8-25-2016

Machine Learning Approach to Forecast Global Solar Radiation Time Series

Guillermo Terren-Serrano

Guillermo Terren-Serrano

Candidate

Electrical and Computer Engineering

Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

Manel Martinez-Ramon, Chair

Ramiro Jordan, Co-Chair

Christos Christodoulou, Member

Machine Learning Approach to Forecast Global Solar Radiation Time Series

by

Guillermo Terrén-Serrano

Technical Industrial Engineering, University of Zaragoza, 2012

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

July, 2016

Dedication

To my friends, who took me on cool road trips around the USA.

Acknowledgments

I would like to thank my advisor, Professor Manel Martínez-Ramón, for the knowledge that he shared with me. And to my labmate Miguel Angel, for all the time that we spent together during our research.

I would also like to thank my parents and sister, for their unconditional support at any time of the day.

Machine Learning Approach to Forecast Global Solar Radiation Time Series

by

Guillermo Terrén-Serrano

Technical Industrial Engineering, University of Zaragoza, 2012

M.S., Electrical Engineering, University of New Mexico, 2016

Abstract

The integration of Renewable Energy (RE) into Power Systems brings new challenges to the Smart Grids (SG) technologies. The generation output from renewable sources generally depends on the atmospheric conditions. This fact causes intermittences on the power output from renewable source, and hence the power quality of the grid is directly affected by atmospheric phenomena. The increasing advances on technologies for energy storage open a track to the Energy Management (EM). Therefore, the power output from a renewable source can be stored or dispatched in a particular time-instant in order to meet the demand. Scheduling Demand Respond (DR) action on the grid, can optimize the dispatch by reducing over generated energy wastage. The difficulty now is to ensure the availability of energy to supply into the grid by forecasting the Global Solar Radiation (GSR) on a localization where a Photovoltaic (PV) system is connected. This thesis tries to address the issue using Machine Learning (ML) techniques. This eases the generation scheduling task. The work developed on this thesis is focused on exploring ML techniques to hourly forecast GSR and optimize the dispatch of energy on a SG. The experiments present

results for different configuration of Deep Learning and Gaussian Processes for GSR time-series regression, aiming to discuss the advantages of using hybrid methods on the context of SG.

Contents

List of Figures

List of Tables

Glossary

a_{ij}	Coefficients that define a particular element on a vector or matrix.
a	Down-case letters denote a vectors or scalers.
A	Upper-case letters are used to define a matrix.
A^T	Transpose of operator.
a^*	Test sample of data.
a_*	Predicted data.
$\|.\|$	Absolute value of the total magnitude between the straight brackets.
$\sigma(.)$	Sigmoid function.
Δ	Incremental value.
$\bigtriangledown$	Gradient.
$\mathbb{R}^D$	D-dimensional Euclidean space defined on real numbers.
$\mathcal{N}(.,.)$	Normal distribution.
$m(.)$	mean function.
$k(.,.)$	Kernel function.

$\mathbb{E}[.]$	Expectation operator.
$\mathcal{GP}(.,.)$	Gaussian Process.
SG	Smart Grid.
MG	Micro Grid.
DR	Demand Response.
EM	Energy Management.
RE	Renewable Energy.
PV	Photovoltaic.
GSR	Global Solar Radiation.
ML	Machine Learning.
ANN	Artificial Neural Network.
DL	Deep Learning.
SVM	Support Vector Machine.
GP	Gaussian Process.
EML	Extreme Machine Learning.
k-NN	k-Nearest Neighbor.
MM	Maximum Margin.
MLE	Maximum Likelihood Estimator.
MMSE	Minimum Mean Square Error.

CD	Contrastive Divergence.
MLP	Multi-Layer Perception.
RR	Ridge Regression.
RBM	Restricted Boltzmann Machine.
ReLU	Rectified Linear Units.
WRF	Weather Research and Forecasting

Chapter 1

Introduction

The increasing number of relevant publications on Smart Grid (SG) is opening new paths on the research of Power Systems Networks. Since SG technologies allow us using real-time features measured from the Power Grid, they also allow us implementing real-time decisions that can affect the to Grid. Adding these characteristics to common Power System can improve the optimization of the overall performances of a Grid.

1.1 Smart Grids

One of the main goals of the SG and one of the most important challenges for the Micro Grids (MG) is fitting as tightly as possible the Electrical Demand with the Power Generation. In order to achieve this, the system needs to be aware of the availability of the resources. In this way, a SG can redistribute the power generation performing an optimization to maintain continuous the power supply on all the loads. On top of that, a Power System operating on islanded-mode can execute Demand Response (DR) decisions to assure the continuous operation of the main loads. There

are several DR measure that can be taken on the optimization but the most important are essentially Peak Shaving and Load Scheduling [1]. On one hand, the Peak Shaving would disconnect secondary loads in order to maintain active primary loads on the Power System. On the other hand, Load Scheduling would optimize the dispatch of energy and the availability of the resource with the programmed loads, in order to minimize the consumption of energy by just connecting certain loads at the time-instant that energy generation is available in exceed. In the case that the system cannot operate independently from a Grid even taking DR measures, the MG would be connected back to a larger Grid.

Another of the challenges on the SG is the integration of Renewable Energy (RE) with Power Systems. The instability and intermittence of the resources makes difficult to relay the Power Generation on these kind of resources. Furthermore, modeling the weather to preciously forecast the following time-instant is a hard task. In contrast, the decreasing price of the batteries and the continuous improvement of performances and size, considerably ease the integration of RE and therefore actual MG implementations. There are mainly two stage of power generation in which the batteries can assist when RE are used: the generation curve smoothing and shifting of high frequency disruptions on the generation. Since batteries can perform these two function on the generation, RE is a real alternative on the Power Generation.

The combination of different resources will stabilize the Energy Generation ensuring that the MG will meet the energy demanded at any time-instant. For example, Natural Gas Cogeneration plants perform an excellent usage of the energy with high efficiencies on the production because of the reutilization of the heat produced in exceed for heating tasks on thermal loads included on the system. In this way, there is a notable reduction on the overall energy consumed by the system, due to the diversification of the resources. In the case of the electrical loads, the optimization

[1]Source: www.energy.gov

would be done between Natural Gas plant, Photovoltaic Systems and Batteries, aiming to minimize the usage of the electricity supplied by the grid for obvious reasons related with the prizing.

In order to perform an optimization on the energy generation implementations of algorithms for the Energy Management (EM) need to be done. What these algorithms aim to achieve is the load forecasting and generation forecasting. As was stated previously, the availability of the resource on RE can vary considerably during a day; therefore the EM is a difficult task to carry out on MG. In contrast, whether the system has the information of a forecasted time-instant ahead relatively precise, the adaptation of the MG to that case-scenario by selecting the generation sources and performing EM plan will ensure the optimal energy operation of the Power System.

Forecasting power output with an algorithm will allow the optimization of the power generation and its dispatch to the load. The main goal at this stage of the problem is to reduce the fuel usage and CO_2 emission by conventional power plants by maximizing the energy produced by renewable resources [2]. A secondary goal will be to minimize the number of batteries needed by the Power System in order to smooth and shift the output of the renewable sources, which will considerably reduce the initial economic investment in MG . Even though the cost of batteries is reducing significantly, they are still very expensive resources. If the price finally decreases down to a feasible cost, the main energy supply source to Power Systems will be renewable. Another advantage that renewable sources bring to the Power Systems is that their maintenance costs are very small in comparison with conventional power plants, besides that there is not expenditure in combustible along their entire life-cycle. It is also very important to mention that the usage of batteries on the MG is tied to the fact of their short life-cycle. They commonly require replacement between

[2]Source: www3.epa.gov

5 and 8 years after their commissioning [3]. This is another important reason why the number of batteries needs to be properly assessed at the designing stage of a MG project.

1.2 Energy and Power Forecasting on Micro Grids

In particular, we tackle the problem of generation forecasting the point of view of Machine Learning (ML). The implementation of a forecasting model is based on historical data extracted from features of the MG and its surroundings. In this case, devices such as Smart Meter and a Weather Station acquire the data, which records the information about the weather and generation patterns on a particular location. Parameters such as air maximum and minimum temperature or global solar radiation are highly correlated with the power dispatched by Photovoltaic systems (PV) on the MG. Other features such as air humidity or wind speed have less influence on the power output curve. It is also relevant to mention that there are usually seasonal components on the Power Grid, although it would be necessary to explore each case-scenario separately the seasonal component of the data depends on the sub-problem location. For these reasons, the proper Data Processing and Acquisition are very important. The goal at this stage is to filter out the unnecessary features by pre-processing data adequately. In this way, better performances on the regression for a predicted power output by an ML algorithm can be expected. Thus, a pre-training algorithm to find the most important features from the data could considerably improve the performances of the GSR or power output from the developed forecasting model .

Depending on the terminal bus that the PV system is attached to, the problem will be stated at a different level with regards to the transmission system. There are

[3]Source: www.nrel.gov

approaches to the problem that aim to forecast the generation from the point of utility grid with many available resources and the possibility of diversifying the generation. There are also cases which aimed to predict a single household PV system output. In some other cases, the forecasting model is used to predict the generation of a region or an entire country. Therefore, the resolution of the input data used for setting the model will be different for each of these cases. The developed model cannot be generalized and used for many different kinds of PV system. The periodicity and the frequency of the intermissions will significantly vary according to the weather pattern. For these reason, each type of algorithm needs to be assessed separately.

According to the nature of the problem, the data that will be used in the development of a forecasting model will contain features of the weather parameters that can influence on the PV system or Wind turbine generation. This depends on the renewable source that is aimed to be predicted. In this case, only the PV technology is studied. The number of features or dimensions of the dataset for each one of time-instants will define the bandwidth of the forecasting model. In some cases, a combination of PV power output and weather features has been used. Parameters such as air temperature and cloud cover are very representative for forecasting solar radiation. In the case of a MG, the aim is to optimize the supply of energy to the loads. For doing that, the following time-instant is predicted and the Power System will configure itself for meeting the loads by diversifying the supply of energy and scheduling the dispatch and recharge of the Batteries Bank. In other cases, it can be interesting to forecast the next-day curve in order to schedule the supply of energy. Moreover, it can be used as a mechanism of the electrical market for regulating the generation mix plan for the next day. The forecasting model can be developed for medium term GSR predictions, [2]. The horizon in these cases is between a month and a year. Those time-frames are useful when the goal is to analyze the evolution of degradation of the panel along its life cycle. However, this can be also a mechanism for planning upgrades or scheduling maintenance tasks the PV systems. It can

study tendencies on the generation and detect faults on devices. In order to develop a model for hourly solar radiation forecasting, the input data have certain features which have higher correlation with GSR. From this point of view, the GSR curve has a certain periodicity daily, monthly and yearly. On winter, the solar radiation on the surface of the PV panel would have smaller daily average value than the solar radiation expected on a typical Summer day. The same happen in Spring and Fall, when there are typically more rainy and cloudy days, [3], [4], [5]. There are also variations between consecutive months because of weather and seasonal patterns. They also depend on the location. Therefore, the model has to be developed with the data collected on the location, which is usually from the SG and weather station. Another characteristic of the data that we need to look at, is the depth of the historical. In some of studies that have been carried out, several years of historical data have been used. In contrast to other algorithms that use partial year information for implementing the model.

Chapter 2

Forecasting Methods for Power and Energy Systems

The various ML techniques used for forecasting the PV system output can be classify by analyzing cases that have been study previously. There are also several approaches to data processing that are used for implementing the models. In addition to the variation of input data that is actually under study and its final usage. In this part of the thesis, there are discussed different approaches,plus their advantages and drawbacks. This documentation work aims to find the resources that have been already implemented and, using this knowledge, to develop a new approach that produces better results for the particular case of scheduling a mix generation plan on a MG.

2.1 The Approaches for PV Energy Forecasting

The most appropriate algorithm, depends on the output variable that is being forecasted. Despite that, the final goal is the same. For example, [6], [7], [8], [7] and

[9], attempt to estimate the GSR on a surface. The time horizon and their input dataset varies with respect to each paper. In other articles such as [10], [11], [12] and [3], the actual output power from a PV system is the target of the prediction. In those cases, the input data contains the PV system power output and weather variables in order to characterize the forecasting model accurately. The horizon of the prediction is another of the particularities of each one of the articles. There is a classification addressed in [2], that refers to a previous classification of horizons for Load Forecasting. In this case, the resolution framework is defined as short-term (between day and days), medium term (week and weeks), and long-term (between a year and years). According to this resolution, the algorithm is used for different final proposes such as the scheduling of maintenance tasks [2], energy market assessment [13] or generation planning duties. There is a particular case in [5] in which satellite images for classification of the weather conditions are included. This is done using a trained forecasting model that is likely more appropriate to the weather parameters.

2.2 Selection of Features from Weather Data

In the literature reviewed, the input data includes parameters acquired from weather stations for all cases. In articles like [14], [15] and [9], air temperature vectors are used as the main informative weather input for characterizing the predictive model. In the case of [3], the input variable is the aerosol index, which indicates a particular atmospheric condition in that time instant. The variable of sunshine or light intensity is used on the input data on [9], [16], [17] and [10] in combination with other variables. The most generalized case is one in which the input of a large dataset contains many different variables that physically characterize the weather in a particular time instant, such as air temperature, air humidity, wind speed and direction, sunshine hours, precipitation, cloud coverage, etc.... This type of input dataset is used in

[11], [18], [13] and [6].

There are some articles in which the input data used for training the algorithm are also used for classification of the weather patterns of the day and used a particular model that have been developed using data of similar atmospheric conditions in order to improve the performance of the regression. This is the case in [3], the this classification is done by sunny, cloudy, overcast and rainy day. In [5], there are 12 different models that are applied according to the information contained in satellite sky images. The resolution on the sampling of the input dataset varies also from very short-term to long-term intervals, depending on the goal of the paper. For instance, [19], [20], and [21] use a resolution on the sampling of an hour. Others articles processed resolutions on the time-frame of minutes and seconds such as [5], [8] and [22]. In articles [14] and [7], the resolution of the data is daily and hourly respectively. In regards to the depth of the dataset used to develop the predictive algorithm, there are similarities between [14], [23], [13] and [6]. The input data set is composed of weather data collected over several years. In other cases, the dataset is composed of information collected along decades as in [7] and [8]. In contrast, other articles used smaller sampling datasets, in the magnitude order of days such as [2], [13], and [12].

There are different criteria on the size of partitions of data used for the training, test and validation of the algorithm. Beside of the criteria for selecting which portion is used on each one of the phases. There are cases such as [13], [6], [7] and [8], in which are used a percentage of data between 60.% and 80.% for training and 30.% and 10.% for test. A validation dataset is not always required but it usually has a size which represent around the 10.% of the whole dataset. In other articles, the seasonal structure of the data is maintained for testing the performance, for instance in [11] and [14]. In others like [5], the dataset is composed by data from several locations, and it is split in such a way that data for training, corresponds to 65 stations and

the sample for test, to other 18 stations. According to the horizon of the prediction have been found different models on the literature. In [2], the forecasting resolution is daily and the horizon is 6 months ahead. This information pretends to be useful to schedule maintenance tasks. In the case of [17], [3] and [22], the resolution is hourly and the prediction depth is day ahead. The magnitude order in the resolution of the prediction is of minutes in articles such as [12] and [10]. The horizon can vary from an hour to several hours ahead in [13] and [21]. A model for monthly prediction is proposed in the article [4]. It uses 3 different models for sunny, cloudy or rainy conditions.

2.3 Review of Implemented Algorithms

In regression algorithm that is implemented on the experiments, there are several techniques combined showing different results, depending on the training and the parametrization of the algorithm. Although, all reviewed articles used supervised learning techniques on their work.

For instance, Artificial Neural Networks (ANN) are used in articles such as [5], [3], [19], [16] and [4]. The performance of the ANN are improved by optimization algorithms. For instance, in [2] is implemented Genetic Swarm Optimization. In [17], is used a genetic algorithm for parameters identification. In some other cases, the algorithms used on the optimization of the ANN is the well-known Backpropagation, for example in [3], [10], [24], and [7]. However, [19] and [5] used the Leverberg-Marquart algorithm for the minimization of the regression error. In some other articles, there are implementations of hybrid algorithms, such as [9], [13] and [2], in which were tested a Neuro-Fuzzy Computing, that is a combination of an ANN and Fuzzy Inference System. In [24], several variation of ANN are tested for comparing the performance of each one with the same input dataset, including a Deep Learning

(DL) approach for pre-training the network. In particular, this paper implements a Conditional Restricted Boltzmann Machine, a Recurrent Neural Network and Convolutional Network. The article [21] researches a Generalized Regression Neural Network optimized by Particle Swarm Optimization. A Radial Basic Function Neural Network algorithm, which is an ANN that uses a radial basic as activation function, was tested on [18] and [25] for very short-term forecasting.

In this review of literature, were found articles published in which non-linear models, optimized by Minimum Margin, were experimented. For instance, in [22], [14], [15], [6] [26] and [20] were implemented different Support Vector Machines (SVM) in combination of multiple Kernels functions. In [7] and [20] were used linear and non-linear kernels. Radial Basis function kernel was also tested in [7]. In the article [26], the performance of a SVM is compared to an ANN and Gaussian Process (GP). For example in [14], it was tested a SVM and an Extreme Machine Learning (EML) algorithm.

Some articles such as [27], [6] and [14], implemented EML techniques for forecasting GSR and energy generation. EML is a novelty technique inside the ANN that simplifies the training of the algorithm. For optimizing the EML, on [6] is used the Coral Reef Optimization, which is a Meta-Heuristic algorithm based on the reproduction and formation of the coral. In combination with EML in [27], a Entropy Method produces higher accuracy on the output and computes faster. This method essentially works by calculating the entropy of information to the relative degree of change indicators. A kernel Square Exponential is used on [14] at the input of a EML algorithm.

There are articles that show the performance of other ML algorithms such as in [26] and [11]. In [11] particularly, it is implemented the K-Nearest Neighbor (k-NN) method optimized by Gradient Boosting for a deterministic forecasting. The performance of different techniques are tested in [26]. These algorithms are concretely:

Relevance Vector Machine, Ridge Regression, Boosted Trees, Regression Tress, Least Square Linear Regression and Regularized Least Square Linear Regression. In [8], Auto-Regressive, Auto-Regressive Moving Average, Markov Chain, k-NN are implemented and their performances are compared to an ANN.

2.4 Performances of Forecasting Methods

The performance for different techniques found on this review vary considerable depending on the input data. The sampling resolution, historical depth, forecasting horizon and the selection of variables as the most informative for developing the model, have particular interpretation in each one of the article published.

The articles, [28] and [29] show a surveys of the different ML algorithms published for forecasting PV system energy output and GSR and they compare their performances. In the review of literature, were found performances with hourly resolution of SVM and EML. In articles such as [7] and [27], the R^2 are 0.969 and 0.99 respectively. In contrast with the R^2 0.6567 obtained by an ANN in [9]. In [5], ANN performs a Correlation Coefficient of 99.% and in [23] a R^2 of 99.78.% for daily forecasting. Meanwhile in [8] an ANN archives 0.79 of R^2 with a 3 seconds resolution input data by using a weather historical of 17 years. A GP obtains 2.19 a Mean Absolute Error in [26]. The best performance of a DL algorithm is a R^2 of 84,8.% in [24].

Chapter 3

Machine Learning

Machine Learning can be defined as the discipline of knowledge that research algorithms that can learn information from datasets. The enclosed information in data is used for making predictions and decisions. These algorithms essentially develop models from training samples of data, which are simply observations of variables. For instance, another interesting definition for the objective in ML is stated on [30], such as its primary goal is to make predictions as accurately as possible.

3.1 Introduction

The real fact is that there are an increasing number of ML application in the contemporaneous society and its number grows more and more every day. For instances, there are many applications of ML algorithms on medical diagnosis, which by analyzing datasets of brain scanners, those are able to predict or recognized patters on a patient from brain image. On the other hand, there are application more focus on e-commerce or website platforms, in which items are suggested to users based on their interests. Those identified by the information acquiesced from their pat-

tern of usage and visits on a website. There are other uses that are empowering search engines in order to best classify the results of a search to the interests that a user can have when it introduces a particular set of words. Other applications that have successfully found a solution to their problems in ML, was speech recognition. There have been developed machines that are able to transcribe speech into text. In some other areas such as computer vision, machines have been trained to identify objects on images. It is also important to mention that their presence is increasing on consumer products such as cameras and smartphones. This list of new applications is growing and the number of articles published in regards to new ML algorithms implemented and successfully working solving prediction, patterns recognition or classification problems is continuously increasing.

3.1.1 Types of Learning

Depending on the method used on the training of the machine, the algorithms can be classified in supervised learning, unsupervised learning and Reinforcement Learning. On the supervised learning, the machine is trained to map from particular input dataset and a goal, and produces an output from the model. Therefore, there is a pair of input-output label for each sample and the algorithm is trained to meet it as accurately as possible. In the case of unsupervised learning, there is only a set of input data on the training of the machine, so there are not labels or targets. For this reason, the training phase of the machine is particularly tough because it is really difficult to know what are the output patterns that the machines are going to display. Therefore, it is sometimes impossible to find a particular pattern on the data or established an error metric to use along the training to avoid over-fitting problems. Another classification on the training criteria used in ML, is reinforcement learning. On this type of training, the machine interacts directly with a dynamic environment, in which it must perform a particular goal but without any feedback about whether

it is coming close to their goal or target, or not.

The problems that ML aims to solve depends on the characteristics of the output data expected from the model. There are algorithms that are trained for making a regression such as predictive control in robotics. Other problem in ML is classification, in which the machine executes a categorization from the input data. In this case, an example could be to identify the sex of a website user from their track on website. Another output that a ML algorithm can implement is clustering for finding tendencies on the input data. Dimensionally reduction is an output, in which the machine would reduce the input dimensions of the data to a smaller number of the dimensions, but maintaining the information enclosed in the data by reflexing the statistical properties of the input on the output dataset.

3.1.2 Optimization Methods

A different classification on ML can be done from the perspective of the function criteria that the algorithm uses to approximate the output along the training of the machine. In this case, the techniques can be categorized in three algorithms which are Maximum Margin (MM), Maximum Likelihood Estimator (MLE) or Minimum Mean Square Error (MMSE).

On the case of ML algorithms such as SVM and Boosting the criteria for training the machine is to maximize the margin. For instance, on SVM [31] the algorithm aims to find the hyperplane which have maximum distance between the data points inside an established margin. On the other hand, Boosting algorithm [32] attempts to ding a good linear combination of the members of basis functions to optimize a given loss function. This is done iteratively for the basis function which gives the steepest descent in the loss function, and changing its coefficients accordantly. In cases of other ML algorithms such as Deep Learning and Gaussian Process, the

technique used for implementing an output from a model is the MLE. The aim of this method is to estimate the parameters of a statistical model given an input data, this is done by selecting the parameters for the model which maximize the likelihood function. Indeed, they maximize the probability of the observed input data under the resulting distribution. MLE gives a unified approach to an estimation, which is well-defined in the cases of normal distribution and many other problems. In a GP [30], is maximized the likelihood with respect to the hyper-parameters of the model to estimates a posterior which combines information from the prior and the data. A variation of MLE is described on [33], where the Contrastive Divergence (CD) is used as an approximation MLE algorithm for DL. In this case the parameters of the model are learned by maximizing the probability of the input data or equivalently by minimizing the negative log-likelihood of the probabilities of the data. Another criteria in a learning algorithm is the MMSE, which is used in ML algorithms such as Multi-Layer Perceptron (MLP) or Ridge Regression (RR). The MLP is a feed-forward ANN composed by multiple neuronal layers which is fully interconnected between them. The neurons are connected by an activation input function for mapping the output of each neuron from its inputs. The learning algorithm is the well-known backpropagation, which aims to minimize the mean square error produced by the output of the last neuron of the MLP and its training target. The RR is estimator that approximates an output by adjusting the ridge coefficient minimizing the mean square error in the estimation.

3.2 Deep Learning

The algorithm of DL can be classified inside the Artificial Neural Networks because of its multi-layer structure formed by input neurons and output neurones. These multi-layer neural networks are a class of ML algorithm, built by attaching multiple

layers together in such a way that they form an unique machine. The methodology used in DL is to sequence independent machines, in which the output of one layer is the input of the next layer and successively.

3.2.1 Introduction

In the DL algorithm, each layer is trained independently so its internal parameters are adjusted in a way that the model is able to retain the statistical properties of the input nodes. The algorithm essentially generates a representation in the output, by keeping the information contained in the input data, but changing the its dimensionality. The independent layers in DL, are called Restricted Boltzmann Machines (RBM) and they are trained by the well-known algorithm of Contrastive Divergence [34].

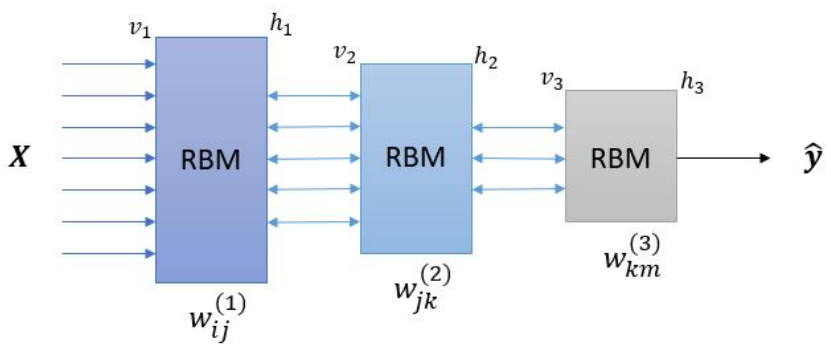

Figure 3.1: DL composed by several RBMs attached together forming a Deep Belief Network.

The most distinctive feature of the DL algorithm is that it is able to discover patters and structures in large input data sets [35]. The optimization algorithm used for training the networks is the Backpropagation. It basically indicates how the internal parameters, in each RBM that composed the network, should change in

order to obtain an output more representative of the input. In this time, from the entire network of attached RBMs.

Each one of the RBMs which compose the network, has a determined number of input nodes and output nodes. Each input node has as input feature from the dataset. It can be reworded as the output of the RBM changes the dimensionality, or the number of features of an input sample of data.

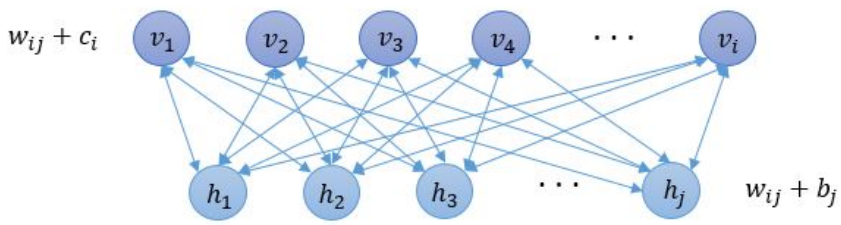

Figure 3.2: Neuron connections between visible (v_i) and hidden (h_j) units in a single RBM.

From now on, the features of the input data are going to be called visible units and they will be denoted by v. The output data from a layer is going to be defined as the hidden units and indicated by an h.

On the implementation of DL, each RBM is trained independent and the joint configuration $(\mathbf{v}, \mathbf{h})$ of the visible and hidden units has an Energy function such as:

$$E(\mathbf{v}, \mathbf{h}) = \sum_{i \in visible} c_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{3.1}$$

this equation is independent for each one of the RBMs that forms the machine. Where v_i and h_j are the states of the visible unit i and the hidden unit j, c_i and b_j, are the visible and hidden biases respectively, and w_{ij} are the weights between visible and hidden units. By computing the Energy, the network assigns a probability to a possible state that a neuron can have [34].

On a RBM the probabilities to every possible join between pairs of visible and hidden units are given by computing the previous stated Energy function as follows:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp\left(-E(\mathbf{v}, \mathbf{h})\right) \tag{3.2}$$

where Z is the summation of all the possible joins between pairs of visible and hidden units on the RBM so that:

$$Z = \sum \exp\left(-E(\mathbf{v}, \mathbf{h})\right) \tag{3.3}$$

The probabilities of each visible units on a RBM, are assigned to the vector v. This is given by simply summation all the possible hidden states:

$$p(\mathbf{v}) = \frac{1}{Z} \sum_{hidden} \exp\left(-E(\mathbf{v}, \mathbf{h})\right) \tag{3.4}$$

The probability given to an input sample of training data can be raised o decreased by adjusting the weights and bias of the connections in the network. Hence, raising the energy of a particular input node will decrease the energy of other input nodes. This methodology is also used in the backpropagation and it will be explained later on this section.

Considering the particular case in which each node of a network has a binary state, and their activation is defined by a logistic function. For a given a training sample, each binary state h_j in the output node, will be set to 1 with a probability:

$$p(h_j = 1|\mathbf{v}) = \sigma\left(b_j + \sum_i v_i w_{ij}\right) \tag{3.5}$$

where $\sigma(\mathbf{x})$ is the logistic function:

$$\sigma(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{x}}} \tag{3.6}$$

and where $\mathbf{x}$ is the previously stated:

$$\mathbf{x} = b_j + \sum_i v_i w_{ij} \tag{3.7}$$

3.2.2 Training a Restricted Boltzmann Machine

From the earliest days of pattern recognition, the aim of researches has been to replace hand-engineered features with trainable multi-layer networks, but despite its simplicity, the solution was not widely understood until on the mid 1980s, when it turns out, that multi-layer architectures can be trained by simply stochastic gradient descent.

CD is a procedure used for training DL machines introduced in [34]. It is essentially an approximation of the gradient of the log-likelihood based on a model of short Markov Chain between probabilities of the hidden units, given the visible units of the model and vice versa, the visible unites given the hidden units. The optimization of the log-likelihood is the learning criteria. This method is also used in many other probabilistic ML algorithms. The implementation of this optimization algorithm consists in initializing the Markov Chain from a distribution which is expected to be closed to the desired. In this case, from the distribution of a training sample of the input data.

In the case of training a single RBM, the gradient of the log-likelihood of the input vector, which are the visible units, with respect to the weights of the connections of

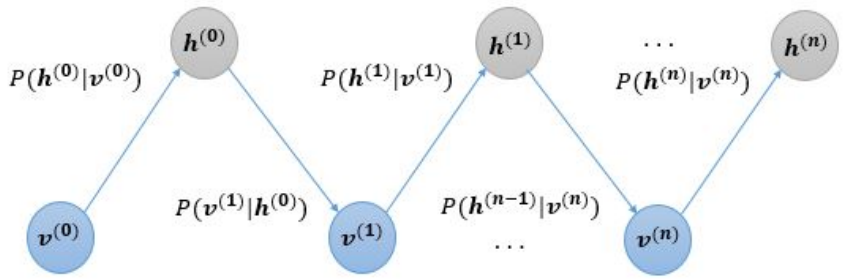

Figure 3.3: Gibbs Sampling steps updating the parameters of the model by iteratively sampling the $p(\mathbf{h}^{(n)}|\mathbf{v}^{(n)})$ and construct from them $p(\mathbf{h}^{(n+1)}|\mathbf{v}^{(n+1)})$, contrasting their divergence.

the network is:

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \mathbb{E}[v_i h_j]_{data} - \mathbb{E}[v_i h_j]_{reconstruction} \tag{3.8}$$

In the case of the biases of the network, the gradient would be stated as follows:

$$\frac{\partial \log p(\mathbf{v})}{\partial b_j} = v_j - \sum_{visible} p(\mathbf{v}) v_j \tag{3.9}$$

$$\frac{\partial \log p(\mathbf{v})}{\partial c_i} = p(h_j = 1|\mathbf{v}) - \sum_{visible} p(\mathbf{v}) p(h_j = 1|\mathbf{v}) \tag{3.10}$$

where $\mathbb{E}$ denote the expectations under the distributions. The probabilities of the hidden units are computed on the forward or positive phase of the CD. For an input sample of the data, that is $\mathbf{v}$, for the binary state h_j of each hidden units j to be equal to 1 have a probability given by:

$$p(h_j = 1|\mathbf{v}) = \sigma\left(\sum_i v_i w_{ij} + b_j\right) \tag{3.11}$$

from the probability of a hidden unit to be 1 is sampled the $\mathbb{E}[v_i h_j]_{data}$. In the reconstruction of the visible unites, that is sometimes called backwards or negative phase, the probabilities are given by:

$$p(v_i = 1|\mathbf{h}) = \sigma\left(\sum_j h_j w_{ij} + c_i\right) \tag{3.12}$$

and from the probabilities of a visible unit to be 1, it is sample the $\mathbb{E}[v_i h_j]_{reconstr}$.

The final step on the formulation of the gradient, it is just to update the parameters of the probabilistic model of an independent RBM. The new sates of the hidden unites are modified interactively along a determined number of epochs and the change of the weights is explicitly given by:

$$\Delta W_{ij} = \epsilon\left(\mathbb{E}[v_i h_j]_{data} - \mathbb{E}[v_i, h_j]_{reconstr}\right) \tag{3.13}$$

where ϵ is the learning rate of the parameters, $\mathbb{E}[v_i h_j]_{data}$ is the fraction of times that input neuron i and the output neuron j are on together driven by the initial sample of data, and $\mathbb{E}[v_i h_j]_{reconstr}$ is the corresponding fraction of asymmetry on the reflexing into the input data set. A simply formulation for the same learning rate is used to update the bias of the network [34].

3.2.3 Training a Deep Belief Network

The Backpropagation algorithm computes the gradient of a cost function and its goal is to minimize its output. It can be referred as objective function [35]. The derivatives are computed with respect to the parameters of the model of each RBM attached together on a multi-layer network. In contrast to the independent training

method of an RBM on the DL, Backpropagation updates the parameters of the model using the chain rule of the derivatives.

This method update interactively the parameters on all ensemble layers on the network, an increment or change on the objective function is propagated backwards throughout all the network. In this way, the derivative of the network's objective function, which is its the error on the output, can be computed from the gradient with respect to the parameters.

Minimization of the Regression Error

In the case of a network configured for regression, the parameters of the different layers are updated in order to match an objective function, which in this case is the error function. The MMSE error of the regression is of the output of the network that is aimed to minimize. It is given by this following equation:

$$E = \frac{1}{2}\sum_{i=1}^{N}\left(|\theta_i - t_i|\right)^2 \tag{3.14}$$

where N is the number of samples on the input vector, θ_i are the outputs of the network and t_i are the labels or target of the regression at each time instant i of the input data set [36].

The parameters of the model, which are the weights and bias, are the only that can be modified in order to make the error as low as possible. Therefore, E is minimized by using an iterative process of gradient descent, for which the gradient is calculated as follows:

$$\bigtriangledown E = \left(\frac{\partial E}{\partial \mathbf{W}^{(1)}}\frac{\partial E}{\partial \mathbf{W}^{(2)}}\cdots\frac{\partial E}{\partial \mathbf{W}^{(\ell)}}\right) \tag{3.15}$$

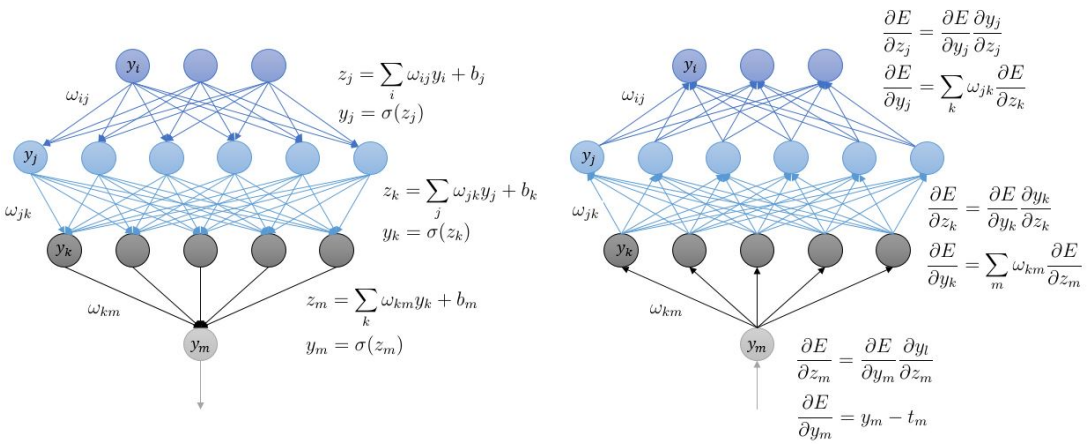

Figure 3.4: Forward phase equations and backwards equation from chain rule of the derivatives.

where $\mathbf{W}$ represents the weights matrix from each one of the ℓ layers of the network.

The Backpropagation algorithm is basically used to find some local minimum in the error function. However, it is not guarantee to find a global minimum. The way in which the gradient of the error function is computed and used to correct the initial weights is such as:

$$\Delta \mathbf{W}_\ell = -\gamma \frac{\partial E}{\partial \mathbf{W}_\ell} \tag{3.16}$$

for each one of the layers ℓ, according to the learning rate γ.

Minimization of the Reconstruction Error

A particularization of the Backpropagation algorithm is called as fine tuning of the weights on [34]. In this case, the algorithm is used for computing the reconstruction

error, and to propagate this error throughout the network. The method used is the gradient conjugate.

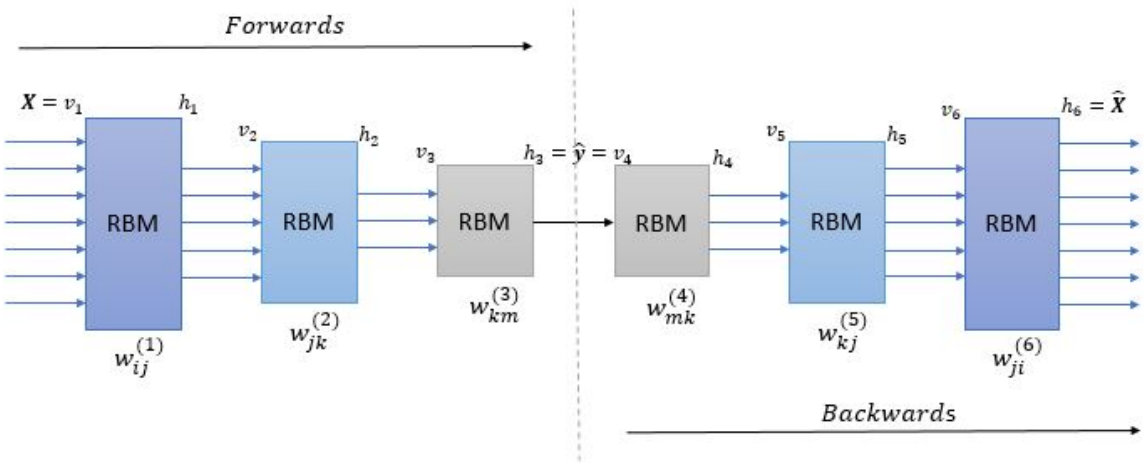

Figure 3.5: Training sequence on a DBN using Backpropagation. This figure shows how on first training, propagates input forwards throughout the network, and reconstructs the output backwards. Propagating the error on the reconstruction on the output.

The error function, which is aimed to minimized, is the reconstruction error instead of the error on the regression. On this algorithm, it is necessary to compute first the reconstruction of the input from the obtained output of the network In this way,it can be computed the reconstruction error of data. Once this objective error function is computed, the error is propagated forward throughout the network updating the parameters of the model at each one of the layers, so there the error is minimized. In this case, the error function is defined by:

$$E = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{D}\left(\hat{X}_{ij} - X_{ij}\right)^2 \tag{3.17}$$

where N are the total number of samples i on the input data and D the number of features or dimensions j on the data set. $Xout_{ij}$ represent the output from the

model of ensemble layers or RBMs and Xin_{ij} is simply the input data.

In this case of Backpropagation algorithm, the visible biases are updated on the backwards phase of the computation and the hidden biases are when the process goes forward once again. The weights are updated on the backwards and forward phase of the computations.

3.2.4 Activation Functions

DL is multi-layer network in which each layer is defined as an independent RBM where the output of one them is the input of the following one. In these networks, either input as output are independent neurons connected between the previous units. However, units on the same level are not connected between them. The activation function of each neuron defines the probability of the states on the output neurons. On this chapter, the DL was explained for the case of binary states on either input and output neurons that compose a layer. This it is just an example of type of input state activated by a particular function, the sigmoid. On this part of the chapter, there are introduced several activation functions that have been implemented, and that they improve significantly the performances of the algorithm for some particular case-scenario.

Logistic Function

For a given training sample, the binaries states of the hidden units are set on with a probability defined by the logistic sigmoid function of its input x_{ij} such as:

$$\sigma(x_{ij}) = \frac{1}{1 + e^{-x_{ij}}} \tag{3.18}$$

where x_{ij} are the weighed inputs of the neuron such as $v_i w_{ij} + b_j$, v_i are each i visible unit of the layer and b_j each visible bias from an output neuron j. From now on x_{ij} is defined in this way for each one of the units explained later in this chapter.

Gaussian Function

The fact of applying a Gaussian function to the input data, or visible units, of a layer produces continuity on the input. This characteristic can be useful in some cases, it depends on the nature of the data. In contrast, it can also produce a poor representation of the input data, in some other cases. A solution to that can be to substitute the binary output units by linear units with intended Gaussian noise [37].

$$f(x_{ij}) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(\frac{-(x_{ij} - \mu)^2}{2\sigma^2}\right) \tag{3.19}$$

where μ and σ are the mean and the standard deviation of the input data set. The Energy function of the layer varies when the activation function changes. In the case of a Gaussian activation function on the visible units and a sigmoid activation function on the hidden units the Energy function is stated as follows:

$$E(\mathbf{v}, \mathbf{h}) = \sum_{i \in visible} \frac{v_i - c_i}{2\sigma_i^2} - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i \sigma_i h_j w_{ij} \tag{3.20}$$

when visible and hidden unites are activated by a Gaussian function the instability problems increase considerably, as it is referred on [37]. The individual activities are retained around the means by quadratic terms by coefficients determined by the standard deviation. In the case of layers in which both visible and hidden units are activated by a Gaussian function, the formulation of the Energy function is such as:

$$E(\mathbf{v}, \mathbf{h}) = \sum_{i \in visible} \frac{v_i - c_i}{2\sigma_i^2} - \sum_{j \in hidden} \frac{h_j - b_j}{2\sigma_j^2} - \sum_{i,j} \frac{v_i}{\sigma_i} \frac{h_j}{\sigma_j} w_{ij} \tag{3.21}$$

Hyperbolic Tangent

The hyperbolic tangent function is a rescaled version of the logistic sigmoid function to an output on the rage of $[-1, 1]$, instead of the sigmoid's output range of $[0, 1]$. This range can be interesting in particular when the input data contends relevant information under the range of 0. On this case a sigmoid function saturates to 0 its output belong that limit. In contrast, a *tanh* function maintains the information inside its range. The hyperbolic tangent is defined by:

$$f(x_{ij}) = \frac{e^{x_{ij}} - e^{-x_{ij}}}{e^{x_{ij}} + e^{-x_{ij}}} \tag{3.22}$$

Rectified Function

The Rectified Linear Units (ReLU) are defined on [37] and [35]. Its main property is that can be used as mathematical half-wave rectifier given by this expression:

$$f(x_{ij}) = max(x_{ij}, 0) \tag{3.23}$$

ReLU have typically the capacity of learning faster on a machine composed by many multiple layers or RBMs on a DL algorithm. On [35] is stated that this characteristic allows to train deep supervised networks without needing unsupervised pre-training of the network. The units at the output of a layer, can be seen as distorting the image from the input in a non-linear way. This is particular useful

on the last layer of the network because it can be used for saturating the output in a way that eases the categorical differentiation between outputs from inputs on the network.

Soft-sign Function

The soft-sign function found its application on DL on an experiment carried out on [38]. This function has similar properties to the tangent hyperbolic, both are inside the range of $[-1, 1]$ and outside that rage both of the functions saturate. Their differences are essentially that soft-sign function has quadratic polynomials as tails instead of exponentials, therefore the approaches to their asymptotes much slower. Its formulation is:

$$f(x_{ij}) = \frac{x_{ij}}{1 + |x_{ij}|} \tag{3.24}$$

Binomial units

Binomial units are essentially as the binary units produced by a sigmoid function previously introduced. The particularity of the units, which were mentioned on [37], are such as their output is modified by introducing an offset value. This fact can produce more interesting neuron models closest to a actual neuron. Furthermore, its implementation can produces more useful results in some cases. This is achieved by adding a constant and a variable to each input so their binary probabilities are biased. A generalization of the offset value introduced on a sigmoid function is $-(N - 0.5)$ where N is the offset value for a variable and 0.5 is a constant offset value for the dataset. The general form for the equation would be:

$$f(x_{ij}) = \sum_{i=1} \sigma(x_{ij} - k + 0.5) \approx \log(1 + e^{x_{ij}}) \tag{3.25}$$

3.3 Gaussian Processes

GP are non-parametric predictive models in which every observation is associated with a normally distributed stochastic process. It produces a prediction for an unseen data point in a high-dimension space. The distribution of a Gaussian process is the join distribution of all the variables that define a data point.

The main advantage of using GP between other methods is that just the hyper-parameters of the covariance matrix needs to be optimized. In this ML technique, the optimization criteria used is the Maximum Likelihood. This algorithm adjusts the hyper-parameters of the model in such a way that they maximize the likelihood under a training dataset. All the parameters are initialized randomly and the algorithm is able to find a minimum for a given set of hyper-parameters. This is an iterative process and these minima search line is done by Conjugate Gradient. However, this approach is very sensible to find some local minima. Therefore, it requires different initializations in order to find a better solution.

3.3.1 Linear Model

A theoretical demonstration for a general case of GP is developed below for a given function or estimator [39] such as:

$$y_i = f(x_i) + \epsilon_i, \qquad f(x_i) = \mathbf{w}^T x_i \tag{3.26}$$

where ϵ_i is an estimation for the error and y_i is the regression obtained for each value of $x_i \in \mathbb{R}^D$, which are the observations. The error is assumed to be Gaussian with the following distribution:

$$\epsilon_i \sim \mathcal{N}(0, \sigma_n^2) \tag{3.27}$$

If y_i is defined as a GP, its mean is given by $f(x)$ and its variance is σ_n^2. The probability distribution of a sample of y_i for given observation x_i and a set of parameters $\mathbf{w}$, is computed for a normal distribution such as:

$$p(y_i|x_i, \mathbf{w}) = \frac{1}{\sqrt{2\pi\sigma_n}} exp\left(-\frac{|y_i - \mathbf{w}^T x_i|^2}{2\sigma_n^2}\right) \tag{3.28}$$

and the join process for this normal distribution can be computed from:

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{N} p(y_i|x_i, \mathbf{w}) \tag{3.29}$$

The parameters of the model $\mathbf{w}$ are a linear combination of each pair of elements x_i and y_i. Hence, the distribution of the $\mathbf{w}$ is a process which depends only on $\mathbf{X}$ and $\mathbf{y}$. In this way, the prior distribution of the model parameters is given by the following general equation for normal distribution of D dimensions:

$$\mathbf{w} \sim \mathcal{N}(0, \mathbf{\Sigma_p}) = \frac{1}{(2\pi|\mathbf{\Sigma_p}|)^{(D+1)/2}} exp\left(-\frac{1}{2}\mathbf{w}^T \mathbf{\Sigma_p^{-1}} \mathbf{w}\right) \tag{3.30}$$

where $\mathbf{\Sigma_p}$ is the covariance matrix of the process and $\mathbf{w}$ are the parameters of the process.

According to the Bayes Theorem is straight forward that the posterior distribution of the process with respect to $\mathbf{X}$ and $\mathbf{y}$ is stated as follows:

$$p(\mathbf{w}|\mathbf{X},\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})} \tag{3.31}$$

For this process, the probabilities on the denominator does not depends on the parameters of the model, and as it is simply aimed to maximize the distribution of the posterior, the Bayes Theorem can be simplified to this form:

$$p(\mathbf{w}|\mathbf{y},\mathbf{X}) \propto p(\mathbf{y}|\mathbf{X},\mathbf{w})p(\mathbf{w}) \tag{3.32}$$

The marginal likelihood of the process, which acts simply as a normalization constant because it independent to the model's weights, is given by the following integral:

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{w})p(\mathbf{y}|\mathbf{X},\mathbf{w})\partial\mathbf{w} \tag{3.33}$$

The final form of the posterior distribution is the product of two normal probability distributions. The prior and the parameters normal distribution. Hence, the posterior is also a normal probability distribution defined by this product:

$$\begin{aligned} p(\mathbf{w}|\mathbf{y},\mathbf{X}) &\propto \exp\left(-\frac{\left|\mathbf{y}-\mathbf{X^T}\mathbf{w}\right|^2}{2\sigma_n^2}\right)\exp\left(-\frac{1}{2}\mathbf{w^T}\mathbf{\Sigma_p^{-1}}\mathbf{w}\right) \\ &\propto \exp\left(-\frac{1}{2}(\mathbf{w}-\bar{\mathbf{w}})^T\left(\frac{1}{\sigma_2^2}\mathbf{X}\mathbf{X^T}+\mathbf{\Sigma_p^{-1}}\right)(\mathbf{w}-\bar{\mathbf{w}})\right) \end{aligned} \tag{3.34}$$

where $\mathbf{A} = \sigma_{\mathbf{n}}^{-\mathbf{2}}\mathbf{X}\mathbf{X}^{\mathbf{T}} + \mathbf{\Sigma}_{\mathbf{p}}^{-\mathbf{1}}$, and $\bar{\mathbf{w}}$ is the mean of the parameters $\mathbf{w}$. That it is defined as $\bar{\mathbf{w}} = \sigma_n^{-2}\mathbf{A}^{-\mathbf{1}}\mathbf{X}\mathbf{y}$. The final form for the Gaussian posterior is like:

$$p(\mathbf{w}|\mathbf{y}, \mathbf{X}) \sim \mathcal{N}\left(\frac{1}{\sigma_n^2}\mathbf{A}^{-\mathbf{1}}\mathbf{X}\mathbf{y}, \mathbf{A}^{-\mathbf{1}}\right) \tag{3.35}$$

The Maximization of the Likelihood over the posterior distribution finally defines the optimum parameters for the GP. This is done by Conjugate Gradient line search and it is stated as follows:

$$\mathbf{w}_{ML} = \arg\max_{\mathbf{w}} p(\mathbf{w}|\mathbf{y}, \mathbf{X}) \tag{3.36}$$

In order to make predictions for a test case, we average over all possible parameters values. weighted by their posterior portability. For a new sample $\mathbf{x}^*$, which does not belong to the training dataset. The final regression model will output a prediction such as:

$$y^* = f_*(\mathbf{x}^*) + \epsilon^* \qquad\qquad f_*(\mathbf{x}^*) = \mathbf{w}^T\mathbf{x}^* \tag{3.37}$$

with a probability defined by $p(f_*|\mathbf{x}^*, \mathbf{w})$. The predictive probability distribution of the posterior is given by averaging the output of all possible linear models with respect to the Gaussian posterior, computing following integral:

$$\begin{aligned} p(f_*|\mathbf{x}^*, \mathbf{X}, \mathbf{y}) &= \int p(f_*|\mathbf{x}_*, \mathbf{w})p(\mathbf{w}|\mathbf{X}, \mathbf{y})\partial\mathbf{w} \\ &= \mathcal{N}\left(\frac{1}{\sigma_n^2}\mathbf{x}_*^T\mathbf{A}^{-\mathbf{1}}\mathbf{X}\mathbf{y}, \mathbf{x}_*^T\mathbf{A}^{-\mathbf{1}}\mathbf{x}_*\right) \end{aligned} \tag{3.38}$$

3.3.2 Model on Feature Space

Another but equivalent method that conducts to similar results is to do inference in a space defined by functions. This function $\phi(\mathbf{x})$, maps the input vector to a N dimensional feature space. Therefore, the model is stated on a features space as:

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w} \tag{3.39}$$

where $\mathbf{x}$ is the vector which contains a number of N observations.

The development of the model is done in the same way that for a liner model. The only different is that now, instead of having a matrix input $\mathbf{X}$, we have the function $\Phi(\mathbf{X})$, which represent the mapping of the matrix $\mathbf{X}$ to a new feature space. The final predictive distribution on a features space, now is given by the equivalent expression:

$$p(f_*|\phi(\mathbf{x}^*), \Phi(\mathbf{X}), \mathbf{y}) \sim \mathcal{N}\left(\frac{1}{\sigma_n^2}\phi(\mathbf{x}_*^T)A^{-1}\Phi(\mathbf{X})\mathbf{y}, \phi(\mathbf{x}_*^T)A^{-1}\phi(\mathbf{x}_*)\right) \tag{3.40}$$

with $\mathbf{A} = \sigma_n^{-2}\Phi(\mathbf{X})\Phi(\mathbf{X})^T + \mathbf{\Sigma}_p^{-1}$. The final dimensions of A are $N \times N$. As N is the number of observations in the input vector $\mathbf{x}$, the computation of A^{-1} for obtaining a predicted output will be very difficult for large datasets.

A GP can also describe a distribution over functions. In this case, the GP is specified fully by the mean $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ of the process $f(\mathbf{x})$, such as:

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[\left(f(\mathbf{x}) - m(\mathbf{x})\right)\left(f(\mathbf{x}' - m(\mathbf{x}'))\right)\right] \end{aligned} \tag{3.41}$$

the definition for a GP is such as:

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right) \tag{3.42}$$

This is just a generalization of a Gaussian distribution whose mean and covariance functions are vector and matrix respectively.

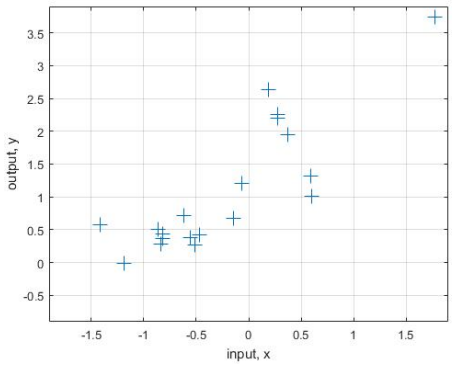

Figure 3.6: Example of GP for regression. 2 Dimensions data distributed on the space.

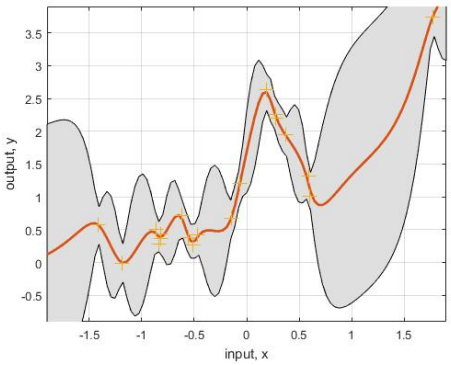

Figure 3.7: Iteration 1 of MLE for finding the model's w, confidence interval in shadowed.

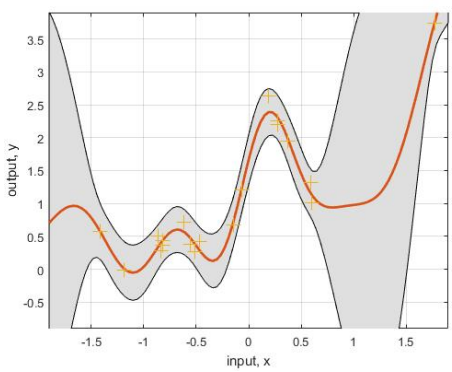

Figure 3.8: Iteration 2 of MLE, model confidence interval fitting.

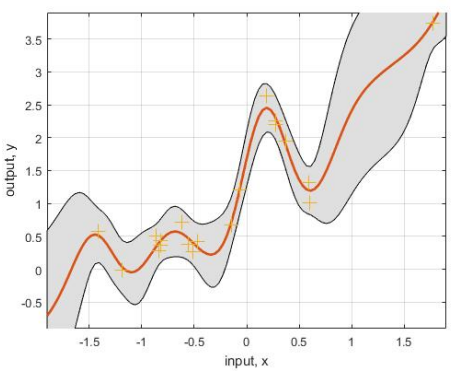

Figure 3.9: Iteration 3 of MLE, model confidence interval matching tightly.

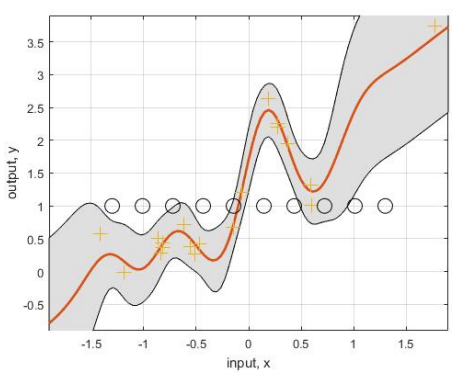

Figure 3.10: End of computing, optimal w. These captures belong to the example code on [1].

Chapter 4

Experiments

This chapter shows experiments base on the ML algorithms previously introduced on the chapter 3. The results from the tested of two DL structures, were compared with the results obtained from a SVM. The DL structure was similar to used on the SVM. The objective of the experiments is to research whether a pre-training on the weights that compose the DL produces better results on the prediction.

4.1 Weather Data

The model used is a Weather Research and Forecasting (WRF) meso-scale version 3.6. WRF is a powerful meso-scale numerical weather prediction system designed for atmospheric research and also for operating forecasting models. The model was developed in collaboration of several institutions in the USA, the National Center for Atmospheric Research, the National Center for Environmental Prediction, the Forecast Systems Laboratory, the Air Force Weather Agency, the Naval Research Laboratory, the University of Oklahoma, and the Federal Aviation Administration. This weather model has been used previously for weather forecasting in the article

[40]. Additionally, it has been implemented successfully with the propose of power output forecasting in the context of SG and RE on [41].

WRF version 3.6 runs every 12 hours since being commissioned in 2011. The meteorological data is collected from several nodes sprawled over an are that goes from the latitude N34°33′43″ to N44°28′12″ and from the longitude W4°25′12″ to E4°23′2″. The atmospheric values are measured from sensors vertically localized at 37 different levels above ground. Also at the ground, and on four levels under the ground. This weather station is located in Toledo, Spain.

The aim for the features contended within the dataset is to train and test the algorithms that were previously introduced in this document. This weather station in Toledo belongs to the *"Agencia Española de Meteorología (AEMET)"* and it is exactly located at (N39°53′5″, W4°2′45″) on the Spanish territory at 515 meters above the sea level. The dataset contains weather features collected over a complete year with sampling resolution of an hour. The beginning of the data is May 1st 2013 and ends on April 20th 2014. This is the dataset used to carry out the experiments measuring the performance of the different predictive models in this document.

The data collected from the 25 nodes of the meso-scale model forms the input dataset for the experiments. Each one of the nodes sense 423 physical features of the weather. Among other variables, the set includes wind speed and air temperature, measured at several altitudes; plus other variables such as atmospheric long-wave solar radiation, cloud covering, air humidity, etc. These variables together form a $25x423$ matrix of sensor measurement, and there are 5,840 samples along a year.

On the data processing task, the 3-dimensional matrix (height, feature and time) was reshaped into a 2-dimensions matrix, which contains all measurements for each time-instant. Hence, the final dimension of the input matrix is $5,840 \times 10,575$. For regularization of the parameters contained in the matrix, the mean of the measure-

ments was calculated for each dimension, and subtracted from all its samples on the dataset. The matrix contains some parameters that are irrelevant for the propose of the regression. That is because its magnitude remains constant along all the sampling length. The standard deviation was calculated for each one of the different variables. The variables with 0 standard deviation were deleted from the set, in order to alleviate the computational load. After the pre-processing, the final dimensions of the input matrix was $5,840x9,864$, which it is a 7% reduction from initial matrix.

4.2 Structures

The experiments aim to study the performances of two different DL structures for performing a independent regressions but using the same dataset. The DL tested is a 3 layer structure. These layers are machines trained independently, the RBMs previously mentioned. Before training the structure with the goal of regression, the weights of each RBM are pre-trained. There are used two different cascaded methods. These methods are Contrastive Divergence and minimization of MSE reconstruction error by backwards propagation of the error through the network.

The training of the DL structures for estimating a prediction, is done by using two different methods. The first structure is trained by Backpropagation algorithm in order to adjust the pre-trained weights to produces a regression. On the experiments, the RBMs that compose the DL structures, have binary visible and hidden units in their layers with exception of the last layer. These hidden units, are generated by a liner function. On the second structure, a GP substitutes the last RBM, so the 2 RBM have also linear units. In this way, the GP is independently trained having as input the output units generated by the RBM. The covariance function using on the GP is a square exponential Kernel. This function is introduced on an appendix.

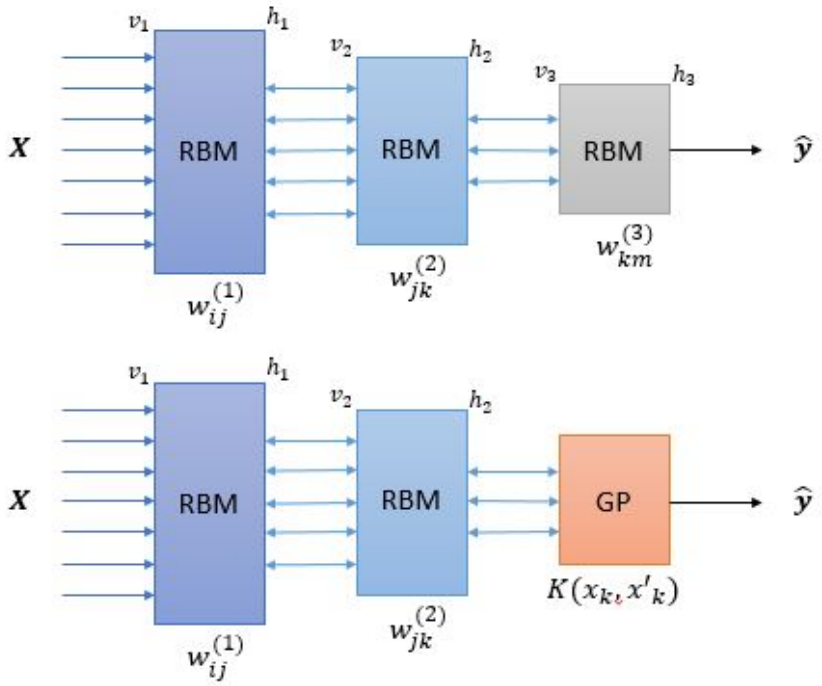

Figure 4.1: The figure shows both of the structures experiment on this document. The first one is a DL composed by 3 RBMs. The second one is a DL with GP on its last layer, instead of a RBM.

4.3 Methodology for the Experiments

The first set of experiments, consists on testing the pre-training methods on both structures. On the first structure has 3 RBM that are trained independently. In contrast, the second structure have only 2 RBMS that need to be trained, because its last layers is GP.

The experiment for the pre-training consists of performing 10 different initialization on DL structure. First, a pre-training using CD, followed by a seconds pre-training by a minimization of the reconstruction error by propagating forwards and backwards the error. The samples collected from the experiments, are split into two subsets. On one hand, the subset of parameters trained by CD. On the other

hand, the subset of parameters trained by both methods CD and minimizing the reconstruction error. The objective of these experiments, is to discover which one of the pre-training methods perform better results, when the aim is to produce an estimation. From the collected samples, it is selected the one that hassmallest reconstruction error.

The method for the pre-traning by CD, is runs during 10 epochs on each RBMs. In the case of the pre-traning by minimization of the reconstruction error, the algorithm runs iteratively until that the reconstruction error is constant along 10 epochs. This is done for reducing the computing time. Although, number of iterations could be extended until that the reconstruction error on the validation dataset starts to increase. As we consider that the change on the weights is insignificant from the standstill to the increasing, the criteria used for stopping the computation is that the reconstruction error is constant during 10.

Onces a sample is selected from the pre-traning methods tested, the next step is to experiment the regression error on the DL structures. The method follows in this experiment, is to compute 10 different initialization with seasonal disorder applied to the data. The criteria to select a model, is the best R^2 on these 10 initializations.

On the implementation, the data have been split into 3 different samples. This data have been previously disordered in order to dismount the seasonal component on the data. The GSR follows a trend that match with the seasons of a year. The objective of the disorder, is to develop a unique model for forecasting GSR valid for any day of a year.

The partitions of the data are: train, test and validation dataset. The training subset have been only used to train the machines that compose the structures. The test subset have been used for testing the final performances of each machine. The subset of validation have been used to check the performances of the machine along

their training. This subset is basically used to control the training, and thus to know when to stop the training. Depending on the type of learning of the machine, the subsets of data needed are different.

	Pre-training		Regression	
Data — Method	**CD**	**Recons. Error**	**BP**	**GP**
Training	X	X	X	X
Test	X	X	X	X
Validation		X	X	

Table 4.1: This table shows, for what have been used each one of the 3 different subsets, on the algorithms implemented.

4.4 Results

The experiments described, are been tested for 5 different configurations on the structure. The configuration are basically variation on the number of hidden neurones at the output of the RBMs. The aim is to explore what is the number of hidden units have a positive impact on algorithm for regression.

The training of the machine have been done and tested with different subsets of data. The values to measure the performance, are MSE for the reconstruction error, and MSE and R^2 for the regression. The MSE on the reconstruction, is a quality value commonly used on pre-training algorithms, its expression is as follows:

$$MSE = \frac{1}{ND}\sum_{i=1}^{N}\sum_{j=1}^{D}(\hat{x}_{ij} - x_{ij})^2 \tag{4.1}$$

where N and D are the number of samples and the dimensions of the input data. x_{ij} is the observations on the data, and its reconstruction from the output is $\hat{x}_{ij}$. The subindex i and j denote the element on the dataset and its reconstruction.

On the training of the regression, the criteria on the optimization is to minimize the R^2 to produce on the validation. The value of MSE has been computed as informative parameter for evaluating the quality of the regression. The MSE is described such as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{4.2}$$

the number of samples is denoted by N, The predicted output is expressed by $\hat{y})i$ and a training target value used for the prediction is y_i.

These parameters evaluate the performances, and also they give a criteria to know when an algorithm have been trained. In this way, the model generated will forecast a time instant with small error. The coefficient is the R^2, and it is computed such a:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2} \tag{4.3}$$

in this expression, the prediction is denote as $\hat{y}_i$, and the target is y_i. N is the number of samples, and $\bar{y}_i$ is the mean of the target.

The results of the experiments are compared with a SVM. The DL structure that compose the SVM has 2 layers, as on the GP. Therefore, the DL is used again for reducing the dimensionality of the data. The results are shown on R^2 values. The hidden node on the RBMs are not the same, but the results are just orientation, to visualize how this other structure performance essentially. This SVM is not trained using the minimization of the reconstruction error on the per-traning.

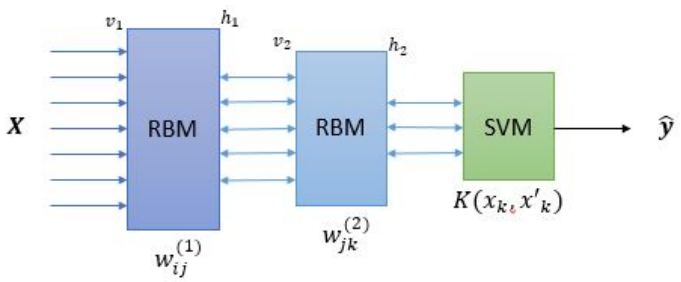

Figure 4.2: DL structure with a SVM on its layer.

4.4.1 Configuration I

The configuration experimented on these structures, is 9864 visible nodes and 2500 hidden nodes. The second layer has 2500 nodes at the input and 50 nodes on the output. The last layer reduces the dimensions from 50 to 1.

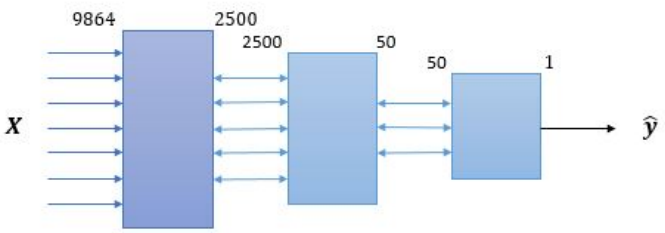

Figure 4.3: Structure experimented: c

The training of the DL is done during 10 epochs. DL has de characteristic of reducing the dimensionality of the data. This is done by calculating the correlation between visible and hidden units. Therefore, the weights on the model will have higher value for the variables that have also high correlation on the dataset.

The output of the first layer will extract the information on most important variables on the data. From the figures, can be seen that two strips on the weights' model are starting to show up. The value on the weights that are more informative increases as the machine is over trained.

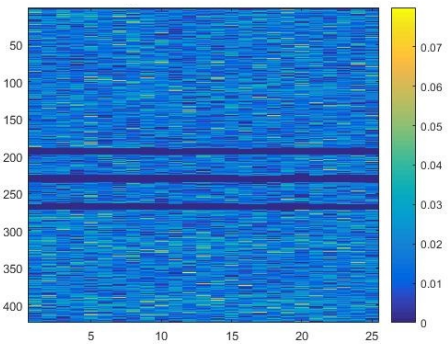

Figure 4.4: Map of the weights of the first RBM for 10 epochs.

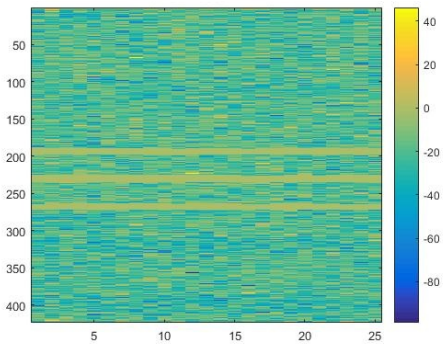

Figure 4.5: Map of t-student compute from the weights' map.

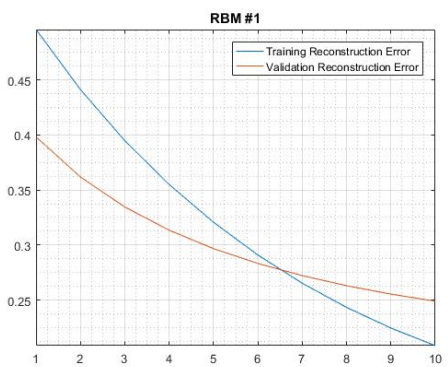

Figure 4.6: Reconstruction error on the training of the first RBM.

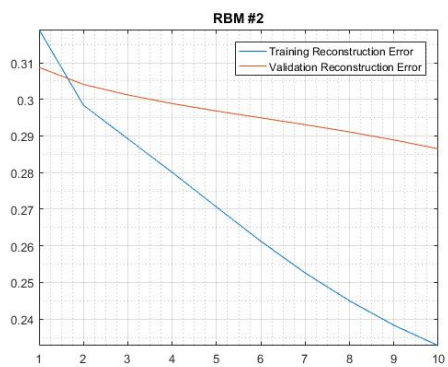

Figure 4.7: Evolution of the error along the training of the second RBM.

Along the CD the training is monitored at each initialization. On the structure attached to a GP, the weights trained on the third RBM are not used. The best MSE reconstruction error obtained after pre-traning by CD, is 0.1180.

The after the pre-traning by CD, start the pre-traning to minimize of the reconstruction error. The weights are updated once again, to minimize the error. The MSE reconstruction error obtained on this structure is 0.0680.

As the reconstruction error has decrease on this configuration after running the minimization algorithm. The weights on the structure has likely suffered modifica-

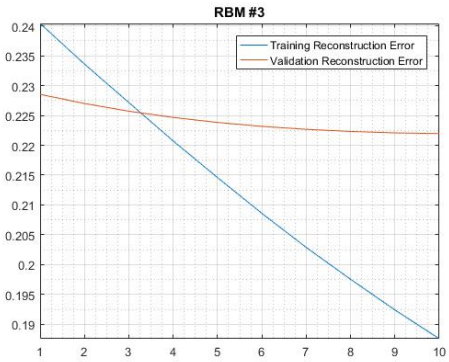

Figure 4.8: Training and validation error for the third RBM.

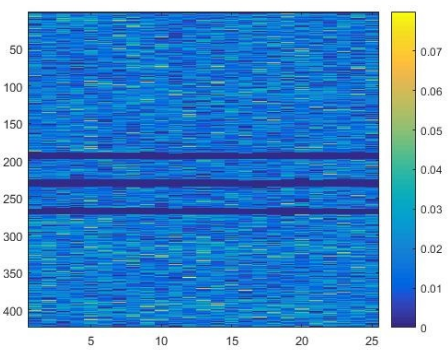

Figure 4.9: Final map of weights on the first layer.

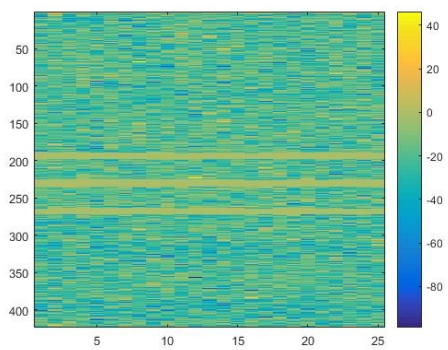

Figure 4.10: T-student map on the first layer.

tions. It is interesting to have a look over the final weights of the pre-traning.

The structure after the weights pre-traning, is trained by Backpropgation for producing regressions. The weights of the model will be updated again. Although it is aimed now to decrease the regression coefficient R^2. The training is evaluated by computing the MSE after each iteration on validation. It can be seen on the graph that there are instabilities on the training.

On parallel, it is also experimented the GP for regression. The results of both algorithms are compared with the performances of SVM attached on a DL structure.

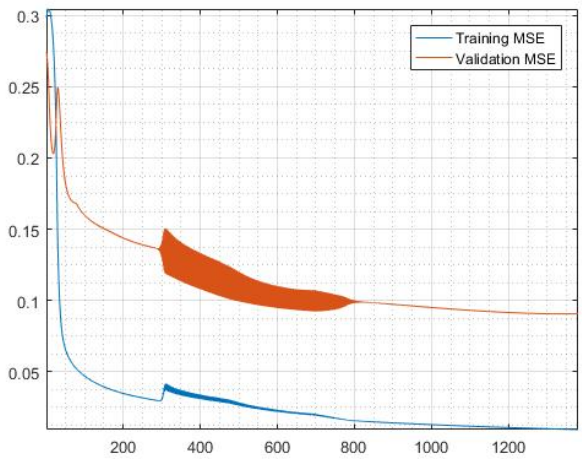

Figure 4.11: Convergence evolution on the Backpropagation algorithm for the configuration I.

The experiments are resume on the following table. The best R^2 is obtained by Backpropagation after two weights pre-tranings.

DL Configuration: 9860 2500 50				
Pre-training	**MSE RE**	**Training**	**MSE**	$\mathbf{R^2}$
CD	0.1180	**BP**	0.1146	0.5354
		GP	0.1213	0.5082
CD + RE	0.0680	**BP**	0.0862	0.6504
		GP	0.1210	0.5080
DL Config.: 9860 1000 50		**SVM**		0.8711

Table 4.2: Results for $v_1 = 9864|v_2 = 2500|v_3 = 50|h_3 = 1$.

4.4.2 Configuration II

On the second configuration, the number of output nodes on the second RBM is increased up to 100. The number of hidden units on the first layer, is still constant.

The pre-traning methodology is the same that on the previous experiment.

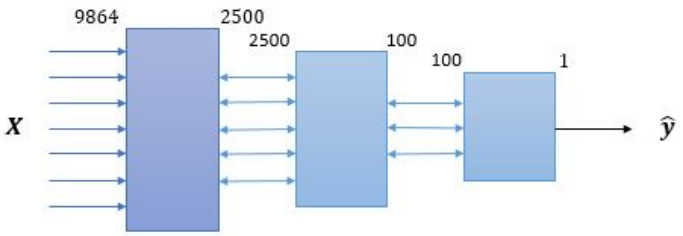

Figure 4.12: Structure experimented: $v_1 = 9864|v_2 = 2500|v_3 = 100|h_3 = 1$

After the pre-traning by CD, the weights have values similar to the obtained on the configuration I. The variations are because of the random initialization on the weights that DL uses to approximates the hidden probabilities by the Gibbs Sampling. CD is no deterministic algorithm.

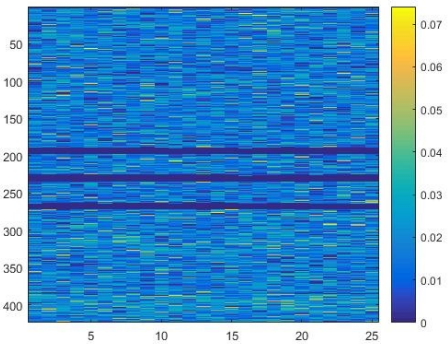

Figure 4.13: Weight's map for configuration II.

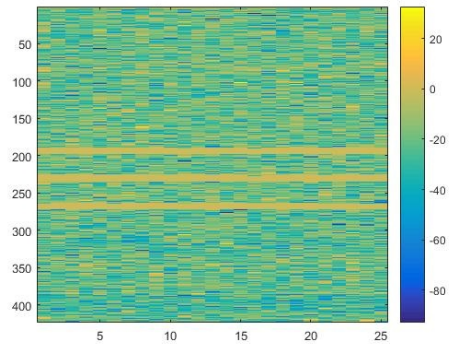

Figure 4.14: Map of t-student after the 10 epochs CD training.

Each RBM was trained along 10 epochs. The training is essentially the same on all the experiments. .

The weights after the fast convergence on a minima by CD, are updated by propagating the error forward and backwards iteratively. The propagation is a iterative methods and it stops when MSE on validation reach the minima. This is the seconds part of the per-training.

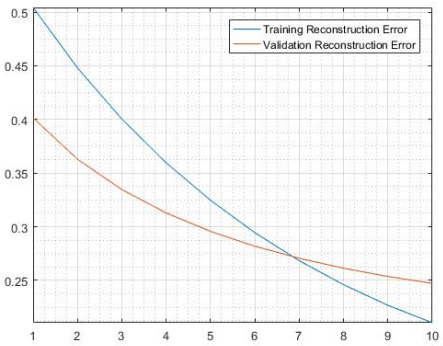

Figure 4.15: Convergence of CD for the first RBM.

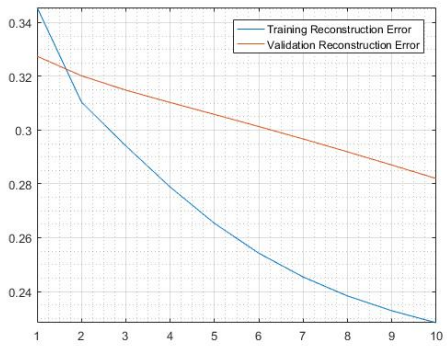

Figure 4.16: Validation and training error on second RBM.

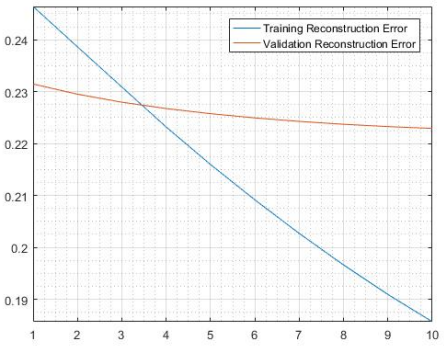

Figure 4.17: Evolution of the training on the third RBM

The weights of the configuration II are trained by Backpropagation. The performance on the regression is 0.7966. The performance for regression after the pretraning by minimizing the reconstruction error, is 0.0491. The minimization is clearly producing worst results.

The GP performs an increment on R^2 coefficient to 0.6902 with respect to previous experiments. Therefore, increasing the number of output nodes from 50 to 100 improves the result on the regression. In contrast, the R^2 after pre-traning by minimizing the reconstruction error, is just 0.1184. The second pre-training is affecting negatively.

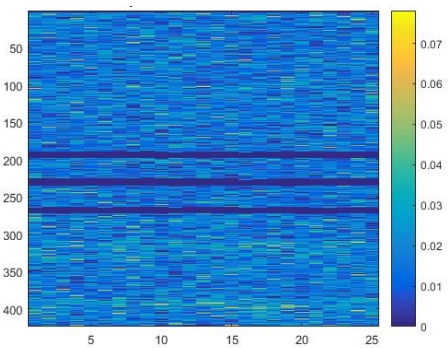

Figure 4.18: Map of the weights after Backpropagation

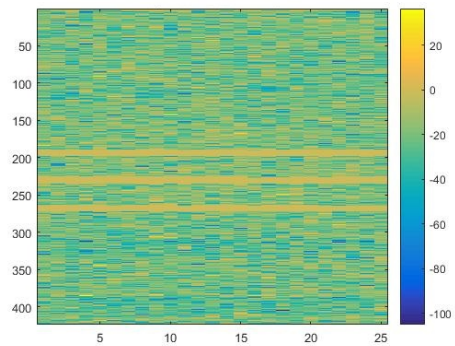

Figure 4.19: T-student over the map of weights on configuration II

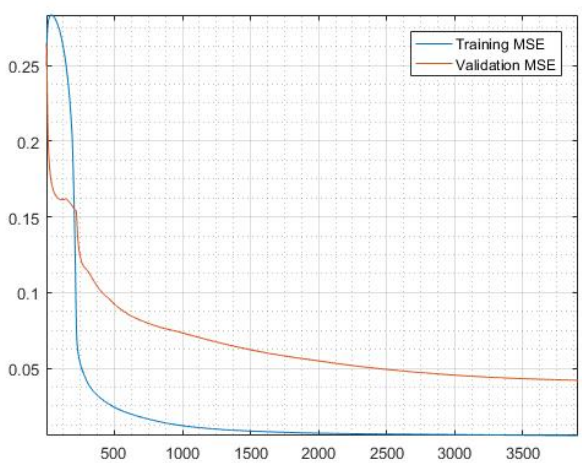

Figure 4.20: Convergence of the Backpropagation algorithm.

DL Configuration: 9860 2500 100				
Pre-training	**MSE RE**	**Training**	**MSE**	**R^2**
CD	0.118	**BP**	0.0502	0.7966
		GP	0.0764	0.6902
CD + RE	0.068	**BP**	0.2345	0.0491
		GP	0.1184	0.5201
DL Config.: 9860 1000 100		**SVM**		0.9033

Table 4.3: Results for $v_1 = 9864|v_2 = 2500|v_3 = 100|h_3 = 1$.

4.4.3 Configuration III

On this configuration, the output nodes at the last layer was increased up to 1000. The goal is to know whether the GP and Backpropagation regression performs better for doubling the output nodes.

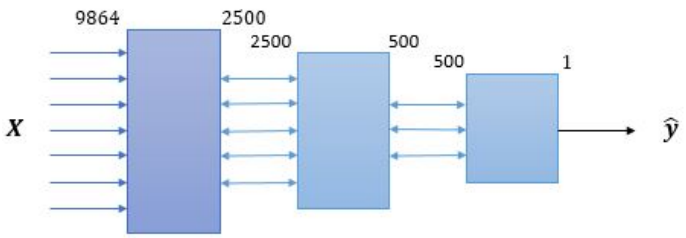

Figure 4.21: Structure experimented: $v_1 = 9864|v_2 = 2500|v_3 = 500|h_3 = 1$

The best weights on the pre-training are selected by the criteria of smallest reconstruction error. The CD training is not supervise, so there is not target values to train the machine. The best reconstruction error is 0.2297 after 10 initializations trained along 10 epochs.

The map of weights is very similar to the previous configuration, that means that the CD algorithm is also converging to similar weights values. It is difficult to monitoring the over-fitting on the DL, because it is unsupervised learning. A solution could be to look at the weights map to see whether it is deleting many variables.

The training of the RBM converged on the same MSE for reconstruction error. The validation tends to the same minima. In case of over-training, the MSE would start to increase at some point of the training.

In this case, Backpropagation diverges on the training. The regression output that it produces has a R^2 of 0.1154, this means that the correlation between target and prediction on the test dataset, is very low. Therefore, the quality the output is not useful for doing regression. In contrast, the GP increases the R^2 for 500 nodes at

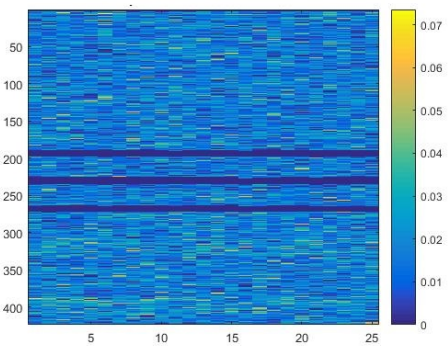

Figure 4.22: Map of weights from the first RBM.

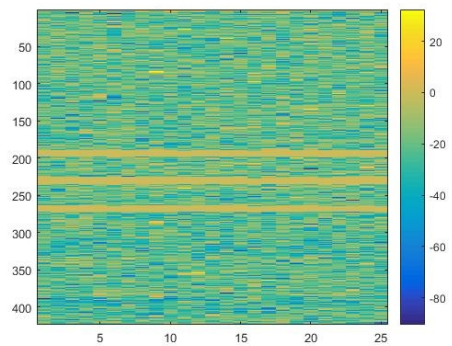

Figure 4.23: map of t-student from the weights.

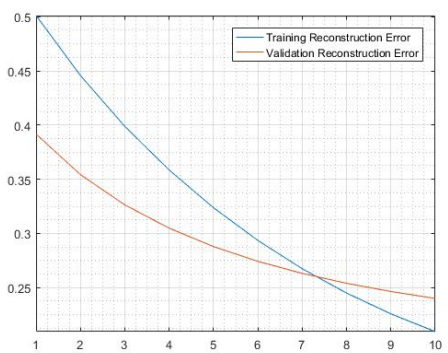

Figure 4.24: Training of the first RBM on the configuration III

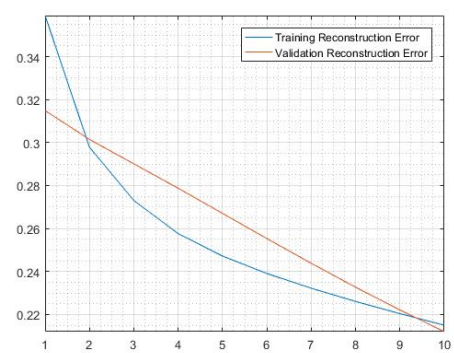

Figure 4.25: Second RBM is training along 10 epochs.

the last RBM. DL could be deleting features from the input dataset and this could be useful to train a regression model.

The performances of the GP with a input pre-traning by CD and minimizing the reconstruction error, it is clearly biasing the regression. Although, The performance improved with respect to the previous experiment. R^2 is 0.6693, and the MSE for pre-training by CD, improved up to 0.0526. The MSE produced after pre-traning by both methods, decreased to 0.0816 from 0.1180. Result obtained on the configuration II.

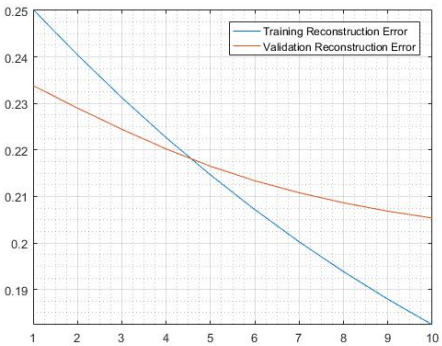

Figure 4.26: Training and validation MSE for the third RBM.

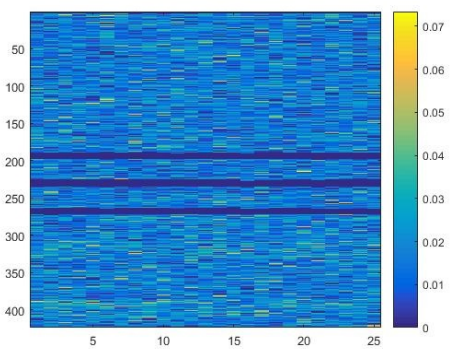

Figure 4.27: Map of weights after minimizing the MSE for reconstruction.

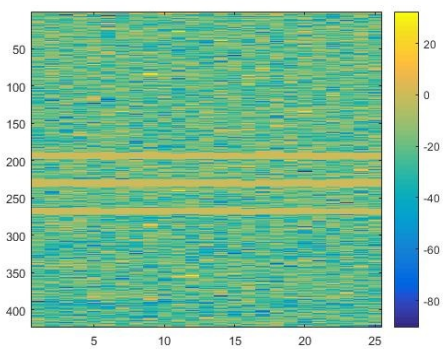

Figure 4.28: Map of the t-student from the backpropagation's weights.

DL Configuration: 9860 2500 500				
Pre-training	**MSE RE**	**Training**	**MSE**	**R^2**
CD	0.2297	**BP**	0.2182	0.1154
		GP	0.0526	0.7867
CD + RE	0.0502	**BP**	0.2338	0.0520
		GP	0.0816	0.6693
DL Config.: 9860 1000 500		**SVM**		0.9410

Table 4.4: Results for $v_1 = 9864|v_2 = 2500|v_3 = 500|h_3 = 1$.

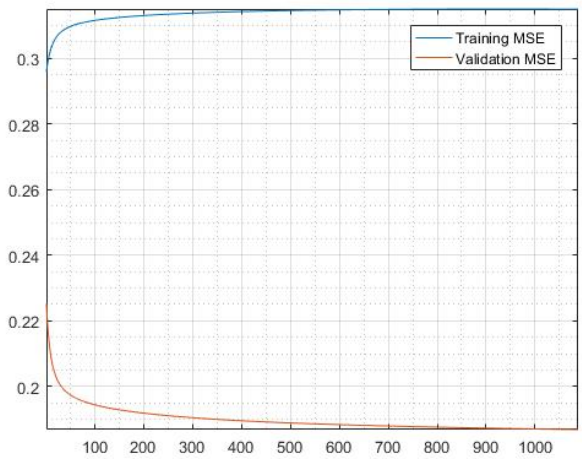

Figure 4.29: Backpropagation algorithm diverges due to the large number of neurons at the output RBM.

4.4.4 Configuration IV

On the Configuration IV, the number of hidden node at the seconds RBM, was increased to 1000. It was aimed to see whether the performance of a GP improved increasing the hidden nodes. Backpropagation performs worst when the number of nodes increased. The step down on nodes is probably to big to converge successfully.

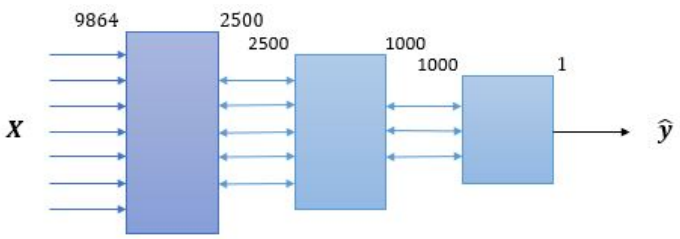

Figure 4.30: Structure experimented: $v_1 = 9864|v_2 = 2500|v_3 = 1000|h_3 = 1$

On the pre-training by CD, the map of weights is still the same. Therefore, it can be conclude that after the initialization, CD rapidly finds a local minima. The

weights have similar values on a different initialization. The t-student maps are not showing significant variations on the 3 configuration tested.

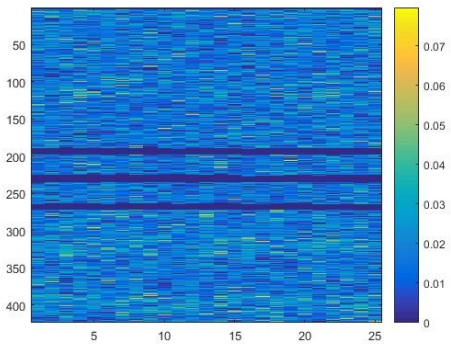

Figure 4.31: Wights of the training of the first RBM for the Configuration IV.

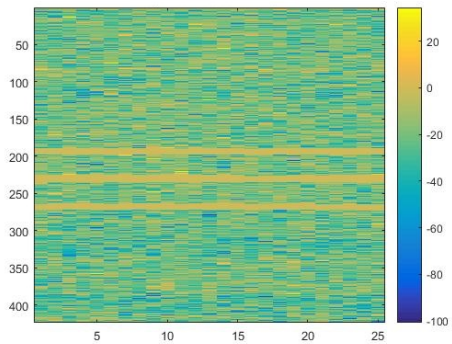

Figure 4.32: Values for the t-student from the first RBM after the training by CD.

The CD algorithm, trained along 10 epochs converges quickly. The MSE on the reconstruction error is 0.2841 on this experiment. The value increased from 0.2297 on configuration II, to 0.1350 on configuration I. Therefore, the MSE obtained increased by increasing the output nodes at the second RBM.

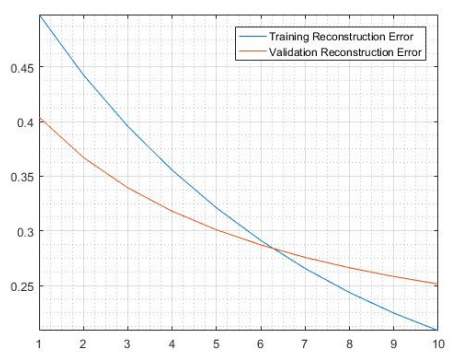

Figure 4.33: Pre-traning by CD for the first RBM, on the acse of smallest MSE.

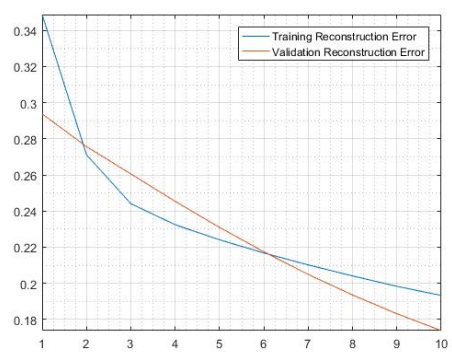

Figure 4.34: Visualization of the evolution of the pre-traning by CD on the second RBM.

The pre-traning by minimizing the reconstruction error, produced the MSE on

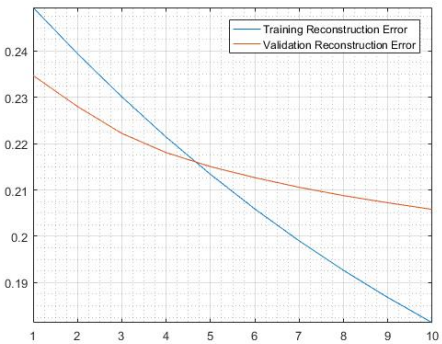

Figure 4.35: Third RBM's MSE is increasing as the number of neurons does.

this configuration. This means that both configurations are converging to a global minima, or that there is probably the same local minima. The value of the MSE is 0.0520.

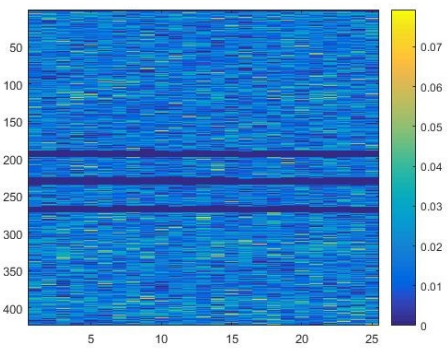

Figure 4.36: Backpropagation weights are similar, the algorithm converges to the same minims for MSE.

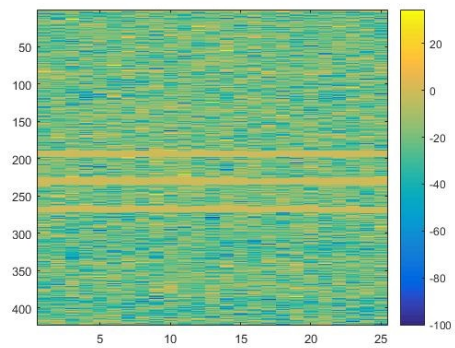

Figure 4.37: The t-student map does not show relevant variation on the weights.

As it is shown on the configuration IV, Backpropagation diverged when the output node increased tp 500. The correlation coefficient R^2 is dropping up to -0.1045, and up to 0.1177 on the case of pre-traning by CD.

The GP produces better results on the regression as the number of hidden nodes

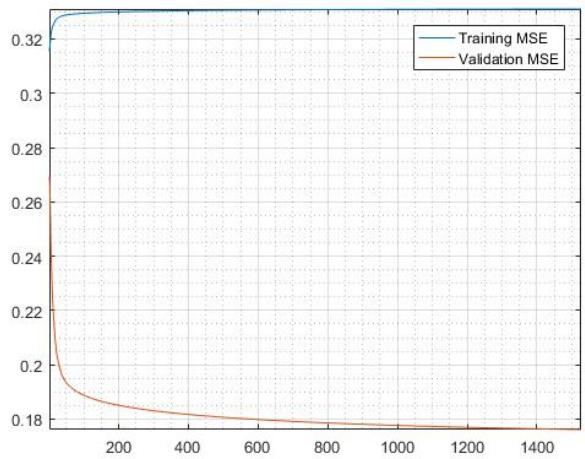

Figure 4.38: Backpropagation diverges for 1000 neurons at output of the second RBM

increases in the middle RBM. The R^2 coefficient reaches 0.8136 on the case of training the machine by CD. For the case that the structure is pre-trained by both algorithms, the correlation on the regression dropped to a R^2 of 0.3682. The minimization of the reconstruction error affects negatively to thw regression. The SVM performs the best results on all the configuration reviewed so far. The MSE is 0.0460 on the regression.

DL Configuration: 9860 2500 1000				
Pre-training	**MSE RE**	**Training**	**MSE**	**R^2**
CD	0.2841	**BP**	0.2176	0.1177
		GP	0.0460	0.8136
CD + RE	0.0502	**BP**	0.2724	-0.1045
		GP	0.0705	0.714
DL Config.: 9860 1000 500		**SVM**		0.9410

Table 4.5: Results for $v_1 = 9864|v_2 = 2500|v_3 = 1000|h_3 = 1$.

4.4.5 Configuration V

On the structures that were tested on this configuration, nodes at the first RBM where increase up to 5000. The number at the hidden nodes at seconds layer were increase also up to 2500. The objective of experimenting this setup, is to check whether the GP produces a smaller MSE on the regression than the previous configurations.

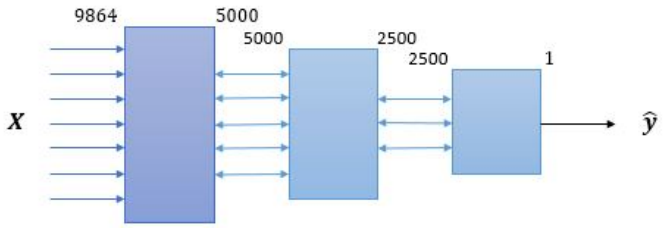

Figure 4.39: Structure experimented: $v_1 = 9864|v_2 = 5000|v_3 = 2500|h_3 = 1$

The weights after 10 epochs of training by CD, have a different map that on previous configuration. The weights of the variables that are more informative for characterizing model has an higher value. The algorithm is deleting the variables that are less correlated on the input dataset.

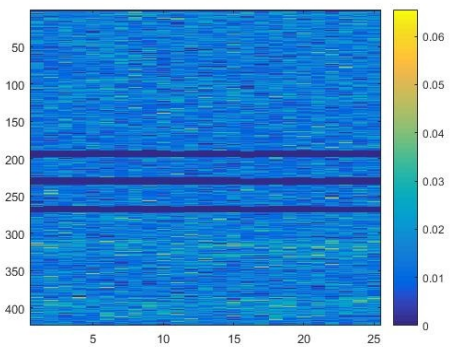

Figure 4.40: Map of weights from 5000 hidden units at the first RBM.

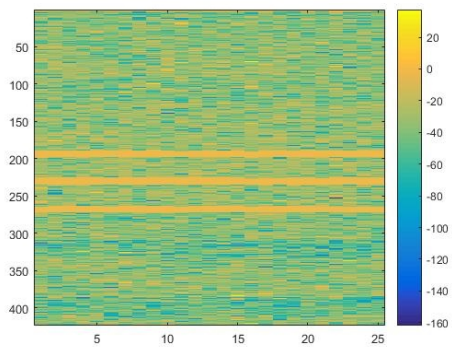

Figure 4.41: Map of t-student from the first RBM on the structure.

The pre-traning on the first RBM by CD is smoothly converging to a local minima in just 10 epochs. The seconds layer had a gently training but the minima is not reached in just 10 epochs. In contrast, on the third RBM, the MSE was still decreasing after the 10 epochs steeply. This is likely because of difference between the quantity of input nodes and output nodes is too large.

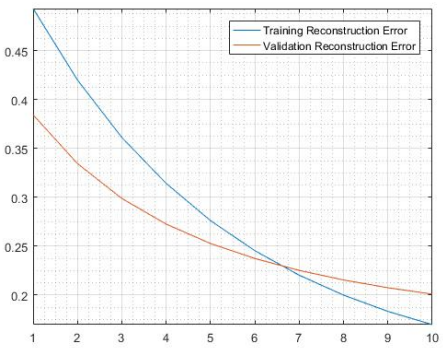

Figure 4.42: Training of the first RBM by CD for configuration V.

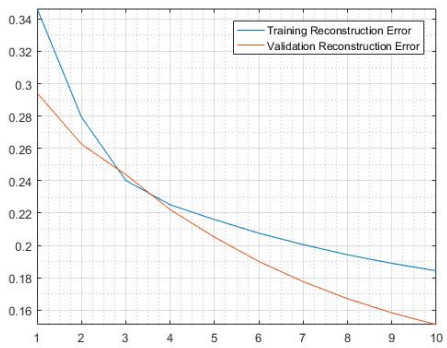

Figure 4.43: Motorization of the pre-traning on the seconds RBM.

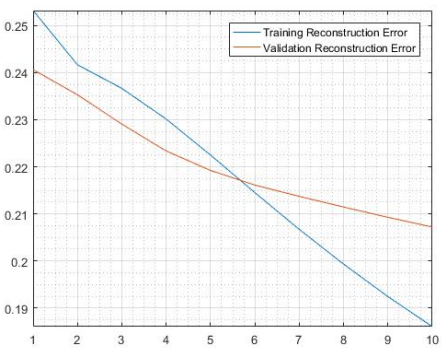

Figure 4.44: Training of the last layer for the configuration V.

The map of weights after the pre-traning, shows that the variables that contends more information are weighted heavily. The forward and backward error propagation training, went further down on the minima reached by on CD. This is meaningful

because it is pointing to the minima which reduces the MSE. Therefore, it is reached by training the algorithm with the most correlated variables on the dataset.

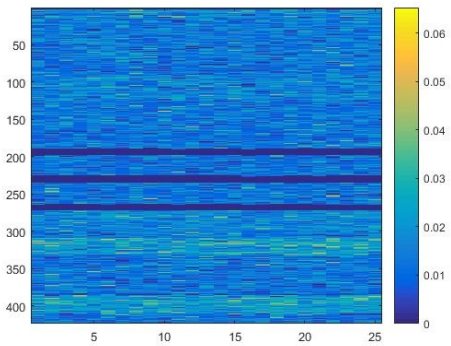

Figure 4.45: Backpropagation weights map after 330 epochs of training.

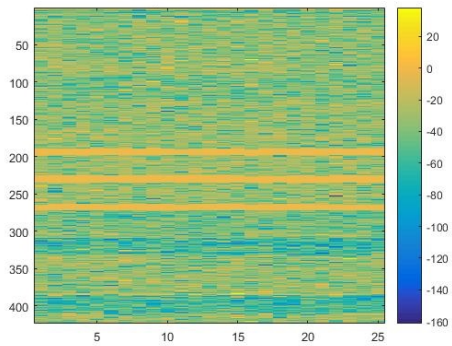

Figure 4.46: T-student of the weights of the first layer.

In this configuration the pre-traning by CD, had MSE of 0.3226. After the minimization, MSE decreased to 0.0502. The minimization of the MSE performs better results than in the previous configurations.

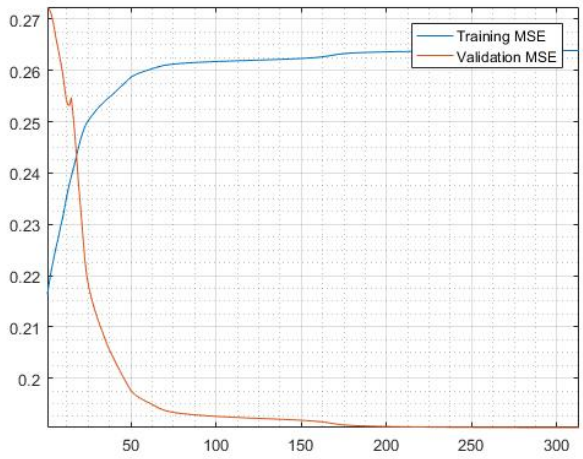

Figure 4.47: Backpropagation for regression diverges for an output of 2500 units on the second layer.

Backpropagation diverges on this configuration. This algorithm performs a neg-

ative R^2, which means that the values on the regression and on the target are completely independents. On the case of using just the CD, R^2 is 0.2220.

The Gaussian Process had a MSE of 0.0517 on the regression. The R^2 decreased from 0.8136 to 0.7905. The GP converged on a model using a pre-training by CD and minimizing the MSE on the reconstruction. In this configuration, the SVM is also performing better results than a GP or Backpropagation.

DL Configuration: 9860 5000 2500				
Pre-training	**MSE RE**	**Training**	**MSE**	**R^2**
CD	0.3226	**BP**	0.1919	0.2220
		GP	0.0517	0.7905
CD + RE	0.0502	**BP**	0.3024	-0.2260
		GP	0.1558	0.3682
DL Config.: 9860 1000 500		**SVM**		0.9410

Table 4.6: Results for $v_1 = 9864|v_2 = 5000|v_3 = 2505|h_3 = 1$.

4.5 Analysis of the Results

The results obtained on the experiments, show that Backpropagation performs better MSE and R^2 on structures with small number of output nodes at the second RBM. The performance on regression decrease as the hidden units at second RBM increases. The training of the weights become difficult because of reducing the dimensions steeply. The difference between visible and hidden units should not be large at the third RBM.

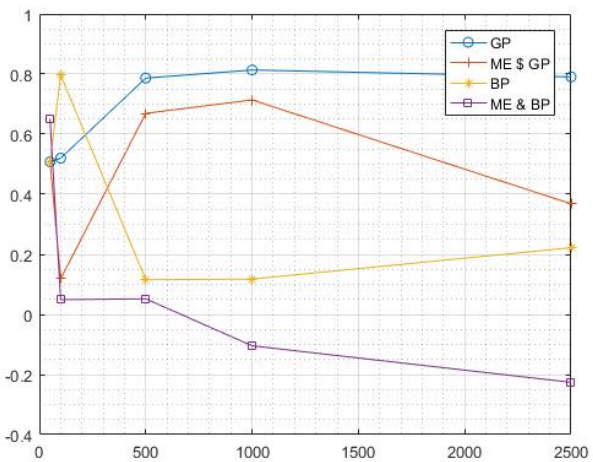

Figure 4.48: Results of the algorithms tested on each configuration.

The GP performed the best result on the configuration IV. The MSE is 0.046 and the R^2 is 0.8136. This result does not differ considerably from the results on the configuration II and IV, which performs MSE 0.0526 and 0.0517 receptively. The values are 0.7867 and 0.7905 for R^2. Therefore, the regression by GP will generate worst prediction when the number of hidden nodes at the second RBM increases from 1000 and decreases from the 500.

On the configurations that have been tested, SVM is the method that performs better results on R^2 and MSE. This is a symptom that the GP can likely perform

better regression on other structures that have not been tested on this document.

The results on the regression, which have been pre-training by minimizing of the reconstruction error, performs the worst. The R^2 coefficient reached negative values in some cases. Therefore, the prediction of GSR was completely uncorrelated from its target. This happened despite the fact that the weights on the RBMs were not significantly updated after converging. This is probably because the local minima on MSE of reconstruction error does not match with the local minima on MSE of regression.

The training by CD over-fits faster than other methods, when the difference between visible and hidden node is small. This training method needs more epoch to converge when difference between units is high. Therefore, the objective is to reduce the dimensionality avoiding over-fitting.I trick that helps do not decrease steeply the dimensions.

The pre-training by CD, increases the performance on the regression. This pre-training reduces considerably the computing time of the algorithm from days to minutes. This technique reduces the size of the matrices computed on parameters optimization by deleting from the data uncorrelated variables.

Chapter 5

Future Work

The results showed higher values for the correlation coefficient on the SVM. This means that is likely to find a GP that can also reach values closer to the obtained by the SVM. There are several paths that can be followed to explore this. There are different kernel functions that have not been tested yet. Moreover, the covariance matrix that defines a GP can be a product of kernel functions, which project different features from the data on a higher dimensions space.

The ML techniques applied on the SG have mainly been used in experiments for Load Demand, GSR and power output Forecasting. There are applications that have been not explored yet such as classification on fault detections. There are Markov models for cascade failure on transmission system that are currently being research. Plus, other application on Electrical Engineering as Cognitive Radio or Predictive Maintenance on materials that can be deteriorated by the exposure to magnetite fields and high currents.

In particular, the DL is a ML technique that makes tractable computing high dimensional datasets. The main disadvantage is that it is an unsupervised learning, so that it very difficult to monitor the over-fitting along the training. There many

interesting activation functions that can be explored in order to find configuration that performs better results.

The kernel functions is probably the key to find a GP that fits better the forecast on this case of GBS. The day light obviously has a daily periodicity. Therefore, if the hyper-parameters which defined this periodicity can be found, the performance on the regression from a GP will probably increase. The inconvenient is that the computational time needed on a GP increases rapidly because of the matrix computation on the optimization. Therefore, the adding kernel functions together could increase the complexity of the model up to untranslatable covariance matrices.

Another interesting ML technique is the Bootstrapping. This a tool thank cool make tractable problems that now are difficult to face because of the computation of high dimensional matrix on the optimization. The bootstrapping would delete the variable that are less correlated, reducing the dimension of the observations and ease the computation. The challenge on this methods is to find the right trade-off between performances and computational speed. This is probably a technique that can be combined with other methods and producing better or similar results lighting the process.

Appendices

Appendix A

Kernel Functions

The covariance functions, or kernels, are parametric functions for the model input. They define the covariance between pairs of input variables. A covariance matrix is really important on the GP because it contends the information for model generated by the process. The selection of a particular kernel function can improve or deteriorate the performance of a GP. Therefore, its selection and parameters optimization requires particular attention from the point of view of the performance and computational speed.

In this section, the parameters of kernel functions that are interesting to optimize, are denoted as θ. They are useful to introduce information about components that generate uncertainties on the model.

Linear Kernel

This kernel function is useful when the input data is already in a high-dimensions space, and the original features from the data are individually important for the model.

$$K(x, x') = x^T x' \tag{A.1}$$

Square Exponential Kernel

The square exponential is considered as a default kernel function on different ML techniques. It is used when there is no prior knowledge about the nature of the input data because of its infinite number of derivatives. This fact produces a smoothness effect on the covariance matrix of GP. However, this characteristic is often useless for tackling problems in which physical parameters have to be modeled.

$$K(x, x') = \theta_1^2 exp\left(-\frac{(x - x')^2}{\theta_2^2}\right) \tag{A.2}$$

where θ_1 is the parameter which models the magnitude of the kernel and θ_2 is the length-scale of the Euclidean distance between the sampled data points.

Periodic Kernel

The periodic kernel function is useful for modeling highly periodic trends on the input dataset. This function can be parametrized in order to fit the model with the different frequencies components in the sampled data. For instance, the oscillation on the observations can be due to seasonal variations. The periodic kernel has the following form:

$$K(x, x') = \theta_1^2 \left(-\frac{2sin^2(\pi(x - x')}{\theta_2^2}\right) \tag{A.3}$$

Rational Quadratic Kernel

The rational quadratic covariance is a stationary kernel function. It can be seen as scalable mixture of square exponential kernels with different length-scale parameter for each one of them. There is a practical implementation on a GP in the article [42].

$$K(x, x') = \theta_1^2 \left(1 + \frac{(x - x')^2}{2\theta_3\theta_2^2}\right)^{-\theta_3} \tag{A.4}$$

where θ_3 is a shape parameter, which describes as the tails decay on the kernel function.

Independent Gaussian and Dependent Noise Kernel

For modeling error on the prediction and Gaussian noise on the sampling that can affect the enclosed information on the data and that can also influence the prediction, an independent Gaussian covariance matrix can be used. In this way, perturbations on the trend are considered in the model such as:

$$K(x, x') = \theta_1^2 exp\left(-\frac{(x - x')^2}{2\theta_2^2}\right) + \theta_3^2\delta_{xx'} \tag{A.5}$$

where θ_3 is the magnitude parameter for the noise and $\delta_{xx'}$, is the independent short-term correlation noise between data points.

The properties of a covariance function can be improved by adding or multiplying different together covariance matrices. The new covariance has a combination of characteristics from the root functions. This is particularly interesting for ap-

proaching to problems with many seasonal components or harmonics due to cyclical variation on the input variable of the process.

References

[1] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[2] F. Grimaccia, M. Mussetta, and R. Zich, "Neuro-fuzzy predictive model for pv energy production based on weather forecast," in *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, pp. 2454–2457, June 2011.

[3] J. Liu, W. Fang, X. Zhang, and C. Yang, "An improved photovoltaic power forecasting model with the assistance of aerosol index data," *IEEE Transactions on Sustainable Energy*, vol. 6, pp. 434–442, April 2015.

[4] C. Chen, S. Duan, T. Cai, and B. Liu, "Online 24-h solar power forecasting based on weather type classification using artificial neural network," *Solar Energy*, vol. 85, no. 11, pp. 2856 – 2870, 2011.

[5] A. Linares-Rodriguez, J. A. Ruiz-Arias, D. Pozo-Vazquez, and J. Tovar-Pescador, "An artificial neural network ensemble model for estimating global solar radiation from meteosat satellite images," *Energy*, vol. 61, pp. 636 – 645, 2013.

[6] S. Salcedo-Sanz, C. Casanova-Mateo, A. Pastor-Sánchez, and M. Sánchez-Girón, "Daily global solar radiation prediction based on a hybrid coral reefs optimization – extreme learning machine approach," *Solar Energy*, vol. 105, pp. 91 – 98, 2014.

[7] J.-L. Chen, H.-B. Liu, W. Wu, and D.-T. Xie, "Estimation of monthly solar radiation from measured temperatures using support vector machines – a case study," *Renewable Energy*, vol. 36, no. 1, pp. 413 – 420, 2011.

[8] C. Paoli, C. Voyant, M. Muselli, and M.-L. Nivet, "Forecasting of preprocessed daily solar radiation time series using neural networks," *Solar Energy*, vol. 84, no. 12, pp. 2146 – 2160, 2010.

[9] L. Olatomiwa, S. Mekhilef, S. Shamshirband, and D. Petković, "Adaptive neuro-fuzzy approach for solar radiation prediction in nigeria," *Renewable and Sustainable Energy Reviews*, vol. 51, pp. 1784 – 1791, 2015.

[10] Y. Tao and Y. Chen, "Distributed pv power forecasting using genetic algorithm based neural network approach," in *Proceedings of the 2014 International Conference on Advanced Mechatronic Systems*, pp. 557–560, Aug 2014.

[11] J. Huang and M. Perry, "A semi-empirical approach using gradient boosting and -nearest neighbors regression for {GEFCom2014} probabilistic solar power forecasting," *International Journal of Forecasting*, vol. 32, no. 3, pp. 1081 – 1086, 2016.

[12] B. Tuyishimire, R. McCann, and J. Bute, "Evaluation of a kalman predictor approach in forecasting pv solar power generation," in *2013 4th IEEE International Symposium on Power Electronics for Distributed Generation Systems (PEDG)*, pp. 1–6, July 2013.

[13] L. A. Fernandez-Jimenez, A. M. noz Jimenez, A. Falces, M. Mendoza-Villena, E. Garcia-Garrido, P. M. Lara-Santillan, E. Zorzano-Alba, and P. J. Zorzano-Santamaria, "Short-term power forecasting system for photovoltaic plants," *Renewable Energy*, vol. 44, pp. 311 – 317, 2012.

[14] S. Shamshirband, K. Mohammadi, H.-L. Chen, G. N. Samy, D. Petković, and C. Ma, "Daily global solar radiation prediction from air temperatures using kernel extreme learning machine: A case study for iran," *Journal of Atmospheric and Solar-Terrestrial Physics*, vol. 134, pp. 109 – 117, 2015.

[15] J.-L. Chen, G.-S. Li, B.-B. Xiao, Z.-F. Wen, M.-Q. Lv, C.-D. Chen, Y. Jiang, X.-X. Wang, and S.-J. Wu, "Assessing the transferability of support vector machine model for estimation of global solar radiation from air temperature," *Energy Conversion and Management*, vol. 89, pp. 318 – 329, 2015.

[16] M. Bilgili and M. Ozgoren, "Daily total global solar radiation modeling from several meteorological data," *Meteorology and Atmospheric Physics*, vol. 112, no. 3, pp. 125–138, 2011.

[17] Y. Zhao, J. Rong, B. Wang, B. Liu, and X. Zha, "17th ifac symposium on system identification sysid 2015 parameter optimization of pv based on hybrid genetic algorithm," *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 568 – 572, 2015.

[18] A. I. Negash, A. Hooshmand, and R. Sharma, "A wavelet-based method for high resolution multi-step pv generation forecasting," in *2014 IEEE PES T D Conference and Exposition*, pp. 1–5, April 2014.

[19] A. Yona, T. Senjyu, A. Y. Saber, T. Funabashi, H. Sekine, and C. H. Kim, "Application of neural network to 24-hour-ahead generating power forecasting for pv system," in *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, pp. 1–6, July 2008.

[20] Z. Li, Y. Zhou, C. Cheng, Y. Li, and K. Lai, "Short term photovoltaic power generation forecasting using rbf neural network," in *The 26th Chinese Control and Decision Conference (2014 CCDC)*, pp. 2758–2763, May 2014.

[21] D. AlHakeem, P. Mandal, A. U. Haque, A. Yona, T. Senjyu, and T. L. Tseng, "A new strategy to quantify uncertainties of wavelet-grnn-pso based solar pv power forecasts using bootstrap confidence intervals," in *2015 IEEE Power Energy Society General Meeting*, pp. 1–5, July 2015.

[22] J. Shi, W.-J. Lee, Y. Liu, Y. Yang, and P. Wang, "Forecasting power output of photovoltaic system based on weather classification and support vector machine," in *Industry Applications Society Annual Meeting (IAS), 2011 IEEE*, pp. 1–6, Oct 2011.

[23] M. Behrang, E. Assareh, A. Ghanbarzadeh, and A. Noghrehabadi, "The potential of different artificial neural network (ann) techniques in daily global solar radiation modeling based on meteorological data," *Solar Energy*, vol. 84, no. 8, pp. 1468 – 1480, 2010.

[24] A. G. Salman, B. Kanigoro, and Y. Heryadi, "Weather forecasting using deep learning techniques," in *2015 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pp. 281–285, Oct 2015.

[25] J. Li and J. K. Ward, "Irradiance forecasting for the photovoltaic systems," in *Modelling, Identification Control (ICMIC), 2014 Proceedings of the 6th International Conference on*, pp. 346–350, Dec 2014.

[26] S. Salcedo-Sanz, C. Casanova-Mateo, J. Muñoz-Marí, and G. Camps-Valls, "Prediction of daily global solar irradiation using temporal gaussian processes," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, pp. 1936–1940, Nov 2014.

[27] P. Tang, D. Chen, and Y. Hou, "Entropy method combined with extreme learning machine method for the short-term photovoltaic power generation forecasting," *Chaos, Solitons and Fractals*, vol. 89, pp. 243 – 248, 2016.

[28] T. Khatib, A. Mohamed, and K. Sopian, "A review of solar energy modeling techniques," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 5, pp. 2864 – 2869, 2012.

[29] Y. Ren, P. Suganthan, and N. Srikanth, "Ensemble methods for wind and solar power forecasting—a state-of-the-art review," *Renewable and Sustainable Energy Reviews*, vol. 50, pp. 82 – 91, 2015.

[30] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Massachusetts Institute of Technology: MIT Press, 2006.

[31] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, Aug. 2004.

[32] S. Rosset, J. Zhu, and T. Hastie, "Boosting as a regularized path to a maximum margin classifier," *J. Mach. Learn. Res.*, vol. 5, pp. 941–973, Dec. 2004.

[33] M. A. Carreira-Perpinan and G. E. Hinton, "On contrastive divergence learning."

[34] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks.," 2006.

[35] B.-Y. LeCun, Y. and G. E. Hinton, "Deep learning," May 2015.

[36] R. Rojas, *Neural Networks: A Systematic Introduction*. New York, NY, USA: Springer-Verlag New York, Inc., 1996.

[37] G. Hinton, "A practical guide to training restricted boltzmann machines," *Momentum*, vol. 9, p. 1, 2010.

[38] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," vol. 9, pp. 249–256, May 2010.

[39] M. Martinez-Ramon, "Regression with gaussian process networks." ECE595 Advance Machine Learning, 2015.

[40] T. M. Giannaros, V. Kotroni, and K. Lagouvardos, "Predicting lightning activity in greece with the weather research and forecasting (wrf) model," *Atmospheric Research*, vol. 156, pp. 1 – 13, 2015.

[41] D. Carvalho, A. Rocha, M. Gómez-Gesteira, and C. S. Santos, "Sensitivity of the {WRF} model wind simulation and wind energy production estimates to planetary boundary layer parameterizations for onshore and offshore areas in the iberian peninsula," *Applied Energy*, vol. 135, pp. 234 – 246, 2014.

[42] A. Solin and S. Särkkä, "Gaussian quadratures for state space approximation of scale mixtures of squared exponential covariance functions," in *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, Sept 2014.