what we can expect from this pre-treatment, we must add, to significantly reduce the amount of data, we can extend the requirements to terminate the training or the number of feedback, thus if the results follow the theoretical behavior it should reduce the minimum MAPE at least a couple of tenths.

Another important factor is the case where the network learns the dataset, but does not understand it, which will obtain some very low, even close to 0, MAPEs however, it won't know how to predict outside the field of behavior or circumstances that it has learned. For this reason we always add a column with the MAPE of the testing month. In the table below, we see how in the case of holidays - been reduced in comparison with the capacity of our network and algorithms - surely has happened. Then we will see a better example of this possibility, but it will also help us to see how it affects in different circumstances the use of other algorithms.

Division	Average MAPE		Minimum MAPE	
	Total	Test	Total	Test
Workweek	1.47	3.58	1.33	2.56
Weekend	2.38	7.90	2.15	5.91
Holidays	3.09	40.88	2.23	18.50
Monday	1.92	11.27	1.52	9.16
Tuesday	1.19	10.13	0.95	6.87
Wednesday	1.03	8.31	0.88	5.55
Thursday	1.08	8.19	0.92	5.54
Friday	0.94	5.00	0.81	2.37
Saturday	1.29	10.57	1.11	7.76
Sunday	2.27	13.65	1.89	7.65
19:24 Hours	2.24	10.57	1.50	6.37
1:18 Hours	1.81	9.50	1.39	7.09

Table 12 - Comparison between scenarios

The first defense that will be made is the use of post-preprocessed of data with an unsupervised clustering, to subsequently make the prediction with a supervised learning.

*This approach could be seen as a network of considerable size, taught by a competitive methodology where a not-small number of neurons are modified each time that a neuron of the subgroup is the one that best defines a particular pattern or scenario as a dynamic series of memories; however, there is a big difference in the interconnections with neurons outside the specialized subgroup. The idea is to make the mapping of networks in an unsupervised way (force and relations between neurons) while the calculation of weights is done in an independent quasi supervised way. So we get a network with many non-totally independent small networks which, in turn, have several centroids, which, along with their brothers (strongly related neurons within their own group), redirected to children (linked neurons with their parents, or top layer) of the previous sub integrated network. Annex 5 - Hybrid model concept proposal*

In this way multiple benefits are achieved, without a priori higher contraindication as long as the quantity and quality of data is enough:

1. Significantly improves the forecast for the scenarios previously presented as well as its generalization capability.
2. Considerably reduces the time of training and adaptation of weights according to the availability of new observations.
3. Lets you enhance and analyze separately behaviors at a glance which are unappreciable.
4. Facilitates decision support oriented processing to generate ranges of probabilities for their results. (According to the number of times that a subgroup has been activated)
5. Simplify part of the optimization process. To be able to identify clearly dimensional groups of neurons that do not add value to the model, you can identify the patterns of memories that can be improved or eliminated, similar to the process of optimization in a microprocessor.

To demonstrate, a clustering of the data set used in the project and an analysis of the weights is shown, although there's still a broad field of study and analysis in both respects. Take note that to achieve an efficient understanding between unsupervised analyzer section (Brendan van Rooyen, 2015), dynamic memory, and the decision-maker guided by supervised learning. The network generated by a modified version of the radial basis network should be easily understandable and changeable by the modified deep competitive learning network.

In the first picture you can see how the machine splits the data according to its proximity by means of the modified algorithm for k-means explained in the chapter on Data

mining. We can then automatically apply a simple decision model as seen in the following point, to try to separate the stages or patterns of behavior according to our limitations.

If we separate the twenty patterns data set, we have sufficient capacity to differentiate each behavior, while not increasing the size of the model in excess. In our case the cluster has been performed in a crude way, but a pre-processing more oriented to this analysis will improve the results without a doubt. Due to the significant differences between each training on the same network, the scarce field of improvement due to the exposed limitations makes it unnecessary to separate such pre-processing. Therefore, we will use a basic preprocessing for unsupervised learning and we'll add to another more differentiated once the supervised learning is performed.

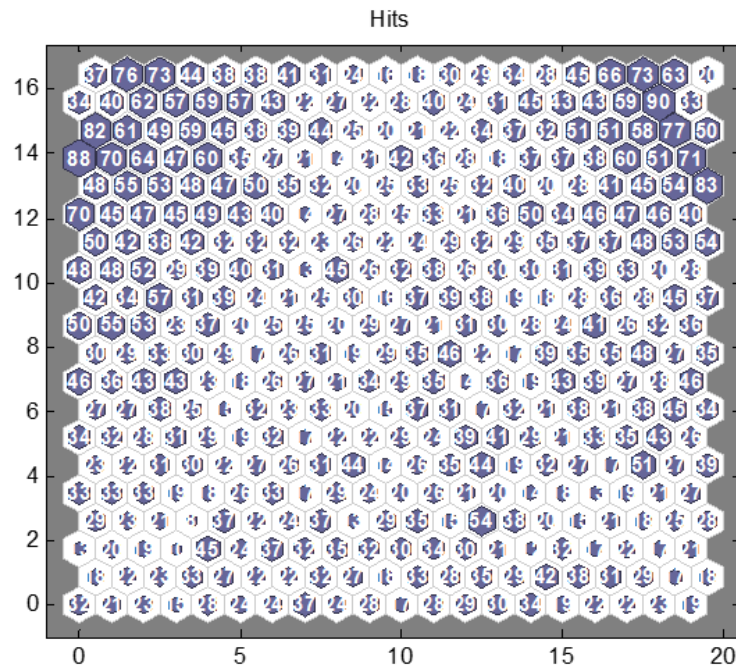


Figure 7.69 - Number of times each pattern is activated in a 400 division

Following the approach of the document, in order to optimize the configuration of all aspects that come into play in our model we must analyze how each parameter affects it and what importance it has.

As expected and because of its purpose, the parameter called SpecialFlag adds additional value in certain cases in a similar way that it happens with festive days. With maxTemp and minTemp we can see that it is a range of values that really affect the scenarios, it is conceivable that while we are in mean values of temperature, this factor does not affect the stage, but as we get closer to extreme it gains greater importance. Unfortunately, our values are average daily temperature values, but are littering our model partly because hourly values would be the minimum desirable or weighted average historical consumption would also be desirable.



Different conclusions can be obtained in a similar manner from the importance of the rest of the weights in the face of improving both the network and the data set provided. Progressive weights and values of the interconnections to the hours, weekends and weekday that link to the load of the previous day or the previous week, months (seasons). In the examples of this point there has been a 20 by 20 matrices, but higher or lower divisions may facilitate understanding of data and reduce the size of the full model.

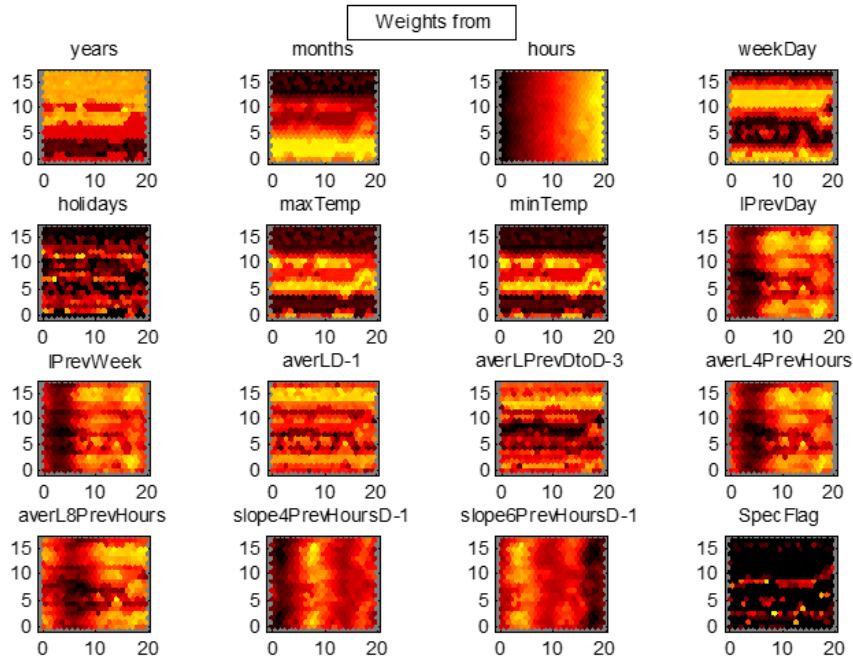


Figure 7.70 - Neuron's weights for each variable

In the image below you can see our first set of explanatory data represented with a topographic hexagon where the red lines represent the connections between neurons and the color of each side refers to the distances between them, being higher in the darkest areas. In the example shown, it is difficult to find relations, and if it should be separated more it would get very similar to those of the previous image. Dark areas are very loose links so we would use them to differentiate between behavior patterns; however, there is no doubt that a dynamic process with their respective algorithm is required. In the picture on the right, we have the same case, but only for vector objective in order to perform reverse engineering in certain cases. Due to the extension of the document we won't enter into a deep analysis of the results of this example, but we can in the second case where on top there is a zone of significant or total division, this analyzed in more detail could provide us with some degree of certainty the possibility of separating these scenarios, not depending on the explanatory values but rather on the values that we hope to find, giving us a certain range of probability for a given scenario.

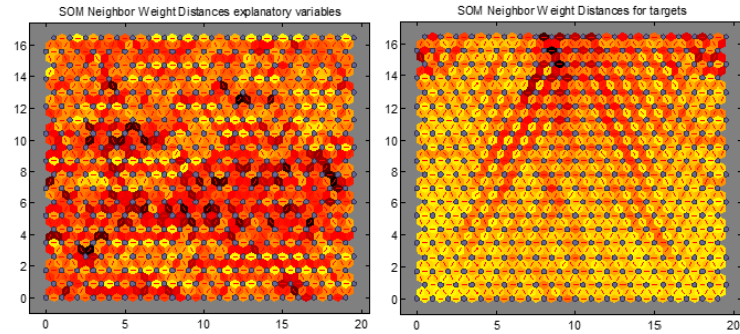


Figure 7.71 - Distance and relationship between neighbors

#### 7.2.4.8 Radial Basis and k-means Classification

Separately or after our unsupervised clustering, we must make the decision of how to split the dataset to be able to later analyze it in a better way.

The operation and significance of this type of network, mainly its transformation function has been previously explained. Now let's see their performance as well as some examples of the results that should be obtained. First, the hypothetical centers are determined, and then the beta ratio is calculated, which is the radius of space covered by each radial network neuron. These, together with the weights are re-calculated on each iteration after presenting the entire data set with the gradient descent algorithm also explained previously.

The following image shows the centers and the spaces they cover; our initial approach would be to conduct a supervised learning training with a modified algorithm an optimized configuration of a neural network for each behavior or scenario. Looking at it another way, for each site that covers enough information.

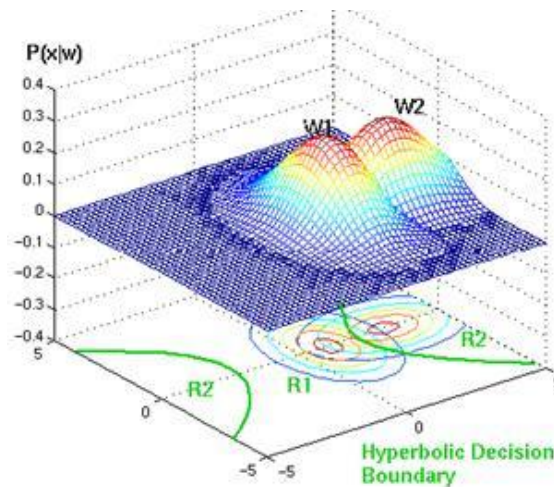


Figure 7.72 - Example of decision boundary

Under certain limitations, it is not practical to have multiple modifications of a general model. Therefore, we can apply decision boundary to help us, according to our requirements; to reduce the number of models that once trained will only have to be updated. In the next picture we have the described example applied to the above scenario, dividing all the subspaces of behavior in two patterns.

Finally, we mention the main difference when applying RBF and KNN. In the first case we have multiple areas of space where neurons have a considerable weight though with k-means we will have a well-defined space with a single category assigned by a point.

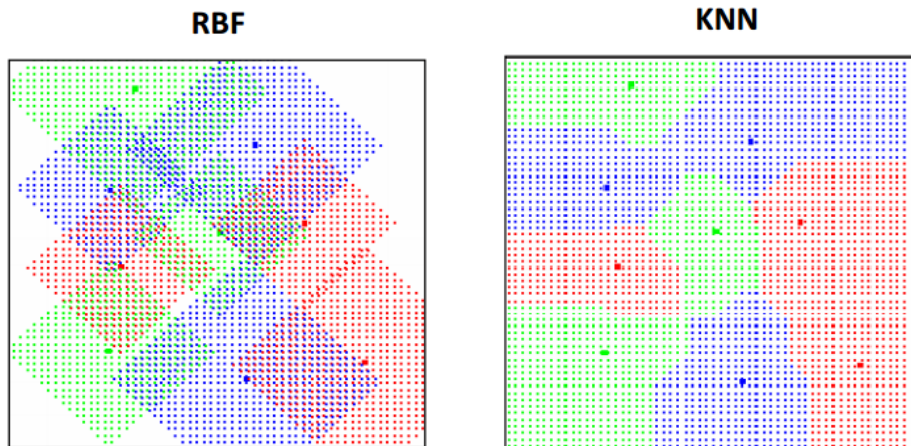


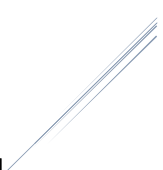
Figure 7.73 - Important difference between RBNN and k-mean NN

#### 7.2.4.9 Shallow vs Deep Architecture and customized connections

Due to the limitations previously exposed we won't study other types of networks or search its optimal configuration in depth. However to defend the use of the chosen network, we'll show a few examples where while there is no significant difference it can be concluded that at least the referenced NARX network does not behave worse than other types.

In the table we have the used NARX network as reference as well as its performance, which reflects the minimum mse - mean square error - retrieved. In both cases the smaller the number the better our model will be, more specifically more accurately will have. The data that are loaded to the three networks has been pre-processed, since in the third case they are loaded directly in intermediate layers and a gross load will possibly erase any virtue in our network.

The second exposed network is a neural network with more than one hidden layer, it should be understood that this is the future on the part of the artificial intelligence, mostly if we



focus on deep learning, or reaching machine super intelligence. The advantages and disadvantages of using networks with more than one hidden layer do not enter within the extension of this document. Very briefly, we got a considerable increase in the capacity of understanding the problem even though the training time increases considerably when raising the number of interconnections between the neurons that the model has. Thus, in our case, would reduce the cost variance at the expense of increasing the bias; in our case we are looking for a point where both are as small as possible. In our case the time factor and other constraints prevent us from deeply analyzing this type of network, although an advanced setting would ensure better results.

There is considerable documentation on Shallow vs Deep Architecture (Depth-Breadth Tradeoff) which analyzes the consequences of the complexity of the circuit which forms the network, considerably increased in the case of convolutional layers (Anon., 2015). In the case of a deep architecture, we are giving ways of nonlinear modules, all those composed of trainable parameters. In this way we can make compact models since we are changing depth for breadth. A good summary of some complications associated with having multiple convolutions layers and a summary of as GEMM and Basic Linear Algebra Subprograms is based on what can be found in the following reference. (Warden, 2015)

Following this reasoning, an important factor is to decide the size of the hidden layers, many studies have been performed (Anon., 2014) although none has come to get any concrete value. In our case it has been decided based on the advice of the paper studied using 50 and 20, to see how it affects the result in the provision of the same number of neurons. With the same reasoning, the loaded parameters and functions have been chosen so that they are as similar as possible.

The algorithm between layers is another decisive factor, sometimes it is recommended to use a hidden layer with a particular function because the values in this layer have a mathematical meaning or can be analyzed separately.

In terms of outcome, as we reflected, theoretically the same results can be achieved either with a single or a finite number of them. Standard general regression with non-elevated complexity problems is not recommended, is not necessary, to increase the number of layers due to the increase in complexity of the model and the small increase in the results when losing bias.

In recent years, the efficiency in training networks has been improved with more than one couple of hidden layers. However, most of the studies have shown empirically that although they can represent complex problems, they don't get significantly better results than a model with one or two layers with the sufficient number of neurons to represent the problem. This is consistent with the fact that most efficient training algorithms in most of the studies are used to determine the number of layers, the number of neurons in each of them as well as the

interconnections between them and how they affect different functions in each layer. (Yann LeCun, 2015) (David Eigen, 2013) (Yoshua Bengio, 2007)

The third example is a network with two hidden layers where NARX network's input values are used as parameters to update the values of the other layers. It is a simple example for our purpose, to demonstrate the almost total freedom that we have to configure our neural network according to the requirements of our problem.

Ideally, we could have a NARX entry in the first layer, and later various configurations and feedback to finally close network, linking the output with any layer in such a way that we can considerably improve the behavior of the network. Both, in bias and variance, since, although we trained in open mode, without that last commented recurrence due to the increase in the required time, achieving that the weights fit according to the results obtained, reducing the bias-variance. Our results indicate that we can obtain MAPEs below the unit for the training set and very close to one for relatively seen scenarios, i.e. with a reasoned understanding, the problem can be predicted. (Graves, 2012)

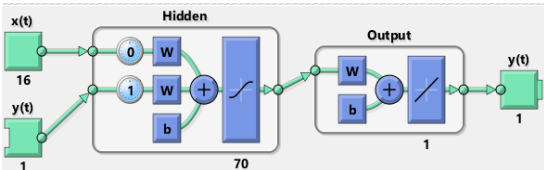
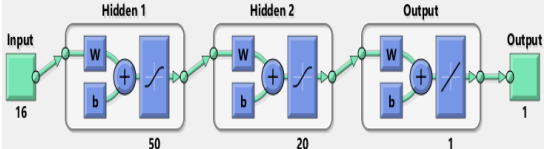
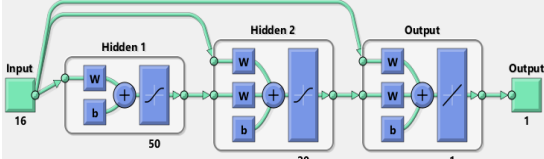
Network	MAPE	Performance
	2.23	1.1882e-04
	2.29	1.2153e-04
	2.27	1.3095e-04

Table 13 – Comparison between architectures

#### 7.2.4.10 More configured learning algorithms

Due to the large number of algorithms that exist in training, the modifications that each can have and its main scope of use, at this point we will only will represent the algorithm that has been finally chosen for the training: Bayesian regularization backpropagation. The following table is attached in order to compare the results with some of the other most used algorithms today.

The operation method of our algorithm will be to minimize the linear combination of error and weights, in our case, and for the reasons stated previously, we will use mean square error in measuring the error, although after that we use mean average percentage error to check their goodness. (Sohl-Dickstein, 2014)

In order not to lose generalization ability, as it goes understanding the model in each iteration the linear combination will be modified so that it does not increase in a high way. If you want to understand in greater depth the functioning of the algorithm you can read about the Bayesian interpretation of regularization for kernel methods. Whether the target value is a scalar or several of them, as well as the perspective of the regularization and Bayesian perspective of the process significantly exceeds the level of this document.

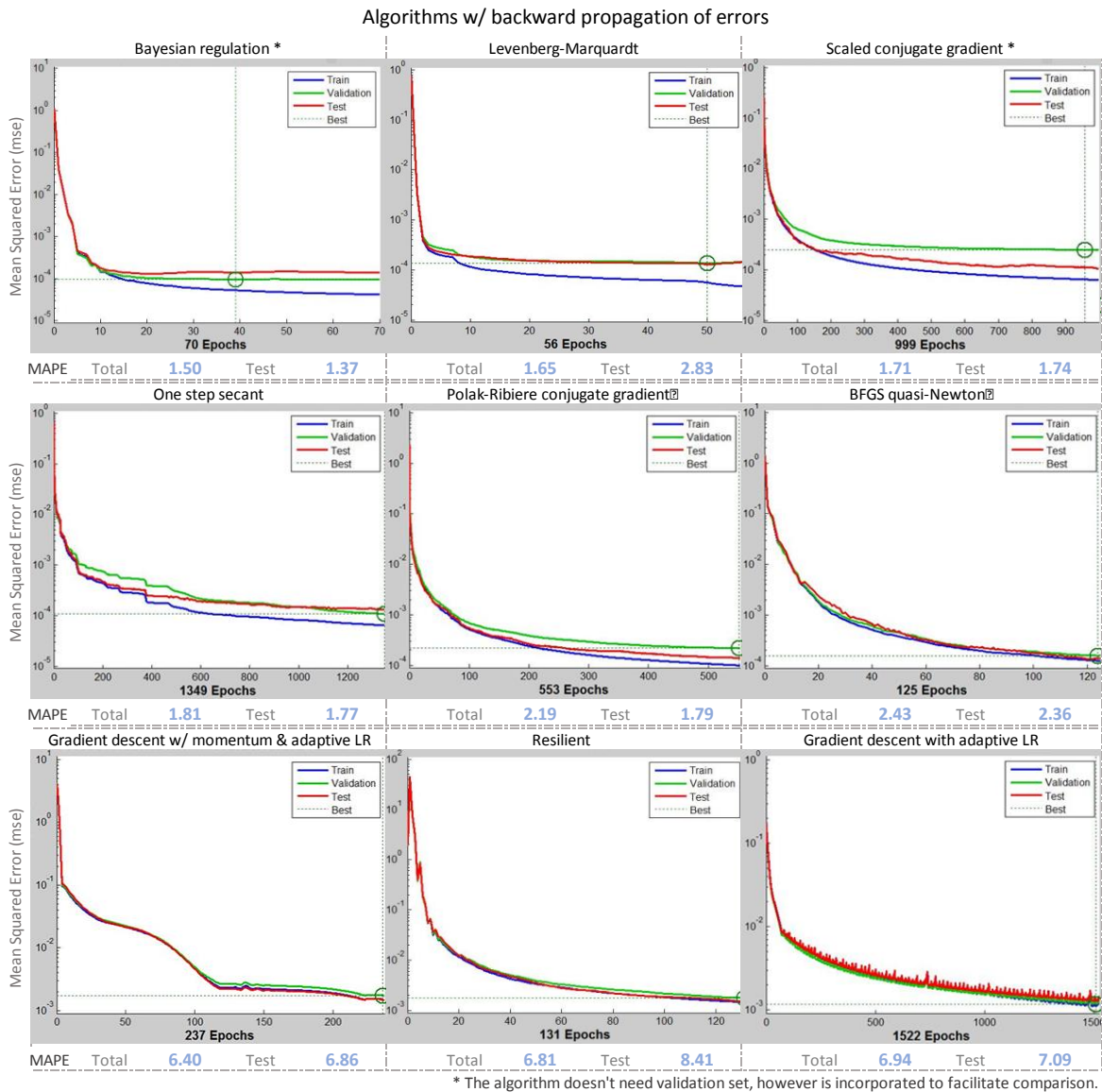


Table 14 - Comparison between learning algorithms



With the bases previously exposed Levenberg-Marquardt algorithm is used (Anon., 2015), then the Jacobian of the objective function is calculated with respect to the weights and bias  $jX$  and the parameters are updated by backpropagation. Where  $E$  is the matrix of the errors,  $I$  is the matrix that identifies the values to update,  $mu$  is a value that fits in each iteration as the improvement or worsening of the objective function at each stage and  $dX$  is the increase made in the parameters. "Automatically chooses the weighting ratio that multiplies the sum of squared weights that is added to the sum of squared errors to form the objective function. The choice depends on the effective number of weights" (Hagan, 1997)

$$jj = jX \cdot jX \quad (7.3)$$

$$je = jX \cdot E \quad (7.4)$$

$$dX = -(jj + I \cdot mu) \setminus je \quad (7.5)$$

#### 7.2.4.11 Integration in decision support

The following example shows the application of the same model in predicting the needs of auxiliary services by the transport net operator. In order to vindicate the benefits of the models based on this methodology no change will be made within the previously load oriented forecasting model. However and because the problem goes from being continuous in time and value to being quasi discrete in both respects, a direct and simple postprocessing of the values returned will be required. If this model is integrated into this post-processing is not required as it will define the range of confidence with which to predict certain value and therefore the decision whether in the short or medium term will have some reference.

There are a lot of different algorithms and programs focused on such problems, however, with this example we demonstrate the virtues of our approach. While the fitting problem is easily overcome by increasing the number of weights and increasing the time, the important point is to optimize the capacity of forecasting. As we will see in the results, we again have the dilemma of bias variance tradeoff.

Regarding the data loaded, Annex 3, has been for the requirements of the problem, values obtained at least two hours in advance. The major issue in this case would be the consideration or not of the latest technological mix with available programs or simply base it on aspects that can be previewed with this or other models, understanding load forecasting, pricing, renewable supplies... A change in this decision would predict the needs of the system operator with more than 24 hours horizon and as we will see, it doesn't significantly affect the results, so it will be the approach with the greatest benefit for forecasting.

In order to see how it affects the inclusion of this important element, a comparison will be made, using as a reference a value we'll name MAP2, calculated with the same formula that

the ASM/10. The name change will be so as not to cause confusion, since the results cannot be compared to the rest of the document are considered to understand the level reach by the results. These values are only useful for comparisons between themselves. The following results, they make sense and are in line with expectations. The third column indicates the percentage of hours, which returns with a certain probability of having or not a reservation and is not the right thing in the use of prediction. You should bear in mind that this value includes many hours that with a greater postprocessing would be cleaned, the reason why the values or error expected are around 5%.

<i>MAP2</i>	Fitting	Forecasting	Periods
<2 hours	3.14	3.66	8.1%
>24 hours	2.51	4.76	10.2%

Finally, before moving on to the results, it should be noted that in this example, seeing the results obtained in 7.2.4.6 and the proposals made in 7.2.4.7 and 7.2.4.8 you'd expect significantly better results, like by adjusting the number of neurons, parameters, network architecture and optimization of the global model. In the first picture the prediction vs the real needs in the first tier forecasted are shown. For its part on the volume of requirements, the average negative error is considerably higher than the positive, varying substantially between months, so there is room for improvement. By the hour, the average error except in H18 is less than 500MW so is good enough to be able to take decisions with considerable certainty. If we believe that when the requirements are for groups of hours, the average error of [H18,H20:H23] is lower than this reference value. Therefore our decisions are going to be in the vast majority of the hypothesis (a minimum of three sessions in the case of being required) within a range of confidence lower to 100MW of error, represented in the graph under the title of hypothesis.

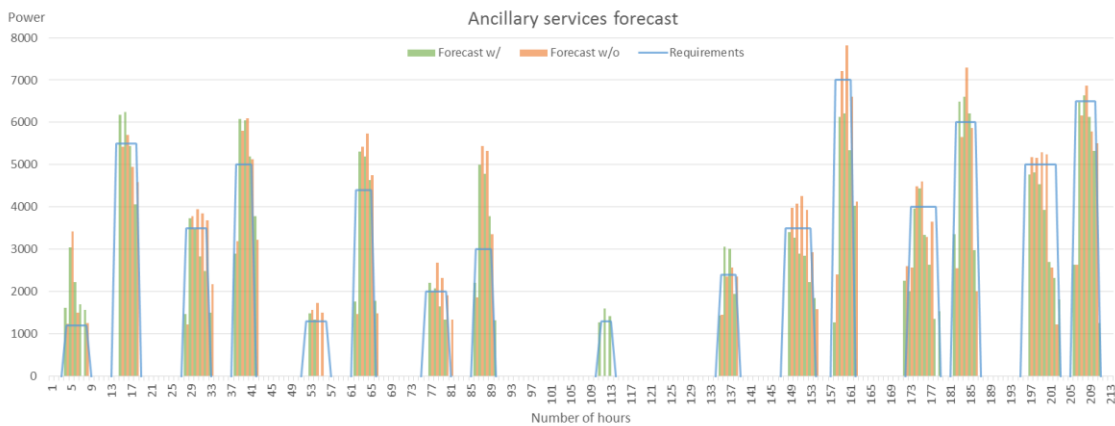


Figure 7.74 - Ancillary services requirements forecast



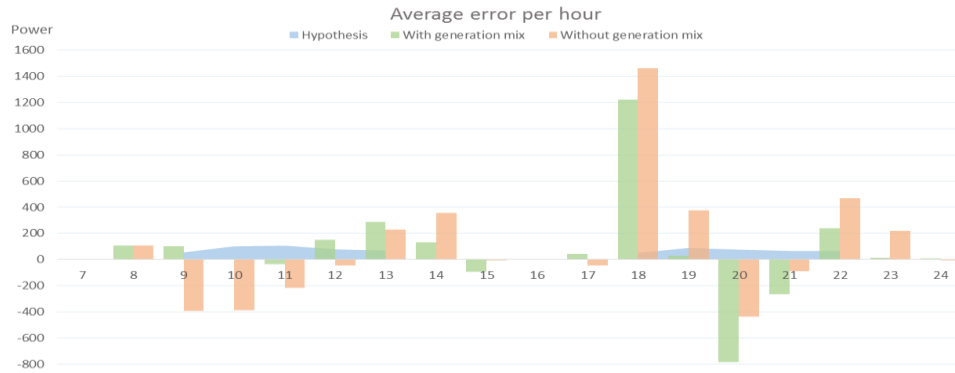


Figure 7.75 - Ancillary services requirements average error

#### 7.2.4.12 Results

Before understanding the results we have to take into account the dilemma between adjusting the time series and forecasting capacity. Similarly, it has been mentioned previously, some techniques that improve the short-term forecast, for example, giving more values to the changes made to the weights for the latest observations. In our case, the extent of the adjustment has been made over a period of nearly two years and the prediction period refers to an entire month, so the 30th of the month the ability of prediction is considerably worse than in the first days outside of the time series provided. This factor significantly affects the results, but in exchange we get a more realistic average value to the actual capacity of the model, both in prediction in the case of load curves as in the case of ancillary services.

The limitations as will be further discussed in the next chapter, significantly affect the outcome and process. The noise obtained can be understood as white noise, but arguably could be reduced if we improve the capabilities of the model along with the input data.

There are other features of the learning process of the evolution of the parameters in the algorithm that help indicate the state and aspirations that can be expected in the process. The evolution of these values has been taken into account both in the case of early stopping as to consider the learning as finished, see the summary in annex 4.

In certain cases, where the behavior is significantly different, and every time we have enough observations, it has been proved that specialize networks or groups of neurons for different scenarios can significantly reduce the error. However, high errors in the prediction aspect are obtained with much greater ease. As an example of a Saturday, we may have a MAPE of average prediction of 1.11 while for the Sunday we get values close or even greater than 5.

The values shown below are as we reflect on our methodology the ones that obtain the lowest average values between the adjustment and forecasting. If we focus on each feature separately, we can easily reduce the value of adjustments with values lower than the unit, reaching 0.39 without it being our main objective, but in terms of the average prediction it has not been reduced below a MAPE of 1.0687.

<i>MAPE</i>	Fitting Forecasting	
TSO day ahead forecast	0.892	1.145
Weekdays model average	1.060	0.890
Actual demand	2.990	2.864

## 8. Conclusions

### 8.1 Achievements and limitations

In a brief way, we have been able to achieve in a manner superior to the requirements the objectives initially raised. In addition, we have proposed and demonstrated variations that allows the model to grow and develop in various problems and applications. All cases exposed in the document have been substantially limited by some factors that are discussed below.

1. Time. The time to be able to study and apply other algorithms and approaches, in order to configure and optimize the network model chosen in greater detail, in order to be able to devote more time to process and better refine the weights of the network and be able to optimize the input data.
2. Input data. Information to predict with, both in quality and in quantity. This has arguably been the most important factor constraining the respective results. In our case, we have the hourly value of demand and only a weighted value of the maximum and minimum temperature in Spain in the day. In addition, using observations from two years ago, quite limited to the time of specialize groups of neurons in different scenarios, and making the capacity of prediction more difficult because in the greater number of occasions we are predicting outside of history.
3. Processing capacity and memory. In certain algorithms this factor not only increases inefficiently the time required, but it also restricts the use of algorithms and architectures desired. We don't get the same values in each iteration if we only take into account the values for that hour, that if in each iteration we use the set of values for the entire day; we are increasing the efficiency of the algorithm at each stage.

Focusing on the achievements, we consider the initial objectives reached by far. On the one hand, we have learned that this field of science has spent years growing, and today it seems very difficult to know all aspects that affect even a single approach. We have studied more than a dozen algorithms and tried to use methodologies with relative ease of implementation and great results, serving for a first layer on machine learning and data analysis.

Focusing on the results obtained, we have achieved amazing results even with reduced learning time. We have managed to adjust time series close to the 20,000 elements with hardly a dozen neurons with a MAPE around the unit. However, the greatest achievement has been to optimize the model to face the prediction of different problems. In the case of the load curve published by REE on the previous day, we have obtained a MAPE slightly higher than the unit, under very significant limitations. On the other hand, we have been able to reduce this value in five of the seven days of the week, and everything points out that with greater amount of data it could be possible to reduce the error at specific periods in a manner that is appreciable.

With regard to the anticipation of the actual demand, it is necessary to include other factors that cannot be predicted statistically today. If we understand a MAPE less than 2.3 as a result with an exceptional amount of time, data, models and professional equipment. We have gotten closer in a great way with an insignificant investment in all aspects.

The conclusions obtained continue in the following two points, especially in matters relating to machine learning in point 8.2.3 Business Intelligence and Learning Intelligent Optimization (LION).

## 8.2 Further developments and research

### 8.2.1 Energy Industry Applications

Possibly the most interesting aspect of these models in the short term is the wide range where they can be applied: forecasting, market analysis, trading, decision support, risk management and scenarios, some optimization problems, competence behaviors, supervision and regulation are some of the aspects where you have data and a supervised or unsupervised analysis can help us to understand the underlying problem and make better decisions. Specialized machines can follow self-learning, behaving by their own account and enriched by their or our decision-making for later use in our strategies.

In our case, we have demonstrated that these methodologies have a good behavior in predicting, but its main use today is in the classification. For this reason, in certain circumstances where the problem and objectives can be particularly complicated, we can base on the mixed model submitted to help us both in analysis and decision-making. However, to put into practice a model with these features it generally requires a large amount of time, which moves at a faster rate than the results once certain values are obtained.

The mixed use of algorithms, with modifications and approaches in some of its features gives an immeasurable number of different models that can be studied and also designed for certain problems. Here lies part of its progress, the improvement of models directed to certain requirements and with their respective limitations and tradeoffs.

### 8.2.2 Business Intelligence and Learning Intelligent Optimization (LION & RSO)

As it has been mentioned in the previous point, there is a large amount of learning-oriented algorithms. On the other hand, there are many methodologies of optimization, as a general rule, it tends to minimize some objective function. The problem in the application to various real-life problems lay in the objective function and its complexity for internalizing behaviors or real situations. If we gather the whole experience in optimization, more specifically

Reactivate Search Optimization (RSO), with machine learning, we can focus on problems considerably more complex in a more practical and automatic manner.

As illustrated in (Roberto Battiti, 2014): “Reactive Search Optimization (RSO) advocates the integration of sub-symbolic machine learning techniques into search heuristics for solving complex optimization problems” and “Intelligent optimization, a superset of Reactive Search, refers to a more extended area of research, including online and offline schemes based on the use of memory, adaptation, incremental development of models, experimental algorithmic applied to optimization, intelligent tuning and design of heuristics.” These approaches promise in a joint fashion to greatly reduce the main problems of ML in its development, implementation and optimization.

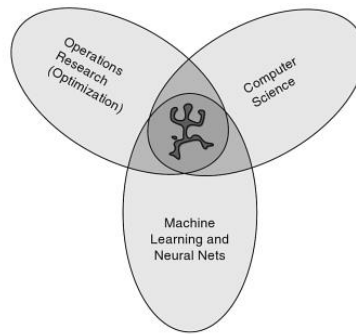


Figure 7.76 - ML and intelligent optimization as the intersection

The main problem when applying these analysis and decision support systems in professional environments is the lack of staff with experience and ability. This is in addition to the difficulties mentioned throughout the document, as a summary and according to their order as well as the process carried out in our model can be summed up as.

1. Identifying the most effective machine learning technique
2. Accurate parameter tuning
3. Optimization problem
4. Final integration

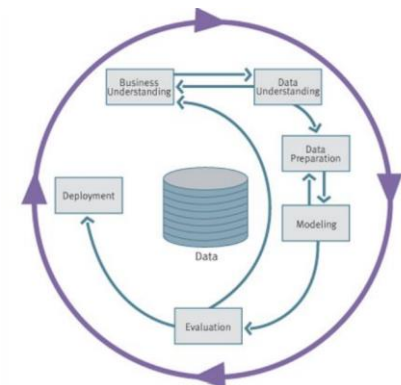


Figure 7.77 - Business integration

Another problem, which was inherited from the great complexity and difficulty of the point two, is the factor of the small but significant increase in the results with different but similar approaches. Seeing the actual example done in the elaboration of this document. We used a clustering and NARX network relatively optimized to achieve our objectives. However, we have mentioned

- More than ten techniques to optimize their behavior
- More than ten techniques to improve the results
- More than ten algorithms and approaches with which better results can be expected.
- Optimization methods, both for the selection of parameters as for the improvement of final results
- Various changes in the algorithm to adjust the behavior of the model for our particular problem.

These are the same problems faced by a professional development, there are a lot of improvements, but its implementation and enforcement are too costly in time with respect to the measurable improvements. In our case we have taken several weeks to reduce a MAPE of 2.2 to one of 1.1. How long would it take reduce it to 0.7? Or better said, how long would it take to apply all improvements mentioned and how would they influence? Surely years, if we need to optimize the hundreds of parameters and possible changes that we will get in the final model.

Finally, today there are many companies that offer highly complex models of these systems, either only ML or LION that can be applied to a huge amount of problems and surely in all of them excellent results are obtained, but it seems difficult to believe they can be optimized in detail for each problem. They are excellent models, able to generalize and adapt in a certain way to the requirements presented. It is in this together with the advantages of the processing where the main virtue in the short term in this field and at the same time one of their biggest problems lies. (Roberto Battiti, 2010)

### 8.3 Open tools and libraries

None of the next document would have been possible without all the information and tools available online to the general public. In the last few years progress in these methodologies has gained considerable popularity, and has left the academic world thanks in part to the multitude of online courses (mathematicalmonk, 2014) (Aksoy, 2013) (LIONos, 2011) (OCWmit, 2006) (Standford, 2012) and the great amount of various libraries, modules, tools and programs both open as well as various paid versions. (Haddop, 2014) (Spark, 2014)

The online community continues to grow progressively, supported by the initiatives of various companies and entrepreneurs which facilitate both the learning and the advantages of

the application of these algorithms. (Theano, 2014) (Torch7, 2014) (SciKit, 2014). There are also paid tools for many companies that under certain licenses are free, and include manuals that allow you to apply and learn with programs targeting the private sector. (Microsoft, 2014) (IBM, 2014)

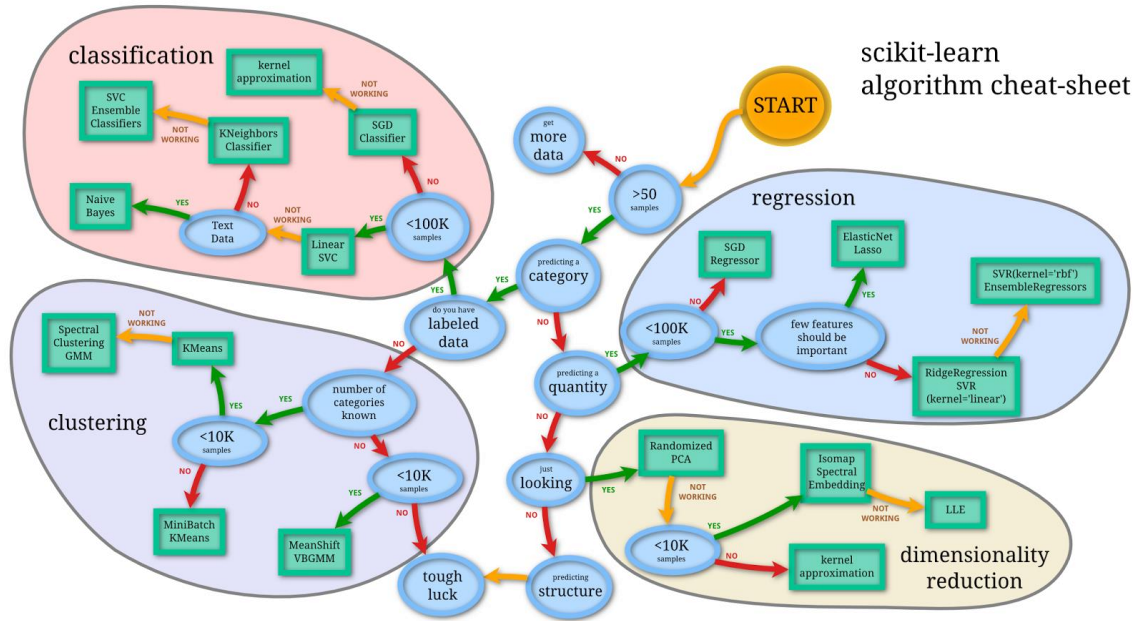
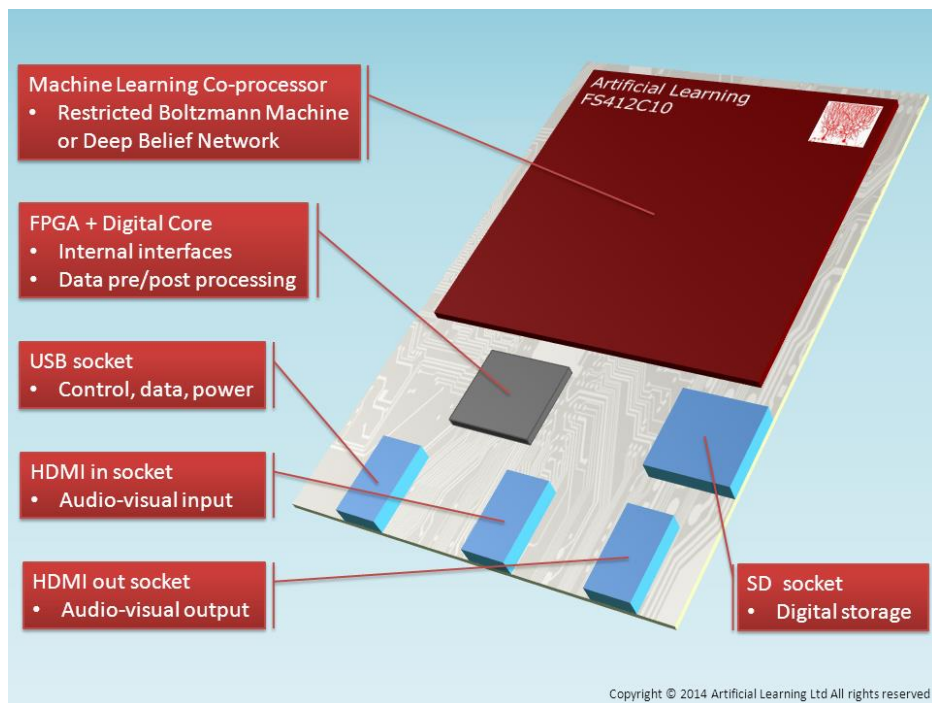
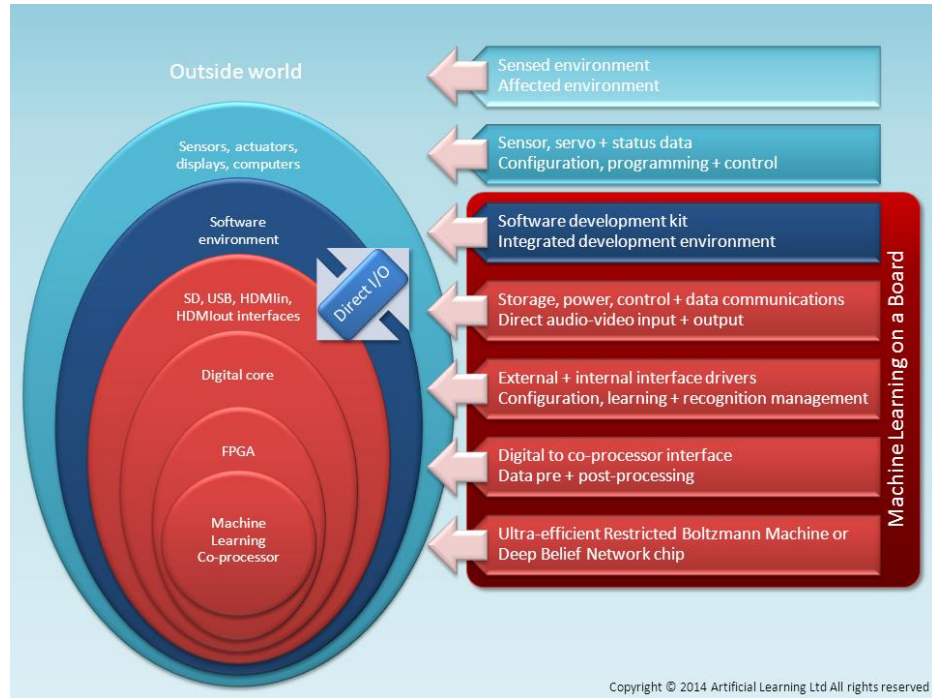


Figure 7.78 - Algorithms available in SciKit

For reasons of extension we will mention only some of the most popular online free tools. Its approach and limitations gives each one a particular market, but in most cases, because they are open and use common languages, several can be used depending on the requirements. This facilitates drastically any approach to this world and allows any user to explore and implement their advantages with the only limitation of the time, quantity, and quality of data. In the last two cases and for certain problems there are platforms and projects available which dispose and inform about data as well as the processes and reference advances.

# Annexes

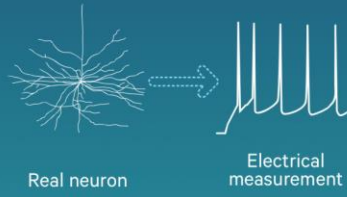
## Annex 1 - Machine learning environment and on board example





# Replicating the brain architecture

Developing complex neuron models that can be implemented in hardware

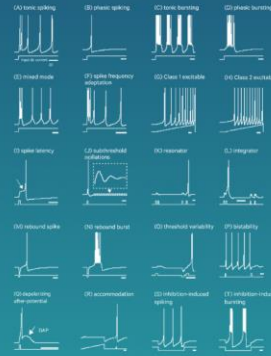


$$v' = 0.04v^2 + 5v + 140 - u + I$$

$$u' = a(bv - u)$$

if  $v = 30$  mV, then  $v \leftarrow c, u \leftarrow u + d$

**Individual neuron modeling**

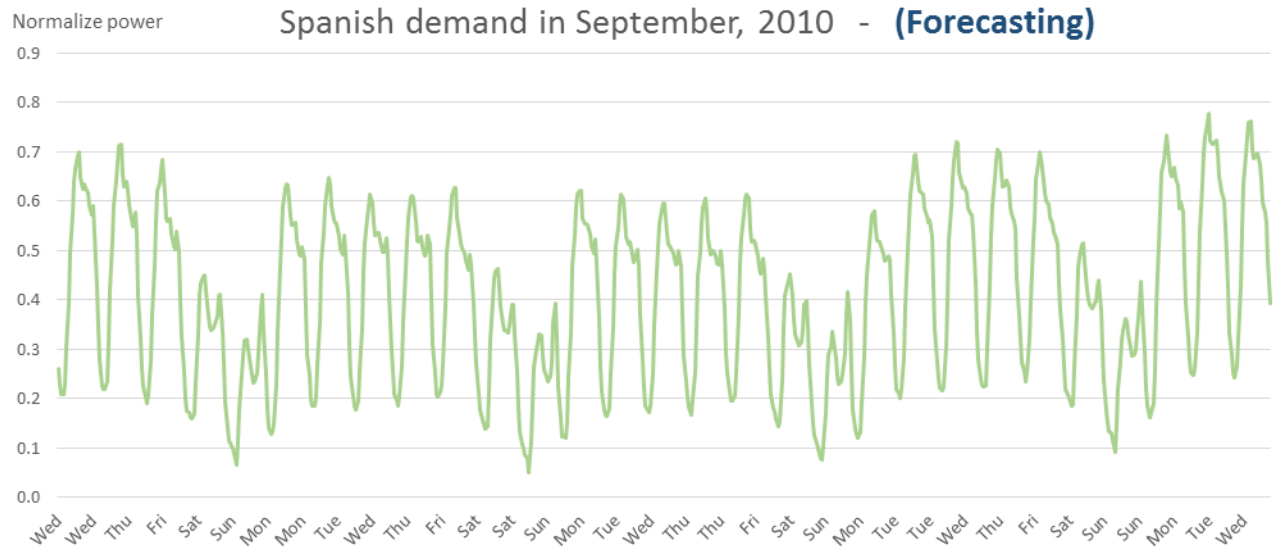
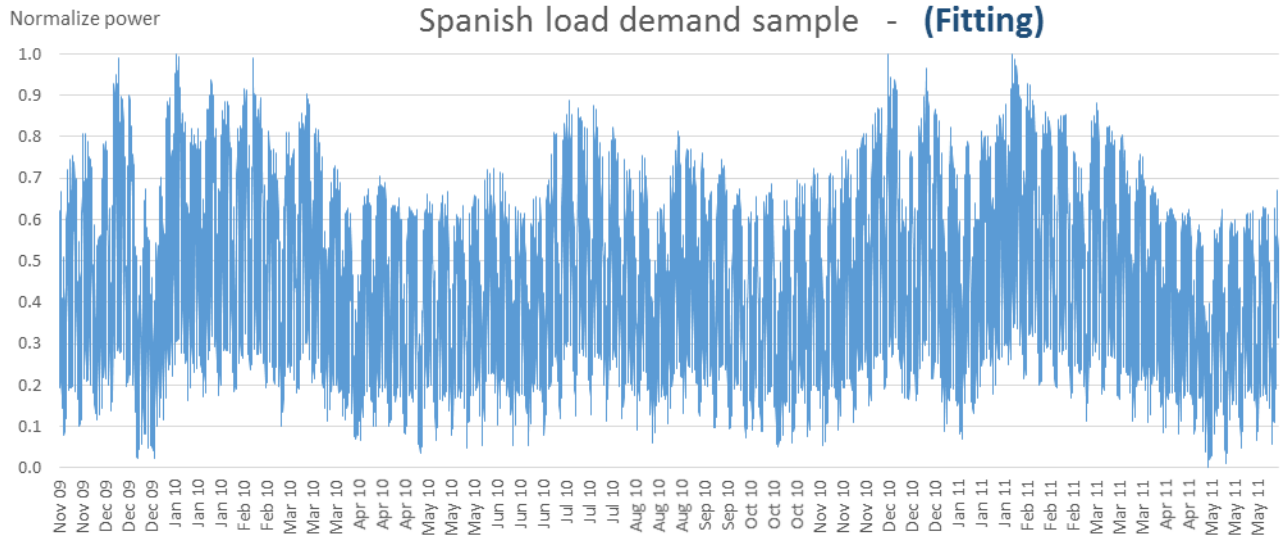


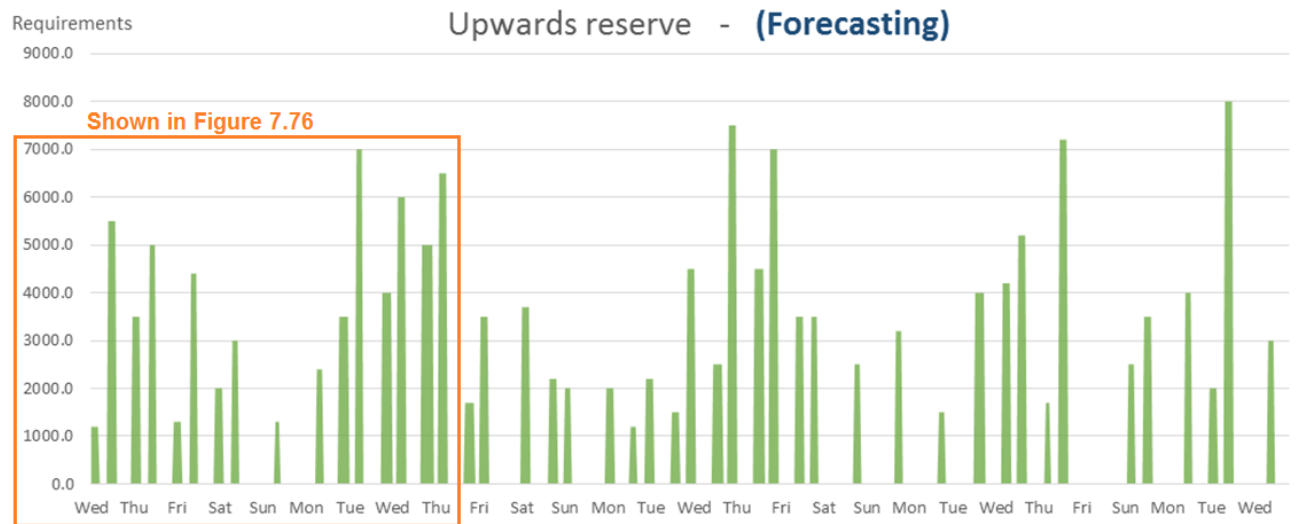
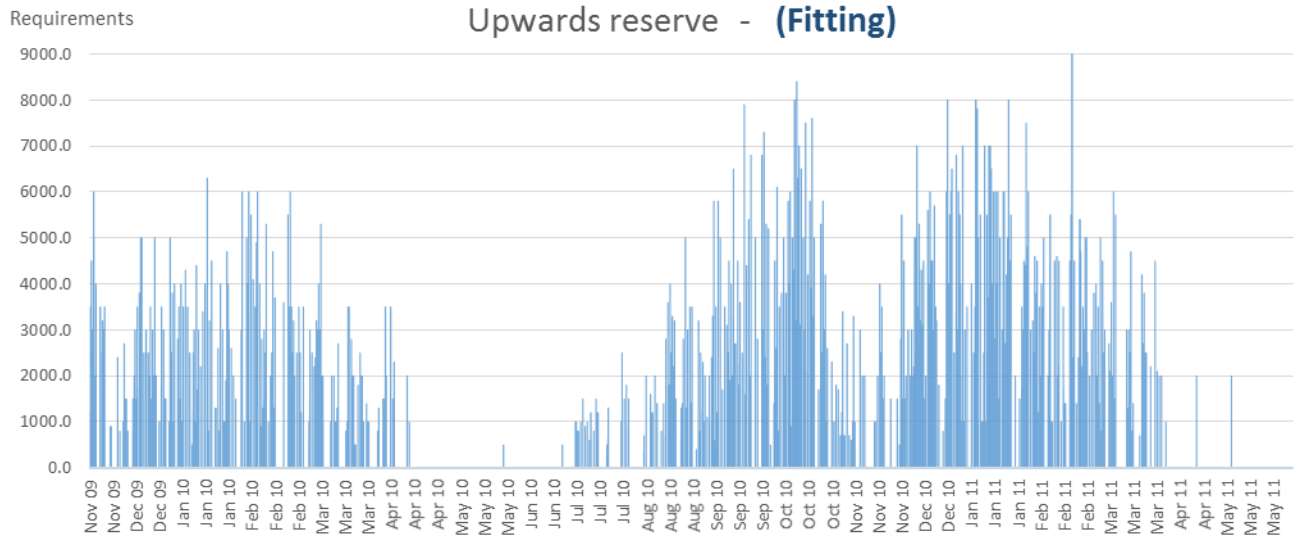
**Family of models**



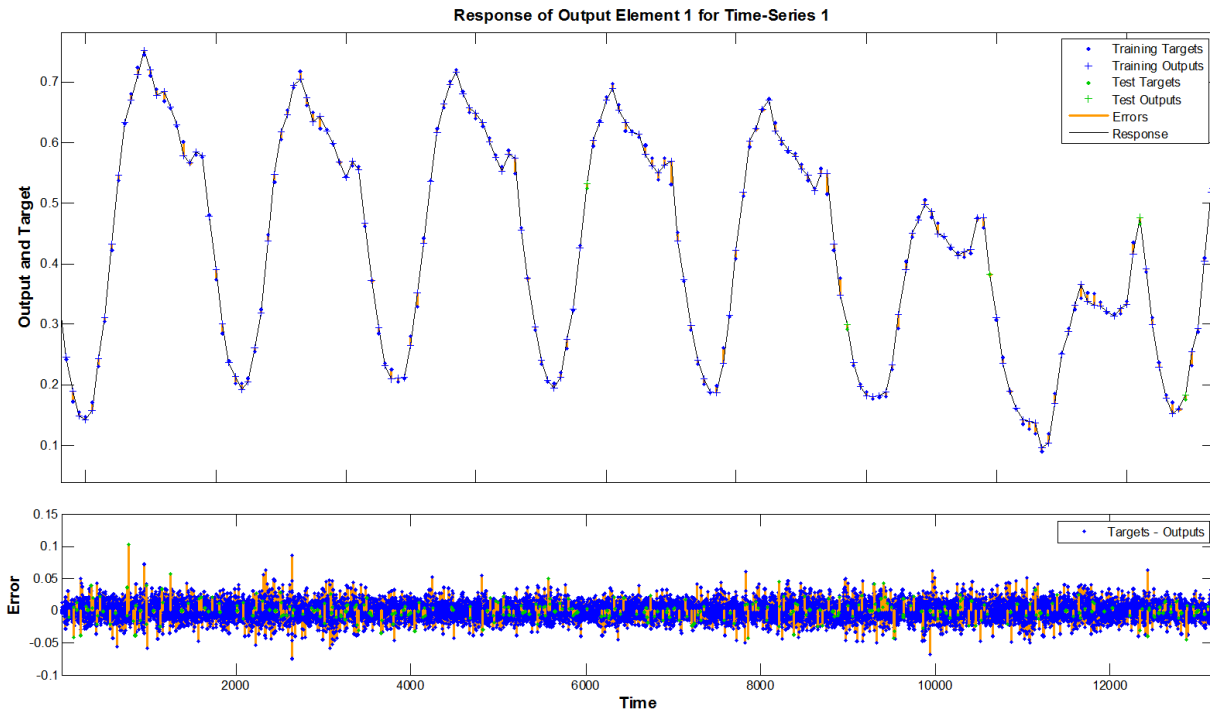
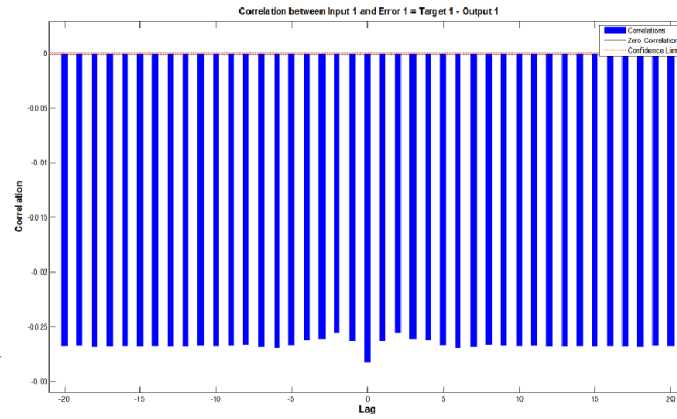
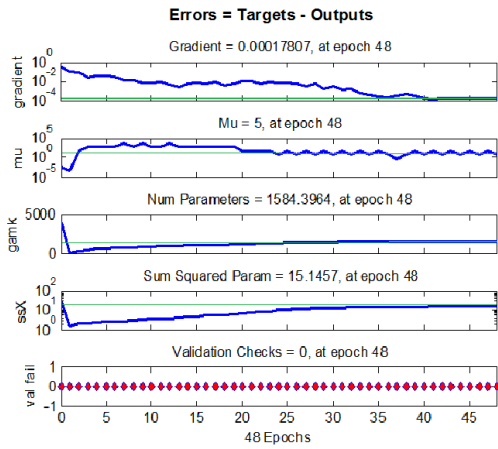
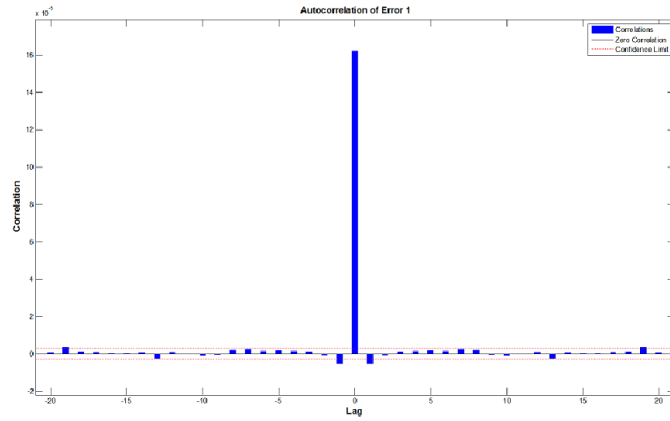
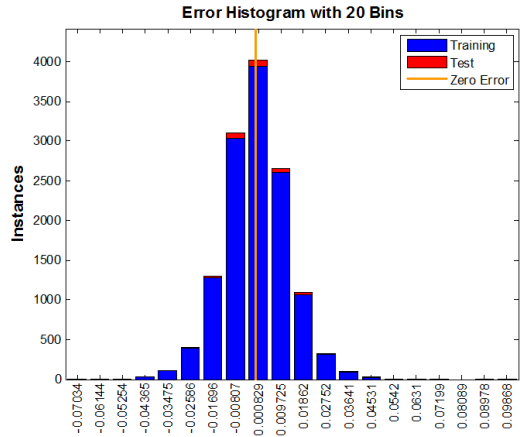
**Tools and simulation in software or FPGA\* hardware**

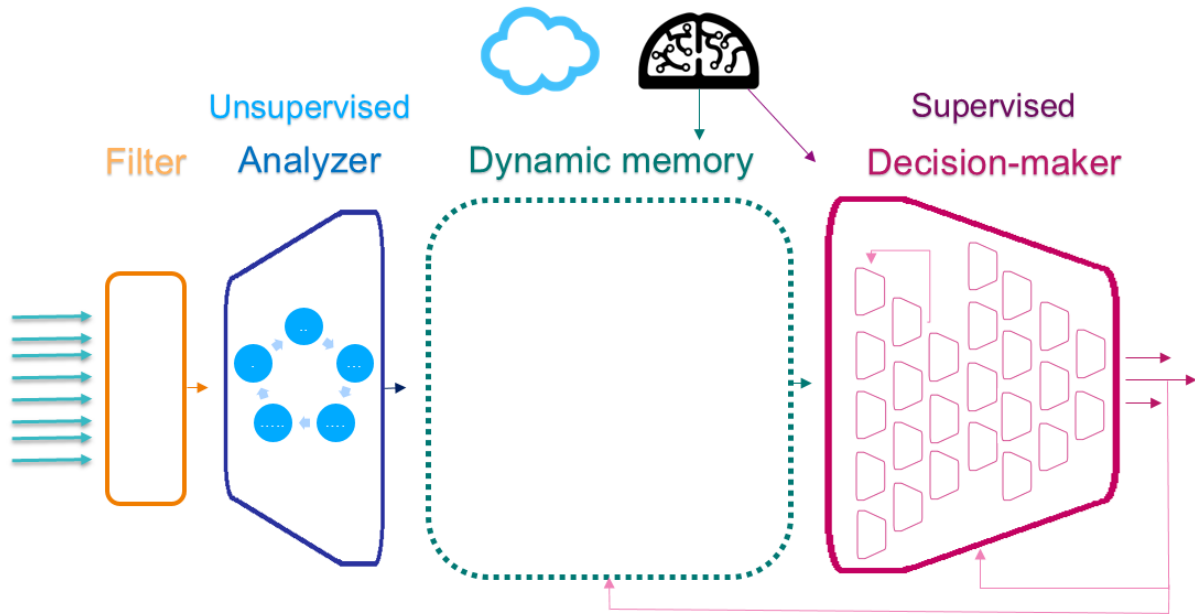
\* Field-programmable gate array





# Annex 4 - NARX learning and error summary





- Lots of information in one step of a time series creates an image of that scenario.
- We have a sequence of scenarios with their respective patterns.
- The model has to decide within a confidence interval which is the following scenario based on the **understanding** of the **SEQUENCE**,
- considering the **never seen been before environment**.

## List of Figures

Figure 2.1 - Probability density example	Figure 2.2 - Different probability densities .....	22
Figure 2.3 - Risk minimization trade-off.....		24
Figure 2.4 - Vapnik-Chervonenkis dimension .....		26
Figure 2.5 - Clustering example .....		28
Figure 2.6 - Association rule example .....		30
Figure 2.7 - Reinforcement learning example .....		31
Figure 2.8 - Inputs examples	Figure 2.9 - Graphical dependencies .....	32
Figure 4.10 - Probability equality example .....		37
Figure 4.11 - Event space example with two continuous variables.....		38
Figure 4.12 - Example of relationships between neurons .....		39
Figure 5.13 - Weighted sum of Radial Basis Function.....		47
Figure 5.14 - Process of SOM learning .....		47
Figure 5.15 - Boltzmann nodes interconnections .....		50
Figure 5.16 - Reservoir Network architecture example.....		51
Figure 5.17 - RN with Sparse Online postprocessing .....		52
Figure 5.18 - Bayes Network dependencies.....		55
Figure 5.19 - Example of discrete-time Markov process .....		58
Figure 5.20 - Decision Tree and Random Forest.....		59
Figure 5.21 - Time and Frequency domains.....		60
Figure 6.22 - Simple Neuron .....		64
Figure 6.23 - Common Transfer Functions.....		65
Figure 6.24 - Vector Inputs.....		65
Figure 6.25 - Neuron .....		66
Figure 6.26 - One layer	Figure 6.27 - Weight Matrix .....	66
Figure 6.28 - Simple Multiple Layers.....		67
Figure 6.29 - Jacobian Learning with adaptative $\beta$ .....		74
Figure 6.30 - LF comparison #1, convergence vs speed	Figure 6.31 - LF comparison #1, mse evolution	. 76
Figure 6.32 - LF comparison #2, convergence vs speed	Figure 6.33 - LF comparison #2, mse evolution	. 76
Figure 6.34 - LF comparison #3, convergence vs speed	Figure 6.35 - LF comparison #3, mse evolution	. 77
Figure 6.36 - Overfitting example	Figure 6.37 - Selection of early stopping .....	78
Figure 6.38 - Output analysis example .....		79
Figure 6.39 - Input example	Figure 6.40 - Output example with non-dynamic network.....	81
Figure 6.41 - Output example with dynamic network.....		81
Figure 6.42 - Output example with a dynamic and recurrent network.....		82
Figure 6.43 - Simple dynamic network .....		82
Figure 6.44 - Network with Time-Delay .....		83
Figure 6.45 - Example of a simple Time-Delay Network response and error .....		84
Figure 6.46 - NARX Network with Time-Delay .....		85
Figure 6.47 - Network response with exponential weight of errors over time .....		87
Figure 6.48 - Basic Radial Basis Network .....		88

Figure 6.49 - Graphical representation of RBN centers.....	89
Figure 6.50 - Alternatives to sigmoid function.....	92
Figure 6.51 - Bootstrap aggregating example.....	94
Figure 6.52 - GANN, GA and NN w/ Back-Propagation learning rate .....	95
Figure 7.53 - Examples of same correlation coefficient.....	97
Figure 7.54 - Load time series divisions .....	98
Figure 7.55 - Load with simple pre-processing .....	102
Figure 7.55 - Example of fit error .....	104
Figure 7.56 - Zoom in L2 error .....	105
Figure 7.57 - Example of error due to long weekend .....	106
Figure 7.58 - Example of time series used to get expected errors .....	108
Figure 7.59 - Initial and first derivative autocorrelation and partial autocorrelation .....	111
Figure 7.60 - Errors and correlations after ARIMA .....	111
Figure 7.61 - Cook's distance .....	113
Figure 7.62 - Histogram and plot vs lagged residuals .....	114
Figure 7.63 - Effect of each variable on the outcome.....	114
Figure 7.64 - Load in $t$ vs $t-24$ vs $t-7\cdot24$ and fit.....	115
Figure 7.65 - Small section of the Bagged Tree .....	116
Figure 7.66 - Actual bias-variance dilemma.....	117
Figure 7.67 - Comparison under different number of neurons.....	119
Figure 7.68 - Number of times each pattern is activated in a 400 division .....	126
Figure 7.69 - Neuron's weights for each variable .....	127
Figure 7.70 - Distance and relationship between neighbors.....	128
Figure 7.71 - Example of decision boundary.....	128
Figure 7.72 - Important difference between RBNN and k-mean NN.....	129
Figure 7.73 - Ancillary services requirements forecast.....	134
Figure 7.74 - Ancillary services requirements average error.....	135
Figure 7.75 - ML and intelligent optimization as the intersection.....	139
Figure 7.76 - Business integration.....	139
Figure 7.77 - Algorithms available in SciKit.....	141

## List of Tables

Table 1 - MAPEs comparison between different inputs .....	100
Table 2 - Pearson correlation and variance for inputs.....	100
Table 3 - MAPEs values with and without PCA .....	103
Table 4 - Coefficients and significant values of ARIMA(3,1,5) .....	112
Table 5 - Statistical coefficients for inputs.....	113
Table 6 - Several inputs delays with feedback delay equal to one .....	120
Table 7 - Several inputs delays with feedback delay from one to twenty four .....	120
Table 8 - Comparison between inputs subdivisions .....	121
Table 9 - Comparison between divisions for learning, testing and validation.....	122
Table 10 - Comparison between preprocessing .....	122
Table 11 - Comparison between learning requirements .....	123
Table 12 - Comparison between scenarios .....	124
Table 13 – Comparison between architectures .....	131
Table 14 - Comparison between learning algorithms.....	132

## References

Adam P. Piotrowski, J. J. N., 2013. *A comparison of methods to avoid overfitting in neural networks training in the case of catchment runoff modelling*. [Online]

Available at: <http://www.sciencedirect.com/science/article/pii/S0022169412008931>

Agosta, J. M., 2004. *Bayes Network "Smart" Diagnostics*. [Online]

Available at: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.369>

Aksoy, M., 2013. *NN - Bayesian Decision Theory*. [Online]

Available at: [http://www.byclb.com/TR/Tutorials/neural\\_networks/ch4\\_1.htm](http://www.byclb.com/TR/Tutorials/neural_networks/ch4_1.htm)

Alex Graves, G. W. I. D., 2014. *Neural Turing Machines*. [Online]

Available at: <http://arxiv.org/abs/1410.5401>

Amjady, N., 2011. *A New Neural Network Approach to Short Term Load Forecasting of Electrical Power Systems*, s.l.: s.n.

Andrew Ng., 2014. [Online]

Available at: <http://cs229.stanford.edu/>

Anon., 2014. *Deep learning and Deep neural networks*. [Online]

Available at: [http://en.wikipedia.org/wiki/Deep\\_learning#Deep\\_neural\\_networks](http://en.wikipedia.org/wiki/Deep_learning#Deep_neural_networks)



Anon., 2015. *Artificial\_neural\_network*. [Online]

Available at: [http://en.wikipedia.org/wiki/Artificial\\_neural\\_network](http://en.wikipedia.org/wiki/Artificial_neural_network)

Anon., 2015. *Autoencoder*. [Online]

Available at: <http://en.wikipedia.org/wiki/Autoencoder>

Anon., 2015. *Bayesian\_network*. [Online]

Available at: [http://en.wikipedia.org/wiki/Bayesian\\_network#Structure\\_learning](http://en.wikipedia.org/wiki/Bayesian_network#Structure_learning)

Anon., 2015. *Computational\_complexity\_theory*. [Online]

Available at: [http://en.wikipedia.org/wiki/Computational\\_complexity\\_theory](http://en.wikipedia.org/wiki/Computational_complexity_theory)

Anon., 2015. *Convolutional\_neural\_network*. [Online]

Available at: [http://en.wikipedia.org/wiki/Convolutional\\_neural\\_network](http://en.wikipedia.org/wiki/Convolutional_neural_network)

Anon., 2015. *Levenberg–Marquardt\_algorithm*. [Online]

Available at: [http://en.wikipedia.org/wiki/Levenberg%E2%80%93Marquardt\\_algorithm](http://en.wikipedia.org/wiki/Levenberg%E2%80%93Marquardt_algorithm)

Anon., 2015. *Levenberg–Marquardt\_algorithm*. [Online]

Available at: [http://en.wikipedia.org/wiki/Levenberg%E2%80%93Marquardt\\_algorithm](http://en.wikipedia.org/wiki/Levenberg%E2%80%93Marquardt_algorithm)

Anon., 2015. *Machine\_learning*. [Online]

Available at: [http://en.wikipedia.org/wiki/Machine\\_learning](http://en.wikipedia.org/wiki/Machine_learning)

Anon., 2015. *Mathematical\_optimization*. [Online]

Available at: [http://en.wikipedia.org/wiki/Mathematical\\_optimization](http://en.wikipedia.org/wiki/Mathematical_optimization)

Anon., 2015. *Memristor*. [Online]

Available at: <http://en.wikipedia.org/wiki/Memristor>

Anon., 2015. *Predictive\_analytics*. [Online]

Available at: [http://en.wikipedia.org/wiki/Predictive\\_analytics](http://en.wikipedia.org/wiki/Predictive_analytics)

Anon., 2015. *Predictive\_modelling*. [Online]

Available at: [http://en.wikipedia.org/wiki/Predictive\\_modelling](http://en.wikipedia.org/wiki/Predictive_modelling)

Anon., 2015. *Statistical\_model*. [Online]

Available at: [http://en.wikipedia.org/wiki/Statistical\\_model](http://en.wikipedia.org/wiki/Statistical_model)

Anon., 2015. *Statistics*. [Online]

Available at: <http://en.wikipedia.org/wiki/Statistics>

Awan, S. M., 2012. *Application of NARX based FFNN, SVR and ANN Fitting models for long term industrial load forecasting and their comparison*, s.l.: s.n.

Baggenstoss, P., 2001. *A modified Baum-Welch algorithm for hidden Markov models with multiple observation spaces*. [Online]

Available at: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=917686>

Baldi, P., 2012. *Autoencoders, Unsupervised Learning, and Deep Architectures*. [Online]

Available at: <http://www.jmlr.org/proceedings/papers/v27/baldi12a/baldi12a.pdf>

Benjamin Schrauwen, D. V. J. V. C., 2007. *An overview of reservoir computing: theory, applications and implementations*. [Online]

Available at: <https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2007-8.pdf>

Bloom, J., 2014. *Supernovas and Novel Insight: Where Machine Learning is Headed Next*. [Online]

Available at: <http://a16z.com/2015/01/07/supernovas-and-novel-insight-where-machine-learning-is-headed-next/>

Boden, M. A., 1988. [Online]

Available at: <http://mitpress.mit.edu/books/artificial-intelligence-psychology>

Boden, M. A., 1988. *Artificial Intelligence in Psychology*. [Online]

Available at: <http://mitpress.mit.edu/books/artificial-intelligence-psychology>

Bontempi, G., 2013. *Machine Learning Strategies for Time Series Forecasting*, s.l.: s.n.

Bostrom, N., 2014. *What happens when our computers get smarter than we are?*. [Online]

Available at:

[https://www.ted.com/talks/nick\\_bostrom\\_what\\_happens\\_when\\_our\\_computers\\_get\\_smarter\\_than\\_we\\_are](https://www.ted.com/talks/nick_bostrom_what_happens_when_our_computers_get_smarter_than_we_are)

Bousquet, O., 2003. *Statistical Learning Theory. Lecture 3: VC theory*. [Online]

Available at: [http://www.kyb.mpg.de/fileadmin/user\\_upload/files/publications/pdfs/pdf2522.pdf](http://www.kyb.mpg.de/fileadmin/user_upload/files/publications/pdfs/pdf2522.pdf)

Breiman, L., 2001. *Statistical Modeling: The Two Cultures*. [Online]

Available at: <http://projecteuclid.org/euclid.ss/1009213726>

Brendan van Rooyen, R. C. W., 2015. *A Theory of Feature Learning*. [Online]

Available at: <http://arxiv.org/pdf/1504.00083v1.pdf>

Brynjolfsson, E., 2013. *TED*. [Online]

Available at:

[http://www.ted.com/talks/erik\\_brynjolfsson\\_the\\_key\\_to\\_growth\\_race\\_em\\_with\\_em\\_the\\_machines](http://www.ted.com/talks/erik_brynjolfsson_the_key_to_growth_race_em_with_em_the_machines)

Bryson, A. E., n.d. *Applied Optimal Control: Optimization, Estimation and Control*. s.l.:s.n.

Bullinaria, J. A., 2004. *Self Organizing Maps: Fundamentals*. [Online]

Available at: <http://www.cs.bham.ac.uk/~jxb/NN/l16.pdf>

Ceperic, E., 2013. *Machines, A Strategy for Short-Term Load Forecasting by Support Vector Regression*, s.l.: s.n.

Chang, 2007. [Online]

Available at: <http://www.stat.yale.edu/~pollard/Courses/251.spring09/Handouts/Chang-notes.pdf>

CheGuan, 2013. *Very Short-Term Load Forecasting: Wavelet Neural Networks With Daya Pre-Filtering*, s.l.: s.n.

Chien-Cheng, 2002. *A backpropagation algorithm with adaptive learning rate and momentum coefficient*. [Online]

Available at: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1007668>

Cukier, K., 2014. *Big data is better data*. [Online]

Available at: [http://www.ted.com/talks/kenneth\\_cukier\\_big\\_data\\_is\\_better\\_data](http://www.ted.com/talks/kenneth_cukier_big_data_is_better_data)

D.C. Park, M. E.-S. R. M., 2001. *Electric Load Forecasting Using An Artificial Neural Network*. [Online]

Available at: [http://marksmannet.com/RobertMarks/REPRINTS/1991-05\\_ElectricLoadForecasting.pdf](http://marksmannet.com/RobertMarks/REPRINTS/1991-05_ElectricLoadForecasting.pdf)

Dalto, M., 2014. *Deep neuronal networks for time series prediction with applications in ultra-short-term wind forecasting*, s.l.: s.n.

Dataconomy, 2014. *10 Machine Learning Experts You Need to Know*. [Online]

Available at: <http://dataconomy.com/10-machine-learning-experts-you-need-to-know/>

David E. Rumelhart, D. Z., 1985. *Feature discovery by competitive learning*. [Online]

Available at: <http://www.sciencedirect.com/science/article/pii/S0364021385800100>

David Eigen, J. R. Y. L., 2013. *Understanding Deep Architectures using a Recursive Convolutional Network*. [Online]

Available at: <http://arxiv.org/pdf/1312.1847.pdf>

Dean, J., 2012. *Large Scale Distributed Deep Networks*, s.l.: s.n.

DeepMind, 2014. [Online]

Available at: <http://www.technologyreview.com/view/532156/googles-secretive-deepmind-startup-unveils-a-neural-turing-machine/>

Dimitrios Mantzaris, G. A. A. A., 2011. *Genetic algorithm pruning of probabilistic neural networks in medical disease estimation*. [Online]

Available at: <http://www.sciencedirect.com/science/article/pii/S0893608011001638>

Eirola, E., 2014. *Machine learning methods for incomplete data and variable selection*. [Online]

Available at: <http://lib.tkk.fi/Diss/2014/isbn9789526058719/isbn9789526058719.pdf>

Enzo Busseti, I. O. S. W., 2012. *Deep Learning For Time Series Modeling*. [Online]

Available at: <http://cs229.stanford.edu/proj2012/BussetiOsbandWong-DeepLearningForTimeSeriesModeling.pdf>

- Eric B. Baum, D. H., 1989. *What Size Net Gives Valid Generalization?*. [Online]  
Available at: <http://www.mitpressjournals.org/doi/abs/10.1162/neco.1989.1.1.151>
- Evans, P., 2013. *TED*. [Online]  
Available at: [http://www.ted.com/talks/philip\\_evans\\_how\\_data\\_will\\_transform\\_business](http://www.ted.com/talks/philip_evans_how_data_will_transform_business)
- FutureOfLife, 2014. *Research Priorities for Robust and Beneficial Artificial Intelligence: an Open Letter*. [Online]  
Available at: [http://futureoflife.org/misc/open\\_letter](http://futureoflife.org/misc/open_letter)
- Galván, D. V., 2011. *Optimización de un modelo de previsión de demanda eléctrica para clientes minoristas*. [Online]  
Available at: <http://www.iit.upcomillas.es/pfc/resumenes/4ded21becbdb.pdf>
- Geiger, D., 1990. *Identifying independence in Bayesian Networks*. [Online]  
Available at: [http://ftp.cs.ucla.edu/pub/stat\\_ser/r116.pdf](http://ftp.cs.ucla.edu/pub/stat_ser/r116.pdf)
- Ghahramani, Z., 2001. *An Introduction to Hidden Markov Models and Bayesian Networks*. [Online]  
Available at: <http://mlg.eng.cam.ac.uk/zoubin/papers/ijprai.pdf>
- Graves, A., 2012. *A Comparison of Network Architectures*. [Online]  
Available at: [http://link.springer.com/chapter/10.1007/978-3-642-24797-2\\_5](http://link.springer.com/chapter/10.1007/978-3-642-24797-2_5)
- Guoqiang Zhang, B. E. P. M. Y. H., 2007. *Forecasting with artificial neural networks: The state of the art*. [Online]  
Available at:  
[http://www.researchgate.net/publication/222489987\\_Forecasting\\_with\\_artificial\\_neural\\_networks\\_The\\_state\\_of\\_the\\_art](http://www.researchgate.net/publication/222489987_Forecasting_with_artificial_neural_networks_The_state_of_the_art)
- Haddop, 2014. *Apache*. [Online]  
Available at: <https://hadoop.apache.org/>
- Hagan, F. a., 1997. *Matlab*. [Online]  
Available at: <http://es.mathworks.com/help/nnet/ref/trainbr.html>
- Harold Soh, Y. D., 2012. *Iterative Temporal Learning and Prediction with the Sparse Online Echo State Gaussian Process*. [Online]  
Available at: [http://haroldsoh.files.wordpress.com/2012/02/oesgp\\_soh\\_camready.pdf](http://haroldsoh.files.wordpress.com/2012/02/oesgp_soh_camready.pdf)  
[Accessed 12 2014].
- Harold Soh, Y. D., 2014. *Iterative Temporal Learning and Prediction with the Sparse Online Echo State Gaussian Process*. [Online]  
Available at: <https://spiral.imperial.ac.uk/bitstream/10044/1/12655/4/ijcnn.pdf>
- Heckerman, D., 1996. *A Tutorial on Learning With Bayesian Networks*. [Online]  
Available at: <http://research.microsoft.com/pubs/69588/tr-95-06.pdf>

- Heckerman, D., 1996. *A Tutorial on Learning With Bayesian Networks*. [Online]  
Available at: <http://research.microsoft.com/pubs/69588/tr-95-06.pdf>
- Hong, T., 2012. *IEEE Working Group on Energy Forecasting*. [Online]  
Available at: <http://www.drhongtao.com/ieee-wgef>
- Hongzhan Nie, G. L. X. L. Y. W., 2012. *Hybrid of ARIMA and SVMs for Short-Term Load Forecasting*. [Online]  
Available at: <http://www.sciencedirect.com/science/article/pii/S1876610212002391>
- Howard, J., 2014. *The wonderful and terrifying implications of computers that can learn*. [Online]  
Available at:  
[http://www.ted.com/talks/jeremy\\_howard\\_the\\_wonderful\\_and\\_terrifying\\_implications\\_of\\_computers\\_that\\_can\\_learn](http://www.ted.com/talks/jeremy_howard_the_wonderful_and_terrifying_implications_of_computers_that_can_learn)
- HP, 2014. *HP plans to launch memristor, silicon photonic computer within the decade*. [Online]  
Available at: <http://arstechnica.com/information-technology/2014/06/hp-plans-to-launch-memristor-silicon-photonic-computer-within-the-decade/>
- Huang, G.-B., 2005. *A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation*. [Online]  
Available at: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1388458>
- Hugo Larochelle, Y. B., 2009. *Exploring Strategies for Training Deep Neural Networks*. [Online]  
Available at: <http://www.cs.toronto.edu/~larocheh/publications/jmlr-larochelle09a.pdf>
- Hutchinson, J. M., 1994. *A Radial Basis Function Approach to Financial Time Series Analysis*. [Online]  
Available at: <ftp://publications.ai.mit.edu/ai-publications/pdf/AITR-1457.pdf>
- IBM, 2014. *IBM Big Data & Analytics*. [Online]  
Available at: [http://www.ibm.com/big-data/us/en/big-data-and-analytics/?lnk=bd\\_hm\\_top](http://www.ibm.com/big-data/us/en/big-data-and-analytics/?lnk=bd_hm_top)
- IBM, 2014. *Million neurons in 28nm CMOS processor*. [Online]  
Available at: [http://www.theregister.co.uk/2014/08/07/ibm\\_synapse\\_chip/](http://www.theregister.co.uk/2014/08/07/ibm_synapse_chip/)
- Intel, Q., 2014. *Intel Follows Qualcomm Down Neural Network Path*. [Online]  
Available at: <http://electronics360.globalspec.com/article/4318/intel-follows-qualcomm-down-neural-network-path>
- J. Leonard, M. K., 1990. *Improvement of the backpropagation algorithm for training neural networks*. [Online]  
Available at: <http://www.sciencedirect.com/science/article/pii/0098135490870706>
- Jaakkola, T., 2010. *Lecture 11: model selection, density estimation*. [Online]  
Available at: <http://www.ai.mit.edu/courses/6.867-f01/lectures/lecture-11-ho.ps>

Janardan Misraa, I. S., 2010. *Artificial neural networks in hardware: A survey of two decades of progress*. [Online]

Available at: <http://www.sciencedirect.com/science/article/pii/S092523121000216X>

Jonathan Eastep, D. W. A. A., 2006. *Smart Data Structures: An Online Machine Learning Approach to Multicore Data Structures*. [Online]

Available at: <http://groups.csail.mit.edu/carbon/wordpress/wp-content/uploads/2011/03/eastep-smart-data-structures-icac11.pdf>

Joyce, J. Z. E. N., 2003. *Bayes' Theorem*, *The Stanford Encyclopedia of Philosophy*. [Online]

Available at: <http://plato.stanford.edu/entries/bayes-theorem/>

Kakade, S. M., 2012. *Online Bounds for Bayesian Algorithms*. [Online]

Available at: [http://research.microsoft.com/en-us/um/people/skakade/papers/ml/online\\_bayes.ps](http://research.microsoft.com/en-us/um/people/skakade/papers/ml/online_bayes.ps)

Ke-Lin Du, S. S., 2014. *Neural Networks and Statistical Learning*. [Online].

Kim, C.-H., 2012. *Short-term Electric Load Forecasting Using Data Mining Technique*, s.l.: s.n.

Kon, M. A., 2006. *Neural Networks, Radial Basis Functions, and Complexity*. [Online]

Available at: <http://math.bu.edu/people/mkon/nnpap3.pdf>

Krichmar, J. L., 2002. *Machine Psychology: Autonomous Behavior, Perceptual Categorization and Conditioning in a Brain-based Device*. [Online]

Available at: <http://cercor.oxfordjournals.org/content/12/8/818.abstract>

Kumar, S., 2008. *Electricity price forecasting in deregulated markets: A review and evaluation*, s.l.: s.n.

Kumar, V., 2012. *Cluster Analysis: Basic Concepts and Algorithms*. [Online]

Available at: <http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf>

Kurzweil, R., 2005. *TED*. [Online]

Available at: [http://www.ted.com/talks/ray\\_kurzweil\\_on\\_how\\_technology\\_will\\_transform\\_us](http://www.ted.com/talks/ray_kurzweil_on_how_technology_will_transform_us)

L. Suganthia, A. A. S., 2012. *Energy models for demand forecasting - A review*. [Online]

Available at: <http://www.sciencedirect.com/science/article/pii/S1364032111004242>

Langkvist, M., 2014. *A Review of Unsupervised Feature Learning and Deep Learning for Time-Series Modeling*, s.l.: s.n.

LeCun, Y., 2007. *Scaling Learning Algorithms towards AI*. [Online]

Available at: <http://yann.lecun.com/exdb/publis/pdf/bengio-lecun-07.pdf>

Leung H, L. T. W. S., 2001. *Prediction of noisy chaotic time series using an optimal radial basis function neural network*. [Online]

Available at: <http://www.ncbi.nlm.nih.gov/pubmed/18249942>

LIONos, 2011. *LIONos*. [Online]

Available at: <http://intelligent-optimization.org/LIONbook/>

Lloyd, S., 2013. *Quantum algorithms for supervised and unsupervised machine learning*. [Online]

Available at: <http://arxiv.org/pdf/1307.0411v2.pdf>

Mao K., 2004. *Feature subset selection for support vector machines through discriminative function pruning analysis*. [Online]

Available at: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1262482>

Markos Papadonikolakis, C.-S. B. G. C., 2009. *Performance Comparison of GPU and FPGA architectures for the SVM Training Problem*. [Online]

Available at: <http://cas.ee.ic.ac.uk/people/gac1/pubs/MarkosFPT09.pdf>

Markram, H., 2009. *TED*. [Online]

Available at: [http://www.ted.com/talks/henry\\_markram\\_supercomputing\\_the\\_brain\\_s\\_secrets](http://www.ted.com/talks/henry_markram_supercomputing_the_brain_s_secrets)

mathematicalmonk, 2014. [Online]

Available at: <https://www.youtube.com/playlist?list=PLD0F06AA0D2E8FFBA>

MathWorks, 2015. *Econometrics Toolbox*. [Online]

Available at: <http://es.mathworks.com/help/econ/index.html>

MathWorks, 2015. *Neural Network Toolbox*. [Online]

Available at: <http://es.mathworks.com/help/nnet/index.html>

Matthew M. Botvinick, Y. N., 2008. *Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective*. [Online]

Available at: <http://www.princeton.edu/~yael/Publications/BotvinickNivBarto2008.pdf>

Mehdi Khashei, M. B., 2010. *A novel hybridization of artificial neural networks and ARIMA models for time series forecasting*. [Online]

Available at: <http://www.sciencedirect.com/science/article/pii/S1568494610002759>

Microsoft, 2014. *Azure - ML*. [Online]

Available at: <http://azure.microsoft.com/en-us/services/machine-learning/>

Mishra, S., 2008. *Short Term Load Forecasting Using Computational Intelligence Methods*, s.l.: s.n.

MIT\_Press, 2015. *A Better Way to Build Brain-Inspired Chips*. [Online]

Available at: <http://www.technologyreview.com/news/537211/a-better-way-to-build-brain-inspired-chips/>

Narayanan Sundaram, B. C., 2010. *Accelerating Machine Learning Applications on Graphics Processors*. [Online]

Available at: [http://www.cs.berkeley.edu/~kubitron/courses/cs258-S08/projects/reports/project7\\_talk\\_ver2.ppt](http://www.cs.berkeley.edu/~kubitron/courses/cs258-S08/projects/reports/project7_talk_ver2.ppt)

- Ng, A., 2012. *Machine Learning course*. [Online]  
Available at: <http://openclassroom.stanford.edu/MainFolder/CoursePage.php?course=MachineLearning>
- Norvig, P., 2012. *Channeling the Flood of Data*. [Online]  
Available at: [http://library.fora.tv/2012/10/14/Peter\\_Norvig\\_Channeling\\_the\\_Flood\\_of\\_Data](http://library.fora.tv/2012/10/14/Peter_Norvig_Channeling_the_Flood_of_Data)
- OCWmit, 2006. *Rohit Singh*. [Online]  
Available at: <http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/>
- Olshausen, B. A., 2004. *Bayesian probability theory*. [Online]  
Available at: <http://redwood.berkeley.edu/bruno/npb163/bayes.pdf>
- Ong, H.-C., 2011. *A Comparison on Neural Network Forecasting*, s.l.: s.n.
- P. Rivas-Perea, G. R., 2011. *Short term electric power consumption forecasting using linear programming support vector regression*. [Online]  
Available at: [http://cs.ecs.baylor.edu/~rivas\\_perea/pdfs/rivas2011short.pdf](http://cs.ecs.baylor.edu/~rivas_perea/pdfs/rivas2011short.pdf)
- Pankratz, A., 1983. *Forecasting with Univariate Box - Jenkins Models: Concepts and Cases*. [Online]  
Available at: <http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0471090239.html>
- Peter Sussner, E. L. E., 2011. *Morphological perceptrons with competitive learning: Lattice-theoretical framework and constructive learning algorithm*. [Online]  
Available at: <http://www.sciencedirect.com/science/article/pii/S0020025510001283>
- Petr Berka, J. R., 2010. *Machine Learning and Association Rules*. [Online]  
Available at: [https://www.rocq.inria.fr/axis/COMPSTAT2010/TU\\_Berka-Rauch\\_paper.pdf](https://www.rocq.inria.fr/axis/COMPSTAT2010/TU_Berka-Rauch_paper.pdf)
- Pham, D. T., 2004. *Selection of K in K-means clustering*. [Online]  
Available at: <http://www.ee.columbia.edu/~dpwe/papers/PhamDN05-kmeans.pdf>
- Pitt, L. D., 1971. *A Markov property for Gaussian processes with a multidimensional parameter*. [Online]  
Available at: <http://link.springer.com/article/10.1007%2FBF00252003>
- Prechelt, L., 1998. *Automatic early stopping using cross validation: quantifying the criteria*. [Online]  
Available at: <http://www.sciencedirect.com/science/article/pii/S0893608098000100>
- Qualcomm, 2013. *Processors: Brain-Inspired Computing*. [Online]  
Available at: <https://www.qualcomm.com/news/onq/2013/10/10/introducing-qualcomm-zeroth-processors-brain-inspired-computing>
- Richard Maclin, D. O., 1997. *An empirical comparison of bagging and boosting*. [Online]  
Available at: [www.d.umn.edu/~rmaclin/publications/maclin-aaai97.pdf](http://www.d.umn.edu/~rmaclin/publications/maclin-aaai97.pdf)
- Riedmiller, M., 1994. *Advanced supervised learning in multi-layer perceptrons — From backpropagation to adaptive learning algorithms*. [Online]  
Available at: <http://www.sciencedirect.com/science/article/pii/0920548994900175>



Rob Hyndman, S. F., 2010. *Short-term load forecasting based on a semi-parametric additive model*. [Online]

Available at: <http://robjhyndman.com/conference/aupec2010/>

Roberto Battiti, M. B., 2010. *Reactive Search Optimization: Learning while Optimizing*. [Online]

Available at: <http://rtm.science.unitn.it/~battiti/archive/handbook-of-metaheuristics-in-press.pdf>

Roberto Battiti, M. B., 2014. *LIONlab*. [Online]

Available at: <http://intelligent-optimization.org/LIONbook/>

Roberto Battiti, M. B. F. M., 2014. *Reactive Search and Intelligent Optimization*. s.l.:s.n.

Roweis, S., 2010. *Boltzmann Machines*. [Online]

Available at: <https://www.cs.nyu.edu/~roweis/notes/boltz.pdf>

SAS, 2013. *Big Data in Big Companies*. [Online]

Available at: <http://www.sas.com/resources/asset/Big-Data-in-Big-Companies.pdf>

Schilling, R., 2002. *Approximation of nonlinear systems with radial basis function neural networks*. [Online]

Available at: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=896792>

Search Engine Optimization, 2015. *Deep Learning And The Future Of Search Engine Optimization* [Online]

Available at: <http://techcrunch.com/2015/06/18/deep-learning-and-the-future-of-search-engine-optimization>

Schmidhuber, J., 2014. *Deep Learning in Neural Networks: An Overview*. [Online]

Available at: <http://arxiv.org/abs/1404.7828>

SciKit, 2014. *github*. [Online]

Available at: <https://github.com/scikit-learn/scikit-learn>

Sharon-Use Normanda, D. T., 1992. *Parameter Updating in a Bayes Network*. [Online]

Available at: <http://amstat.tandfonline.com/doi/abs/10.1080/01621459.1992.10476266>

Shogun, 2014. *Github*. [Online]

Available at: <https://github.com/shogun-toolbox/shogun>

Siemens, 2015. *Big Data: Magnifying America's Energy Future*. [Online]


Available at: <http://online.wsj.com/ad/article/siemans-big-data-magnifying-americas-energy-future>

Snijder, F., Dept. of Electr. Eng., D. U. o. T. N., Babuska, R. & Verhaegen, M., 1998.

<http://ieeexplore.ieee.org/>. [Online]

Available at:

[http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=686019&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxppls%2Fabs\\_all.jsp%3Farnumber%3D686019](http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=686019&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxppls%2Fabs_all.jsp%3Farnumber%3D686019)

- 
- Sohl-Dickstein, J., 2014. *Minimum Probability Flow Learning*. [Online]  
Available at: <http://redwood.berkeley.edu/jascha/pdfs/icml.pdf>
- Spark, 2014. *Apache*. [Online]  
Available at: <https://spark.apache.org/>
- Stanford, 2012. *Machile Learning Course*. [Online]  
Available at: <https://www.coursera.org/course/ml>
- Stanley C. Ahalt, A. K. K., 1990. *Competitive learning algorithms for vector quantization*. [Online]  
Available at: <http://www.sciencedirect.com/science/article/pii/089360809090071R>
- StatLearn, 2003. *StatLearn Reading List*. [Online]  
Available at: <http://web.media.mit.edu/~ash/StatLearn/>
- Stuart, G., 1992. *Neuronal Networks and the Bias\|Variance dilemma*. [Online]  
Available at:  
<http://www.dam.brown.edu/people/geman/Homepage/Essays%20and%20ideas%20about%20neurobiology/bias-variance.pdf>
- Suvrit Sra, S. N., 2012. *Optimization for Machine Learning*. [Online].
- Tapas Kanungo, D. M. M., 2002. *An Efficient k-Means Clustering Algorithm: Analysis and Implementation*. [Online]  
Available at: <http://www.cs.umd.edu/~mount/Projects/KMeans/pami02.pdf>
- Torch7, 2014. *Github*. [Online]  
Available at: <https://github.com/torch/torch7>
- Trevor Hastie, R. T. J. F., 2009. *Data Mining, Inference, and Prediction..* [Online]  
Available at: <http://statweb.stanford.edu/~tibs/ElemStatLearn/>
- Triola, M. F., 2006. *Bayes' Theorem*. [Online]  
Available at: <http://faculty.washington.edu/tamre/BayesTheorem.pdf>
- WahHe, 2014. *Deep neural network based load forecast*, s.l.: s.n.
- Warden, P., 2015. *Why GEMM is at the heart of deep learning*. [Online]  
Available at: <http://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/>
- White, H., 1989. *Learning in Artificial Neural Networks: A Statistical Perspective*. [Online]  
Available at: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6796856>
- Xiao, D., 2009. *New Perspectives on the Complexity of Computational Learning, and Other Problems in Theoretical Computer Science*. [Online]  
Available at: <http://www.liafa.univ-paris-diderot.fr/~dxiao/docs/phd-thesis.pdf>

Xiaofeng, 2014. *Research and Application of Data Mining and NARX Neural Networks in Load Forecasting*, s.l.: s.n.

Yann LeCun, A. C. M. H., 2015. *The Loss Surfaces of Multilayer Networks*. [Online]  
Available at: <http://arxiv.org/pdf/1412.0233.pdf>

Yann LeCun, L. B., 1998. *Gradient-Based Learning Applied*. [Online]  
Available at: <http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf>

Yoshua Bengio, Y. L., 2007. *Scaling Learning Algorithms towards AI*. [Online]  
Available at: <http://yann.lecun.com/exdb/publis/pdf/bengio-lecun-07.pdf>

Zhang, J., 2002. *Sequential training of bootstrap aggregated neural networks for nonlinear systems modelling*. [Online]  
Available at: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1024861>

Zheng, Z., 2006. *Boosting and Bagging of Neural Networks with Applications to Financial Time Series*. [Online]  
Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.6651&rep=rep1&type=pdf>

I would like to take the opportunity to **thank** all MOOC, tutorials, Wikipedia, mathematicalmonk and thousands of people who have devoted part of their time to publish for free a lot of basic information in relation to machine learning, data mining and optimization.

