Universidad de Alcalá
COMISIÓN DE ESTUDIOS OFICIALES
DE POSGRADO Y DOCTORADO

ACTA DE EVALUACIÓN DE LA TESIS DOCTORAL

Año académico 2016/17

DOCTORANDO: **GASCÓN MORENO, JORGE**
D.N.I./PASAPORTE: ******91488Q**

PROGRAMA DE DOCTORADO: **D347-TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES**
DEPARTAMENTO DE: **TEORÍA DE LA SEÑAL Y COMUNICACIONES**
TITULACIÓN DE DOCTOR EN: **DOCTOR/A POR LA UNIVERSIDAD DE ALCALÁ**

En el día de hoy 6/10/2016, reunido el tribunal de evaluación nombrado por la Comisión de Estudios Oficiales de Posgrado y Doctorado de la Universidad y constituido por los miembros que suscriben la presente Acta, el aspirante defendió su Tesis Doctoral, elaborada bajo la dirección de **SANCHO SALCEDO SANZ // JOSE ANTONIO PORTILLA FIGUERAS.**

Sobre el siguiente tema: *NOVEL EVOLUTIONARY-BASED METHODS FOR THE ROBUST TRAINING OF SVR AND GMDH REGRESSORS*

Finalizada la defensa y discusión de la tesis, el tribunal acordó otorgar la CALIFICACIÓN GLOBAL[3] de (**no apto, aprobado, notable** y **sobresaliente**): SOBRESALIENTE

Alcalá de Henares, 6 de Octubre de 2016

EL PRESIDENTE
Fdo.: SILVIA JIMÉNEZ

EL SECRETARIO
Fdo.: Enrique Alexandre Cortizo

EL VOCAL
Fdo.: LEOPOLDO CARRO

EL VOCAL
Fdo.: CARLOS GARCÍA

EL VOCAL
Fdo.: JAVIER DEL SER

FIRMA DEL ALUMNO
Fdo.: Jorge Gascón Moreno

Con fecha 27 de octubre de 2017 **la Comisión Delegada de la Comisión de Estudios Oficiales de Posgrado, a la vista de los votos emitidos de manera anónima por el tribunal que ha juzgado la tesis, resuelve:**

☒ Conceder la Mención de "Cum Laude"
☐ No conceder la Mención de "Cum Laude"

La Secretaria de la Comisión Delegada

[3] La calificación podrá ser **"no apto" "aprobado" "notable" y "sobresaliente"**. El tribunal podrá otorgar la mención de "cum laude" si la calificación global es de sobresaliente y se emite en tal sentido el voto secreto positivo por unanimidad.

INCIDENCIAS / OBSERVACIONES:

Universidad de Alcalá

COMISIÓN DE ESTUDIOS OFICIALES
DE POSGRADO Y DOCTORADO

En aplicación del art. 14.7 del RD. 99/2011 y el art. 14 del Reglamento de Elaboración, Autorización y Defensa de la Tesis Doctoral, la Comisión Delegada de la Comisión de Estudios Oficiales de Posgrado y Doctorado, en sesión pública de fecha 27 de octubre, procedió al escrutinio de los votos emitidos por los miembros del tribunal de la tesis defendida por *GASCÓN MORENO, JORGE*, el día 6 de octubre de 2016, titulada *NOVEL EVOLUTIONARY-BASED METHODS FOR THE ROBUST TRAINING OF SVR AND GMDH REGRESSORS*, para determinar, si a la misma, se le concede la mención "cum laude", arrojando como resultado el voto favorable de todos los miembros del tribunal.

Por lo tanto, la Comisión de Estudios Oficiales de Posgrado **resuelve otorgar** a dicha tesis la

MENCIÓN "CUM LAUDE"

Alcalá de Henares, 31 de octubre de 2016
EL PRESIDENTE DE LA COMISIÓN DE ESTUDIOS
OFICIALES DE POSGRADO Y DOCTORADO

UNIVERSIDAD DE ALCALÁ · COMISIÓN DE ESTUDIOS OFICIALES DE POSGRADO Y DOCTORADO

Juan Ramón Velasco Pérez

Copia por e-mail a:
Doctorando: GASCÓN MORENO, JORGE
Secretario del Tribunal: ENRIQUE ALEXANDRE CORTIZO
Directores de Tesis: SANCHO SALCEDO SANZ // JOSE ANTONIO PORTILLA FIGUERAS

Universidad de Alcalá

Sancho Salcedo Sanz
Dpto. de Teoría de la Señal y Comunicaciones
Campus Universitario s/n
28805 Alcalá de Henares (Madrid)
Telf: +34 91 885 67 31
Fax: +34 91 885 66 99
sancho.salcedo@uah.es

DR. D. SANCHO SALCEDO SANZ, Profesor Titular de Universidad del Área de Conocimiento de Teoría de la Señal y Comunicaciones de la Universidad de Alcalá, y DR. D. JOSÉ ANTONIO PORTILLA FIGUERAS, Profesor Titular de Universidad del Área de Conocimiento de Teoría de la Señal y Comunicaciones de la Universidad de Alcalá.

CERTIFICAN

Que la tesis "Novel Evolutionary-based Methods for the Robust Training of SVR and GMDH Regressors", presentada por Jorge Gascón Moreno y realizada en el Departamento de Teoría de la Señal y Comunicaciones bajo nuestra dirección, reúne méritos suficientes para optar al grado de Doctor, por lo que puede procederse a su depósito y lectura.

Alcalá de Henares, Junio 2016.

Fdo.: Dr. D. Sancho Salcedo Sanz

Fdo.: Dr. D. José Antonio Portilla Figueras

Universidad de Alcalá

Dpto. de Teoría de la Señal y Comunicaciones
Campus Universitario s/n
28805 Alcalá de Henares (Madrid)
Telf: +34 91 885 88 99
Fax: +34 91 885 66 99

Jorge Gascón Moreno ha realizado en el Departamento de Teoría de la Señal y Comunicaciones y bajo la dirección del Dr. D. Sancho Salcedo Sanz y del Dr. D. José Antonio Portilla Figueras, la tesis doctoral titulada "Novel Evolutionary-based Methods for the Robust Training of SVR and GMDH Regressors", cumpliéndose todos los requisitos para la tramitación que conduce a su posterior lectura.

Alcalá de Henares, Junio 2016.

EL DIRECTOR DEL DEPARTAMENTO

Fdo: Dr. D. Javier Acevedo Rodríguez

Escuela Politécnica Superior

Departamento de Teoría de la Señal y Comunicaciones

Ph. D. Thesis

Novel Evolutionary-based Methods for the Robust Training of SVR and GMDH Regressors

Author:

Jorge Gascón Moreno

Supervisores:

Sancho Salcedo Sanz, Ph. D.

José Antonio Portilla Figueras, Ph. D.

Alcalá de Henares, Junio 2016

Abstract

This Ph.D. Thesis elaborates on several novel improvements for two specific state-of-the-art Machine Learning algorithms: the Support Vector Regression (SVR) approach, and the Group Method of Data Handling. In the case of the SVR approach, a new multi-parametric evolutionary SVR is proposed. This new algorithm takes into account a different value of the γ parameter for each dimension of the feature space. In this case, it is not possible to apply a classic grid search, due to computational requirements of such an algorithm, and therefore in this thesis an evolutionary approach is successfully applied to obtain the optimal values for these SVR parameters. Regarding the GMDH network, in this thesis a novel construction algorithm based on a hyper-heuristic approach is proposed. Hyper-heuristic is a novel concept related to evolutionary computation, in which the algorithm encodes several smaller heuristics which can be applied in a sequential fashion to solve a given optimization problem. In this specific application, several basic heuristic are encoded in an evolutionary algorithm to form a hyper-heuristic approach which constructs robust versions of GMDH networks for regression problems. A final contribution of this thesis is the proposal of new validation methods to better estimate the performance of regression techniques in data-driven problems. The idea is to obtain better models from the training phase of the algorithms, in such a way that the performance in the test set is improved, mainly in training time and overall performance of the system, with respect to classical evaluation methods such as K-Fold cross validation, etc. All the proposed and developed methods of the thesis are experimentally evaluated in benchmark and real-world data-driven regression problems.

Resumen

Esta Tesis plantea nuevas mejoras sobre dos métodos del estado del arte en el área de Aprendizaje Máquina: Máquinas de Vectores Soporte para Regresión (SVR) y el algoritmo conocido como Group Method of Data Handling (GDMH). En el caso de las SVR, se ha desarrollado un nuevo algoritmo de tipo evolutivo para el entrenamiento con kernel multi-paramétrico. Este nuevo algoritmo tiene en cuenta un parámetro γ distinto, para cada una de las dimensiones del espacio de entradas. En este caso, debido al incremento del número de parámetros no puede utilizarse una búsqueda en grid clásica, debido al coste computacional que conllevaría. Por ello, en esta Tesis se propone la utilización de un algoritmo evolutivo para la obtención de los valores óptimos de los parámetros de la SVR. Respecto a las redes GMDH, esta Tesis propone un nuevo algoritmo de construcción de estas redes basado en un algoritmo de tipo hiper-heurístico. Esta aproximación es un concepto nuevo relacionado con la computación evolutiva, que codifica varios heurísticos que pueden ser utilizados de forma secuencial para resolver un problema de optimización. En nuestro caso particular, varios heurísticos básicos se codifican en un algoritmo evolutivo, para crear una solución hiper-heurística que permita construir redes GMDH robustas en problemas de regresión. Como contribución final de esta Tesis, se proponen nuevos métodos de validación que mejoren el rendimiento de las técnicas de regresión en problemas *data-driven*. La idea es obtener mejores modelos en la fase de entrenamiento del algoritmo, de tal forma que el desempeño con el conjunto de test mejore, principalmente en lo que a tiempo de entrenamiento se refiere y en el rendimiento general del sistema, con respecto a otros métodos de validación clásicos como son K-Fold *cross validation*, etc. Todas las propuestas y métodos desarrollados en esta Tesis han sido evaluados experimentalmente en problemas *benchmark*, así como en aplicaciones de regresión reales.

Todo empezó cuando Sancho (director de esta Tesis) me ofreció un día realizar una beca de investigación a tiempo parcial dentro de su grupo de investigación (GHEODE), que dirige junto con J.A. Portilla. A partir de este momento comenzó mi camino en el mundo de la investigación, y lo que iba a ser una beca de 6 meses se acabó alargando 2 años mas. En este tiempo he conocido a grande profesores como Silvia y J.A. Portilla, muy buenos doctorandos que me ayudaron a empezar (Ángel y Emilio), inmejorables compañeros de Tesis (Beatriz, Arturo, Juan y Leo) y también dos geniales doctorandas que estuvieron de estancia en nuestro laboratorio (Itziar y Diana). Gracias al gran ambiente que existía en el laboratorio, me quedé allí durante todo es tiempo. Esto junto con la confianza y el apoyo que siempre he tenido de Sancho, me han ayudado en la consecución de esta Tesis.

Por supuesto, agradecer a mis padres que sin su determinación y empuje no hubiese sido igual. Ya fuese con el "Pero vas a acabar el doctorado, ¿No?" en los primeros años justo al dejar la universidad y empezar a trabajar, o con el casi diario "¿Cómo llevas la Tesis" de los últimos meses. Ahora en serio, gracias a ambos por esforzaros tanto, sé que no siempre ha sido fácil para vosotros mantener a los dos hijos estudiando y aun así nunca nos ha faltado aquello que hemos necesitado en cada momento y por eso esta Tesis es gracias en gran parte a vuestro trabajo. También quería agradecer a mi hermano, Álvaro, por siempre haber estado ahí cuando lo he necesitado y haberme ayudado tanto desde que dejé el pueblo y me fui a la universidad a estudiar.

Finalmente, no quería dejar de agradecer a Rebeca todo lo que me ayudado, el tiempo que le ha dedicado a esta Tesis y la gran paciencia que ha tenido mientras la escribía. Gracias por todos los consejos dados y correcciones hechas, que me han ayudado a mejorar el texto. Gracias también por aguantar las horas dedicadas a la tesis: a la vuelta del trabajo y los fines de semana sin salir de casa. También agradecer los ánimos (H H G M D H!!!!) que me ha dado cuando menos ganas tenía de seguir escribiendo. Por todo esto, esta Tesis es en gran parte también suya.

Jorge Gascón Moreno

Acknowledgements

- This work has been partially supported by the project TIN2014-54583-C2-2-R of the Spanish Ministerial Commission of Science and Technology (MICYT), and by the Comunidad Autónoma de Madrid, under project number S2013ICE-2933_02.
- I would like to acknowledge University of Alcalá, that supported this work through several research fellowships:
  - "Formación Personal Investigador" (FPI) call 2010.
  - "Beca Investigación" call 2010.
  - "Beca Introducción a la Investigación" call 2009.

Table of contents

List of figures

List of tables

Acronyms

CV Cross-Validation.

EA Evolutionary Algorithm.

ECMWF European Centre for Medium-Range Weather Forecast.

EP Evolutionary Programming.

ES Evolutionary Strategies.

GMDH Group Method of Data Handling.

GS Grid Search.

HBC Hess Brezowsky Classification.

HH Hyper-Heuristic.

HH-GMDH Hyper-Heuristic GMDH.

KGP Kolmogorov-Gabor Polynomial.

KKT Karush-Kuhn-Tucker.

KS-Test Kolmogorov-Smirnov Test.

MCP Measure-Correlate-Predict.

MEI Multivariate ENSO Index.

ML Machine Learning.

MLP Multi-Layer Perceptron.

MSE Mean Square Error.

NAO North Atlantic Oscillation.

NOAA National Oceanic and Atmospheric Administration.

OLR Ongoing long-wave radiation.

PNN Polynomial Neural Network.

QBO Quasi-Biennial Oscillation.

RMCPO Real MCP Operations.

RMSE Root Mean Square Error.

SC Solar Cycle.

SVM Support Vector Machines.

SVR Support Vector Regression.

TOC Total Ozone Content.

TOMS Total Ozone Mapping Spectrometer.

Chapter 1

Introduction

In today's world, it is possible to save all kind of information generated in a company, University, government, mobile application, etc. This information gives the opportunity to analyze different phenomena and find new useful applications [1], such as tumor detection, stock market value prediction, Internet personal advertising, tasks automatization or improve the decision making in very different areas. In order to get this knowledge, it is necessary to carefully analyze the data and find out the relationships between our desired outputs and the input data. This analysis could be carried out with traditional methods where it is necessary to have an expert on the area who knows or has the ability to find these relationships. However, this is not always possible because the amount of data is huge or there is a lack of knowledge in order to build a robust model[2].

Due to this fact, and the tremendous increasing of computation capabilities which has been producing lately, an area called *data mining* (currently also known as Big Data) has became very popular [3]. Data mining allows building an expert system without detailed knowledge about the area[1]. Data mining involves two main parts, data preparation and machine learning. The first part is on charge, among others, of the acquisition, cleaning and filtering of the data. The set of data created by the first part is then passed to the ML algorithm, which takes care of the mathematical model building.

This thesis is focused on the improvement of a specific number of ML techniques for prediction and regression problems. In a more specific way, this work is devoted to the development of hybrid evolutionary neural computation techniques, in a broad sense (including as neural computation some approaches such as kernel methods or support vector algorithms). Hybrid techniques involving evolutionary computation and neural techniques have been applied to a large amount of classification and regression problems, obtaining excellent results in many application areas. There is, however, a large margin

[1]This does not mean that there is not necessary previous knowledge in the area. This knowledge is necessary in order to provide proper data and guide the Machine Learning (ML) to find the best model.

of improvement in many of these applications, and it has been tried to build this gap with this research work, in which it has been put together ML regression algorithms with evolutionary computation techniques.

Since the number of hybrid approaches involving evolutionary techniques and neural computation algorithms is huge, we have focused on a reduced set of algorithms, according to their applicability in real problems. First, we have tried to improve hybrid approaches involving Support Vector Regression (SVR) approaches. In this case, we have tackled different problems involving standard and multi-parametric SVR approaches, and how to improve them with a good design of evolutionary algorithms to estimate their hyper-parameters. Second, we have also considered a fast-training algorithm called Group Method of Data Handling (GMDH), where we have tried to improve it with a recently proposed evolutionary technique called Hyper-Heuristic (HH). All the algorithms proposed can be interpreted in the frame of ML, with different possible problems to be tackled: supervised classification or regression, clustering (unsupervised learning), etc. In the following subsection we revise the main concepts associated to ML methods, we also outline the main concepts of the algorithms involved in this thesis (though in the next chapter we provide a full description of these approaches), and finally we describe the main objectives of the thesis and the structure of the rest of the work.

1.1 Machine Learning Overview

ML is a field of computer science that focuses on pattern recognition and developing algorithms which are able to learn and create estimation models from data. There are three main groups of ML algorithms:

- **Supervised Learning**: These algorithms need a data set formed by a vector of inputs and outputs which represents the results for a given phenomenon. The aim is to build a mathematical model based on this experience which will be able to reproduce with a grade of error the phenomenon behavior. This study is focused in these kind of learning methods and it is described more deeply in the next section.

- **Unsupervised Learning**: In this case we have a set of inputs but there is not any information about the desired outputs, so there is not a feedback which indicates us about the quality of our approach. The goal of these methods usually is to find patterns between the data and clustering them in several groups, where the samples have some similar features.These type of learning is very popular as recommendation algorithm which suggests you a film, a product to buy, etc. based on the consumption of people with similar profiles.

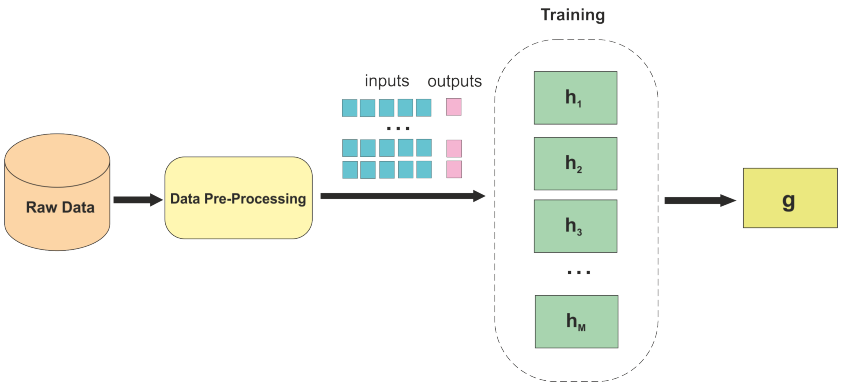


Figure 1.1: Supervised learning process

- **Reinforcement Learning**: These kind of algorithms are not based on a set of outputs and/or inputs which have been saved previously. Instead of that, the algorithm starts from a initial state and apply an action, this action moves the agent to a new state. Every state has associated a reward that evaluates it. The aim is to collect the maximum cumulative rewards on a path. This technique is similar to the way animals acquire new knowledge with a learning based on punishments and recompenses.

1.1.1 Supervised Learning Methods

Supervised learning methods are the most popular ones in ML and have been applied to a large amount of different applications. These algorithms are methods which infer an estimation function by learning from a set of labeled data. These data consist of a group of samples which are formed by a vector of inputs and the desired output. In other words, there is an unknown target function (f) which represents the problem, the learning method tries to generate a model (g) as similar as possible to f, with the assistance of a data set $S = [(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)]$. The building process of a supervised machine is described in Figure 1.1:

1. Data collection. As a first task, a set of relevant information about the phenomenon is gathered in order to create the training set. The more number of samples the better to get a generalized function, avoiding *over-fitting*. However, a big data size implies large computation time. Therefore it is necessary to look for a trade off, having enough samples to cover the entire sample space but also keeping an affordable training time. It is important to have samples distributed uniformly in the sample space, and not concentrate too many points in a small area and too

few samples in the rest, since the model will focus on the most populated area and ignore the rest (over-fitting).

2. Data Preprocessing. Before applying any ML algorithm, we have to clean the data in order to eliminate errors, missing values, noise, apply some interesting transforms or just convert alphanumerical values to numerical ones.

3. Training process. This step consists in applying the learning algorithm which in most cases is based on generating different functions (hypothesis $h_i(\mathbf{x})$) and check their performance over the training set S. Some algorithms have a finite amount of hypothesis and it is possible to test all of them. However, most of them have a function model with several parameters and infinite number of combinations, so the algorithm tunes these parameters trying to improve the **training error** until the algorithm converges or a specific condition is satisfied.

4. Keep the best model: From the hypothesis set generated by the learning process $H = [h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_M(\mathbf{x})]$, it is selected the best one under a specific criteria. This best hypothesis becomes our model $g(\mathbf{x})$.

The training process is the core of the ML, step where the learning takes place generating batch of hypothesis with different configurations. Every $h_i(\mathbf{x})$ is tested with the data set, for each sample $\mathbf{x}_i$ the hypothesis makes a prediction $h(\mathbf{x}_i)$ and is compared with the real value $\mathbf{y}_i$.

Each input of the problem is a random variable X_i, so there is a set of random variables given by $\mathbf{X} = X_1, X_2, \ldots, X_K$. Therefore the output is also a random variable given by $\mathbf{Y} = f(\mathbf{X})$, but in most of the cases there is not a deterministic relationship and there is additive noise which represents unknown variables or measurement error, hence the output is expressed as $\mathbf{Y} = f(\mathbf{X}) + noise$. The learning process tries to find the function $f(\mathbf{X})$ with the minimum estimation error (EE):

$$EE = \mathrm{E}_x[L(f(\mathbf{X}), \mathbf{Y})]. \tag{1.1}$$

where $L(f(\mathbf{X}), \mathbf{Y})$ is the loss function, usually it is applied the squared error loss function $L(f(\mathbf{X}), \mathbf{Y}) = (f(\mathbf{X}) - \mathbf{Y})^2$. The expected value of the squared error loss function is given by Equation 1.2:

$$\mathrm{E}_x[L(f(\mathbf{X}), \mathbf{Y})] = \int (f(\mathbf{x}) - \mathbf{y})^2 P(\mathbf{x}, \mathbf{y}) dx \cdot dy \tag{1.2}$$

The solution is $f(\mathbf{x}) = \mathrm{E}(\mathbf{Y}|\mathbf{X} = \mathbf{x})$ (see [4]). However, as it was previously said the distributions $\mathbf{Y}$ and $\mathbf{X}$ are unknown, otherwise the whole problem would be directly solved.

The different supervised learning algorithms try to implement a recipe to approximate $f(\mathbf{x})$ with the available training data S.

In most cases, the training process consists in generating a batch of hypothesis with different configurations. Every $h_i(\mathbf{x})$ is tested with the data set and the predictions $\hat{\mathbf{y}}$ are compared with $\mathbf{y}$ in order to get the **In Sample Error** (E_{in}) [5]. Hereby, we can rank all the hypothesis and choose the one who has the least E_{in} which is supposed to be the best approximation of $f(\mathbf{x})$.

$$E_{in} = error(h_i(\mathbf{x}), \mathbf{y}). \tag{1.3}$$

$$error(h_i(\mathbf{x}), \mathbf{y}) = \frac{1}{N}\sum_{n=1}^{N}(h_i(\mathbf{x}_n) - \mathbf{y}_n)^2. \tag{1.4}$$

Usually, $error(h_i(\mathbf{x}_n), \mathbf{y}_n)$ is calculated as the average of the squares differences, also called Mean Square Error (MSE)[2] (Equation 1.4). The MSE is the average of the squared loss function for a vector of predictions $\mathbf{y}$. This E_{in} is an approximation of the real error of h_i also called **Out of Sample Error** (E_{out}). The E_{out} is the hypothesis error committed over samples outside the data set and it has the expression showed in Equation 1.5.

$$E_{out} = \mathrm{E}_x[error(h_i(\mathbf{x}), \mathbf{y})]. \tag{1.5}$$

where $\mathrm{E}_x[error(h_i(\mathbf{x}), \mathbf{y})]$ is the expected value over $\mathbf{x}$. This error cannot be calculated because the probability distribution $\mathbf{X}$ and $\mathbf{Y}$ are unknown. Therefore, it is essential that E_{in} has a similar behavior to E_{out} and thus the best hypothesis ($g(\mathbf{x})$) fulfill $E_{out}(g(\mathbf{x})) \approx 0$. This simple statement $E_{out}(g(\mathbf{x})) \approx 0$ implies two different conditions:

1. $E_{in}(g(\mathbf{x})) \approx 0$. **Error minimization**. The estimation function generated has to have a low level error.

2. $E_{out}(g(\mathbf{x})) \approx E_{in}$. **Generalization**: The out of sample error has to be an error close to the real error. Otherwise, it can be achieved an very small error with the data set, and then get huge error with samples out of this data set. This problem is called *over-fitting* and is one of the main problem that we have to deal in ML.

It is critical to have an error metric which satisfies these two conditions at the same time, because the selection of the best estimation function depends on it. Most of the

[2]MSE could not be a good error measure when there are outliers, since large errors have more weights than the small ones. This problem could be reduced by Root Mean Square Error (RMSE).

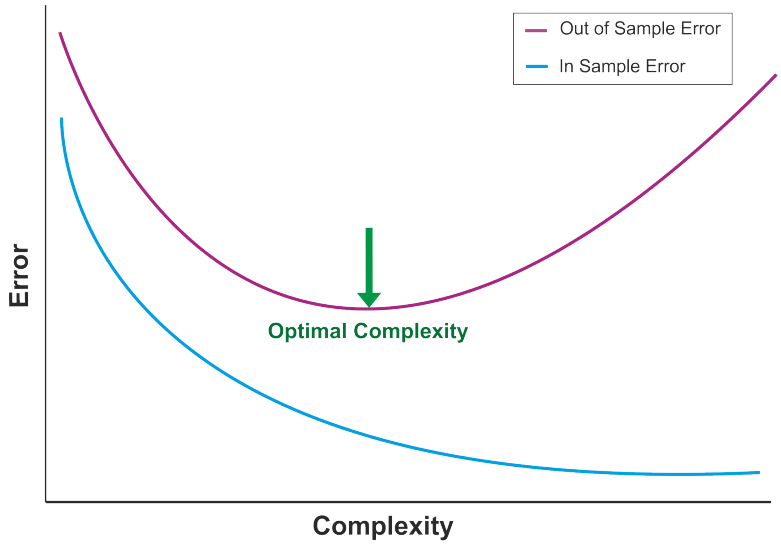


Figure 1.2: In Sample Error and Out of sample error.

times, when we improve one of the condition the other one gets worse, it is necessary to make a trade off. This trade off is represented in Figure 1.2, the more complex is the model the less in sample error but also the bigger the out of sample error is. Sometimes it is better to sacrifice one of the conditions in order to get closer to the other one. For example, an investor needs to have a model which $E_{out}(g)$ approximates closely to E_{in}, despite of E_{out} is high. If we are able to build a classification model to invert which has a high in sample error like for example 52%, but we can assure that this measure is close to the reality, we can invest a great amount of money in our system and get important profits.

1.1.2 Over-Fitting and Generalization

A ML algorithm creates an estimator which makes predictions from an infinite input space to a finite space (classification) ($f : \mathbb{R}^n \rightarrow \mathbb{S}^m$) or infinite (regression) output space ($f : \mathbb{R}^n \rightarrow \mathbb{R}^m$). The algorithm has to work with a small portion of reality given in a data set and deal with the problem of building a generalizable model with only the available information. If the algorithm does not take care about generalization and it is only focused on getting a small error, the result would be a very good model in terms of the available data (low E_{in}), but with a poor behavior with new samples (high E_{out}). Figure 1.3 shows how a over-fitted model has a more complex shape which fits to all the samples so takes into consideration the sample noise too.

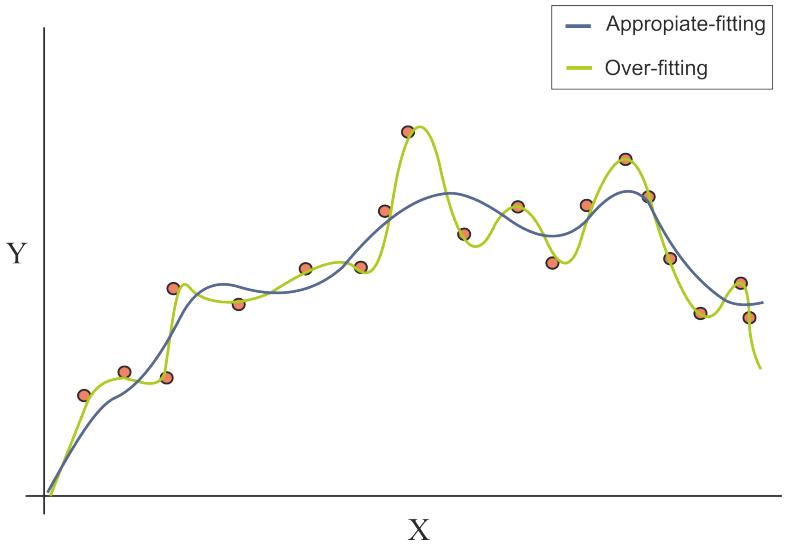


Figure 1.3: Graph with a over-fitted system and the objective function $f(\mathbf{x})$.

In most of the algorithms, the learning process cannot handle the over-fitting and needs an extra step to care about this problem. In this step the so called **Validation Method** is applied in order to measure and penalize this over-fitting, thus it is selected the model with less **Validation error**.

A naive approach for the calculation of E_{in} is to get this estimation error using the training data, thus we will get a model with a low E_{in} but we do not know if it is over-fitted or not. One simple approach could be splitting the data into two subsets: the training set and the validation set. The training set is used for building the hypothesis and the validation set for calculating the E_{in}, thus if a $h_i(\mathbf{x})$ is over-fitted the error will be high. However, this solution does not exploit all the data efficiently and when this has just a few samples and split into two smalls groups of data, the learning algorithm could have a poor behavior choosing a poor hypothesis.

This problem could be mitigated with Cross-Validation (CV) methods. These kind of validation methods use the whole data set for training and validating hypothesis. One of this methods is **Leave-One-Out** which consists in training with the whole data set and leave one sample out to estimate it with the resultant model. This process is repeated N times (one for each sample), therefore the computational cost of this method is very high and can make the training part too heavy (see Figure 1.4). There is another method called **K-Fold CV** which is less time consuming, instead of leaving just one sample out, each iteration of the algorithms leaves $\frac{N}{K}$ samples out so this process has to be repeated only K times[3].

[3]Leave-One-Out is a particularization of K-Fold with $K = N$.

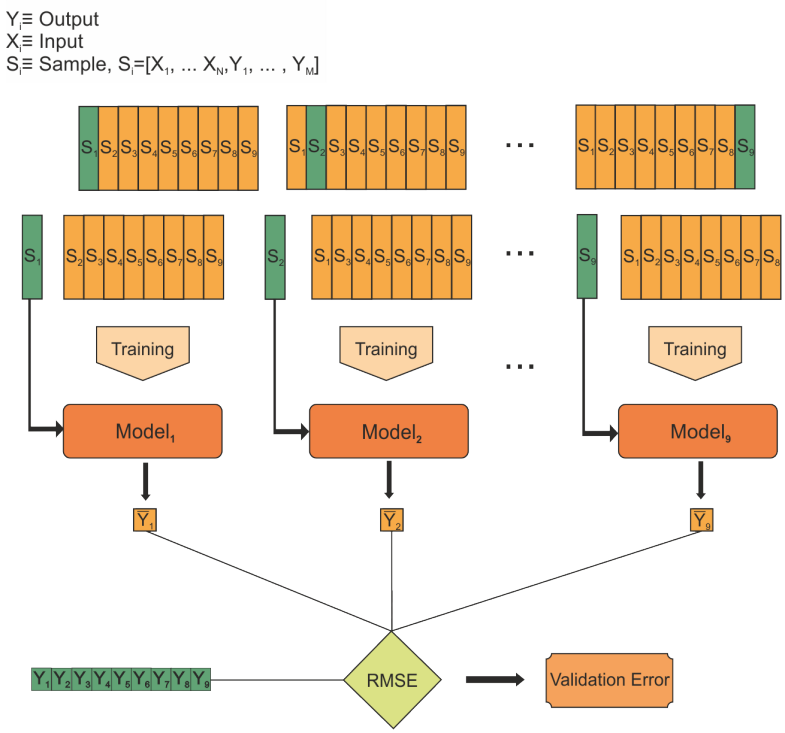


Figure 1.4: Leave One Out cross-validation diagram.

The computational cost of these techniques is high, doing the training process very heavy. Nevertheless, the advantages of the application of these techniques compensates this increment on time. In this Thesis we present new validation methods which reduce considerably this training time.

1.1.3 Overview Machine Learning Algorithm

ML algorithms have became very popular thanks to the explosion of *Big Data*. Nowadays, there is a huge amount of different learning algorithms which have been applied to too many different areas. In this section, we show a summary of some of the most important techniques:

- **Artificial Neural Networks**: Definitely, neural networks are the most used and popular ML algorithms. These networks are based on the behavior of the neurons in our brain, which are the responsible of the information transmission. A neuron consists of a set of several inputs called *dendrites* connected to other neurons, they receive stimulus from these neurons and transmit them to the core also called *soma*. The soma processes the inputs from the different dendrites and elaborates a response which is transmitted by the *axon* to other neurons or finally to an organ.

The artificial neural networks try to replicate this behavior in different ways, the best known is called Multi-Layer Perceptron (MLP). A Perceptron is the name of the MLP basic unit. These perceptrons consist of several inputs $\mathbf{x} = [x_1, x_2, \ldots, x_N]$, which are weighted $\mathbf{x}_w = [w_1x_1, w_2x_2, \ldots, w_Nx_N]$. Then, these weighted inputs are summed up and the result is processed by an activation function and finally send this output $y = f(\sum_{i=1}^{N} w_i x_i)$ to others perceptrons or take it as the final result. A MLP consists of several layers of these elements, usually the standard configuration has an input layer, one or several hidden layers and the output layer. These kind of networks need a training process in order to calibrate the weights of each perceptron. There are many different methods to calibrate these weights, being Back-propagation error one of the most popular [6].

Neural networks have been implemented and are currently being applied with success to a wide range of problems, such as: Rainfall forecasting in Australia [7], breast cancer detection [8], traffic signal recognition [9], etc.

- **Bayesian Networks**: Also called as belief networks, is a method to find probabilities which look for the relationship between different variables. Bayesian Networks are graphical models, where each node of the graph represents a random variable and the arcs represents the probabilistic dependence between two variables, the lack of arcs indicates conditional independence. The graph generated is called directed acyclic graph. For a set of variables $\mathbf{X} = X_1, X_2, \ldots, X_N$ a Bayesian Network needs to define the conditional probability distribution (CPD), $\Theta_{x_i,D_i} = P_B(x_i|D_i)$, where $D_i \in X_1, X_2, \ldots, X_{i-1}$ is the subset of nodes that X_i depends on.

$$P(X_i|D_i) = P(X_i|X_1, X_2, \ldots, X_{i-1}) \tag{1.6}$$

With the conditional probabilities and applying the probability chain rule, it can be calculated the joint probability as

$$P(X_1, X_2, \ldots, X_N) = \prod_{i=1}^{N} P(X_i|D_i) \tag{1.7}$$

Bayesian networks are not a black box algorithm: the models created can be easily interpreted and each output has a probability associated. These conditional probabilities can be calculated in different ways: with a supervised learning algorithm, Maximum-likelihood, expectation maximization algorithm, Bayesian estimation, etc. Sometimes the structure of the Bayesian Network is also unknown and we do not have idea of the relationships between the variables. In these cases, there are

techniques which are able to deduce possible structures from the data: scored-based method or constraint-based method.

Bayesian Networks have been used for a long time. NASA developed a diagnosis health system [10]. But also they are very popular in financial risk management [11]. Furthermore, Bayesian networks have resulted in many others successful techniques such as Naive Bayes, Hidden Markov Model which is used in Kalman filters that are very popular for tracking problems and turbo codes which are used in 3G and 4G mobile communications [12].

- **Support Vector Machines (SVM)**: it is a very popular technique based on Kernel theory. During the learning process, the SVM has to solve a convex optimization problem with a unique solution, this is an advantage with respect to Artificial Neural Networks since they can fall into a local minimum during the learning process. Another advantage of SVM is that the models are less over-fitted than neural networks, because SVM takes care of generalization during the learning process. However, SVM also needs an extra regularization process to avoid the over-fitting. In Chapter 2 the theory behind Support Vector Machines and the learning process is explained.

- **Group Method of Data Handling (GMDH)**: It is a self-organized technique, which does not need to tune any parameter before the training process. The algorithm is this way able to create the model by itself. The algorithm builds a network of nodes which are the combination of different inputs into a n order polynomial, one of this polynomial could be the output or the input for a new layer of polynomials. This process is repeated until a certain condition is achieved. The Group Method of Data Handling is not such popular as the previous techniques, however it has two main advantages making inputs filtering and a training algorithm faster than the other techniques, since it is not necessary to repeat the training for a set of parameters. Section 2.2 gives a deeper introduction to this algorithm.

- **Evolutionary Algorithm (EA)**: are a set of techniques which are based on the method of natural selection described by Darwin in 1859, whereby the individuals best fitted to the environment are the ones more likely to survive and reproduce. Thus each new generation is best prepared to the environment than the previous one.

  Evolutionary philosophy can be applied to solve optimization problems. The individuals are candidate solutions to the optimization problem, each of these

solutions is evaluated with a cost function which gives a mark called fitness. This fitness is used to assign a probability which indicates how likely is an individual to be elected for the crossover process, where the new generation is created as a combination of the chosen individuals. This process is repeated until the algorithm converge to a solution good enough.

The time consumption of this kind of algorithms is considerably smaller than a simple brute force algorithm. For this reason evolutionary algorithms have become very popular and have been applied to very different applications such as: Public transport network optimization [13], mobile network deployment [14], optimal positioning of wind turbines in wind farms [15], etc.

1.2 Objectives of the Thesis

As mentioned before, the main objective of this thesis is to develop a number of novel hybrid approaches involving evolutionary algorithms with neural computation techniques. Specifically, we propose the following objectives:

1. Obtaining robust approaches for training multi-parametric Gaussian SVR algorithms.

2. Developing novel validation methods for improving standard and also multi-parametric SVR approaches.

3. Improving the performance of GMDH training algorithms.

All these objectives have been approached by evolutionary computation, with hybrid solutions which improve the performance of the original algorithms.

1.3 Structure of the rest of the work

The structure of the remainder of the thesis is the following: next chapter summarizes a good description of the main techniques we use in this Thesis. The main concepts of SVR, GMDH and EAs are fully described at this point. Chapter 3 describes how to improve the performance of multi-parametric SVR training (hyper-parameters search) with new evolutionary algorithms. Chapter 4 extends the improvement of SVR algorithms by means of proposing novel validation methods, for both standard and multi-parametric SVR approaches. Chapter 5 discusses the improvement of GMDH algorithms using hyper-heuristics. All these chapters are intended to be self-explanatory, including a brief

introduction to the hybrid technique, an experimental section to show the improvement obtained with the hybrid algorithm, and also some conclusions that can be extracted in each chapter. Chapter 6 provides some final remarks and future lines of research, whereas this thesis is closed with the Appendix A and B where the main contributions of the work (in terms of publications and alternative research production) are presented.

Chapter 2

State of art

2.1 Support Vector Machines

Support vector machines are a robust methodology in statistical machine learning that have been successfully applied to solve classification and regression problems in several fields. The theory of support vector machines was developed by Vapnik and his team in AT&T [16][17]. The reason behind the popularity of the SVM is the structural risk minimization, which not only takes care about fitting the problem, but also its over-fitting. The next subsections explain the SVM theory. For further information about SVM the reader is encouraged to review [18] [19] [20].

2.1.1 Introduction to SVMs

Let suppose the classification problem shown in Figure 2.1, there are two different classes: red triangles and blue circles. It is straightforward to find a line which separates the data perfectly into two classes. However there are multiples lines which make a perfect separation (see Figure 2.1). Each separation line shown in Figure 2.2 does not have classification error, but which one of the three separators is the best? Intuitively we would choose the one with the largest distance between the two groups of samples. This reasoning makes the solution more generalizable for future samples, minimizing futures over-fitting problems.

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{2.1}$$

Equation 2.1 represents a general hyper plane h where $\mathbf{w} = (w_1, w_2, \ldots, w_M)$ is the vector of weights, $\mathbf{x} = (x_1, x_2, \ldots, x_M)$ is the vector of inputs and b is the offset. A

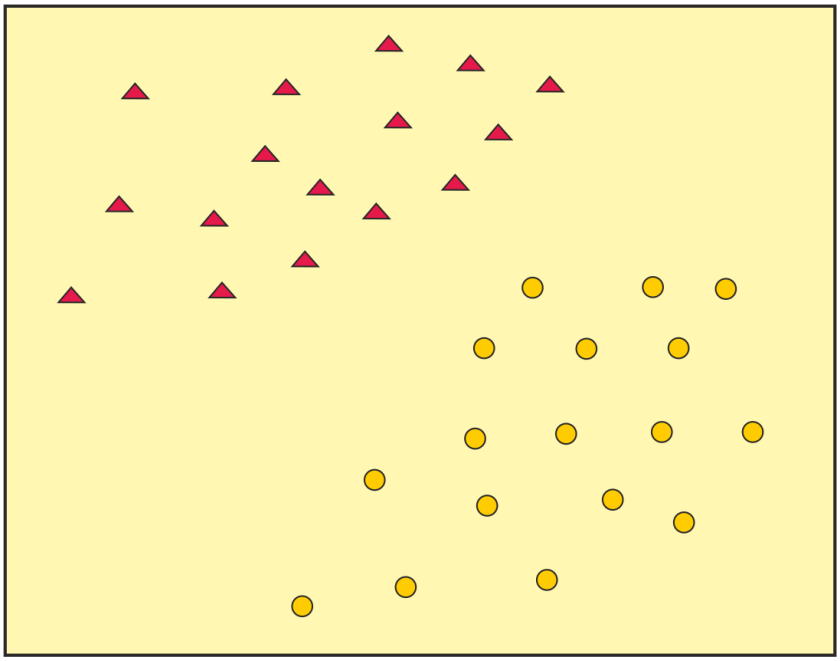


Figure 2.1: Example of classification problem.

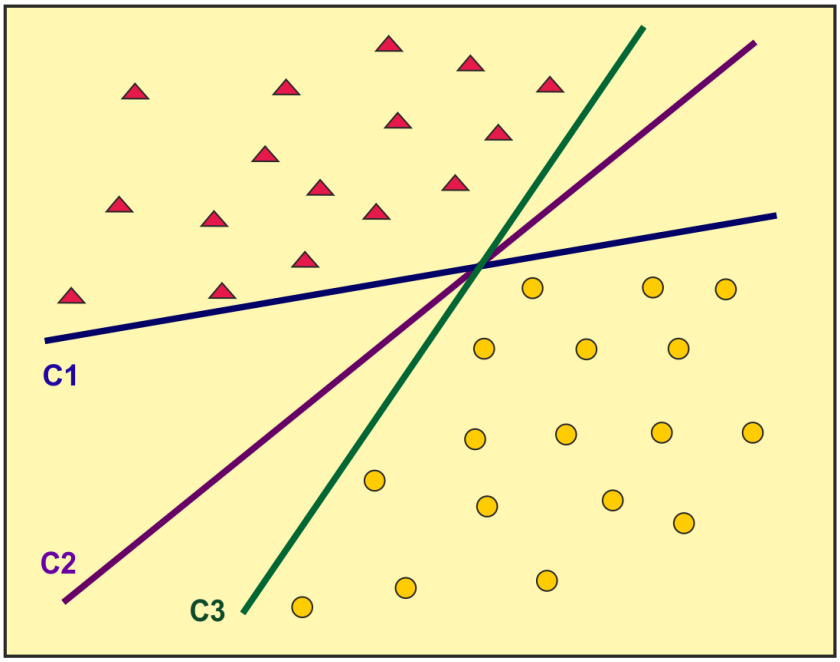


Figure 2.2: Perfect classifiers.

support vector machine algorithm tries to find the hyper-plane with the largest distance between classes. The distance between a sample and a given hyper plane is expressed as:

$$d(x,h) = \frac{|\mathbf{w}^T\mathbf{x} + b|}{||\mathbf{w}||} \tag{2.2}$$

Since in a classification problem $y_n = \pm 1$ and $y_n(\mathbf{w}^T\mathbf{x} + b) \geq 0$ when it is a separable problem:

$$\frac{|\mathbf{w}^T\mathbf{x} + b|}{||\mathbf{w}||} = \frac{|y_n(\mathbf{w}^T\mathbf{x} + b)|}{||\mathbf{w}||} = \frac{y_n(\mathbf{w}^T\mathbf{x} + b)}{||\mathbf{w}||} \tag{2.3}$$

The aim is to maximize the distance between the hyper plane to the closest point, expressed in Equation 2.4. The solution of this optimization problem is not straightforward, however it can be set $\min_n[y_n(\mathbf{w}^T\mathbf{x} + b)]$ to 1 and this converts Equation 2.4 in a more straightforward problem [18]. Furthermore, maximize the inverse of the weight is equivalent to minimize the $\frac{1}{2}\mathbf{w}^T\mathbf{w}$, which is an easier problem (later we will see the reason for the $\frac{1}{2}$) . Finally, we have to deal with an optimization problem expressed in Equation 2.5.

$$\arg\max_{w,b} = \{\frac{1}{||w||}\min_n[y_n(\mathbf{w}^T\mathbf{x} + b)]\} \tag{2.4}$$

$$\begin{aligned} \text{minimize:} \qquad & \frac{1}{2}\mathbf{w}^T\mathbf{w}, \\ \text{subject to:} \qquad & \min_{n=1,\ldots,N} y_n(\mathbf{w}^T\mathbf{x}_n + b) = 1 \end{aligned} \tag{2.5}$$

The equation of constraint can be modified for a new one which makes the problem easier to solve. This new constraint is $y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1$ for $n = 1, \ldots, N$. The solution for the new optimization problem (Equation 2.6) it is also a solution for Equation 2.5, since the nearest point $\mathbf{x}_k$ satisfied $y_n(\mathbf{w}^T\mathbf{x}_k + b) = 1$ otherwise $\mathbf{w}$ would not be the minimum.

$$\begin{aligned} \text{minimize:} \qquad & \frac{1}{2}\mathbf{w}^T\mathbf{w}, \\ \text{subject to:} \qquad & y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 \qquad (n = 1, \ldots, N). \end{aligned} \tag{2.6}$$

Those points which satisfy the condition $y_n(\mathbf{w}^T\mathbf{x}_n + b) = 1$ are the ones that define the maximum margin and are called **support vectors**. This subset of points has all the necessary information to get the optimum hyper plane, if the rest of the points are removed and just kept the subset with only support vectors the result of solving the optimization problem would be exactly the same.

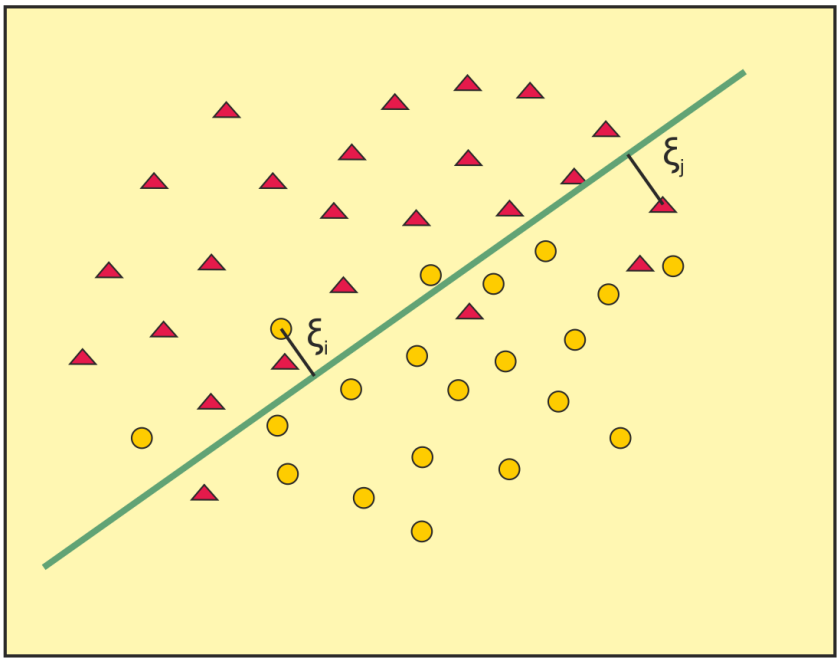


Figure 2.3: Best linear classifier for a non separable data.

2.1.2 Non Separable Data

So far we have been talking about an idyllic world where the data is linearly separable. However, most of the times this is not the case and the data is not separable (see Figure 2.3), so we have to find the best hyper plane which makes fewer mistakes. This means that the margin would be violated for some data samples, that is why is called **soft-margin**. We introduce a new variable ξ which represents the amount exceed for each point. Equation 2.7 represents the so called soft-margin optimization problem.

$$\begin{aligned} \text{minimize:} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{n=1}^{N}\xi_n, \\ \text{subject to:} \quad & y_n(\mathbf{w}^T\mathbf{x}_n + b) \geq 1 - \xi_n \qquad (n = 1, \ldots, N) \text{ and } \xi_n \geq 0. \end{aligned} \tag{2.7}$$

The objective function has a new term $C\sum_{n=1}^{N}\xi_n$ which is the sum of all the excesses multiplied by cost of this margin violation **C**. This new penalty parameter C is defined by the user. A large value of C means a heavy penalization and the margin would be small with very few violations, otherwise when C is small the margin would be larger. A very small margin could produce over-fitting so it is important to define carefully this value.

2.1.3 Non Linear SVM

So far we have been working with linear problems, but most of the time the data is not linearly separable. Figure 2.4 represents a problem which cannot be solved by linear SVM. However, it is possible to work with the same philosophy of SVM by applying a transformation $\Phi : \mathbb{R}^d \to \mathbb{R}^d$ in order to move the data into a higher dimension where they could be linearly separable [20]. Figure 2.5 shows the transformation into a higher dimension space where the problem is linearly separable.

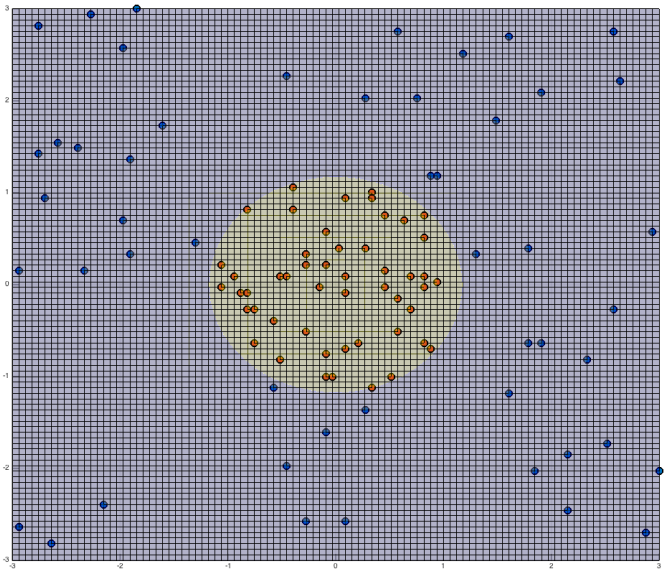


Figure 2.4: Non linear data set.

Consider a function $\Phi : \mathbb{R}^d \to \mathbb{R}^d$ which transforms the data from the input space $\mathcal{X}$ to the new transformed space $\mathcal{Z}$. This new data set will be the input for our SVM, and all the theory before still remains the same. Hence, the optimization problem can be rewritten as Equation 2.8, where $\mathbf{u}$ is a new vector of weights different than $\mathbf{w}$ because in this new feature-space we get a completely different hyper plane.

$$\begin{aligned} &\text{minimize:} && \frac{1}{2}\mathbf{u}^T\mathbf{u} + C\sum_{n=1}^{N}\xi_n, \\ &\text{subject to:} && y_n(\mathbf{u}^T\Phi(\mathbf{x}_n) + b) \geq 1 - \xi_n \qquad (n = 1, \ldots, N) \text{ and } \xi_n \geq 0. \end{aligned} \tag{2.8}$$

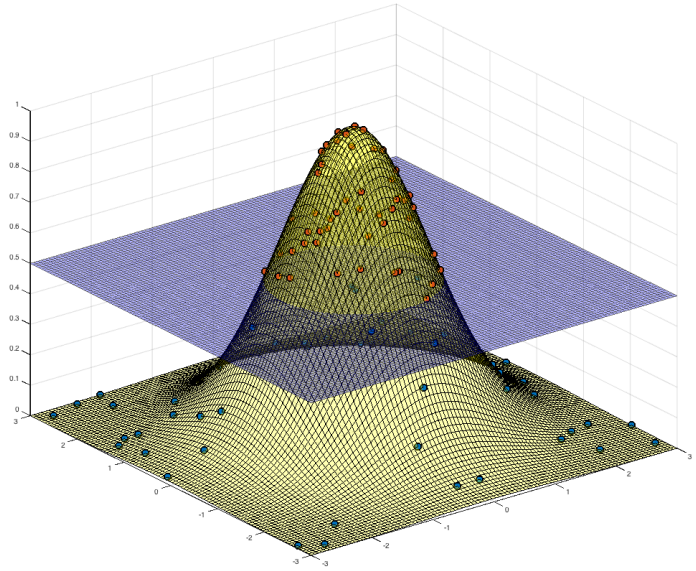


Figure 2.5: Example of a linearly separable problem in the new transformed space.

Equation 2.8 is an example of quadratic programming problem. In order to obtain the solution we can work with Lagrange multipliers [21]. Equation 2.9 is the Lagrange function of our optimization problem.

$$L(\mathbf{u}, b, \xi, \alpha, \mu) = \frac{1}{2}\mathbf{u}^T\mathbf{u} + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n(\xi_n - 1 + y_n(\mathbf{u}^T\Phi(\mathbf{x}_n) + b)) - \sum_{n=1}^{N}\mu_n\xi_n \quad (2.9)$$

Minimize this Lagrange function is similar to optimization problem 2.8. However, this new approach with KKT [22] conditions gives us the tools to solve easily the optimization problem. The KKT conditions should be satisfied for constrained optimization problem, in case of a convex problem like the SVM the KKT conditions are necessary and sufficient. These are the KKT conditions:

- Stationarity:

$$\nabla_{u,b,\xi}L(\mathbf{u}, b, \xi, \alpha, \mu)|_{\mathbf{u}=\mathbf{u}*,b=b*,\xi=\xi*,\alpha=\alpha*,\mu=\mu*} = 0 \quad (2.10)$$

- Complementary slackness:

$$\begin{aligned}\alpha_n y_n(\mathbf{u}^T\Phi(\mathbf{x}_n) + b) &= 0\\ \mu_n\xi_n &= 0\end{aligned} \quad (2.11)$$

Stationarity condition establishes that the derivative of $L(\mathbf{u}, b, \xi, \alpha, \mu)$ respect to $\mathbf{u}$, b and ξ, has to be zero. Hence, applying this condition we get next useful equations:

$$\begin{aligned} \frac{\partial L}{\partial u} &= \mathbf{u} - \sum_{n=1}^{N} \alpha_n y_n \Phi(\mathbf{x}_n) \\ \mathbf{u} &= \sum_{n=1}^{N} \alpha_n y_n \Phi(\mathbf{x}_n) \end{aligned} \tag{2.12}$$

$$\frac{\partial L}{\partial b} = \sum_{n=1}^{N} \alpha_n y_n = 0 \tag{2.13}$$

$$\frac{\partial L}{\partial \xi_n} = C - \alpha_n - \mu_n \tag{2.14}$$

$$L(\alpha, \mu) = -\frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{N} y_n \Phi(\mathbf{x}_n) \alpha_n y_k \Phi(\mathbf{x}_k) \alpha_k + \sum_{n=1}^{N} \alpha_n \tag{2.15}$$

Equations 2.12, 2.13 and 2.14 allow simplifying the expression 2.9 leaving the equation in terms of the Lagrange Multipliers α_n as the only unknown variables. This new representation is called the dual problem[1], as in Equation 2.5 this dual representation takes the form of a quadratic programming of α_n. The problem has changed from one with M variables (input dimension) to a problem of N variables (number of inputs). According to Bishop [18] this change may be seem disadvantageous. However, this new representation allows the SVM working with kernels and it performs efficiently in feature spaces whose dimensionality exceeds the number of data points. In order to get the solution to the primal problem we must maximize $L(\alpha, \mu)$ [2], therefore the optimization problem has this new form:

$$\begin{aligned} &\text{maximize: } -\frac{1}{2} \sum_{n=1}^{N} \sum_{k=1}^{N} y_n \Phi(\mathbf{x}_n) \alpha_n y_k \Phi(\mathbf{x}_k) \alpha_k + \sum_{n=1}^{N} \alpha_n, \\ &\text{subject to: } \quad \begin{aligned} &\alpha_n \geq 0, \\ &\textstyle\sum_{n=1}^{N} \alpha_n y_n = 0. \end{aligned} \end{aligned} \tag{2.16}$$

[1] Equation 2.8 is the primal problem.

[2] See [23][21] to understand why the dual problem has to be maximized

We apply a QP-solver to the problem 2.16 to solve this new dual formula. After that we will have the solution $\hat{\alpha}$ and we can easily compute $\mathbf{u}$:

$$\mathbf{u} = \sum_{n=1}^{N} y_n \hat{\alpha_n} \Phi(\mathbf{x}_n) \tag{2.17}$$

At least one of the sample data will be a support vector so that we can use the KKT condition to get the value of b:

$$y_s(\mathbf{u}\Phi(\mathbf{x}_s) + b) = 1 \tag{2.18}$$

Solving b:

$$b = y_s - \sum_{n=1}^{N} y_n \hat{\alpha_n} \mathbf{x}_n \mathbf{x}_s \tag{2.19}$$

Finally, we have solved the dual problem and the estimation function has the next expression:

$$g(\mathbf{x}) = sign(\mathbf{u}^T \Phi(\mathbf{x}) + b) \tag{2.20}$$

$$g(\mathbf{x}) = sign(\sum_{n=1}^{N} y_n \alpha_n \Phi(\mathbf{x}_n) \Phi(\mathbf{x}) + b) \tag{2.21}$$

At first glance, we see that the dual problem and estimation function always depend on the inner product $\Phi(\mathbf{x}_n)\Phi(\mathbf{x})$. In order to avoid applying the transform to the whole data set and store them, it is used Kernel theory (see Equation 2.22).

$$g(\mathbf{x}) = sign(\sum_{n=1}^{N} y_n \alpha_n K(\mathbf{x}_n, \mathbf{x}) + b) \tag{2.22}$$

Kernel functions allow applying the inner product in the transformed space $\mathcal{Z}$ to inputs from $\mathcal{X}$, without the transformation Φ to the whole data set, just applying directly the kernel function to get the inner products. There is a large amount of Kernels, next lines summarizes some of the most popular ones:

- Gaussian radial basis function: Probably the most employed kernel in the literature, it takes the form:

$$K(\mathbf{x}, \mathbf{y}) = \mathrm{e}^{(\frac{||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2})} \tag{2.23}$$

where $||\mathbf{x}-\mathbf{y}||$ is the Euclidean distance between x and y, and σ is a parameter which control the kernel width. For the rest of the text it is substituted $\gamma = \frac{1}{2\sigma^2}$ in order to work with a simpler kernel expression.

$$K(\mathbf{x},\mathbf{y}) = \mathrm{e}^{(\gamma||\mathbf{x}-\mathbf{y}||^2)} \tag{2.24}$$

- Polynomial: The definition for n-degree polynomial kernel:

$$K(\mathbf{x},\mathbf{y}) = (\mathbf{x}^T\mathbf{y}+c)^n \tag{2.25}$$

Usually, the polynomial degree is $n = 2$, since polynomials with a degree greater than 2 are more likely to have over-fitting problems.

- Sigmoid:

This kernel comes from the Neural Networks where the sigmoid function is the most popular activation function.

$$K(\mathbf{x},\mathbf{y}) = tanh(\alpha\mathbf{x}^T\mathbf{y}+c) \tag{2.26}$$

Parameter α controls the slope of the function and c is a constant.

2.1.4 Support Vector Machines for Regression

Support Vector Machines can be also applied to regression applications. The theory is almost the same, there are just a few differences that are clarified in this section.

Similar to the case of non separable data with the margin violation, we have to take into consideration the error and minimize it. Furthermore, in order to keep the sparseness of the support vector machine an **ϵ-insensitive** loss function similar to the Figure 2.7 is applied to measure the error committed:

$$L_\epsilon(x) = \begin{cases} 0 & \text{if } |\mathbf{w}^T\Phi(\mathbf{x}_n) - y_n| \leq \epsilon \\ |\mathbf{w}^T\Phi(\mathbf{x}_n) - y_n| - \epsilon & otherwise \end{cases} \tag{2.27}$$

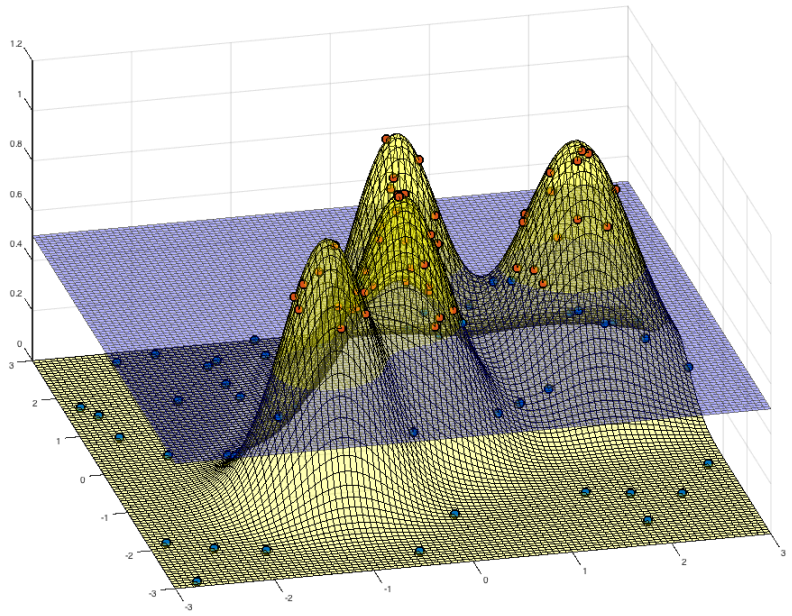


Figure 2.6: Support vector machine with Gaussian Kernel

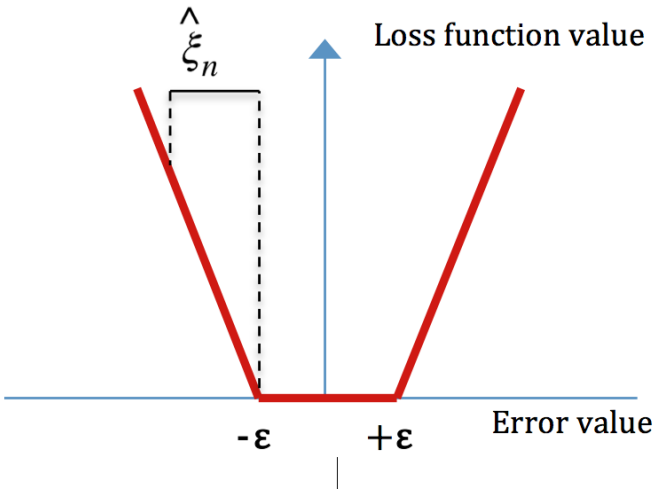


Figure 2.7: Linear ϵ-insensitive Loss Function

Two slack variables are defined to measure this ϵ-insensitive error, $\xi_n \geq 0$ and $\hat{\xi}_n \geq 0$. Where $\xi_n > 0$ correspond to points for which $y_n > g(\mathbf{x}_n) + \epsilon$ and $\hat{\xi}_n > 0$ correspond to points for which $y_n < g(\mathbf{x}_n) - \epsilon$. These changes leave the objective function as:

$$\begin{aligned} \text{minimize: } & C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) + \frac{1}{2}||\mathbf{w}||^2 \\ & y_n \leq \mathbf{w}\Phi(\mathbf{x}_n) + b + \epsilon + \xi_n, \\ \text{subject to: } & y_n \geq \mathbf{w}\Phi(\mathbf{x}_n) + b - \epsilon - \hat{\xi}_n \\ & \xi_n, \hat{\xi}_n \geq 0 \end{aligned} \tag{2.28}$$

Doing the same process done previously with SVM for classification, we can apply the KKT conditions and get the dual representation of the SVR problem.

$$L(\mathbf{w}, b, \xi, \hat{\xi}, \epsilon, \alpha, \hat{\alpha}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{n=1}^{N}(\xi_n + \hat{\xi}_n) - \sum_{n=1}^{N}\alpha_n(\mathbf{w}\Phi(\mathbf{x}_n) + b - y_n + \epsilon + \xi_n) + \sum_{n=1}^{N}\hat{\alpha_n}(\mathbf{w}\Phi(\mathbf{x}_n) + b - y_n - \epsilon - \hat{\xi}_n) - \sum_{n=1}^{N}(\eta_n\xi_n + \hat{\eta}_n\hat{\xi}_n) \tag{2.29}$$

$$\begin{aligned}\frac{\partial L}{\partial w} =& \mathbf{w} - \sum_{n=1}^{N}\alpha_n\Phi(\mathbf{x}_n) + \sum_{n=1}^{N}\hat{\alpha}_n\Phi(\mathbf{x}_n) = 0,\\ \mathbf{w} =& \sum_{n=1}^{N}(\alpha_n - \hat{\alpha}_n)\Phi(\mathbf{x}_n)\end{aligned} \tag{2.30}$$

$$\frac{\partial L}{\partial b} = \sum_{n=1}^{N}(\hat{\alpha} - \alpha) = 0 \tag{2.31}$$

$$\frac{\partial L}{\partial \xi_n} = C + \alpha_n - \eta_n = 0 \tag{2.32}$$

Equations 2.30, 2.31 and 2.32 are replaced in 2.29. Then the Lagrangian is reduced to a simpler expression:

$$L(\alpha, \hat{\alpha}) = -\frac{1}{2}\sum_{n=1}^{N}\sum_{j=1}^{N}(\alpha_n - \hat{\alpha}_n)(\alpha_j - \hat{\alpha}_j)\Phi(\mathbf{x}_n)\Phi(\mathbf{x}_j) + \sum_{n=1}^{N}y_n(\alpha_n - \hat{\alpha}_n) - \epsilon\sum_{n=1}^{N}(\alpha_n + \hat{\alpha}_n) \tag{2.33}$$

The dual variables $\eta, \hat{\eta}$ have been removed, thus the optimization problem has the form:

$$\begin{aligned}\text{maximize: } & -\frac{1}{2}\sum_{n=1}^{N}\sum_{j=1}^{N}(\alpha_n - \hat{\alpha}_n)(\alpha_j - \hat{\alpha}_j)\Phi(\mathbf{x}_n)\Phi(\mathbf{x}_j) + \sum_{n=1}^{N}y_n(\alpha_n - \hat{\alpha}_n) - \epsilon\sum_{n=1}^{N}(\alpha_n + \hat{\alpha}_n)\\ \text{subject to: } & \sum_{n=1}^{N}(\alpha_n - \hat{\alpha}_n) = 0\\ & \alpha_n, \hat{\alpha}_n \geq 0\end{aligned} \tag{2.34}$$

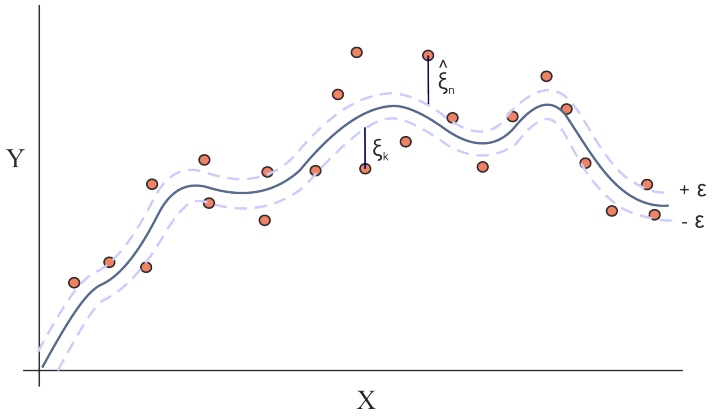


Figure 2.8: Support vector for regression with epsilon - insensitive function.

Solving this dual problem we get the values $\alpha_n*, \hat{\alpha}_n*$ and they can be replaced on 2.30 to get the value of $\mathbf{w}$:

$$\mathbf{w} = \sum_{n=1}^{N}(\alpha_n * -\hat{\alpha}_n*)\Phi(\mathbf{x}_n) \tag{2.35}$$

Hence, the expression for SVR model is:

$$g(\mathbf{x}) = \sum_{n=1}^{N}(\alpha_n - \hat{\alpha}_n)k(\mathbf{x}, \mathbf{x}_n) + b \tag{2.36}$$

Again, it is necessary to calculate the value of $\mathbf{b}$ which is calculated applying Karush-Kuhn-Tucker slackness condition over a support vector ($\xi_n = 0$, $\hat{\xi_n} = 0$). Finally, the regression model is completely defined.

$$b = y_n - \epsilon - \sum_{m=1}^{N}(\alpha_m - \hat{\alpha}_m)k(x_n, x_m) \tag{2.37}$$

2.1.5 Training Method for Support Vector Machines

A Quadratic Programming problem (QP) can be solved in many ways, however there is one method that stands out over every any other for solving Support Vector Machine QP problem. This method is called *Sequential Minimal Optimization* (SMO). SMO is an algorithm proposed in 1998 by Platt in [24] which solves the Support Vector Machine optimization problem in a very fast way in comparison with other previous methods as *chunking algorithm* proposed by Vapnik in [25].

SMO divided the QP problem in the smallest possible optimization problems. Each one of this problems involves only two Lagrange multipliers. The algorithm focuses on one sub-problem and tries to find the optimal values for the two multipliers. The solution

is calculated analytically instead of numerically, and the process is very fast and that is why the whole process is very quick.

For SVM training we use an open source library called LIBSVM developed by Chang and Lin [26] and implements a SMO-type algorithm proposed in [27] which enhance the original one. These library is able to train SVM for classification and regression. It is implemented in Java and C++, but also has numerous interfaces and extensions for Matlab, Weka, Python, etc. For all these reasons this library has been chosen to work with SVM in this Thesis.

2.2 Group Method of Data Handling

Group Method of Data Handling is a self-organized heuristic technique that was developed in the earliest sixties by Ivakhnenko [28]. Self-organized methods do not need a model structure previously defined by the user, but is automatically defined by an algorithm which builds the model based on the cause and effect relationship. Thus, the model is constructed just with the data and not with any personal assumption about the network structure.

The learning algorithm creates a set of elementary nodes whose inputs are the data or the outputs of others elementary nodes, these inputs are processed by a base function $f(\mathbf{x})$ which gives the node output. This algorithm increases the complexity of the model each iteration, until a certain criterion reach a minimum which means the model has an optimal degree of complexity. This **external criterion** has to take care not only of the estimation error but also of the over-fitting, otherwise the model will perform poorly with new samples.

Next section describes with detail the fundamental of GMDH and different kind of GMDH algorithms.

2.2.1 Introduction to GMDH

It is well known that the relationship between any sets of input-output variables can be approximated by Volterra functional series. Volterra series are very similar to Taylor series, the difference between these two systems is that Taylor series give us an output which strictly depends on the input in a particular time. However, Volterra series output depends on past inputs, in other words Volterra series have memory [29]. The discrete Volterra series, also called Kolmogorov-Gabor Polynomial (KGP) has the expression:

$$p = a_0 + \sum_{i=1}^{m} a_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk} x_i x_j x_k + \dots \tag{2.38}$$

where, $\mathbf{x} = (x_1, x_2, \ldots, x_m)$ are the inputs and $\mathbf{A} = (a_0, a_1, a_2, \ldots, a_m)$ are the corresponding coefficients (weights). The KGP is an universal approximator for non-linear functions, since it can approximate any discrete function on a compact data set with the enough polynomial terms, for an specific precision given by the measure of the Mean Square Error [30]:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - p(x_i))^2 \tag{2.39}$$

where y_i are the known outputs and $p(x)$ is the KGP.

The KGP has an important drawback: it is necessary to have a large number of samples and computation time in order to calculate all the coefficients a_i [28]. In order to overcome this drawback, Ivakhnenko proposes a new algorithm which approximates the KGP by using low order polynomials, in an iterative method very similar to the multilayer perceptron neural network. Building a network layer by layer in a bottom-up way until, the output of the current layer has a worse performance than the previous layer. This method does not need so many samples and the time consuming is much lower. In fact, Ivakhnenko proved that using second order polynomials, the complete KGP can be reconstructed. This is the key idea behind the GMDH neural network.

The GMDH has a great variety of types, depending on the constructive network algorithm and elemental function. In the next sections, some of the most representative ones are summarized.

2.2.2 Combinatorial GMDH

This is the basic GMDH algorithm, also called as COMBI, this algorithm explores all the possible inputs combinations generating a bunch of models which are tested with the external criterion. This criterion could be a simple error measure or some metric more complex. There are criterion that takes into consideration the complexity of the model (avoiding over-fitting). Combinatorial algorithm uses single-layered structure and the building process is the next [31]:

- All summation combinations of input arguments are generated $\mathbf{x} = x_1, x_2, \ldots, x_m$.

$$\begin{aligned} y_1 &= a_0^{(1)} + a_1^{(1)} x_1, \\ y_2 &= a_0^{(2)} + a_1^{(2)} x_2, \\ &\vdots \\ y_m &= a_0^{(m)} + a_1^{(m)} x_1 + a_2^{(m)} x_2 \\ &\vdots \\ y_{2^m-1} &= a_0^{(2^m-1)} + a_1^{(2^m-1)} x_1 + \ldots + a_m^{(2^m-1)} x_m \end{aligned} \tag{2.40}$$

  where $a_i^{(k)}$ are the polynomial weights.

- Calculate the coefficients by using least squares error method.

- Check the performance of each partial description by the external criterion with a test set different than the training set used in the second step.

- Choose the model with the minimal value of the criterion. Also it can be chosen the best models and apply another criterion to select the final one.

The combinatorial GMDH has the structure shown in Figure 2.9. The number of base nodes K depends on the number of inputs and it is equal to $K = 2^m - 1$. For example, in Figure 2.9 $K = 7$ since we have 3 inputs. However, if there are 10 the number of nodes will be $K = 1023$. This became a problem when the number of inputs is high because the computational cost of calculate the coefficients of each node is huge.

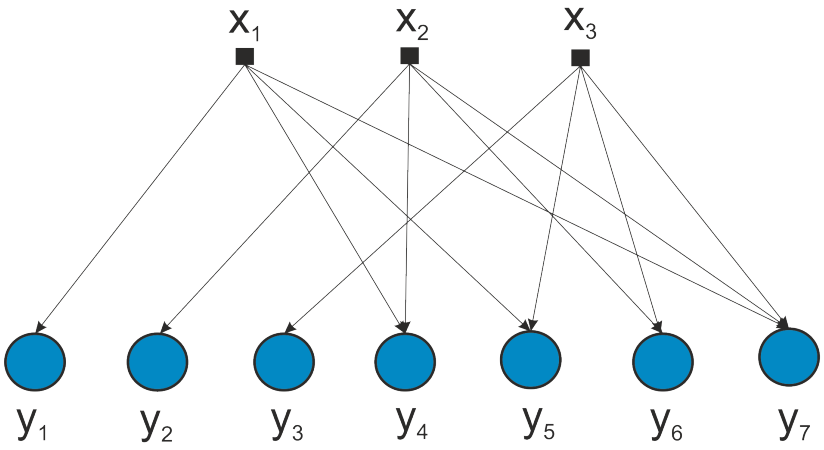


Figure 2.9: Example of COMBI GMDH network.

Ivakhnenko in [31] proposes a recursive method in order to reduce the training time, based on *bordering method* [32] [31] which can optimize the calculation of the coefficients with least squares method:

$$A = (X^T X)^{-1} X^T Y \tag{2.41}$$

The inversion of the matrix $X^T X$ is the hardest part, taking a look to the defined polynomial we see that each one is the same to the previous one but with some extra terms. With the bordering method, it is possible to calculate the inverse in an iterative form, where knowing the matrix $M_k = X_k^T X_k$ it is straightforward to calculate the inverse of M_{k+1} which can be expressed in terms of M_k as:

$$M_{k+1} = \begin{bmatrix} M_k & \mathbf{m}_{k+1} \\ \mathbf{m}_{k+1}^T & a_{k+1} \end{bmatrix} \tag{2.42}$$

Thus the inverse of M_{k+1} takes the form

$$M_{k+1}^{-1} = \begin{bmatrix} M_k^{-1} + \frac{M_k^{-1} m_{k+1}^T m_{k+1} M_k^{-1}}{c_{k+1}} & -\frac{M_k^{-1} m_{k+1}^T}{c_{k+1}} \\ -\frac{m_{k+1} M_k^{-1}}{c_{k+1}} & \frac{1}{c_{k+1}} \end{bmatrix} \tag{2.43}$$

where $c_{k+1} = a_{k+1} - m_{k+1} M_k^{-1} m_{k+1}$. This way increases the calculus speed of each node coefficients and thus reduces the computational cost.

2.2.3 Harmonic GMDH

Harmonic GMDH is another algorithm very useful when we works with time series which show an oscillatory behavior. The base function is composed of the sum of several trigonometric functions, with different frequencies which do not have to be multiple to each others:

$$y = a_0 + \sum_{k=1}^{m} (a_k sin(w_k t) + b_k (cos(w_k t)) \tag{2.44}$$

where a_k and b_k are the polynomial coefficients and w_k, $0 < w_k < \pi$ are the frequencies. This frequencies are unknown and we cannot use least square errors as before because this frequencies are nonlinear parameters. There are several calculation methods to find how many frequencies are needed and estimate their values, such as [31] and [33].

Once the frequencies are known, the calculus of the coefficients in each node can be easily driven with least square error, since these parameters are linear.

In article [30] the authors propose an alternative method to deal with oscillatory processes. Instead of having a trigonometric base functions, they propose to use a multilayered polynomial network adding as an input several Harmonic Terms of the form:

$$H_i(t) = C_i \cdot cos(w_i t - \phi_i) \tag{2.45}$$

In this case the frequencies w_i are calculated by applying Discrete Fourier Transform (DFT) to the data. This harmonics terms are added as input variables thus this trigonometric function are mixed with the data inputs into the polynomial base function. With this scheme the authors assure that:

1. Polynomials are taken as they approximate better the monotonic curvatures as well as the discrepancies and gaps in the time series

2. Harmonics are taken as they approximate better oscillating components, spikes, and critical changes in the series curvature

2.2.4 Multilayer GMDH

This was the first algorithm developed by Ivakhnenko [34] and by far the most popular one. This algorithm builds up a multilayer structure where each node consists of several inputs processed by a polynomial and pass the output to the next layer of nodes. The difference with other algorithms such as neural networks is that the structure of the network (number of layer and nodes per layer) is not a predefined parameter by the user, but the training process defines the structure based on internal and external criteria.

Usually the base function of the node is a second order polynomial of the form:

$$y = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2 \tag{2.46}$$

Equation 2.46 represents the general form of a bivariate second order polynomial, where a_0, a_1, a_2, ..., a_5 are the coefficients and x_i, x_j are two inputs from the data set.

But it is also common the use of first and third order polynomials.

$$y = a_0 + a_1 x_i + a_2 x_j \tag{2.47}$$

$$y = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2 + a_6 x_i x_j^2 + a_8 x_i^2 x_j + a_8 x_i^3 + a_9 x_j^3 \tag{2.48}$$

An interesting read about which polynomial order is better for our application can be find in [35] [36] [37] where the authors study the behavior of these polynomials over multiples applications.

Whatever is the polynomial selected as base function, the multilayer algorithm works in the same way:

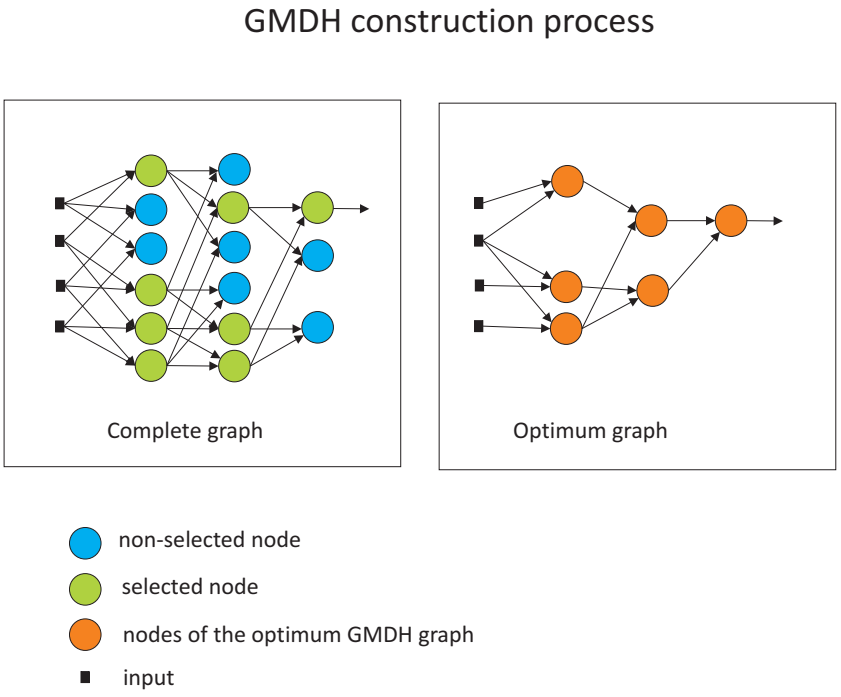


Figure 2.10: Example of multilayer GMDH network.

1. Split the training set into two subsets A and B[3]. Subset A is used to calculate the polynomial coefficients of each node. The subset B, called validation set, checks out the goodness of each polynomial using the external criterion.

2. Make all the possible combination between pairs of inputs. As a result there will be $N(N-1)/2$ nodes. The polynomial coefficients of these nodes are calculated using **Least Squares Error** over the subset A.

3. Next step consists in choosing the best nodes. Every node is tested with the validation set (B) and the external criterion that gives an objective mark to it. Once the layer nodes are ranked, those with the best marks are selected for the next

[3]This division is essential in order to avoid the well-known problem of over-fitting, we always have to avoid use the same set of data for training and testing.

layer. However, there is a question that has not been answered yet, *¿ How many nodes it has to be saved in each iteration?*. In most of the cases, this parameter has to be chosen by the developer, it could be a fixed or a different number each iteration. The number of selected nodes L_i (where i is the layer) affect considerably the performance since the number of nodes to process in the next step depends on it. Furthermore, we cannot take very few nodes, since this could leave some important data out of the process and get a model with a poor behavior.

4. Once we know the best L_i nodes we take their outputs as inputs for the next layer $(i+1)$.

This process is repeated until the minimum value of the external criterion in the current layer is bigger than the previous layer. This means that the new layer does not improve with respect to the previous one. Finally, we get a network complex enough to make good estimations of our problem. Figure 2.10 shows an example of the GMDH construction, and the final optimum GMDH obtained at the end of the process.

2.2.5 External Criterion

Along the description of the different algorithms has been introduced the importance of the external criterion, choosing the *best* nodes of each layer. In the literature there have been presented different alternatives for this criterion.

The most popular is the so called **Regularity Criterion**. This criterion is a simple square error (Equation 2.49) obtained over the samples in the validation set (B) with the polynomial calculated with the subset A.

$$SE = \sum_{k=1}^{B} (y_B - \hat{y}_B)^2 \tag{2.49}$$

where y_B is the output of sample form the data set B and $\hat{y}_B$ is the output of the model created with the set A for that particular sample. The main disadvantage of this criterion is the dependency with the data partition made previously, the results could change greatly depending on this partition.

Several solutions have been planned to face this problem, one of the most popular is the **Bias Criterion**. This criterion is based on choosing those polynomials (nodes) more unbiased, i.e depending less on data partition done. With Bias Criterion the whole data set has to be divided into two subsets with the same size. Each of these subsets is used

to build a model and check the performance of this model with the whole training data. Then the two error measure are subtracted to get the final node score Equation 2.50.

$$BS = |SE_{W/A} - SE_{W/B}| \tag{2.50}$$

where $SE_{W/A}$ and $SE_{W/B}$ are the models squared error over the complete training set $W = A \cup B$.

By using this criterion it is possible to achieve nodes with coefficients highly independent of the partition taken. However, the bias criterion does not give a measure of how good is the model, because this metric is a comparison between two errors. Usually this bias criterion is implemented with others which can measure the error committed.

In our case, we will use bias and regularity criteria in order to get the goodness of both criteria. In [31] the author explains some others interesting criteria.

2.2.6 Coefficients Calculation

In order to calculate the polynomial coefficients in the GMDH, we follow the well known least squares method that can be described as follows: Let $D = (x_i, y_i)_{i=1}^N$ be the set of samples of the problem data set, and let $\{f_j(x)\}_{j=1}^m$ be a set of functions, linearly independent, that will be used as a base of the least squares methods. The goal is to find a function $f(x)$ formed by linear combinations of functions $\{f_j(x)\}$,

$$f(x) = \sum_{j=1}^{m} a_j f_j(x), \tag{2.51}$$

that best represents the samples in D, i.e., in such a way that $f(x)$ minimizes the root mean square error, defined as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(y_k - \sum_{j=1}^{m} a_j f_j(x_k)\right)^2}, \tag{2.52}$$

This is equivalent to directly minimize

$$E_c = \sum_{k=1}^{N}\left(y_k - \sum_{j=1}^{m} a_j f_j(x_k)\right)^2, \tag{2.53}$$

leading to

$$\frac{\partial E_c}{\partial a_i} = \sum_{k=1}^{N} 2\left(y_k - \sum_{j=1}^{m} a_j f_j(x_k)\right)(-f_i(x_k)) = 0,\ i = 1, 2, \ldots, m. \tag{2.54}$$

This is a system of m equations with m unknowns, known as *Gauss normal equations*. Their matrix representation is the following:

$$\begin{bmatrix} \sum_{k=1}^{N} f_1(x_k)f_1(x_k) & \dots & \sum_{k=1}^{N} f_1(x_k)f_m(x_k) \\ \sum_{k=1}^{N} f_2(x_k)f_1(x_k) & \dots & \sum_{k=1}^{N} f_2(x_k)f_m(x_k) \\ \vdots & \vdots & \vdots \\ \sum_{k=1}^{N} f_m(x_k)f_1(x_k) & \dots & \sum_{k=1}^{N} f_m(x_k)f_m(x_k) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{N} f_1(x_k)y_k \\ \sum_{k=1}^{N} f_2(x_k)y_k \\ \vdots \\ \sum_{k=1}^{N} f_m(x_k)y_k \end{bmatrix} \tag{2.55}$$

Note that Expression (2.55) can be in turn rewritten as

$$F^T \cdot F \cdot A = F^T \cdot Y \tag{2.56}$$

where:

$$F = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \dots & f_m(x_1) \\ f_1(x_2) & f_2(x_2) & \dots & f_m(x_2) \\ \vdots & \vdots & \vdots & \vdots \\ f_1(x_N) & f_2(x_N) & \dots & f_m(x_N) \end{bmatrix} \tag{2.57}$$

$$A = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \tag{2.58}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \tag{2.59}$$

Hence

$$A = (F^T \cdot F)^{-1} \cdot F^T \cdot Y \tag{2.60}$$

2.2.7 Regularized least squares errors

A classical way of avoiding overfitting and obtaining less complex and highly generalizable models is to include a regularization penalty in the least squares error method [38]. This leads to the Regularized Average Error (RAE):

$$RAE = \frac{1}{N}\sum_{k=1}^{N}\left(y_k - \sum_{j=1}^{m} a_j f_j(x_k)\right)^2 + \lambda\frac{1}{2}\sum_{m=1}^{M} a_m^2, \tag{2.61}$$

where λ is a parameter that controls the smoothness of the polynomial (the higher λ, the smoothest the polynomial), and should be previously estimated. By optimizing the RAE expression, we obtain the following final expression for the coefficients:

$$A = (F^T \cdot F + \lambda \cdot I)^{-1} F^T \cdot Y \tag{2.62}$$

2.2.8 Multicollinearity

One of the problems that arises with GMDH, and particularly with multilayer GMDH, is the so-called multicollinearity issue [39]. This problem occurs when the independent variables in a regression model, are highly correlated, and they may create instabilities in the regressor coefficients, that produce high unexpected errors with small data or model changes. This is common with multilayer GMDH, since it tends to produce inputs highly correlated in the last layers where the inputs for the regressors are a mix from the original ones. The use of regularized techniques like the one described in the previous section can help minimize the multicollinearity problem.

2.3 Evolutionary Algorithms

Evolutionary algorithms try to simulate the natural selection process explained by Darwing in 1859, through which a population of living beings evolves during generations selecting in each one the best fitted individuals. In other words, the individuals with best features for a particular environment are more likely to survive and to have offsprings for the next generation. This concept is an optimization algorithm itself, where each individual is a possible solution for a given problem. Thus evolving the population during a few generation the algorithm is able to find good solutions with a low computational cost.

Evolutionary computation starts in the 1950s when biologists and computer scientists studied the evolutionary process in nature to find a system who can reproduce that behavior. It was not until the 1960s when Rechenberg [40] introduced the **Evolutionary**

Strategies (ES), a simple evolutionary algorithm that have one single individual and mutates this individual in order to generate a new one. An evaluation function (fitness function) scores both individuals, the best one passes to the next mutation and the other is discarded. Later the algorithm has evolved and there are a great amount of different approaches based on the Rechenberg principles [41].

Almost at the same time, a group of computer scientists L.J. Fogel, A.J. Owens and M.J. Walsh [42] developed another evolutionary algorithm called **Evolutionary Programming (EP)**. This approach is quite similar to evolutionary strategies, mutation is also the only operator that evolutionary programming has. They also are Phenotypic algorithms because can work directly with system parameters instead of create strings representing the system parameters to be optimized (genotypic algorithms). The main difference is that evolutionary programming starts with a population of more than one individual (L), mutation is applied to each individual and after that the population increases to $2 \cdot L$. The fitness of each individual is calculated and the best L individuals are selected for the next generation.

Evolutionary programming and evolutionary strategies are two algorithms which have been applied during decades and today are still being applied in different optimization problems. However, they have a drawback which is the long convergence time since mutation operator is slow finding good solutions. This drawback is overcome by **genetic algorithms** and the crossover operator. The first approach to genetic algorithm was proposed in 1970s by J. Holland [43], who describes an algorithm called **fundamental theorem of genetic algorithms**. Genetic algorithms use also mutation and add two new operators **crossover** and **selection**. Figure 2.11 shows a block diagram of a genetic algorithms with all the steps:

1. Algorithm starts with a population of L individuals (chromosomes) generated randomly.

2. The fitness function evaluates the whole population.

3. Selection operator chooses K chromosomes from the entire population. The fitter the chromosome, the more probability to be selected.

4. Crossover operator generates L new chromosomes by crossing in pairs the K chromosomes selected in the last step.

5. These entire new population pass throw the mutation operator. The mutation operator does not change all the individuals, instead of that it has a probability of mutation which defines the number of chromosomes to be mutated.

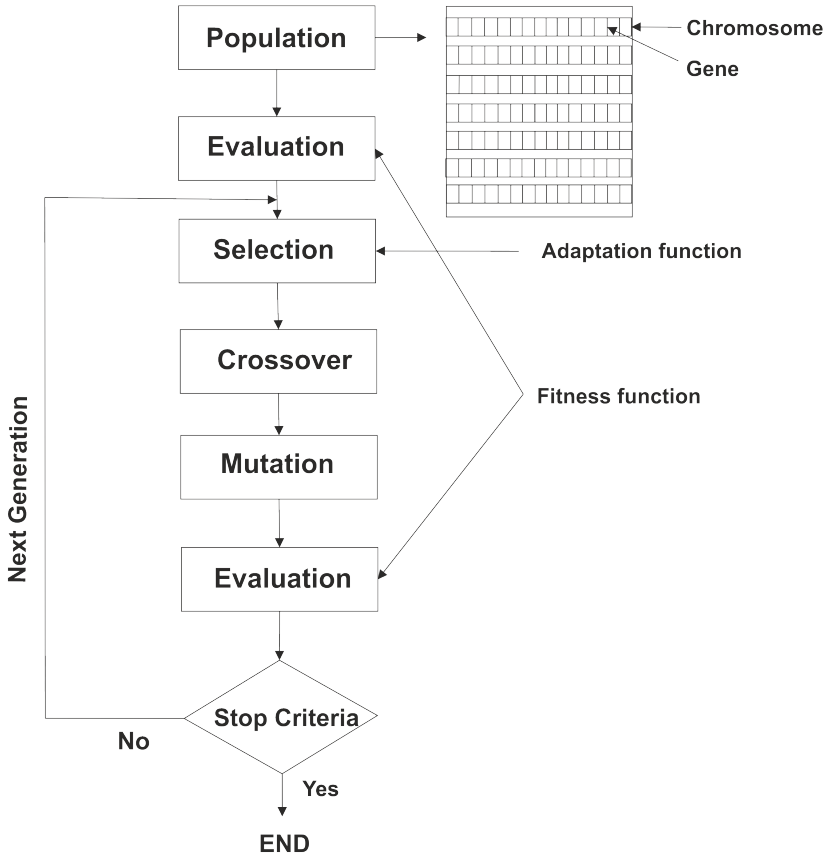


Figure 2.11: Evolutionary Algorithm diagram.

6. The fitness operator evaluates these new chromosomes, if the stopping criterion is satisfied the algorithm finish and the best chromosomes conform the best solution found. Otherwise, the process is repeated.

EA are used as search algorithm in problems where there are several parameters to be tuned. Each chromosome represents a specific value for those parameters. The success of the EA depends on how these parameters are codified: binary, real, natural codification, etc.

These is a general description of EA, this kind of algorithm are very problem dependent that is why there are too many different possible configuration depending on the operators implementation, fitness function, chromosome codification, etc. In each chapter there are a description with the operators and parameters values used in each EA implementation.

2.4 Hyper-Heuristics

Many times the heuristic building process requires complex parameter tune process or the building process depends greatly on the specific problem. Because of this tricky process some inexperience users avoid using more complex machine learning systems and employ simple heuristic easier to implement [44]. Hyper-Heuristics is a new search methodology which has appeared to overcome this problem and creates more generalizable systems. Hyper-Heuristics try to find the configuration or combination of heuristics more suitable for a given problem instead of solving it directly. In other words "a heuristic to choose heuristics" [45].

The first approach to an HH algorithm was made in 1960s by Crowston et al in [46]. Similar approaches were developed in the 80's and 90's. It was not until early 2000s when the term HH was first time mention in a paper by Cowling in [47]. Since then, the popularity of this concept began to increase and it has been applied to many problems, such us production scheduling [48], timetable and rostering [44], vehicle routing [49], channel assignment in cellular communication [50], etc.

According to [51] "A hyper-heuristic can be seen as a (high-level) methodology which, when a particular problem instance or class of instances, and a number of low-level heuristics (or its components) produces an adequate combination of the provided components to effectively solve the given problem(s)". In order to classify Hyper-Heuristic methods there are two different criteria, first one based on the nature of the search and the other based on how the HH receives the feedback. According to the first criteria Hyper-heuristic can be divided into two groups: **HHs for heuristic selection** where the HHs look for the best heuristic from an existing set of heuristics and **HHs for heuristic generation** which are able to build new heuristics employing parts from existing ones. Other classification can be done taking into account the nature of the feedback received by the HH. On the other hand, the HH is called **online** when receives the feedback during the solving process and HH **offline** are those whose feedback is given by the result of a set of training instances.

2.5 Statistical Hypothesis Tests

In order to further study the performance of the different algorithms proposed in this Thesis, it have been carried out statistical comparative analysis. These tests allow concluding if a new algorithm approach has statistical difference on performance compared to the other methods. This section introduces a brief explanation of those methods that will be applied throughout the Thesis.

2.5.1 Student's t-Test

Student's t-test (or just t-test) is a parametric hypothesis test which checks whether two data set have statistically different means, assuming that the samples means follows a normal distribution. This assumption of normality could be a limiting factor for some data set whose distribution is unknown. However for large sets of data the mean can be considered distributed normally by the central limit theorem [52]. There are several implementations of t-test to compare two different samples, in this Thesis is applied **paired t-test**, because of the sample dependence [53]. The paired t-test metric is given by the following expression:

$$t = \frac{\overline{d}}{\frac{s_d}{\sqrt{N}}} \tag{2.63}$$

where $\overline{d}$ and s_d are the mean and standard deviation of the difference between the samples ($d_i = x_i - y_i$), and N is the sample size. T-test is a ratio between the means difference and the standard deviation [54].

After getting the t-test value, we have to check out whether the value is large enough to consider different the two set of data. We need to set a *significance level* α for the test. This value is usually set as $\alpha = 0.05$. Looking at this value in standard normal table we set that $t_{1-\alpha/2} = 1.9673$. This means that we can reject the null hypothesis when

$$|t| > t_{1-\alpha/2} \tag{2.64}$$

which means a p-value less than 0.05.

2.5.2 Kolmogorov-Smirnov Test

Kolmogorov-Smirnov Test (KS-Test) is a non-parametric test, this means that it is not necessary to do any assumption about data distribution. The test can compare a data set with a probability distribution (one sample KS-Test) or compare two samples (two sample KS-test). In our case we use one sample KS-Test to further probe the normality of the samples. Thus it can be decided to use a paired t-test when the KS-Test confirms that the mean data set follows a normal distribution. Using this test, it is also possible to get graph and visualize if the data has or not a normal distribution. Figure 2.12 shows a comparison between an example data set taken it from MATLAB ®[55] and a normal distribution.

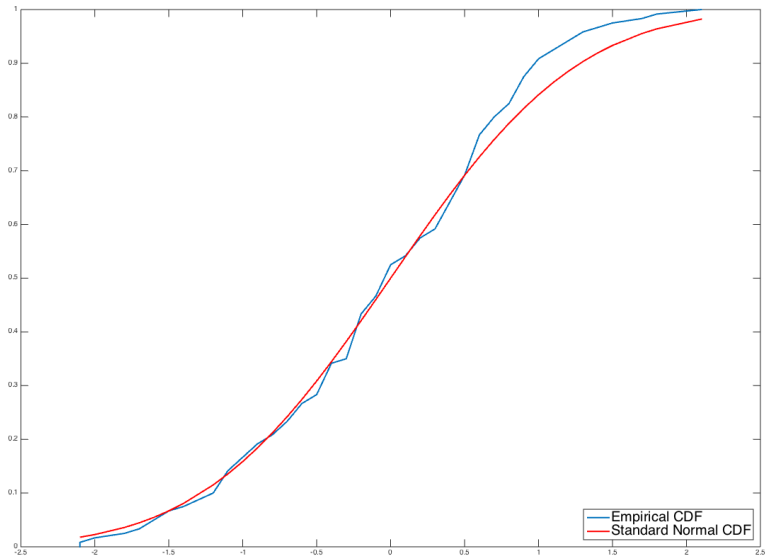


Figure 2.12: Distribution comparison between an example data set and the standard normal distribution.

2.5.3 Sign Test

Sign test is also another non-parametric statistical test [56], which compares the performance of the algorithms in terms of wins, loss and ties. The null hypothesis establishes that one algorithm estimation sample has a probability of $P = 0.5$ of being better than the estimation from the other algorithm. Suppose we have two predictors and the results on terms of win-loss are 13-2. The results are distributed according to a binomial distribution, so the probability of get a result as extreme as this one is given by the sum of probabilities of all these cases:

- Algorithm #1 win 15 loss 0, $P = 3.0518E - 05$.
- Algorithm #1 win 14 loss 1, $P = 4.5776E - 04$.
- Algorithm #1 win 13 loss 2, $P = 3.2043E - 03$.
- Algorithm #2 win 15 loss 0, $P = 3.0518E - 05$.
- Algorithm #2 win 14 loss 1, $P = 4.5776E - 04$.
- Algorithm #2 win 13 loss 2, $P = 3.2043E - 03$.

Thus $P = 0.007385$ is the probability of get a result as extreme as the one of the example. With a level of significance of $\alpha = 0.05$ we can reject the null hypothesis. This test is applied when is not possible to ensure that our sample has a normal distribution, taking into consideration the result of Kolmogorov-Smirnov Test.

2.5.4 Friedman Test

Friedman test [57] is another non-parametric test, similar to one-way ANOVA. Friedman test null hypothesis states equality of medians between the compared algorithms [56]. This test does not work directly with the samples values, but it works with ranks values. Therefore we have to convert the original results to a rank, r_j^i where $i = 1, \ldots, K$ is the algorithm and $j = 1, \ldots, N$ refers to the sample. The best algorithm is ranked with 1, the second with 2 and so on. In case of ties, it is recommended to use the average rank. Once we have ranked the results, it is apply the following expression:

$$F = \frac{12}{N \cdot K(K+1)} \sum_{i=1}^{K} \left(R_i - \frac{N(K+1)}{2} \right)^2 \tag{2.65}$$

where $R_i = \frac{1}{K} \sum_{j=1}^{N} r_j^i$. Friedman statistic is distributed according to a χ^2 distribution with $K - 1$ degrees of freedom.

Iman and Davenport [58] proposed a derivation in Friedman statistic in order to correct the undesirable conservative behavior:

$$F_{ID} = \frac{(N-1)\chi^2}{N(K-1) - \chi^2} \tag{2.66}$$

If one by one, an algorithm is better than an specific control algorithm it is necessary a post-hoc procedure. This post-hoc test give us a p-value to see if the null hypothesis can be rejected or not. One approach is the conversion of the rankings computed by each test by using a normal approximation [56][59]:

$$z = \frac{R_i - R_j}{\sqrt{\frac{K(K+1)}{6N}}} \tag{2.67}$$

This simple post-hoc procedure does not take into consideration any adjustment in α due to the effect of multiple comparison. We use the Holland procedure [60] as p-value adjusted method. Holland adjust α with a process which consists in sorting the p-values $(p_1, p_2, \ldots, p_K)$ from the smallest to the largest. The Holland test rejects the hypothesis $(H_1, H_2, \ldots, H_K)$ if *'i'* is the smallest integer so that $p_i > 1 - (1 - \alpha)^{K-i}$.

Chapter 3

Evolutionary Optimization of Multi-Parametric Kernel ϵ-SVR

In Section 2.1 we explain deeply the theory behind the Support Vector Machine for classification and regression (SVR). However, note that calculation of the parameters **C**, ϵ (in case of regression) and γ (in case of Gaussian kernel) is an open question which was not tackled in that description. These parameters also called, **hyper-parameters**, cannot be calculated by an exact method, so a search algorithm must be applied in order to get the best possible set. It is important to note that the training time heavily depends on the choice of these hyper-parameters so the more efficient is the search algorithm the better performance of the SVM in terms of computation time.

Usually, the search algorithms used to obtain SVM hyper-parameters are based on Grid Search (GS) [61], where the search space of parameters is divided into groups of possible parameters to be tested (an uniform partition of the search space is generally considered). This algorithm can be easily implemented, but it has an important drawback: since the number of combinations is large, the training time becomes very high, even considering only the three standard hyper-parameters defined before **C**, ϵ and γ.

Different works have dealt with the problem of considering different kernels in SVMs in general ([62], [63], [64], [65]) and also specifically for SVR algorithms ([66], [67], [68]). Depending on the problem, the choice of the kernel (or group of kernels) function is key to obtain good results. As mentioned, several works have tackled the inclusion of multi-parametric kernels in SVMs. There are a number of approaches to multi-parametric kernel optimization in SVM that use GS [67] or gradient-based approaches [69][70], though it has been seen that these approaches have some problems such as convergence to suboptimal solutions. For this reason, evolutionary-based approaches are the most used techniques in the optimization of multi-parametric kernel, obtaining good results in this task [71][72]. In fact, there are different works in the literature tackling similar

problems of multi-parametric kernels optimization that have different names, such as the optimization of *anisotropic* Gaussian Kernels [63][73], where a different γ parameter is used in each feature of the input data, or Mahalanobis kernels [74] [75] that consider the same idea, but formalized by means of the Mahalanobis distance and the associated kernel. These approaches are focussed on the optimization of SVMs for classification problems. In [76] several evolutionary strategies are applied to a problem of tuning multi-parametric generic Gaussian Kernels, considering also a SVM for classification. In [64] several evolutionary strategies are used to solve a slightly different problem of kernel optimization with multiple parameters: in this case, the authors propose the linear combination of several Gaussian Kernels, each one with a different γ parameter. This approach is called by the authors as multi-scale kernel.

A generalization of the standard Gaussian kernel has been applied to SVR machines in this Thesis. This approach follows the methodology described in [63][74][75] for classification problems, where the Gaussian kernel function depends on several γ_i parameters, instead of considering only one general γ. In this case, each dimension of the feature space is related to a different γ_i parameter, i.e, the width of the Gaussian kernels is different depending on the dimension considered (M dimensions or features in the input samples). This generalization can be implemented by considering M + 2 parameters in the SVR (C,ϵ,γ_i, $i = 1, \ldots, M$). Basically the idea is to maintain the C and ϵ parameters which define the SVR optimization problem, and M different γ parameters, one for each dimension of the data. This way the final SVR model should be more effective, since it has better ability to discriminate important data components. Note that the application of a multi-parametric kernel in SVR involves several important aspects: first, there are more SVR parameters to be obtained (M-1 more than in the case of the standard SVR approach). On the other hand, note that GS approach is computationally not affordable in this case, so meta-heuristic approaches such as evolutionary algorithms can also be a good option in the SVR case [77], as they have shown in the case of classification problems [69] [76].

Therefore, in this chapter we propose an evolutionary algorithm to carry out the complete SVR hyper-parameters search when considering a multi-parametric kernel SVR. We present the encoding proposed, the different operators that have been applied, and the validation methodology followed to avoid over-training of the machine. As a novelty, we also propose a reduction of the hyper-parameters search space by applying different lower and upper bounds to the multi-parametric kernel.

3.1 SVR Multi-parametric Gaussian Kernel

The expression obtained in the previous chapter for SVR is the following:

$$g(\mathbf{x}) = \sum_{n=1}^{M}(\alpha_n - \hat{\alpha}_n)k(\mathbf{x}, \mathbf{x}_n) + b \tag{3.1}$$

In this chapter we will work with a Gaussian kernel, whose regular expression is:

$$K(\mathbf{x}, \mathbf{y}) = \exp\gamma(||\mathbf{x} - \mathbf{y}||^2) \tag{3.2}$$

Considering a multi-parametric Gaussian kernel function, with different γ's for each feature of the input space:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\sum_{m=1}^{M}\sum_{n=1}^{M}\gamma_{mn}(x_{im} - x_{jm})(x_{in} - x_{jn})\right) \tag{3.3}$$

where M is the number of dimensions in the feature space, γ_{mn} represents the width of the Gaussian function in each direction in the feature spaces. Without loss of generality, in this work we consider that the matrix with values of γ_{mn} is a diagonal matrix, so the simplified Gaussian kernel considered is then:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\sum_{m=1}^{M}\gamma_m(x_{im} - x_{jm})^2\right) \tag{3.4}$$

Thus, the hyper-parameter of the proposed multi-parametric kernel SVR are C, ϵ and γ_m, $m = 1, \ldots, M$.

3.1.1 Hyper-parameters search space reduction using theoretical bounds

Using this model, we carry out a search space reduction for the multi-parametric Gaussian kernel based in the bounds proposed by [78] for the standard kernel function (only one γ parameter case). Following, the same reasoning we extend the bounds for γ_m to our multi-parametric Gaussian Kernel.

In [78] the authors study the influence of each hyper-parameter in the estimation model and the construction process, in order to determine which values are useless:

- **Parameter C**: It is the regularization parameter which controls outliers weight into the optimization problem. C is always positive so the lower bound is $C \geq 0$. On the other side, when C is very high the training time increases considerably because the model has to fit very well each sample in order to have very few outliers.

Ortiz et al, have found an upper bound based on the relationship between C and γ, which is defined in this expression:

$$C < \frac{y_i^{max} - b - \epsilon}{1 - \frac{1}{N-1} \sum_{m=1, m \neq i}^{N} K(x_i, x_m)} \quad [1] \tag{3.5}$$

- **Parameter** ϵ: This value defines the maximum distance between the estimation value and the real one without any penalty. For this parameter there is an exact bound described in [79]. This bound is:

$$\epsilon = \frac{y_i^{max} - y_i^{min}}{2} \tag{3.6}$$

The reasoning behind the bound is straightforward: when ϵ is greater than a half of the maximum distance between samples, it means that there is not support vector. Thus the SVM is just the bias parameter and if we increase ϵ the result is the same as model with a bias.

- **Parameter** γ: This parameter controls the Gaussian width, low values of γ means wider Gaussian and vice versa. According to [78] the maximum value of γ is the one in which the influence between support vectors is negligible. This influence is considered negligible for a value of 0.1% of the amplitude (A). The closest input vectors are the most extreme case, so we can find them to calculate the upper bound:

$$A \cdot \exp(-\gamma \cdot d_{min}^2) \leq 0.001 \cdot A \tag{3.7}$$

where:

$$d_{min} = min_{i \neq j} d(\mathbf{x}_i, \mathbf{x}_j) \tag{3.8}$$

Thus the parameter bound is:

$$\gamma \leq -\frac{log_e(0.001)}{d_{min}^2} \tag{3.9}$$

[1] y_i^{max} is the highest output value

In our study case with a multi-parametric Gaussian Kernel we need to extend this bound to the vector $(\gamma_1, \ldots, \gamma_M)$. In this case we have the following expression:

$$\sum_{m=1}^{M} \gamma_m \cdot (x_{im} - x_{jm})^2 \leq 0.001 \cdot A \tag{3.10}$$

Now we isolate the value of one hyper-parameter γ_l as a function of the rest of hyper-parameters:

$$\gamma_l \leq -\frac{log_e(0.001) - \sum_{m=1, m\neq l}^{M} \gamma_m \cdot (x_{im} - x_{jm})^2}{(x_{il} - x_{jl})^2} \tag{3.11}$$

As in [78] the bound of each hyper-parameter in the kernel function is related to their closest vectors. In this case, related to the closest vectors considering just the feature m. Note that the previous equation achieves the maximum value if the rest of γ_m parameters are supposed to be zero. In this way, we define the minimum distance between features as:

$$d_{min}(x_{im}, x_{jm}) = min_{i\neq j}(x_{im} - x_{jm}) \tag{3.12}$$

Thus the bound for each γ_m is:

$$\gamma_m \leq -\frac{log_e(0.001)}{d^2_{min}(x_{im}, x_{jm})} \tag{3.13}$$

3.2 Optimization of the proposed multi-parametric Gaussian kernel function

Grid search is perhaps the most used algorithm in SVR to obtain a good set of hyper-parameters. GS establishes a grid of possible points to be explored, keeping parameters which show the best performance in a validation data set. There are recent works in the literature which have proposed the application of meta-heuristic techniques, such as evolutionary approaches, to carry out this search [80][81], obtaining good results even in the case of the traditional SVR with three parameters (C,ϵ and γ). However, note that in terms of training time it would not be manageable to extend the GS to multi-parametric kernels, since the number of parameters grows to $M + 2$ parameters (instead of only 3).

We then propose an evolutionary algorithm to obtain the best possible set of parameters for the multi-parametric Gaussian kernel in SVR. Each individual in the evolutionary

population is defined to be a vector representing the hyper-parameters (C,ϵ,γ_1,...,γ_M) in real encoding.

Several parameters have to be previously defined in order to tune the evolutionary algorithm, these parameters can heavily affect the performance of the algorithm and training time. First of all, we have to define the number of individuals in the population, each individual represents an instance of the vector (C,ϵ,γ_1,...,γ_M). For this implementation a total of 25 individual are generated randomly at the beginning of the algorithm. We also have to bound the number of iterations in order to stop the algorithm in case this does not converge to a solution: this maximum is $\aleph = 35$.

3.2.1 Fitness

This operator is in charge of the individual evaluation in order to get a ranking of the best ones of each iteration. As fitness function, we have used the error obtained by means of a K-Fold cross-validation procedure (note that this is the same scheme used as validation method in the grid search).

$$F_i = \frac{1}{RMSE} = \frac{1}{\sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(Y_k - \hat{Y}_{ik}\right)^2}} \tag{3.14}$$

where Y is the real output, $\hat{Y}_i$ is the predicted output by the individual i and N is the number of samples of the training set. This function has to be executed for every new individual and before the selection process.

3.2.2 Selection operator

This operator is responsible for generating an intermediate population by selecting individuals within a population. This selection is done randomly, where the individuals with better fitness value are more likely to form part of the intermediate population.

The roulette-wheel procedure has been applied as selection mechanism [82] many times in the literature. The roulette wheel is divided in segments where each one represents an individual and the size is proportional to the fitness value of the individual, in other words, the probability of survival is defined to each individual depending on its fitness.

$$f_i = \frac{F_i}{\sum_{k=1}^{N} F_k} \tag{3.15}$$

Note that these values are normalized between 0 and 1. Once the roulette is defined, we can sample it in order to create the intermediate population with the parents of the new population.

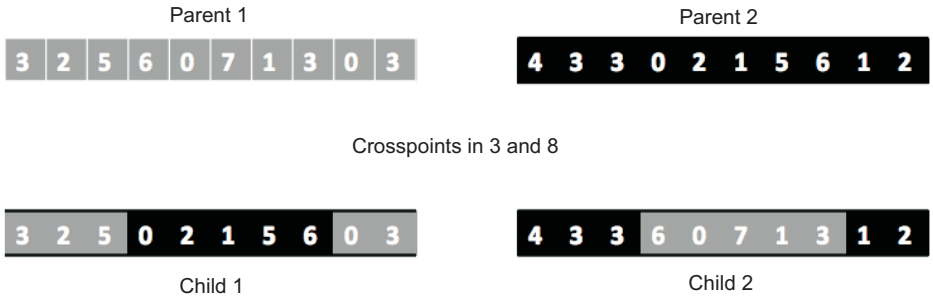


Figure 3.1: Example of the two-point crossover implemented.

3.2.3 Crossover and mutation operators

Once the selection has been made, the crossover operator is applied. This operator selects a couple of individuals and recombine them in order to create a new one by exchanging their genetic material. The crossover probability has been set to a value of 0.7, so the 70% of the individuals in the population will be involved in this operator. The individuals of the population are then chosen in pairs. For each couple, a random number is generated. If it is less than the crossover probability, the crossover is carried out. The process of crossover carried out is a two-point crossover. An example of the crossover operation is in Figure 3.1.

On the other hand, the mutation operator is applied after crossover. In this case, the probability of mutation is set to 10%. When an individual gene must be modified, a new random value different from the current one is generated and assigned to that gene. In this way, diversity in the population is obtained, and possible local minimums can be avoided.

3.2.4 Elitism

A genetic algorithm attempts to reach to the individual with the highest fitness value. In order to avoid the elimination of the optimal individual randomly, it is introduced an operator called Elitism. This operator is based on keep in a privilege position in the population the best individual of every generation, so that it cannot be deleted or modified by other operators. Thus it is ensured that the best individual, which has appeared during the execution survives. In occasions, more individuals are saved in every iteration of the algorithm. In consequence, it is achieved a high level of goodness in the populations, since thanks to the elitism there are always high quality genes. On the other hand, keeping a large number of previous best individuals leads to a poorly diverse

population and it would be easier to fall into a local minimum. In our Genetic Algorithm, the three best individuals are kept as elitism operator for the next generation.

3.2.5 Stopping Criteria

We have three different stopping criteria in our algorithm. The first one consists in evaluating the differences in fitness between the best individual in the population and the third best one. Then, along the evolution, this fitness difference between the best and third best individuals are obtained. If this difference is less than T times the first difference found during at least G generations, we stop the algorithm. Note that this criterion is related to the diversity in the population of the evolutionary algorithm. This diversity-based criterion has a drawback: if the initial population is good enough, it is possible that we cannot reach to the specific reduction in fitness difference between the best and third best individuals in the population.

In order to solve this point, we introduce a second stopping criterion based on the best individual in the population, in such a way that if the fitness of this individual remains constant during $\mathcal{K}$ generations, we also stop the algorithm.

The third stopping criterion is based on a maximum number of generations $\aleph$. If the algorithm reaches this maximum number of generations, then it is stopped.

3.2.6 Repair Function

When the γ_m values are quite different, it means that the chromosome may be generating a model with over-fitting in some of the coordinates. In order to keep the difference between the γ_m parameters of an individual in a reasonable level, a repair function is used to correct high variations among them. To do this, the repair function adjusts the value of the different γ_m in order to keep the maximum deviation among them to be less than a given threshold, in this case 0.2.

Table 3.1 summarizes the evolutionary algorithm specifications.

3.3 Experimental Part

This section presents the experimental part of the chapter, structured in several subsections: first, we consider different experiments in standard repository databases, where we discuss the performance of the evolutionary multi-parametric kernel SVR in comparison to the standard SVR with GS. Section 3.3.1 briefly describes the methodology followed in these experiments, and Section 3.3.2 contains the results obtained in them. Sections

Table 3.1: Summary of the evolutionary algorithm parameters.

Population	25
Selection	Roulette wheel
Crossover	Two-points crossover
Elitism	Yes
P_c	0.7
P_m	0.1
γ_m threshold (repairing)	0.2
T	10
G	5
$\aleph$	35
$\mathcal{K}$	6

3.3.3 and 3.3.4 discuss about two real applications of temperature and ozone forecasts, using the proposed approach in this chapter.

3.3.1 Simulation methodology

In an initial step, prior to the application of the SVR, all the data sets are treated in order to homogenize them. First, the samples with missing values of each data set are eliminated to simplify their treatment, and the non-numerical attributes are substituted by integer numbers. Then, each data set is divided into two sets, the training and test sets, by selecting 80% of instances for the training set and the rest (20%) of the instances for the test set. In a final common step, the data are normalized to zero mean and unit variance.

In order to further study the performance of the different approaches proposed in this Thesis, we have carried out statistical tests. The statistical analysis is based on paired t-test whether the result has a normal distribution (checked with Kolmogorov-Smirnov test), otherwise we use sign-test. The tests have been running over 5 permutations of each data set, with 20 runs per permutation for the evolutionary algorithm. The RMSE and time consumption values presented are the average of these 5 permutations.

Regarding the validation methods for choosing the hyper-parameters, we use a K-Fold cross-validation with $K = 10$, i.e., we divide into K folds the train set and evaluate each fold with the model and training with the rest of the folds. For speeding up the training of the models for each fold, we have modified the functions related to the matrix kernel in the LIBSVM library, in such a way that we keep in memory the complete kernel matrix, and it is only modified when the parameters γ_m are changed. This allows calculating

only one kernel matrix for all K-Fold training and all the range of iterations with the same γ_m parameters.

The experiments have been performed on three different implementations: standard SVR with three hyper-parameters, optimized with a GS and bounds proposed in [78] and two multi-parametric kernel SVR optimized with the proposed EA, one with all the bounds included the new one proposed for the multi-parametric Gaussian Kernel and another without the new bounds proposed.

All the experiments performed in this chapter have been running in the same machine: an Intel Xeon E5645 (6 cores of 2.4 GHz), 12 GB RAM, 500 GB hard disk of 5400 rpm and Windows 7.

3.3.2 Experiments in standard repository data sets

The data sets considered in this Thesis are forecasting problems obtained from the UCI machine learning repository [83] and the data archive of Statlib [84]. Table 3.2 shows the main properties of the selected data sets. Mortpollution measures the age-adjusted mortality rate, taking into account some properties of pollution. Bodyfat estimates the percentage of body fat determined by underwater weighing and various body circumference measurements. Betaplasma and Retplasma study the relationship between personal characteristics and dietary factors, and plasma concentrations of beta-carotene and retinol which might be associated with increased risk of developing cancer. Autompg concerns city-cycle fuel consumption in function of parameters of the car. Housing concerns housing values in suburbs of Boston. Concrete estimates the concrete compressive strength through the components of the mixture. Finally, Abalone set can be handled as a multi-classification or regression problem, and predicts the age of abalones from their physical measurement.

The results obtained with these public repositories with the three different implementations are shown in Table 3.3. As it was said, we use the standard SVR (with 3 hyper-parameters), optimized with a GS approach as comparison algorithm. In terms of accuracy, the different approaches have results very close. In 5 of 8 data set the multi-parametric kernel have lower error than standard SVR, obtaining an error quite close to the standard SVR with GS in the other 3 data sets. The computation time is another important factor to be evaluated in this work. As the data sets are sorted by the number of samples, the increment of training time is evident as the number of samples grows. Note that the computation time with the evolutionary algorithm is much lower than the one provided by the GS in the standard SVR, in all the problems tackled. Of course, this is due to the fact that the evolutionary algorithm evaluates less points in

Table 3.2: Data sets used in experiments carried out.

Data set	Samples	Attributes (N)	Repository
MortPollution	60	15	StatLib
Bodyfat	252	13	StatLib
Betaplasma	315	12	StatLib
Retplasma	315	12	StatLib
Autompg	392	7	UCI
Housing	506	13	UCI
Concrete	1030	16	UCI
Abalone	4177	8	UCI

Table 3.3: SVR performance and training time for the standard SVR (with a GS) and the multi-parametric kernel SVR (with the EA).

	Grid Search (standard SVR)		EA with bounds (multi-parametric kernel SVR)		EA without bounds (multi-parametric kernel SVR)	
Data set	Error	Time (s)	Error	Time (s)	Error	Time (s)
Mortpollution	48.4622	1.7796	**47.0041**	0.8944	49.1451	3.9109
BodyFat	**0.01056**	20.3806	0.01069	12.3272	0.01067	21,9863
Betaplasma	185.7698	22.1930	183.8452	20.5092	**183.5862**	70.2850
Retplasma	221.4400	21.4288	**220.5198**	19.3134	221.5484	66.1389
Autompg	**2.8191**	40.1328	2.8425	29.2307	2.8588	134.5198
Housing	3.6175	64.1164	**3.6107**	44.0433	3.8571	169.3014
Concrete	28.7645	370.2846	28.6867	144.1444	**28.6021**	817.3912
Abalone	**2.0700**	2932.6520	2.0774	1348.7630	2.0738	7870.0225

the search space than the GS. This point, together with the performance of the multi-parametric kernel SVR in RMSE, shows that this approach is able to outperform the performance of the standard SVR in regression problems. Thus it is a very good option to obtain accurate results with a reduced computation time in these kind of problems. Results 3.3 also compare the performance of the evolutionary multi-parametric kernel SVR with and without the new theoretical bounds to reduce the hyper-parameters search space. These results show the good performance of the SVR bounds, which reduce the computation time without modifying its performance in terms of RMSE. Figure 3.2 shows a comparison of computation time of the different algorithms compared (multi-kernel SVR with and without bounds, and standard SVR with GS). It is possible to see the effect of including the new theoretical bounds in the multi-kernel case (optimized with an EA).

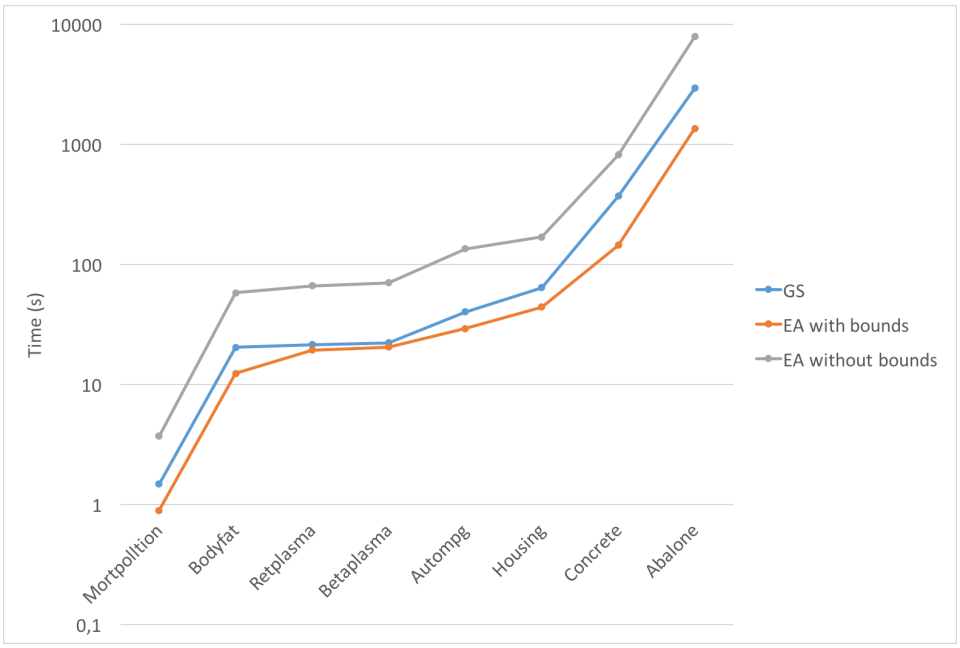


Figure 3.2: Average time (seconds) of the multi-kernel SVR (with and without bounds) and standard SVR (GS) in the UCI data sets.

For a deeper analysis of the multi-parametric kernel SVR, we have carried out an statistical comparative analysis of standard SVR (with GS) and the multi-parametric kernel SVR (optimized with the evolutionary algorithm). Table 3.4 shows the result of the statistical tests, where in five of them there are significant differences. Multi-parametric kernel approach is better in three data sets (Mortpollution, Betaplasma and Retplasama) and worse in two (Bodyfat and Autompg). In the rest of the data sets there are not differences, even in the win-loss-tie metric the results are close.

3.3.3 Temperature forecasting at Barcelona's airport

We further test the multi-parametric kernel SVR performance in a real forecasting problem, specifically we consider the short-term temperature forecasting (1 h,2 h, . . ., 6 h) in the airport of Barcelona ("El Prat"). We have available meteorological data from the Spanish Meteorological State Agency (AEMET), organism that has a measuring station belonging to the network of meteorological observatory at "El Prat" airport (41º17'34"N, 2º04'12"W). This station measures different meteorological variables given in Table 3.5, where we show the variables and their measurement units. Note that we include two extra variables apart from the meteorological ones, i.e., the synoptic condition and a monthly cycle. The synoptic condition or GWL (from the German word Grosswetterlage),

Table 3.4: Statistical tests (t-test or sign-test) with significance ($\alpha = 0.05$) over 10-Fold GS and the multi-kernel evolutionary SVR, for public data sets. W-L-T stands for win, lost, tie in the set of 20 experiments carried out in each permutation.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
Mortpollution	0.0048	t-test	1	0.0357	61-39-0
Bodyfat	0.1042	t-test	1	3.2994e-09	21-79-0
Betaplasma	0.3385	t-test	1	9.9542e-05	73-27-0
Retplasma	0.4488	t-test	1	0.0292	58-42-0
Autompg	0.6357	t-test	1	3.7558e-04	31-69-0
Housing	0.0105	sign-test	0	0.1336	58-42-0
Concrete	0.0077	sign-test	0	0.7642	48-52-0
Abalone	5.4800e-06	sign-test	0	0.9203	49-51-0

represents the climatology situation at a large scale (synoptic scale). We have considered in this case the classical Hess Brezowsky Classification (HBC) to include the GWL condition in the system. Also a monthly cycle to consider variations in solar activity due to the different seasons of the year is included among the prediction variables, given by the following expression:

$$MC_i = \sin\left(\frac{(m_i - 1)\pi}{12}\right), i = 1, \cdots N, \tag{3.16}$$

where m_i represents the month corresponding to the sample i and N represents the number of total samples. The considered data in Barcelona start on January 1st, 2009, and they extend until February 19th, 2009. We consider hourly values of all the prediction variables, and a short-term temperature forecasting problem (from hourly to 6 hour time horizon prediction). Therefore, the data set consists of 1200 samples, where 80% of them form the training set and the other 20% are the test set. The accuracy results obtained are referred to prediction errors in the test set.

Table 3.6 shows the results obtained using the proposed multi-parametric kernel SVR, compared to the results of the classical SVR, with a GS to set its parameters. It is easy to see how the proposed multi-parametric kernel SVR obtains results comparable or better than the standard SVR in terms of RMSE, in almost all the time-horizons predictions considered. The RMSE obtained is quite low, increasing moderately with the time-horizon, as expected. This indicates that the prediction obtained with the SVR is accurate. It is also important that the computation time of the multi-parametric kernel SVR is much better than the classical SVR. This means that the evolutionary algorithm used to set the parameters of the multi-parametric kernel is able to quickly converge to a very good solution. Table 3.6 also shows the effect of the inclusion (or not) of theoretical

Table 3.5: Available data variables, units of measurement and measuring instruments. SC stands for Synoptic Condition, and MC stands for Monthly Cycle.

var. number	Met. variable	units	Instrument
1	Relative humidity	%	Relative humidity probe Vaisala HMP45D
2	Precipitation	mm	Precipitation Transmiter Thies 5.4032.35.008
3	Pressure	hPa	Barometer Vaisala PA21
4	Global radiation	kJ/m^2	Pyranometer Kipp & Zonen CM11
5	Air temperature	^{o}C	Temperature probe Vaisala HMP45D
6	Wind speed	m/s	Anemometer Vaisala WAA15
7	Wind direction	Degrees	Wind vane Vaisala WAV15
8	SC	-	HBC
9	MC	-	Equation (3.16)

Table 3.6: SVR performance and training time for the standard SVR (with a GS) and the multi-parametric kernel SVR (with the EA) in the problem of temperature prediction in Barcelona's airport.

	Grid Search (standard SVR)		EA with bounds (multi-parametric kernel SVR)		EA without bounds (multi-parametric kernel SVR)	
Gap	RMSE	Time (s)	RMSE	Time (s)	RMSE	Time (s)
1 hour	**0.5078**	315.4014	0.5319	157.8238	0.5342	208.7516
2 hours	0.9077	317.2938	**0.8891**	220.3528	0.8894	235.0006
3 hours	1.0578	328.6482	1.0259	220.7146	**1.0251**	272.4376
4 hours	1.1546	358.4986	1.1257	207.3694	**1.1256**	309.7138
5 hours	1.1737	339.6676	1.1595	233.9528	**1.1544**	334.7152
6 hours	1.2380	356.2790	**1.2140**	251.4552	1.2171	349.3596

bounds in the multi-parametric kernel with evolutionary optimization. it is easy to see how the computation time is much lower in the case of considering bounds to reduce the hyper-parameters search space.

Table 3.7 shows the results of the statistical tests carried out. The results show that the multi-parametric kernel SVR performs statistically better than the standard SVR (GS) in all the time horizons except for 1 hour time horizon. These results confirm that multi-parametric evolutionary SVR approach is superior to the standard SVR with GS for this particular problem. Regarding the computation time, multi-parametric evolutionary SVR with EA is much faster, around $30\% - 50\%$ less than the standard SVR with GS. Thus, we can conclude that the proposed multi-parametric kernel SVR offers a good performance in terms of error in the prediction with an excellent computation time for

Table 3.7: Statistical tests (t-test or sign-test) with significance ($\alpha = 0.05$) over 10-Fold GS and the multi-kernel evolutionary SVR, for Barcelona's airport temperature data set.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
1 hours	0.2138	t-test	1	1.1218e-09	26-74-0
2 hours	0.3881	t-test	1	1.1209e-05	62-38-0
3 hours	0.4669	t-test	1	3.8245e-15	85-15-0
4 hours	0.0768	t-test	1	2.3492e-09	79-21-0
5 hours	0.7584	t-test	1	0.0268	64-36-0
6 hours	0.5433	t-test	1	8.6891e-09	80-20-0

its implementation in real prediction systems, such as the temperature prediction shown in this work.

Finally, Figure 3.3 shows the prediction given by the multi-parametric kernel SVR, compared to the real values of temperature. As we can see, the prediction fits quite well to the real signal.

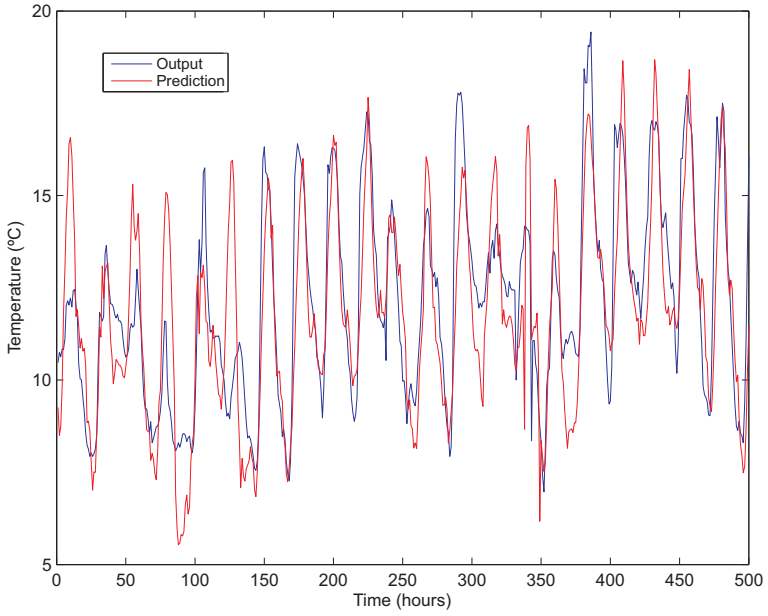


Figure 3.3: Multi-parametric kernel SVR output for 6 hours time horizon temperature prediction at Barcelona's airport.

3.3.4 Total ozone content (TOC) prediction at the Iberian Peninsula

The last real prediction problem that is considered in this chapter is the Total Ozone Content (TOC) prediction at different stations situated in the Iberian Peninsula. Prediction of the ozone series from past data is an important topic that is often addressed in atmospheric physics and chemistry [85, 86]. Specifically, the well documented TOC daily data series over the Iberian Peninsula[2] is available from the Total Ozone Mapping Spectrometer (TOMS), located on board the NASA Nimbus-7 satellite [87], for the period ranging from 1 November 1978 to 6 May 1993 (a total of 173 samples). This data set has been widely used, and it is one of the sources that was used to confirm a linear TOC decrease over that time period. We have chosen overpass data that are calculated with an algorithm that is similar to TOMS Version 8.5 (V8.5) [88]. Specifically, a compact data set of monthly average values has been built that includes TOC over 5 sites in the Iberian Peninsula: Madrid, Arenosillo, Lisbon, Mont-Louis and Murcia (see Table 3.8 and Figure 3.4). In all of the experiments at each site, we used a training period from November 1978 to April 1990, inclusive. The test period comprised 35 months, from June 1990 through April 1993.

Table 3.8: Geographical coordinates of TOMS Nimbus-7 overpass sites.

TOMS Nimbus-7 overpass site	Latitude	Longitude	Altitude
Madrid (Spain)	40.40N	3.68W	548m
Murcia (Spain)	38.00N	1.17W	70m
Lisbon (Portugal)	38.77N	9.13W	105m
Mont-Luis (South France)	42.50N	2.13E	1650m
Arenosillo (Spain)	37.10N	6.73E	41m

The final set of prediction variables that were included in this study (monthly average values over the previous mentioned area) includes the following variables: Ongoing long-wave radiation (OLR), temperature at 50 hPa level (t50), pressure at the tropopause level (TPP), longitudinal component wind at 200 hPa (u200), latitudinal component wind at 200 hPa (v200) and omega vertical velocity at 200 hpa (ω200). Additional data were taken from the Earth System Research Laboratory, which included the Solar Cycle (SC) monthly value expressed as the $10.7cm$ solar radio flux (used as a solar activity proxy in

[2]TOC data are given in Dobson Units ($1DU = 1m \cdot atm \cdot cm$, or $2.69 \times 1016cm^{-2}$).

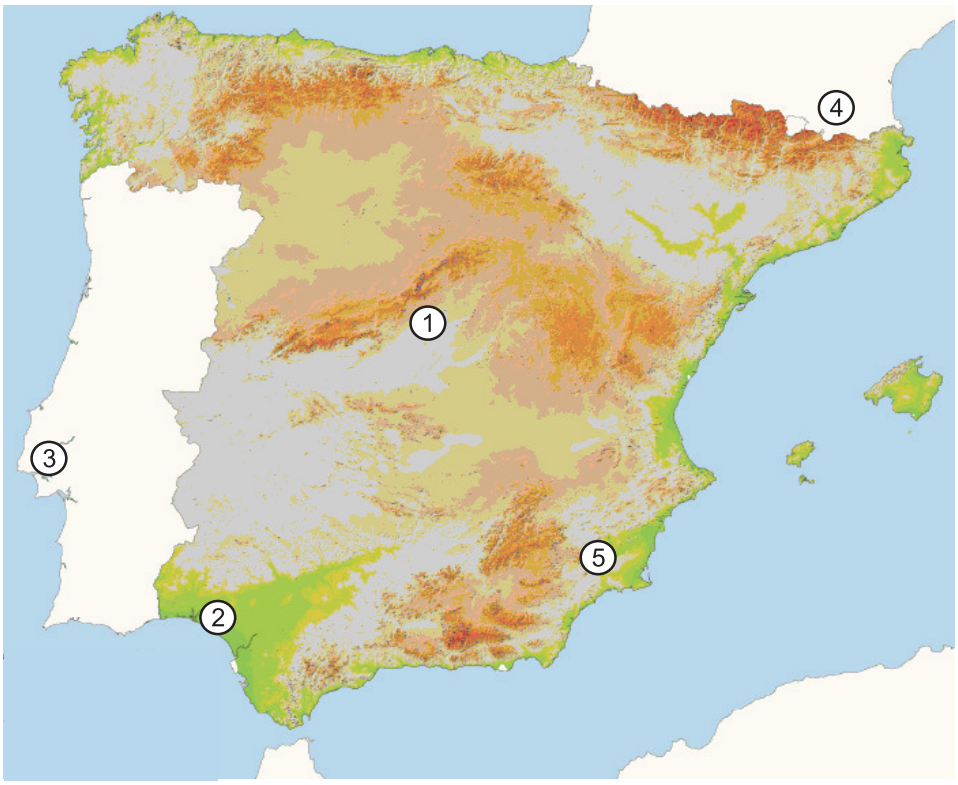


Figure 3.4: TOMS Nimbus-7 overpass observation sites; (1) Madrid National Radiometric Centre; (2) Arenosillo meteorological base; (3) Lisbon; (4) Mont-Louis and (5) Murcia Meteorological Centre. The observation sites are marked by circles.

[89]), the Quasi-Biennial Oscillation (QBO) (as calculated from the zonal average of the 30 mb zonal wind at the Equator, as computed from the NCEP/NCAR Reanalysis), the Singapore observed 30 mb zonal wind (SOZW, related to QBO), and the North Atlantic Oscillation (NAO) Index (as calculated by NOAA and the Multivariate ENSO Index (MEI) [90]).

Table 3.9 shows the RMSE for the three different implementations. The results are not very different from those got it with public repositories where in terms of RMSE the differences are not very high. A further analysis with statistical tests between GS and EA with bounds is presented in Table 3.10, where there is two set with significant differences (Lisbon and Murcia) in favor of the multi-parametric evolutionary SVR. In terms of training time, it is also similar to the others problems, EA improves substantially the grid search training time but also the bounds introduce an important reduction in training time.

Table 3.9: SVR performance and training time for the standard SVR (with a GS) and the multi-parametric kernel SVR (with the EA) in the problem of Ozone at the Iberian peninsula

	Grid Search (standard SVR)		EA with bounds (multi-parametric kernel SVR)		EA without bounds (multi-parametric kernel SVR)	
Data set	Error	Time(s)	Error	Time(s)	Error	Time(s)
Arenosillo	12.1999	10.6800	**12.1691**	4.4607	12.5750	5.4516
Lisbon	**12.8912**	12.0998	13.3267	4.5329	13.4212	6.2281
Madrid	14.3797	12.2650	**13.6795**	4.6033	13.8544	6.8840
Mont-Louis	14.8559	12.4894	**14.8057**	4.6596	15.2393	6.7216
Murcia	11.8828	11.6628	**11.5465**	4.3632	11.6430	6.8315

Table 3.10: Statistical tests (t-test or sign-test) with significance ($\alpha = 0.05$) over 10-Fold GS and the multi-kernel evolutionary SVR, for Ozone data set.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
Arenosillo	0.4278	t-test	0	0.7611	58-42-0
Lisbon	0.7790	t-test	1	0.0071	67-33-0
Madrid	0.0072	sign-test	0	0.0891	41-59-0
Mont-Louis	0.5453	t-test	0	0.6204	45-55-0
Murcia	0.2870	t-test	1	8.8139e-06	66-34-0

3.4 Conclusions

In this chapter, we have proposed a novel evolutionary-optimized multi-parametric kernel Support Vector Regression algorithm (multi-parametric kernel SVR). We have described the main characteristics of the multi-parametric model proposed, and an evolutionary algorithm considered to optimize this kernel, since standard grid search is computationally too expensive to be applied in this task. A reduction of the search space for the multi-parametric kernel has been presented, based on different theoretical lower and upper bounds. We have tested the proposed approach in different regression problems, including databases from popular public repositories and real applications. The results obtained have shown the good performance of the multi-parametric kernel approach against the standard SVR with grid search, both in accuracy and computation time.

Chapter 4

New validation methods for improving regressors training time

In the previous chapter we have explained and demonstrate that the training process for SVM is usually very time consuming. That is why we have tried to introduce evolutionary algorithms which reduce the training time, since EA focuses on the areas where the results are better instead of use a brute force algorithm such as grid search.

In this chapter we have the same objective: *reduce training time of regression algorithms*. However, we are dealing with a different part of the training process: the validation method. In Chapter 1, it is explained the typical problem in machine learning: *over-fitting*. This over-fitting creates models very well tuned to the training set but with a poor behavior with new samples. In order to prevent over-fitting and get the best possible hyper-parameters values we need a validation method during the training process. This validation method must select the best values for the vector (C,ϵ,γ) or (C,ϵ,γ_1,...,γ_M) from the training data. Most of the authors use traditional validation methods such as K-Fold cross-validation, Leave One Out, Bootstrap, etc., but these validation techniques are not focused on improving the SVR training time. They create a bunch of models with different partition in the training set, and calculate the average performance. This produces an important computational overhead. Note that this high training time is specially important in multi-parametric SVR, since it is much harder to find the optimal vector (C,ϵ,γ_1,...,γ_M).

In this chapter, we propose two new validation methods that considerably reduce the training time of the SVR, maintaining in most cases its performance in terms of accuracy. The first new validation technique, called **percentage cross-validation**, is based on splitting the initial training set into two subsets with different percentage of samples, one with $P_{\%}$ of the samples an another with the rest of the samples (100-$P_{\%}$), obtaining two models from these subsets and testing them in the complementary set.

This process is repeated several times by increasing the value of P in each step. The second method is called **generalized predictive cross-validation**, and it is based on testing the behavior of sub-models which are created using predictions of an input set. We will test the proposed CV methods with the same data sets used in the previous chapter: data sets from standard repository and also two real problems.

4.1 New Validation Methods to improve the training time of SVR

Every vector (C,ϵ,γ) in the standard SVR or (C,ϵ,γ_1,...,γ_M) in the multi-parametric case, generated by the search algorithm considered, has to be evaluated in order to know its goodness and thus it can be selected as the best individual. This task is carried out using a validation method, which has to choose those SVR hyper-parameters which may generate a good model in terms of generalization. Note that this process is as important as the search algorithm itself, and both of them determine the quality of the final SVR model. One of the most used procedures for carrying out this validation process is the K-Fold CV approach, which we have already described in Section 1.1.2 and we use it in this chapter as a reference in the experimental part. In the current chapter, we propose two alternative algorithms to carry out the cross validation process: the *percentage cross-validation* approach and the *generalized predictive cross-validation* method.

4.1.1 K-Fold cross validation for reference

K-Fold CV method is one of the most used validation method in the literature. This validation method is able to select hyper-parameters (in most of the cases) with a good generalization performance, in a reasonable amount of time. This method can briefly summarize as follows:

- Let us consider a set of samples $S = \{(\mathbf{x_i}, y_i), i = 1, \ldots, N\}$.
- The set is first partitioned into K equally (or near equally) sized subsets or folds $C_1, C_2, ..., C_K$.
- One fold C_i is selected for validation and the rest of the folds $\overline{C_i}$ are used to create a model $f_{\overline{C_i}}$.
- Once the model $f_{\overline{C_i}}$ is created by the learning process it is tested with the fold C_i. The difference between the predictions and the real outputs generates an error E_i.

- The K-Folds algorithm is carried out by repeating steps 3 to 4 during K iterations, in order to use every fold as validation set.
- The average error is calculated as:

$$E_{av} = \frac{1}{K}\sum_{i=1}^{K}(E_i) \tag{4.1}$$

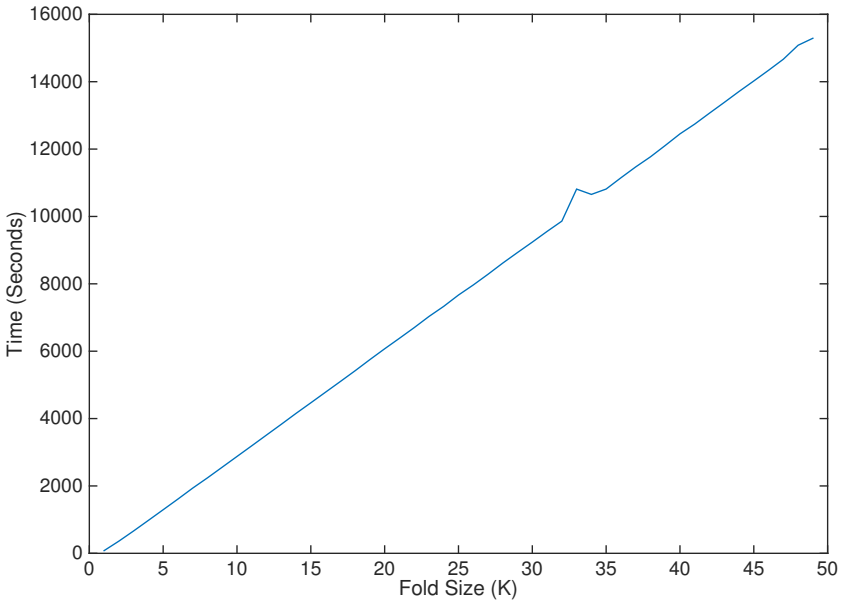


Figure 4.1: Training time in terms of value of the fold size (K). Abalone data set.

Figure 4.1 shows the evolution of training time as a function of K, this training time is clearly linear with the fold size K. On the other hand, the increment of training as a function of number of samples is represented in Figure 4.2. In this case, the training time is not linear is an exponential increase. Hereby, it is necessary to reduce the size of the folds, but also the number of trainings, in order to define new validation methods which can reduce the training time.

4.1.2 Percentage cross-validation

K-Fold CV keeps fixed the size of the training and validation partitions depending on the value of K. If the value of K is high the training time is also elevated. However, if the value is low the training set could not have enough representatives samples. We also know that a high K not always means better performance than a lower value, figure 4.3

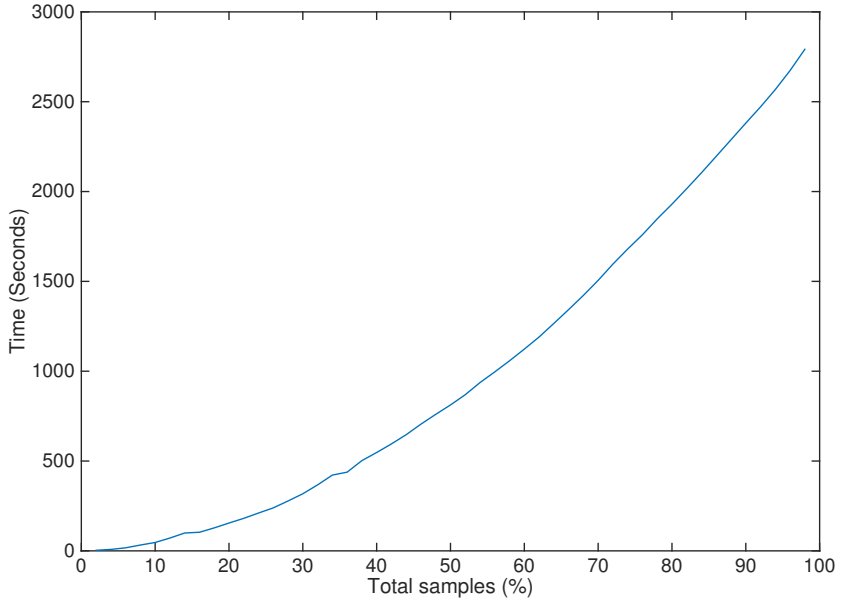


Figure 4.2: Training time in terms of data set size. X axis represents the percentage of total samples of Abalone used for the training

shows the representation of $f(x) = sinc(x)$ and several estimation models. Each model has been created with a different partition size starting from 10% to 90% of the training size and the rest of the training set is used as validation set. This figure shows that partitions with less data have better performance than other with more data.

We propose a new method called **percentage cross-validation**. This method builds several model with training set of different size. The original training set is split into two subsets with different percentage of samples: one with $P^{\%}$ of the samples an another with the rest of the samples $(100 - P)^{\%}$. This process is detailed in the next steps:

1. Let us consider a set of samples $s = (\mathbf{x}_i, y_i)$ where $i = 1, \ldots, N$, $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{iM}]$ and $P^{\%} = 1, 2, \ldots, 100\%$.

2. The set is partitioned into two subsets, one with $P^{\%}$ of samples ($C_{P\%}$) and the other with the rest of the samples ($C_{(100-P)\%}$)

3. Both subset are used to generate two models, obtaining $f_{C_{P\%}}$ and $f_{C_{(100-P)_{\%}}}$.

4. Every model is then tested in the complementary set:

$$f_{C_{P\%}}(\mathbf{x}_i), \mathbf{x}_i \in C_{(100-P)\%} \tag{4.2}$$

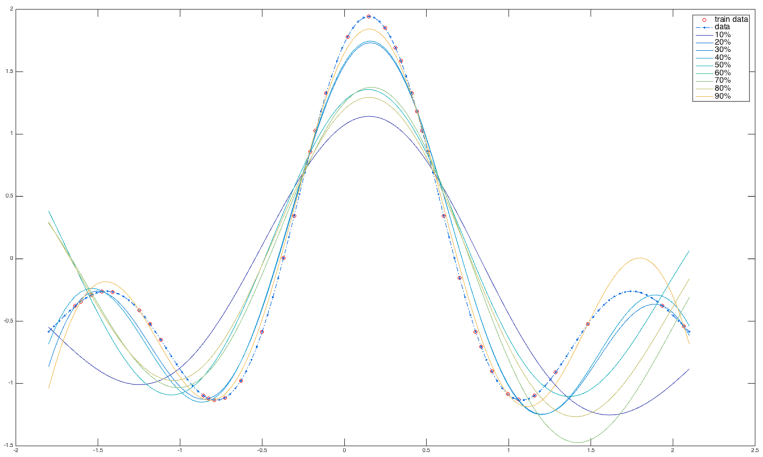


Figure 4.3: Sinc signal estimated by predictors created with different percentage of the training data.

$$f_{C_{(100-P)\%}}(\mathbf{x}_j), \mathbf{x}_j \in C_{P^{\%}} \tag{4.3}$$

5. Percentage CV algorithm is carried out by repeating steps 2, 3 and 4 for $P_k^{\%} = k \cdot P^{\%}$, while $P_k < 100\%$.

6. Both comparisons generate an error $E_{P_k^{\%}}$ where $k = 1, \ldots, K$ and K the number of partitions:

$$E_{av} = \frac{1}{K} \sum_{k=1}^{K} E_{P_k} \tag{4.4}$$

Figure 4.4 represents the block diagram of the process already described.

4.1.3 Generalized predictive cross-validation

This method uses a new idea for model performance testing and over-fitting checking. Most of validation methods are based on building an estimator and measure the error with the validation set. We propose to build a second model based on the estimations made by the first one, and check out the performance of this second model with the validation set. This idea allows increasing the difference between the good and generalizable model with respect to the ones with a poor performance or over-fitted, because in the first case both models would be very similar to each other and the error measure by the second model will be close to the error of the first model. However, in case of poor performance

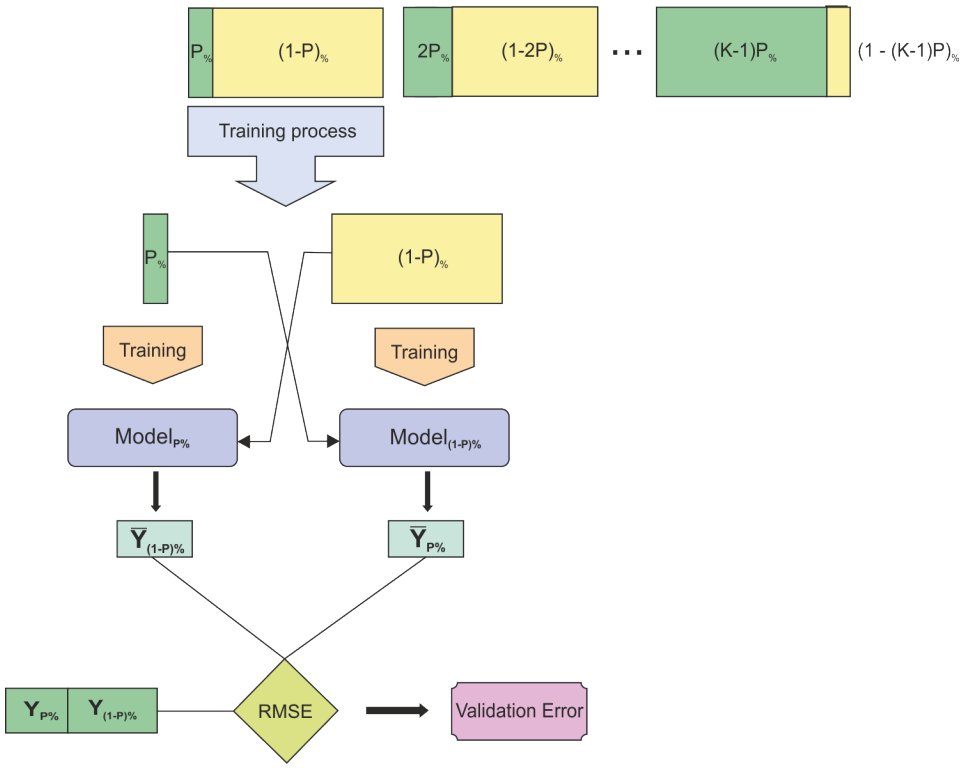


Figure 4.4: Percentage cross-validation block diagram.

the second model will have a worse behavior than the first one in most of the times, since it has been created with an image that significantly distorts the reality.

Based on this concept we have developed a validation method which is summarized in the next steps:

1. Let us consider a set of samples $s = (\mathbf{x}_i, y_i)$ where $i = 1, \ldots, N$ and $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{iM}]$.

2. The set is first partitioned into K equally (or near equally) sized subsets or folds $C_1, C_2, \ldots, C_K$.

3. A subset of data C_i is selected to create a model f_{C_i}.

4. This model makes predictions over the next subset of data C_{i+1}, obtaining $f_{C_i}(\mathbf{x}_j)$ where $\mathbf{x}_j \in C_{i+1}$.

5. A new data set is obtained with the C_{i+1} inputs $\mathbf{x}_j$ and the predictions over these inputs $f_{C_i}(\mathbf{x}_j)$. This new set is defined as $\hat{C}_{i+1} = (\mathbf{x}_j, f_{C_i}(\mathbf{x}_j))$.

6. Another model $f_{\hat{C}_{i+1}}$ is generated by using this new prediction set.

7. The new model is tested over the rest of the original data set $\overline{C_i \bigcup C_{i+1}}$, generating an error E_i.

8. This process is repeated K times, in order to apply the described procedure over all the subsets.

9. The average error is calculated as:

$$E_{av} = \frac{1}{K} \sum_{i=1}^{K} E_i \tag{4.5}$$

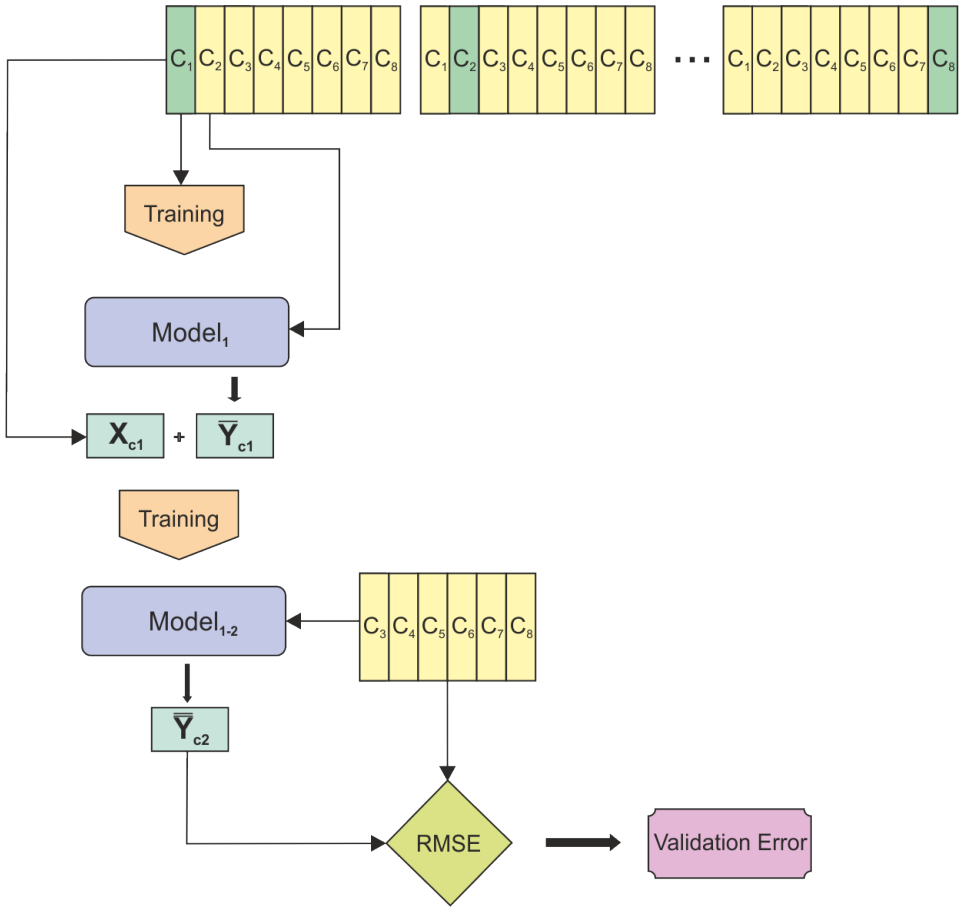


Figure 4.5: Generalized predictive cross-validation block diagram.

The advantage of this method is that we do not need to train almost the whole set of data to create our validation models, instead of that we only need a small subset of data for the two models. This is the reason why even training the double of models the time consumption of this technique is much lower then K-Fold CV. Figure 4.5 shows the diagram process described before.

Similar to K-Fold CV, generalized predictive needs to define the number of folds K. For this purpose, we have carried out several tests to find the best value. First, Table 4.1 shows the error for different number of folds over the public data sets. It seems that values $K = 9, \ldots, 14$ have better results, being $K = 14$ and $K = 9$ the best in terms of RMSE. In order to make a good decision, a further analysis has been made in Table 4.2, where it has been studied the error evolution with the increment of data size (Abalone). The results does not have big differences: having a look at the *Average* column we see that most of the values are in the error interval $2.1000 - 2.1200$, which corresponds to a 1% of error. Finally, in Table 4.3 generalized predictive is evaluated in terms of training time. In this case, there are notorious differences: the higher the number of folds (K), the faster the training is. This fact could be counterintuitive since a higher K means more number of trainings. However, we have shown in Figure 4.2 that the training time exponentially increases with the data size, so with smaller folds the training time decreases even if the number of trainings is higher. Finally, we have decided to select a trade off between low error and training time, for this reason we use $K = 14$ during the experimental part.

Table 4.1: Public data sets RMSE for different values of K.

K	Mortpollution	Bodyfat	Betaplasma	Retplasma	Autompg	Housing	Concrete
	Error	Error	Error	Error	Error	Error	Error
2	48.2712	0.01258	168.5612	285.3060	4.6882	5.3108	31.4700
3	47.5030	0.01132	168.5612	282.5161	4.4532	4.3727	29.1183
4	47.5030	0.01130	166.8480	272.3210	4.5977	4.6502	28.8535
5	48.2712	**0.01127**	162.6510	276.9712	4.7848	4.3923	28.8535
6	47.5030	0.01133	171.2110	262.7470	4.8069	4.7410	28.8590
7	46.6592	0.01132	**162.4712**	254.3715	4.5895	4.3923	28.8234
8	46.6592	0.01134	173.2971	261.4200	4.59767	4.9176	**28.5865**
9	46.6592	0.01129	173.2971	**248.2706**	4.5293	**3.9750**	**28.5865**
10	47.5030	0.01133	168.0370	254.3715	4.7774	4.9176	**28.5865**
11	47.5030	0.01134	168.1570	251.1190	4.5977	4.9374	**28.5865**
12	48.2712	**0.01127**	168.1570	255.0180	4.7041	4.1116	28.6565
13	50.3591	0.01134	167.6731	250.2131	4.7041	4.9176	**28.5865**
14	**46.4561**	**0.01127**	169.0853	254.3715	**4.4446**	4.1155	28.6379
15	48.2712	0.01134	170.1480	260.5500	4.6432	4.3923	**28.5865**

Table 4.2: RMSE with different partitions of Abalone data set, over several values of K.

K	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	Average
	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error	Error
2	2.8628	**2.8037**	2.5693	1.9179	1.6254	2.8933	2.1236	2.2455	2.3036	2.0452	2.3390
3	2.5768	2.8607	2.1280	1.6221	1.5681	2.8028	**1.9761**	1.9036	**2.0058**	1.8372	2.1281
4	2.5176	2.8607	2.0088	**1.6061**	1.5013	2.7684	2.0528	1.9371	2.0297	1.8234	2.1106
5	2.5721	2.8370	2.2108	1.6221	1.5148	2.7958	2.0117	**1.9033**	2.0598	1.8661	2.1393
6	2.5914	2.9420	1.8833	1.7449	1.5148	2.7684	2.0280	1.9111	2.0297	1.8146	2.1228
7	2.5465	2.9148	1.8771	1.6221	1.5681	2.7692	2.0117	1.9267	2.0670	**1.7996**	2.1102
8	2.5176	2.9757	1.8892	1.6221	1.5042	**2.7638**	2.0216	1.9111	2.0598	1.8501	2.1115
9	2.5736	2.8851	**1.8671**	1.6559	1.5124	2.8116	2.0280	1.9267	2.0392	1.8383	2.1138
10	**2.5041**	2.9594	1.9209	1.7213	1.5013	2.7684	2.0369	1.9111	2.0598	1.8234	2.1207
11	2.5214	2.8924	1.8852	1.6559	1.5013	2.7692	2.0280	1.9111	2.0509	1.8146	2.1030
12	2.5214	2.8924	1.8852	1.7086	1.5148	2.7816	2.0216	1.9685	2.0392	1.8146	2.1148
13	2.5110	2.9304	1.8852	1.6221	1.5042	2.7816	2.0280	1.9111	2.0670	1.8383	2.1079
14	2.5098	2.8924	1.8892	1.7213	1.5013	**2.7638**	2.0280	1.9111	2.0509	1.8599	2.1127
15	2.5139	2.9053	1.8892	1.6559	**1.3366**	3.2691	2.0369	1.9267	2.0598	1.9015	2.1495

Table 4.3: Training time with different partitions of Abalone data set, over several values of K.

K	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
	Time(s)	Time(s)	Time(s)	Time(s)	Time(s)	Time(s)	Time(s)	Time(s)	Time(s)	Time(s)
2	26.379	101.29	133.443	205.608	308.147	438.095	573.378	710.268	894.831	1163.35
3	16.036	56.129	138.544	171.756	267.478	358.145	482.43	590.475	735.447	936.124
4	12.168	66.378	100.464	202.628	262.127	383.276	442.79	556.14	681.206	851.292
5	9.828	38.828	73.258	133.817	235.529	290.519	366.397	482.555	582.161	699.691
6	8.845	65.801	59.795	90.855	174.627	294.753	284.342	422.557	481.79	620.381
7	7.909	65.582	50.715	84.755	141.742	205.704	274.311	390.312	413.041	483.912
8	7.035	19.204	39.655	75.956	116.672	198.031	211.817	253.172	315.931	370.578
9	6.63	16.957	40.31	63.82	101.634	140.918	177.652	225.296	271.798	352.685
10	6.443	16.287	33.259	52.525	87.875	138.05	160.399	200.257	249.164	354.089
11	5.678	52.354	29.359	50.731	78.796	109.174	145.641	190.46	241.379	312.452
12	5.351	14.055	28.751	45.022	73.195	110.139	139.963	181.116	245.715	297.336
13	5.304	12.558	27.722	43.758	66.347	98.251	133.692	173.223	240.91	284.762
14	4.789	12.464	25.459	41.309	64.943	94.241	129.152	169.946	235.825	276.963
15	5.741	11.747	23.697	40.077	63.726	91.525	124.41	163.675	222.206	268.491

4.2 Evolutionary Algorithm

In this chapter we consider as well an evolutionary algorithm to obtain the best possible set of parameters for the multi-parametric Gaussian Kernel in SVR. This evolutionary algorithm is similar to the one applied in the previous chapter described in section 3.2.

The algorithm is defined in the following way: each individual in the evolutionary population (with a total of 25 individuals) is defined to be a vector representing the hyper-parameters of the SVR (C,ϵ,γ_1,...,γ_M). We consider a real encoding, with a standard

two-points crossover operator and a random mutation using a Gaussian function. The roulette-wheel selection procedure has been applied as selection mechanism. As objective function, we have used the error obtained by mean of a given validation procedure. In order to avoid the random elimination of the optimal individual, we also apply Elitism saving three individuals each iteration. In addition, we consider the same three stooping criteria described in Section 3.2.5. Finally, it is also used the same repair function from the Section 3.2.6.

4.3 Experimental Part

The procedure to test the new proposed algorithms is similar to the one explained in Section 3.3.1: each data set is divided into two sets, the training and test sets, by selection 80% of the instances for training and the rest of the instances for the test set. The data sets are also the same applied in the previous chapter: UCI Data set, Barcelona airport temperature and total ozone content at Iberian peninsula. The employed machine is the same as the one used also in Chapter 3.

As a benchmark method to compare the two different validation methods proposed, we use the well-known K-Fold CV method with $K = 10$. The same as K-Fold CV, percentage CV and generalized predictive CV have both a parameter to set. For percentage CV method, the P value has been initially set to 20% ($P_i = 20 \cdot i$ with $i = 1, \ldots, 4$). In case of generalized predictive CV the number of subsets have been fixed to 14, which have provided the best balance between error (RMSE) and training time. Point out that in this chapter it also compared the different validation methods with grid search algorithm, in this case we use 20 permutations instead of 5. For evolutionary algorithm tests we still use 5 permutation and 20 repetitions. SVR with GS has been tested in this section with the bounds proposed in [78] and for the multi-parametric kernel the new bounds explained in Chapter 3.

4.3.1 Results on public data sets

In this chapter we use the same data set taken from UCI and Statlib [83, 84]: Abalone, Autompg, Betaplasma, Bodyfat, Concrete, Housing, Mortpollution and Retplasma. First of all, we show the results obtained with the standard version of the SVR (3 hyper-parameters), using a GS to obtain them, with the 10-Fold CV, and the proposed percentage CV and generalized predictive CV. The results are shown in terms of performance (mean RMSE in the test set) and training time in Table 4.4. First of all, we see clear differences in time consuming where the new methods overcome 10-Fold CV. In fact,

this difference in computation is dramatic in large training sets such as Housing, Concrete and Abalone. The comparison between the percentage CV and the generalized predictive CV indicates that the generalized predictive obtains smaller computation times, which in large training sets is about half of the computation time using the percentage CV method.

Table 4.4: UCI results (RMSE and time) of the standard SVR using grid search with 10-Fold, percentage and generalized predictive cross validation methods.

	GS 10-Fold		GS Percentage		GS GP	
Data set	RMSE	Time(s)	RMSE	Time(s)	RMSE	Time(s)
Mortpollution	**48.4622**	1.7796	49.6654	0.7546	57.8183	0.2903
Bodyfat	**0.01056**	20.3806	0.01063	6.7246	0.01064	1.6401
Betaplasma	185.7698	22.1930	**183.2204**	8.9388	188.5184	2.4742
Retplasma	221.4400	21.4288	**218.065**	12.2392	226.9704	2.3712
Autompg	**2.8191**	40.1328	2.9093	16.8562	2.9408	4.3210
Housing	**3.6175**	64.1164	3.7899	26.804	3.8707	11.4040
Concrete	28.7645	370.2846	**28.6222**	164.7522	29.2511	24.3678
Abalone	**2.0700**	2932.6520	2.0786	1313.1100	2.1177	298.0416

In terms of RMSE, there are not great differences between 10-Fold and percentage CV, the best results are shared between this two methods. Generalized predictive has results close to the others methods in most of the data sets, and important differences in Mortpollution. Considering the statistical tests, Table 4.5 shows the comparison between the standard SVR with 10-Fold CV and the percentage CV approach. The results in terms of RMSE are similar using both approaches, and there are only one database with significant differences, Autompg, where the 10-Fold CV performs statistically better than the percentage CV. Also generalized predictive CV has been compared to 10-Fold CV, the results (Table 4.6) show only two significant differences in favor of 10-Fold CV: Concrete and Abalone. The rest of the data sets have small differences regarding Win-Loss-Tie; GP has more wins in Bodyfat and Betaplasma, and 10-Fold in Mortpollution, Retplasma, Autompg and Housing.

In a second round of experiments, we analyze the new validation methods proposed with the evolutionary-trained multi-parametric SVR approach. In Table 4.7, it is shown the results in terms of average RMSE and training time obtained with the three validation methods with the evolutionary algorithm. In terms of training time, we see that the evolutionary algorithm has reduced the training time of 10-Fold and percentage. However, generalized predictive does not improve, in fact in some databases the time is slightly higher. Anyway, GP is the fastest method follows by percentage CV. Figure 4.6 shows

Table 4.5: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over 10-Fold and percentage CV with grid search, for public data sets.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
Mortpollution	0.7917	t-test	0	0.5572	11-7-2
Bodyfat	0.9354	t-test	0	0.1151	11-8-1
Betaplasma	0.5211	t-test	0	0.0962	13-6-1
Retplasma	0.1174	t-test	0	0.1215	13-6-1
Autompg	0.8605	t-test	1	0.0351	4-14-2
Housing	0.0992	t-test	0	0.3264	8-10-2
Concrete	0.1534	t-test	0	0.8399	7-9-4
Abalone	0.0354	sign-test	0	1	7-7-6

Table 4.6: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over 10-Fold and generalized predictive CV with grid search, for public data sets.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
Mortpollution	0.1923	t-test	0	0.0796	7-12-1
Bodyfat	0.6590	t-test	0	0.8400	11-9-0
Betaplasma	0.2945	t-test	0	0.7081	12-8-0
Retplasma	0.4390	t-test	0	0.1238	7-13-0
Autompg	0.6402	t-test	0	0.0545	4-16-0
Housing	0.5541	t-test	0	0.2668	7-13-0
Concrete	0.9391	t-test	1	0.0001	4-16-0
Abalone	0.4212	t-test	1	0.0012	4-16-0

the training time for all the different configurations presented in this section, where generalized predictive GS is the fastest method and 10-Fold is the slowest one.

If we look at mean error the differences are not very significant between 10-Fold CV and percentage CV, however we see that 10-Fold CV has the best mark in most of the data sets. Tables 4.8 and 4.9 show the statistical test for percentage and generalized predictive CV against grid search 10-Fold CV. Table 4.8 shows four statistical differences, percentage CV is better in two Betaplasma and Retplasma, and 10-Fold is better in Bodyfat and Autompg. On the rest of data sets, percentage CV has more wins than 10-Fold. On the other hand, Table 4.9 shows the statistical result between 10-Fold GS and GP with EA, in this case the results confirm that 10-Fold overcomes GP with EA in all the data sets with significant differences.

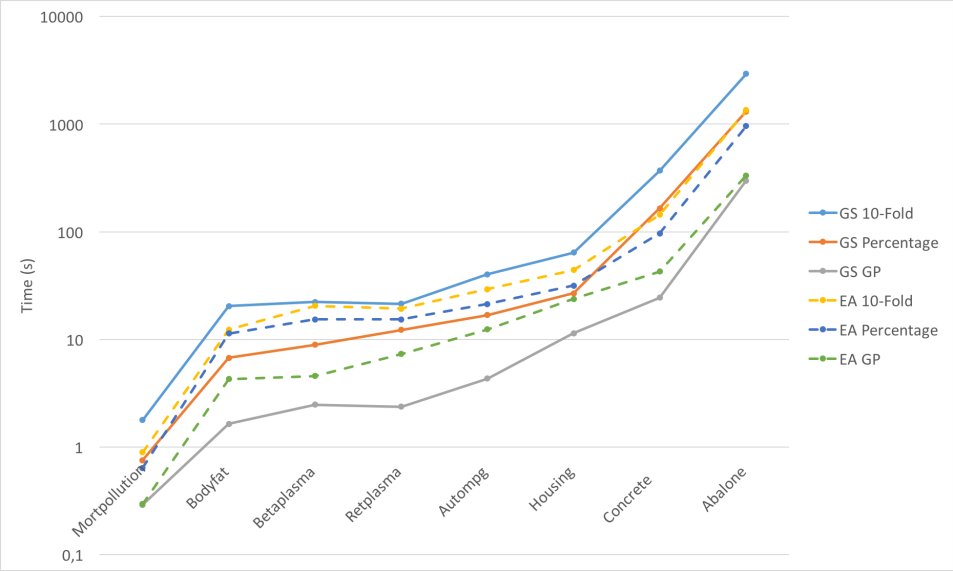


Figure 4.6: Training time for the different validation methods and search algorithms.

Table 4.7: Results (RMSE and time) of the standard SVR using evolutionary algorithm with 10-Fold, percentage and generalized predictive cross validation methods.

	EA 10-Fold		EA Percentage		EA GP	
Data set	RMSE	Time(s)	RMSE	Time(s)	RMSE	Time(s)
Mortpollution	**47.0041**	0.8944	47.5583	0.6381	51.9764	0.2952
Bodyfat	**0.01069**	12.3272	0.01073	11.3683	0.01130	4.2830
Betaplasma	**183.8452**	20.5092	184.2736	15.3602	186.0556	4.5665
Retplasma	220.5198	19.3134	**219.4802**	15.3159	226.6678	7.3043
Autompg	**2.8425**	29.2307	2.9066	21.2836	2.9507	12.3580
Housing	**3.6107**	44.0433	3.8346	31.5425	4.7138	23.7308
Concrete	**28.6867**	144.1444	28.7238	96.5022	29.1270	42.6267
Abalone	2.0774	1348.7600	**2.0764**	958.8950	2.3522	332.6770

4.3.2 Temperature forecasting at Barcelona's airport

Similar to Chapter 3, we also test the validation methods with the same real problems. The first real problem is temperature forecasting at Barcelona's airport. Table 4.10 shows the average error and training time for the three validation methods with GS. In this case, we see a deterioration of the RMSE in the two new validation methods in comparison to 10-Fold CV, over all the time steps 10-Fold has the best RMSE. However, in terms of training time the results are the opposite, the new approaches improve clearly 10-Fold CV procedure and again generalized predictive validation method is the fastest one.

Table 4.8: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over 10-Fold with grid search and percentage CV with the evolutionary algorithm, for public data sets.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
Mortpollution	0.0344	sign-test	0	0.1936	57-43-0
Bodyfat	0.0169	sign-test	1	4.3896e-12	16-83-1
Betaplasma	0.0005	sign-test	1	3.3965e-07	76-24-0
Retplasma	0.1063	t-test	1	8.9677e-05	56-44-0
Autompg	0.0221	sign-test	1	3.6350e-09	20-80-0
Housing	5.4171e-06	sign-test	0	0.2713	56-44-0
Concrete	0.0276	sign-test	0	0.7642	52-48-0
Abalone	2.4576e-07	sign-test	0	0.1936	57-43-0

Table 4.9: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over 10-Fold with grid search and generalized predictive CV with the evolutionary algorithm, for public data sets.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
Mortpollution	0.3836	t-test	1	2.8634e-07	27-73-0
Bodyfat	0.0031	sign-test	1	2.0989e-21	2-98-0
Betaplasma	3.5609e-05	sign-test	1	1.7080e-05	28-72-0
Retplasma	0.2081	t-test	1	2.3452e-10	27-73-0
Autompg	0.0513	t-test	1	2.7017e-08	33-67-0
Housing	8.5197e-05	sign-test	1	3.0150e-22	1-99-0
Concrete	0.0228	sign-test	1	0.0037	35-65-0
Abalone	0.0763	t-test	1	1.8207e-32	0-100-0

Table 4.10: Barcelona's airport results (RMSE and time) of the standard SVR using grid search with 10-Fold, percentage and generalized predictive cross validation methods.

	GS 10-Fold		GS Percentage		GS GP	
Data set	RMSE	Time(s)	RMSE	Time(s)	RMSE	Time(s)
1 hour	**0.5078**	315.4014	0.5266	123.6832	0.5333	22.2836
2 hours	**0.9077**	317.2938	0.9825	124.1448	1.0495	27.0884
3 hours	**1.0578**	328.6482	1.1508	130.3130	1.2682	26.8198
4 hours	**1.1546**	358.4986	1.2274	139.7820	1.4317	32.0800
5 hours	**1.1737**	339.6676	1.3091	138.0010	1.4274	28.6486
6 hours	**1.2380**	356.2790	1.3113	139.9860	1.5240	30.8726

Tables 4.11 and 4.12 show the results of the statistical tests over the two proposed validation methods against 10-Fold CV with GS. These tests confirm the results commented

previously from the Table 4.10. In most of the data sets there are significant differences in favor of 10-Fold CV method, so that the new proposed methods have poorer behavior in terms of accuracy but a better performance in terms of training time.

Table 4.11: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over 10-Fold and percentage CV with grid search, for Barcelona's airport temperature data set.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
1 hour	0.0789	t-test	0	0.4089	3-11-6
2 hours	0.9082	t-test	1	3.3379e-07	0-20-0
3 hours	0.9957	t-test	1	6.7417e-07	1-19-0
4 hours	0.9061	t-test	1	1.4572e-04	3-16-1
5 hours	0.4095	t-test	1	1.6825e-05	1-18-1
6 hours	0.9738	t-test	1	1.2403e-05	2-17-1

Table 4.12: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over 10-Fold and generalized predictive CV with grid search, for Barcelona's airport data set.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
1 hour	0.1181	t-test	0	0.4541	6-11-3
2 hours	0.9077	t-test	1	8.5234e-09	0-20-0
3 hours	0.7816	t-test	1	3.8072e-12	0-20-0
4 hours	0.4952	t-test	1	2.2550e-09	0-20-0
5 hours	0.8388	t-test	1	2.1669e-11	0-20-0
6 hours	0.8185	t-test	1	3.0971e-08	1-19-0

Table 4.13 shows the results of RMSE and training time for the three different validation methods, employing now the evolutionary algorithm proposed. In terms of accuracy, the results are very similar to the one obtained with GS. 10-Fold CV is still the one with the lowest error followed by percentage CV and GP CV. It is also proved what we concluded in Chapter 3: the evolutionary algorithm improves the error over all the methods in comparison with the GS. On the other hand, in terms of training time the there is a mix: 10-Fold CV improves with EA and generalized predictive increases this training time, for percentage cross validation the training time is very similar between both search algorithms.

Table 4.14 presents the statistical test between 10-Fold CV with grid search and percentage CV with the EA. These results show a better performance of 10-Fold CV

Table 4.13: Barcelona's airport results (RMSE and time) of the standard SVR using evolutionary algorithm with 10-Fold, percentage and generalized predictive cross validation methods.

	EA 10-Fold		EA Percentage		EA GP	
Data set	RMSE	Time(s)	RMSE	Time(s)	RMSE	Time(s)
1 hour	0.5318	157.8238	**0.5207**	103.3356	0.6603	46.7727
2 hours	**0.8891**	220.3528	0.9608	138.7492	1.0709	57.0534
3 hours	**1.0259**	220.7146	1.0871	144.7054	1.2636	64.3997
4 hours	**1.1257**	207.3694	1.2029	145.9100	1.3538	70.0105
5 hours	**1.1595**	233.9528	1.2315	164.8830	1.3898	74.8906
6 hours	**1.2140**	251.4552	1.2872	162.4358	1.4457	64.1505

with significance differences in all the time horizons. However, in this case thanks to the EA the RMSE is closer than in the case with GS. Something similar happens with generalized predictive CV (Table 4.15), it has improved the results but still the error is not as good as 10-Fold CV with grid search, and in all the cases there are significant differences where 10-Fold CV is superior.

Table 4.14: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over 10-Fold with grid search and percentage CV with the evolutionary algorithm, for Barcelona's airport data set.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
1 hour	0.3672	t-test	1	1.7079e-04	32-68-0
2 hours	0.3162	t-test	1	1.7409e-17	18-82-0
3 hours	0.3884	t-test	1	2.3956e-07	31-69-0
4 hours	0.1654	t-test	1	2.3296e-12	17-83-0
5 hours	0.0613	t-test	1	3.3290e-06	37-63-0
6 hours	0.7392	t-test	1	2.9975e-06	34-66-0

4.3.3 Total ozone content (TOC) prediction at the Iberian Peninsula

The last real prediction problem that is considered in this chapter is the Total Ozone Content prediction at different stations situated in the Iberian Peninsula, which has been already used on section 3.3.4.

Table 4.15: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over 10-Fold with grid search and generalized predictive CV with the evolutionary algorithm, for Barcelona's airport data set.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
1 hour	0.0131	sign-test	1	2.0989e-21	2-98-0
2 hours	0.2079	t-test	1	7.0774e-19	8-92-0
3 hours	0.2836	t-test	1	3.7420e-29	4-96-0
4 hours	0.6606	t-test	1	5.1249e-32	7-93-0
5 hours	0.7388	t-test	1	4.0286e-24	10-90-0
6 hours	0.0906	t-test	1	4.2004e-31	4-96-0

Table 4.16 shows the average error and training time for the different validation methods with grid search. In terms of training time, the results are similar to the previous data sets, where 10-Fold is the slowest and generalized predictive CV the fastest.

Table 4.16: Ozone results (RMSE and time) of the standard SVR using grid search with 10-Fold, percentage and generalized predictive cross validation methods.

	GS 10-Fold		GS Percentage		GS GP	
Data set	RMSE	Time(s)	RMSE	Time(s)	RMSE	Time(s)
Arenosillo	12.1999	10.6800	12.0945	3.6846	**11.7992**	0.9224
Lisbon	11.8828	11.6628	11.8588	3.6956	**11.2990**	0.9341
Madrid	**12.8920**	12.0998	13.6182	3.8036	13.4253	0.9654
Mont-Louis	**14.8559**	12.4894	15.3384	3.7822	15.9262	0.9754
Murcia	14.3797	12.2650	13.5392	3.7600	**13.1443**	0.9650

Regarding the RMSE there is not a clear method with better results over the two others, 10-Fold and percentage CV methods are quite similar. In fact, Table 4.17 shows that there is just one significant difference, in Mont-Louis, where percentage CV overcomes 10-Fold CV. Table 4.18 presents the statistical test between 10-Fold and generalized predictive, there is again only one data set with significant differences where 10-Fold overcomes GP, in terms of win-loss-tie there are not big differences between the two methods.

Table 4.19 has the RMSE average and time consumption for the same validation methods but employing the evolutionary algorithm. In terms of RMSE, the results are similar to the ones with grid search, there is not a clear improvement with the evolutionary algorithm in any of the validation methods. Table 4.20 shows the statistical tests over 10-Fold CV with grid search and percentage CV with the EA. There are three

Table 4.17: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over 10-Fold with and percentage CV with grid search, for Ozone at Iberian peninsula data set.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
Arenosillo	0.1242	t-test	0	0.0591	10-7-3
Lisbon	0.1491	t-test	0	0.0679	4-13-3
Madrid	0.0748	t-test	0	0.8028	7-4-9
Mont-Louis	0.0771	t-test	1	0.0145	4-11-5
Murcia	0.0551	t-test	0	0.1804	10-5-5

Table 4.18: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over 10-Fold with and generalized predictive CV with grid search, for Ozone at Iberian peninsula data set.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
Arenosillo	0.8165	t-test	0	0.7693	10-9-1
Lisbon	0.9872	t-test	0	0.0522	8-12-0
Madrid	0.4938	t-test	0	0.0822	7-12-1
Mont-Louis	0.3525	t-test	1	0.0004	4-14-2
Murcia	0.7733	t-test	0	0.2260	11-8-1

data sets with significant differences: Lisbon and Murcia have better performance with percentage CV and Mont-Louis with 10-Fold. Statistical tests for generalized predictive CV against 10-Fold (Table 4.21) confirm a slightly better performance of 10-Fold, where three data sets (Arenosillo, Madrid and Mont-Louis) have significant differences in favor of 10-Fold and one (Murcia) with better performance for generalized predictive CV. Regarding training time, it happens something similar to the previous sections where the EA does not reduce the training time in all the validation methods. 10-Fold CV has reduced considerably the time and with EA is three times faster. However, generalized predictive has increased this training time. The reason is the fact that with small data sets the training process is not the bottleneck, but the evolutionary algorithm which is more time consuming than a grid search.

4.4 Conclusions

In this chapter, we have proposed two new validation methods to reduce the SVR training time (search of the SVR hyper-parameters) while maintaining the accuracy and generalization properties of the final machine. These new validation methods can

Table 4.19: Ozone results (RMSE and time) of the standard SVR using evolutionary algorithm with 10-Fold, percentage and generalized predictive cross validation methods.

	EA 10-Fold		EA Percentage		EA GP	
Data set	RMSE	Time(s)	RMSE	Time(s)	RMSE	Time(s)
Arenosillo	**12.1691**	4.4607	12.2146	3.48108	12.5430	1.7334
Lisbon	11.5465	4.3632	**11.3992**	3.06917	11.9130	1.5450
Madrid	13.3267	4.5329	**12.9889**	3.32392	13.5393	1.6713
Mont-Louis	**14.8057**	4.6596	15.2268	3.29758	15.4591	1.5727
Murcia	13.6795	4.6033	**13.4343**	3.30502	13.9272	1.5214

Table 4.20: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over 10-Fold with grid search and percentage CV with the evolutionary algorithm, for Ozone at Iberian peninsula data set.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
Arenosillo	0.7927	t-test	0	0.8984	53-47-0
Lisbon	0.4952	t-test	1	3.4468e-07	75-25-0
Madrid	0.0009	sign-test	0	0.0574	60-40-0
Mont-Louis	0.8709	t-test	1	0.0010	35-65-0
Murcia	0.3502	t-test	1	1.1561e-09	70-30-0

Table 4.21: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over 10-Fold with grid search and generalized predictive CV with the evolutionary algorithm, for Ozone at Iberian peninsula data set.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
Arenosillo	0.7443	t-test	1	0.0032	38-62-0
Lisbon	0.1327	t-test	0	0.8084	57-43-0
Madrid	0.0145	sign-test	1	9.6684e-04	33-67-0
Mont-Louis	0.7570	t-test	1	3.5013e-06	32-68-0
Murcia	0.1766	t-test	1	0.0225	51-49-0

be applied to the standard and multi-parametric versions of the SVR with any type of search algorithm for the hyper-parameters, including grid search and meta-heuristic search approaches such as evolutionary algorithms. In the experimental section of the chapter we have tested the proposed methods in different experiments with grid search and an evolutionary algorithm as a search algorithm, and the K-fold cross-validation as a reference approach. We have shown that the proposed approaches obtain most of the times similar performance than the K-Fold approach in terms of the final SVR

accuracy, but in much shorter time, both when using the grid search and the evolutionary algorithm as SVR hyper-parameters search techniques.

Chapter 5

GMDH Networks construction with Evolutionary-based Hyper-Heuristic

In this Chapter, we present a novel method aiming to construct Group Method of Data Handling (GMDH) networks, assisted by hyper-heuristics algorithms. The proposed approach is based on an evolutionary hyper-heuristic, which completely automates the GMDH construction, by evolving the number of layers, the polynomial order and the number of selected nodes in each layer of the network. It results in a completely self-organized algorithm called Hyper-Heuristic GMDH (HH-GMDH).

5.1 Introduction

During Chapter 2 the GMDH theory and the different kind of algorithms that are implemented has been described. We have seen that the building process is almost automated, but there are some parameters that should be set up at the beginning of the development process, such as the number of nodes in each layer, etc. Note that these parameters will influence not only the performance but also the training time of the algorithm. Another problem that is related to the original GMDH algorithm is the well-known over-fitting, which occurs when the model is too complex, and more, there is another important problem specific to the GMDH architecture, namely multicollinearity: multicollinearity appears when the nodes' coefficients in different layers are highly correlated. This problem can substantially increase the average error of the GMDH approach.

In the last few years, intense research has been reported on GMDH-type networks to improve their performance and to overcome their inherent problems. There are several studies that address useful modifications of the basic GMDH approach: in [30], a polynomial harmonic GMDH is presented for time series modeling. In [91] a generalization

of GMDH called the Polynomial Neural Network (PNN) is described and applied to two different industrial problems: the gas furnace process time series prediction and the pH neutralization process problem. In [92], a modification of the standard GMDH algorithm, leading to good performance reported for different prediction problems [93–96]. However, the most important lines of research to improve the performance of GMDH are related to its hybridization with the use of evolutionary algorithms. Several evolutionary-based approaches have been used to this end, such as genetic algorithms [97–99], evolutionary algorithms [100, 101], genetic programming [102, 103], particle swarm optimization [104] and differential evolution [105]. Usually, these approaches attempt to replace or improve the least squares regression method of the classical GMDH and to design the different layers and layer characteristics of the network. These evolutionary approaches have produced good results in terms of the network performance, at the expense of increasing the computational cost of the algorithm.

In this chapter, we propose an alternative option to optimally construct GMDH networks using hyper-heuristics described in section 2.4. We use an evolutionary-based hyper-heuristic to find the optimal complete GMDH configuration and the polynomial order that is used in each layer, i.e., the hyper-heuristic controls the number of layers, the number of nodes and the polynomial order of each layer. The network is then constructed in a classical way, i.e., by using the least squares approach to set the network parameters. The use of the hyper-heuristic to construct the GMDH decreases the probability of over-fitting and multicollinearity because several models are generated and evaluated during run time. Thus, we choose the best model for the problem that is being solved. Two different versions of the HH-GMDH approach have been defined, depending on whether a regularization parameter (λ) is precalculated or encoded in the algorithm.

5.2 The proposed Hyper-Heuristics GMDH

As mentioned before, when designing a multi-layer GMDH algorithm for a specific problem, it is essential to properly adjust a number of parameters, such as the number of selected nodes in each step, the stopping condition or the polynomial type in each layer. The GMDH performance severely depends on the optimal tuning of these parameters, and this is a task usually carried out off-line by the network designer. We consider the use of a hyper-heuristic algorithm that automatically configures these network's construction parameters.

Specifically we have considered an evolutionary-based hyper-heuristic, in which an EA searches for the best sequence of possible heuristics to configure the number of nodes

in each layer, polynomial type and stopping condition of the GMDH. Note that using this approach, the need for the designer's intervention in the construction of the network is reduced to a minimum. In addition, as it will be shown, the use of hyper-heuristics allows avoiding over-fitting due to the stopping criterion, and also the multicollinearity problem, inherent to classically constructed multi-layer GMDH networks.

It is important to note that, to evaluate the GMDH construction process, the nodes' selection and the hyper-heuristic evolution, we need three different sets of data:

1. A *fitting set* $\mathcal{F}$, in order to obtain the polynomial coefficients for each layer of the network
2. A validation set $\mathcal{V}$, to check the goodness of every node created.
3. An extra set to evaluate the performance of each individual (GMDH network) of the evolutionary algorithm, called *fitness set* $\mathcal{S}$ [1].

In the following subsections we show details on the proposed evolutionary-based hyper-heuristic approach to construct GMDH networks, such as the list of heuristics which form the search space, the encoding, selection, crossover and mutation operators and finally the optimization process carried out to improve the algorithm's computation time.

5.2.1 Basic heuristics considered

We consider two groups of Basic Heuristics (BH) for the network construction, depending on the way the nodes are selected in each layer. Then, these basic heuristics will be combined with different types of polynomials to get the final set of heuristics to be applied to the GMDH construction. These two groups of BH are as follows:

1. **Selection of the $\mathcal{M}$ best nodes:** in each layer, the nodes are sorted in terms of the external criterion used (regularity criteria, i.e. error obtained in a validation set), and the $\mathcal{M}$ first nodes are selected. The value of $\mathcal{M}$ is equal to the number of inputs, and it is varied along the steps depending on the heuristic chosen.

   BH1: The value of $\mathcal{M}$ is not modified. BH2: $\mathcal{M}$ is decreased in a unit. BH3: $\mathcal{M}$ is a half of the previous value.

[1] Note that in situations where the available data is extremely reduced, the correct application of the HH-GMDH may be compromised by the necessity of the training set partition into three subsets. However, the experiments carried out in problems with small data sets available (Mortpollution and Bodyfat from UCI, for example), indicate a good performance of the proposed approach also in these case.

2. $\mathcal{M}$ **Random nodes selection:** the external criterion is not taken into account in this case, but the nodes in each layer are randomly chosen. As in the previous group, the difference among heuristics is the number of chosen nodes.

   BH1: The value of $\mathcal{M}$ is not modified. BH2: $\mathcal{M}$ is decreased in a unit. BH3: $\mathcal{M}$ is a half of the previous value.

The final list of heuristics to construct the GMDH is formed by the BH combined with polynomials of different degrees. To be more explicit, H1 to H6 are the BHs considering the polynomials of degree 1:

$$y = a_0 + a_1 x_1 + a_2 x_2 \tag{5.1}$$

heuristics H7 to H12 are the BHs making use of polynomials of degree 2:

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2 \tag{5.2}$$

and finally, heuristics H13 to H18 are the BHs considering polynomials of degree 3:

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2 + a_6 x_1 x_2^2 + a_7 x_1^2 x_2 + a_8 x_1^3 + a_9 x_2^3 \tag{5.3}$$

Note that, on contrary to the conventional GMDH approach, we consider different heuristics with polynomials of degree different than 2 in the proposed HH-GMDH. Since the GMDH is an approximation to the KGP, the inclusion of polynomials of degree different than 2 may provide more flexibility in the model's construction procedure and, as will be shown in the experimental section of this chapter, it is possible to obtain better results. This technique has been previously used by other authors in [106, 107]. In addition, different experiments including only heuristics related to polynomials of degree 2 have shown an increasing appearance of collinearity events, what makes even more useful the inclusion of these heuristics with polynomials of degree 3 and 1.

5.2.2 The evolutionary algorithm

Evolutionary algorithms has been previously used to evolve hyper-heuristics' solutions [108, 109]. The one developed in this hyper-heuristic has some peculiarities and differences respect to the previous algorithms implemented in chapters 3 and 4. Next sections describe the main characteristics and operators for the evolutionary algorithm implemented to HH-GMDH.

5.2.2.1 Encoding

The encoding of the evolutionary algorithm is based on integer individuals of length K. Each gene in an individual represents the construction of a layer from a previous one (or the input layer in the case of the first layer). In other words, its value labels the type of heuristic to be used to select the nodes in that layer. The zero value indicates the stopping of the network construction. Only the first 0 value is considered. As an example, in an individual defined as [1,3,10,17,0,2,15,0,6,4,8], the network is composed of four layers, with heuristics H1, H3, H10 and H17. In this individual, the GMDH will be constructed as follows: the first layer of the network keeps the same number of nodes as the initial input layer, with linear polynomials (H1). The second layer is constructed by choosing the best $\mathcal{M}/2$ nodes, again with linear polynomials (H3), the third layer is constructed by randomly chosen $\mathcal{M}$ nodes with second order polynomial (H10), and finally the fourth layer is constructed by randomly choosing $\mathcal{M} - 1$ nodes from the previous layer (H17). The final 0 indicates that at this stage the construction of the network is complete, and then the result from the best node in this layer will be set as the network result.

5.2.2.2 Fitness function

The evaluation of each individual in the population is made by using a set of data called *fitness set*, not used during the construction of the network coefficients nor the external criterion to evaluate the network's nodes. The fitness value is calculated as follows:

$$Fitness = \frac{1}{RMSE} = \frac{1}{\sqrt{\frac{1}{N}\sum_{k=1}^{N}(Y - \hat{Y})^2}} \tag{5.4}$$

where Y is the real output, $\hat{Y}$ is predicted output obtained by the GMDH model and N is the number of samples for the fitness set. This function has to be executed for every new individual and before the selection process.

5.2.2.3 Selection operator

In this implementation we use rank-based wheel selection mechanism the same as the SVM implementation for this chapter. First, the individuals are sorted in a list based on their quality. The position of the individuals in the list is called rank of the individual, and denoted R_i, $i = 1, \ldots, \xi$, with ξ number of individuals in the population of the EA. We consider a rank in which the best individual is assigned $R_1 = \xi$, the second best y,

$R_2 = \xi - 1$ and so on. A probability of survival associated to each individual is then defined, as follows:

$$f_i = \frac{2 \cdot R_i}{\xi \cdot (\xi + 1)} \tag{5.5}$$

Note that these values are normalized between 0 and 1, depending on the position of the individual in the ranking list. It is important to note that this rank-based selection mechanism is *static*, in the sense that probabilities of survival (given by f_i) do not depend on the generation, but on the position of the individual in the list. As a small example, consider a population formed by 5 individuals, in which individual 1 is the best quality one ($R_1 = 5$), individual 2 the second best ($R_2 = 4$), and so on, with fitness $F_1 = 3.4, F_2 = 1.8, F_3 = 0.5, F_4 = 0.20, F_5 = 0.10$. In case of rank-based wheel, the probability associated to the individuals are 0.33, 0.26, 0.2, 0.13, 0.06, and the associated probability for the roulette wheel are 0.56, 0.3, 0.083, 0.03, 0.016.

5.2.2.4 Crossover and mutation operators

Once we have a set of selected individuals, the crossover operator is applied. In this case we set the probability of crossover to 60%. For mutation process we set the probability to 5%. The crossover and mutation operators are similar to the ones already explained in section 3.2.3.

5.2.2.5 Individuals correction process

Due to the peculiarities of the problem, it is necessary to implement a correction process to obtain feasible solutions that lead to a proper construction of GMDH networks. For example, it is necessary that all the inputs are passed to the first layer of the GMDH, to obtain all possible combinations of the input variables. Thus, the first gene in every individual must have as heuristic H1, H7 or H13 (heuristics that consider the pass of $\mathcal{M}$ nodes from the previous layer to the current one). Since in the initialization of the algorithm, $\mathcal{M}$ is equal to the number of inputs, then the first layer will always have $\mathcal{M}$ nodes to operate. It is also possible, due to the construction dynamics of the network, that the number of selected nodes reaches 1 before the stopping condition (a 0 in the encoding) appears. In this case a value of 0 is forced in the layer where this problem appears.

5.2.3 Estimation of the regularization parameter λ

In this Thesis, we have considered two different ways of estimation the regularization parameter λ. First, a general procedure, valid for the classical GMDH and HH-GMDH, consists in checking out a range of λ values using the classical GMDH in the whole training set. We found that a good range to fix λ is $[2^{-15}, \ldots, 2^3]$ (18 exponential values). We have called this approach HH-GMDH$_\lambda$ algorithm.

A second approach for the calculation of the parameter λ, in this case specific for the HH-GMDH, consists in including λ in the evolutionary algorithm which conforms the HH-GMDH. In fact, note that we can encode the 18 exponent values of the λ range directly in the considered evolutionary algorithm (since there also 18 heuristics to be selected). Thus, λ is encoded as an extra parameter in the evolutionary algorithm, let us call it l, located at the rightmost part of the evolutionary encoding, and its value is just calculated as $\lambda = 2^{-l+3}$ (to maintain the exponent range between -15 and 3). In this case we call the algorithm HH-GMDH$^*_\lambda$.

5.2.4 Avoiding multicollinearity with HH-GMDH

Multicollinearity in an inherent GMDH-type networks problem, easily detectable in the majority of occasions, which produces outputs that are huge in comparison with the rest of correct ones. We can therefore detect it by using standard deviation of the training set outputs. The procedure we have included in the HH-GMDH is the following: for every sample in the test set (to be predicted), we select the best individual of the evolutionary algorithm, encoding a GMDH. If the prediction value is larger than 4 times the standard deviation of the training set, we assume that multicollinearity has been produced, and then we discard this solution and choose the next best individual in the population. Figure 5.1 shows an example of multicollinearity with Mortpollution data set.

5.3 Experimental part

In this section, we present different results that were obtained by the proposed HH-GMDH approaches (HH-GMDH$_\lambda$ and HH-GMDH$^*_\lambda$) with the same data sets that we have already used in chapters 3 and 4. We compare the results that were obtained by the proposed HH-GMDH algorithms with those obtained by the classical GMDH approach (considering $\mathcal{M} - 1$ nodes from one layer to the next layer and quadratic polynomials).

The methodology chosen for the experiments conducted is as follows: first, the data are divided into a training set and a test set (80% of the data for training and the remaining 20% of the data for testing). The training set is then divided into three subsets

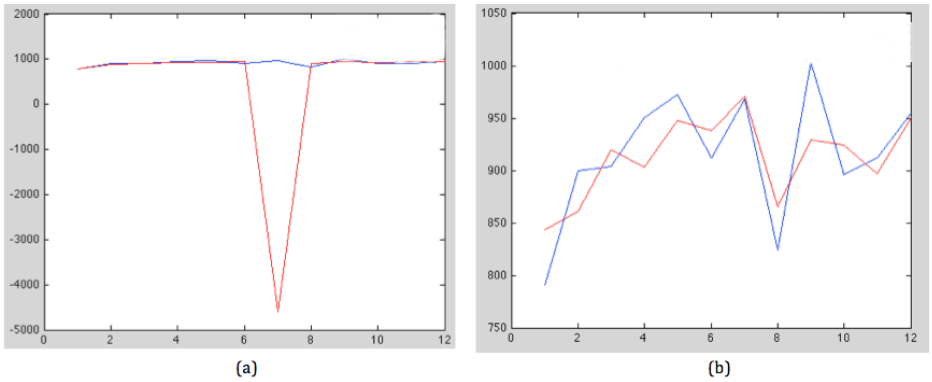


Figure 5.1: a) Standard GMDH estimation over Mortpollution data set: example of multicollinearity problem. b) HH-GMDH$^*_\lambda$ estimation over Morpollution data set.

(fitting ($\mathcal{F}$, 70% of the training data), validation ($\mathcal{V}$, 15% of the training data) and fitness ($\mathcal{S}$, of the training data)). The HH-GMDH approaches are launched to obtained the best method to construct the GMDH, in terms of the training set, and a final evaluation of the problem is conducted on the test set. Note that all of the results that will be shown have been computed over the test set. For this tests, it have been used a different machine which has this features: CPU Intel Core 2 Duo P8600 (2 cores of 2.4GHz), 4GB RAM, 320GB hard disk of 5400 rpm and Windows 7.

Although the aim of this chapter is to evaluate the improvement that is obtained with the HH-GMDH when compared to a classical GMDH, in some of the applications we also include a comparison with the Support Vector Regression algorithm with the standard 10-Fold cross validation for the parameter selection (explained in Chapter 2). As a benchmark, a MLP has been also implemented by using the Neural Network Toolbox from MATLAB [110]. The MLP consists of two hidden layers with log-sigmoid transfer function and an output layer linear transfer function, with a Levemberg-Marquardt training procedure.

5.3.1 Results on public data sets

We first test the proposed HH-GMDH with the well-known UCI data sets. Table 5.1 shows the results that were obtained by the different approaches that were compared on the UCI datasets. It can be seen that the two versions of the HH-GMDH (HH-GMDH$_\lambda$ and HH-GMDH$^*_\lambda$) achieve the best average RMSE in the Mortpollution, Retplasma, Betaplasma and Housing data sets. The SVR obtains the best results, which were compared in terms of RMSE, in Bodyfat and Autompg data sets. For Abalone data set the best result is achieved by the MLP. Regarding the computation time, we see

that the fastest algorithm among all of the tested algorithms is the classical GMDH. The HH-GMDH, in general, performs worse in terms of the computation time compared with the other approach on small data sets, but it appears to be more scalable than the other approaches in comparison; thus, its computation time is equal to or better than those of the MLP and SVR for large data sets. There are also some differences in the computation time between the HH-GMDH$_\lambda$ and HH-GMDH$^*_\lambda$ approaches, in favor of the first approach.

Table 5.1: UCI sets results. * stands for cases in which multicollinearity appeared in at least one permutation with the classical GMDH.

	GMDH		HH-GMDH$_\lambda$		HH-GMDH$^*_\lambda$		SVM		MLP	
Data set	RMSE	Time(s)	RMSE	Time(s)	RMSE	Time(s)	RMSE	Time(s)	RMSE	Time(s)
Mortpollution	43.1*	0.7	42.11	69.6	**41.24**	91.2	45.57	1.9	48.26	13.0
Bodyfat	0.0139*	0.4	0.0112	54.0	0.0113	85.6	**0.0111**	27.0	0.0120	44.6
Betaplasma	173.1	0.8	143.3	47.2	**142.0**	100.2	147.3	41.7	222.7	50.2
Retplasma	216.72*	0.4	211.8	74.5	**209.8**	93.6	218.1	41.9	307.6	50.1
Autompg	4.14	0.4	3.95	45.2	3.99	65.3	**3.67**	84.8	4.73	27.7
Housing	5.23	0.6	4.36	68.8	**4.33**	97.0	4.49	143.2	15.28	94.2
Abalone	2.18	1.04	2.01	96.4	1.91	103.4	2.02	6715.3	**1.83**	368.1

Note that the symbol (*) in the GMDH results of Table 5.1, in different data sets, indicates that the multicollinearity problem has occurred at least in 1 permutation out of the 5 considered. In this case, the sample presents a very high error; thus, the average error for all of the sets is also very high. We have calculated then the average error after eliminating cases of multicollinearity.

Table 5.2: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over GMDH and HH-GMDH$_\lambda$, for public data sets.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
Autompg	2.0539e-06	sign-test	1	6.7953e-06	73-27-0
Betaplasma	0.0604	t-test	0	0.0594	62-38-0
Bodyfat	7.9980e-22	sign-test	1	2.0989e-21	98-2-0
Housing	5.1581e-06	sign-test	1	3.7979e-08	78-22-0
Mortpollution	7.9960e-22	sign-test	1	5.4959e-16	91-9-0
Retplasma	8.7126e-22	sign-test	1	6.7953e-06	73-27-0
Abalone	0.0221	sign-test	1	1.2476e-12	86-14-0

We also conduct a statistical analysis and a discussion of the results that were obtained in each group of problems. The objective of this analysis is to test what extent the behavior for the proposed HH-GMDH approaches (both HH-GMDH$_\lambda$ and HH-GMDH$^*_\lambda$)

Table 5.3: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over GMDH and HH-GMDH$^*_\lambda$, for public data sets.

Data set	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
Autompg	5.2023e-10	sign-test	1	9.5837e-07	75-25-0
Betaplasma	0.0417	sign-test	1	0.0037	65-35-0
Bodyfat	7.9979e-22	sign-test	1	2.0989e-21	98-2-0
Housing	6.2670e-06	sign-test	1	9.5837e-07	75-25-0
Mortpollution	7.9960e-22	sign-test	1	2.7890e-15	90-10-0
Retplasma	8.5866e-22	sign-test	1	0.0037	65-35-0
Abalone	0.0991	t-test	1	0.0105	95-5-0

is superior to that of the classical GMDH approach, e.e, to show that the optimal application of heuristics for the GMDH construction outperforms the classical way of constructing a GMDH network. This calculation will also show that the hyper-heuristic approach for GMDH construction is a robust method that can be accounted for in the design of neural computation algorithms.

Tables 5.2 and 5.3 show pairwise comparison between GMDH/HH-GMDH$_\lambda$ and GMDH/HH-GMDH$^*_\lambda$. Depending on whether the result distribution is normal or not, we use t-test in case of normal distribution and sign-test otherwise. In order to check the distribution Kolmogorov-Smirnov test is applied to the results. Focus on the p-values and Win-Loss-Tie, we see that both Evolutionary GMDH alternatives overcome in almost all the data set to the standard GMDH, in fact, there is not a simple case were standard GMDH improves the hyper-heuristic approaches.

Table 5.4: Friedman ranks (public data sets).

Algorithm	Ranking
GMDH	2.2086
HH-GMDH$_\lambda$	1.9857
HH-GMDH$^*_\lambda$	1.9857

Table 5.4 shows the Friedman ranking that was obtained. The Iman-Davenport statistics and the p-value are, in this case, 0.02082 and 0.9793, which appears to indicate that there is no statistical evidence of the HH-GMDH algorithm's superiority in these problems. Table 5.5 shows the adjusted p-values for the Friedman test and the results that were calculated by using the Holland procedure [56], using the HH-GMDH$_\lambda$ as a control method. The results in this table do not confirm any significant difference in these

Table 5.5: Public data sets prediction. Adjusted p-values for the Friedman test (HH-GMDH$_\lambda$ is the control method).

Algorithm	unadjusted p	$p_{Holland}$
GMDH	0.8577	0.9797
HH-GMDH$_\lambda$	1	1

problems for the HH-GMDH$_\lambda$ over the HH-GMDH$^*_\lambda$ or the classical GMDH, although the ranking values show superior behavior for the HH-GMDH approaches.

5.3.2 Temperature forecasting at Barcelona's airport

We further test the HH-GMDH performance on a real prediction problem; specifically, with the Barcelona airport temperature described in Section 3.3.3. Table 5.6 shows the results that were computed by with the classical GMDH, HH-GMDH (HH-GMDH$_\lambda$ and HH-GMDH$^*_\lambda$), SVR and MLP algorithms. Note that the two versions of the proposed HH-GMDH reached excellent results in terms of RMSE, which were better than the results of the classical GMDH and comparable to or even better than those of the SVM and MLP algorithms. Regarding the computation time, the proposed HH-GMDH outperformed SVM and MLP, although (as expected) its computation time was higher than that of the classical GMDH approach. Thus, we can conclude that the HH-GMDH offers reasonable performance in terms of error prediction, with an affordable computation time for its implementation in real prediction systems. Figure 5.2 shows an example of the HH-GMDH fitness evolution in the 1-hour time prediction case. Note how the high-level evolutionary algorithm improves the fitness (Equation 5.4), i.e., it obtains better GMDH structures during the evolution.

Table 5.6: Temperature prediction in Barcelona's airport results.

	GMDH		HH-GMDH$_\lambda$		HH-GMDH$^*_\lambda$		SVM		MLP	
Gap	RMSE	Time(s)	RMSE	Time(s)	RMSE	Time(s)	RMSE	Time(s)	RMSE	Time(s)
1 hour	0.7525	0.96	0.6796	173.1	0.6783	459.1	**0.6534**	1065.3	0.7391	416.4
2 hours	1.2688	0.60	1.1415	175.4	1.1213	428.7	**1.0655**	959.8	1.1369	498.3
3 hours	1.6792	0.64	1.4235	189.4	1.4652	379.2	**1.3492**	941.0	1.3986	434.9
4 hours	1.8220	0.43	1.5909	181.1	1.6218	473.4	**1.4710**	1026.0	1.6730	415.6
5 hours	1.9253	0.43	1.7656	161.0	1.7671	283.5	**1.6372**	1011.2	1.7136	422.8
6 hours	2.0313	0.60	1.8470	215.7	1.8501	239.5	**1.7496**	1002.8	1.9515	434.3

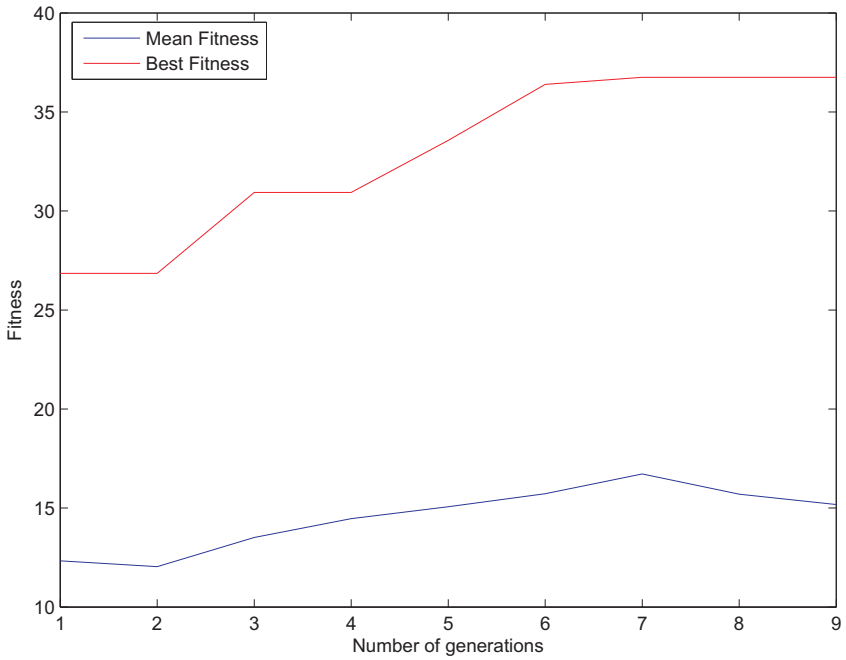


Figure 5.2: HH-GMDH fitness evolution for 1 hour prediction time horizon at Barcelona's airport.

Tables 5.7 and 5.8 have the results of t-test/sign-test. We see that Hyper-Heuristic solutions are better than classical GMDH, the p-value in all the cases reject the null hypothesis so the distribution are statistically different, in fact in almost all the execution the hyper-heuristic approaches win.

Table 5.7: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over GMDH and HH-GMDH$_\lambda$, for Barcelona's airport data set.

Gap	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
1 hour	0.0069	sign-test	1	1.0411e-16	92-8-0
2 hours	0.4926	t-test	1	9.1346e-19	99-1-0
3 hours	0.2404	t-test	1	3.7664e-31	100-0-0
4 hours	0.6877	t-test	1	1.9236e-19	100-0-0
5 hours	0.9718	t-test	1	8.5344e-10	100-0-0
6 hours	0.0994	t-test	1	3.0601e-07	99-1-0

Friedman ranking can be seen in Table 5.9, where the Friedman ranking is reported. The Iman-Davenport statistics and the p-value are, in this case, 89.09 and 0, respectively, which indicates that there are significant differences in the behavior of the different algorithms in these problems. Table 5.10 shows the adjusted p-values for the Friedman

Table 5.8: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over GMDH and HH-GMDH$^*_\lambda$, for Barcelona's airport data sets.

Gap	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
1 hour	0.0567	t-test	1	1.8068e-36	91-9-0
2 hours	0.3568	t-test	1	9.1539e-24	99-1-0
3 hours	0.9906	t-test	1	3.2879e-25	100-0-0
4 hours	0.9826	t-test	1	7.2964e-14	100-0-0
5 hours	0.2072	t-test	1	9.3772e-08	100-0-0
6 hours	0.2301	t-test	1	2.8052e-07	99-1-0

test using the Holland procedure. This table indicates that the classical GMDH performs statistically worse than the HH-GMDH$_\lambda$, whereas the HH-GMDH$^*_\lambda$ appears to perform similarly to the control method (the HH-GMDH$_\lambda$).

Table 5.9: Friedman ranks (Temperature prediction).

Algorithm	Ranking
GMDH	3
HH-GMDH$_\lambda$	1.4333
HH-GMDH$^*_\lambda$	1.5667

Table 5.10: Temperature prediction. Adjusted p-values for the Friedman test (HH-GMDH$_\lambda$ is the control method).

Algorithm	unadjusted p	$p_{Holland}$
GMDH	0	0
HH-GMDH$^*_\lambda$	0.6055	0.6055

5.3.3 Total ozone content (TOC) prediction at the Iberian Peninsula

The last real prediction problem that is considered in this chapter is the Total Ozone Content (TOC) prediction at different stations situated in the Iberian Peninsula. Table 5.11 shows a comparison of the results that were obtained by the classical GMDH and the proposed HH-GMDH$_\lambda$ and HH-GMDH$^*_\lambda$ in terms of the RMSE in the test set considered.

It can be seen that the results that were obtained by the HH-GMDH approaches are consistently better than the results that were offered by the classical GMDH at all of the sites that were considered. The experiments that were conducted show differences between the classical GMDH and the two proposed HH-GMDH, although the differences between the two versions of the HH-GMDH algorithms are small. In this problem, the computation time for the training in all of the approaches was within one second in the case of the classical GMDH and within 10 seconds in the case of the HH-GMDH approaches (note that the number of training samples in this problem is reduced).

Table 5.11: TOC average RMSE obtained with the different GMDH versions (classical and HH-GMDH) discussed in this chapter.

TOMS Nimbus-7 overpass site	GMDH	HH-GMDH$_\lambda$	HH-GMDH$^*_\lambda$
Arenosillo	13.435	12.311	**12.242**
Lisbon	13.474	13.122	**12.890**
Madrid	20.652	**13.598**	13.637
Mont-Louis	15.356	**14.869**	15.115
Murcia	**13.276**	13.337	13.549

Tables 5.12 and 5.13 presents the result of the Kolomogorov-Smirnov test and sign-test. Similar to the other two previous sections the hyper-heuristic approaches outperform the classical GMDH in almost all the repetitions and p-value clearly reject the null hypothesis for all the cases.

Table 5.12: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over GMDH and HH-GMDH$_\lambda$, for Ozone at Iberian peninsula data set.

TOMS Nimbus-7	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
Arenosillo	0.00989953	sign-test	1	6.79535e-06	73-27-0
Lisbon	7.996e-22	sign-test	1	0.00693395	64-36-0
Madrid	5.04323e-10	sign-test	1	8.032e-11	83-17-0
Mont-Louis	1.80974e-21	sign-test	1	3.31884e-18	94-6-0
Murcia	3.64981e-21	sign-test	1	0.0357288	61-39-0

Figure 5.3 finally shows the HH-GMDH$_\lambda$ prediction (over the test set) of TOC at the Madrid overpass site. Note the good reconstruction that the proposed HH-GMDH is able to obtain.

Table 5.14 shows the Friedman ranking that was obtained in these problems from the different algorithms that were considered. The Iman-Davenport statistic and the p-value

Table 5.13: Statistical tests (t-test or sign-test) with statistical significance ($\alpha = 0.05$) over GMDH and HH-GMDH$^*_\lambda$, for Ozone at Iberian peninsula data set.

TOMS Nimbus-7	Kolmogorov-Smirnov(p-value)	test	significance	p-value	W-L-T
Arenosillo	0.0024207	sign-test	1	0.000215599	69-31-0
Lisbon	7.996e-22	sign-test	1	4.1315e-05	71-29-0
Madrid	2.00823e-10	sign-test	1	2.0842e-11	84-16-0
Mont-Louis	1.80974e-21	sign-test	1	5.49592e-16	91-9-0
Murcia	3.65115e-21	sign-test	1	0.0214482	62-38-0

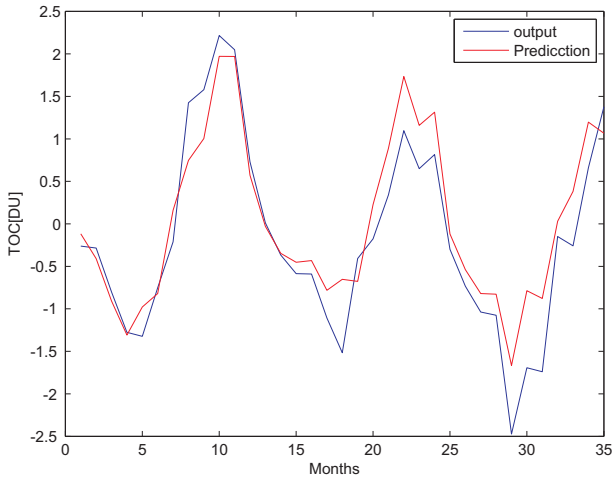


Figure 5.3: Real TOC and HH-GMDH prediction for Madrid TOM Nimbus-7 overpass observation site.

are 0.96 and 0.6279, respectively. Table 5.15 shows the adjusted p-values after application of the post hoc methods. In this case, the Holland procedure discards a significant difference between the GMDH and HH-GMDH$^*_\lambda$ with the control method (HH-GMDH$_\lambda$), although the GMDH appears to perform worse than the proposed HH-GMDH approaches.

Finally, we apply this statistical analysis methodology to all sets of experiments together. Table 5.16 shows the Friedman ranking that was obtained in this case. Table 5.17 shows the adjusted p-values after application of the post hoc methods. The Holland procedure does not show a significant difference between the GMDH and HH-GMDH$^*_\lambda$ with the control method (HH-GMDH$_\lambda$), but the GMDH performs consistently worse than the proposed HH-GMDH approaches.

Table 5.14: Friedman ranks (TOC prediction).

Algorithm	Ranking
GMDH	2.16
HH-GMDH$_\lambda$	1.92
HH-GMDH$^*_\lambda$	1.92

Table 5.15: TOC prediction. Adjusted p-values for the Friedman test (HH-GMDH$_\lambda$ is the control method).

Algorithm	unadjusted p	$p_{Holland}$
GMDH	0.3961	0.6353
HH-GMDH$^*_\lambda$	1	1

Table 5.16: Friedman ranks (all considered problems together).

Algorithm	Ranking
GMDH	2.34
HH-GMDH$_\lambda$	1.82
HH-GMDH$^*_\lambda$	1.82

Table 5.17: All considered problems together. Adjusted p-values for the Friedman test (HH-GMDH$_\lambda$ is the control method).

Algorithm	unadjusted p	$p_{Holland}$
GMDH	0.1984	0.3865
HH-GMDH$^*_\lambda$	1	1

5.4 Conclusions

In this study, we have presented a modification of the construction of group method of data handling networks (GMDH), using hyper-heuristics algorithms (HH-GMDH) driven by evolutionary computation algorithms. Our method allows automatic construction of the network by evolving the number of layers, the polynomial order and the number of nodes in each layer of the network. We have detailed the encoding and the different evolutionary operators specifically designed for the proposed hyper-heuristic approach.

In the experimental part of the chapter, we have shown the performance of the proposed technique on several prediction problems from the UCI public repository and two different real-world applications concerning temperature and ozone prediction. We have compared the HH-GMDH results with those by the classical GMDH. In some applications, we have also compared the HH-GMDH results with those of multi-layer perceptrons and Support Vector Regression algorithms. These results have shown the good performance of the hyper-heuristic approach in the automated construction of GMDH networks.

Chapter 6

Conclusions and future work

This Ph.D. Thesis has tackled different problems related to improvements on two state-of-the-art algorithms: Support Vector Regression (SVR) approaches and Group Method of Data Handing (GMDH). Specifically, an evolutionary-based multi-parametric version of the SVR is proposed, in such a way that the Gaussian kernel shows a different γ_i parameter for each dimension of the features space. This evolutionary version of the SVR allows a more accurate prediction within much less computational time that the classic grid search parameter selection, for the multi-parametric kernel case. In this case, a real-encoding evolutionary approach has been developed and adapted to be effective in the selection of the best SVR parameters for the multi-parametric kernel.

Two new validation methods to calibrate the performance of regression techniques. This new validation framework has been successfully tested for the SVR approach in several specific problems, with accurate results in terms of regression techniques evaluation capabilities, in comparison with a K-Fold cross validation approach.

A final contribution of this thesis consists of proposing a novel hyper-heuristic approach for the optimal construction of GMDH networks is the second major contribution of this thesis. In this case, the hyper-heuristic paradigm has been used to improve the GMDH construction, by encoding several basic heuristic for selecting the nodes for the next layer of the network, and also the network length. This hybrid approach was coined as HH-GMDH, and it was shown that the HH-GMDH was able to improve the performance of the classic GMDH approach in regression problems.

Direct applications of the improvements carried out in the SVR and GMDH techniques are the following:

- Development and analysis of an evolutionary multi-parametric SVR algorithm.
- Development of a new validation framework for data-driven regression techniques.
- Development and analysis of the HH-GMDH approach.

- Application of all these new proposals in two different real problems: Temperature forecasting at Barcelona's airport and Total ozone content (TOC) prediction at the Iberian Peninsula.

- Development of a missing value reconstruction system based on GMDH networks for wind farm management (Appendix B).

- Development of a system for wind speed estimation from pressure data (Appendix B).

6.1 Future work

This thesis opens an important amount of new research lines, in all the topics treated during this research:

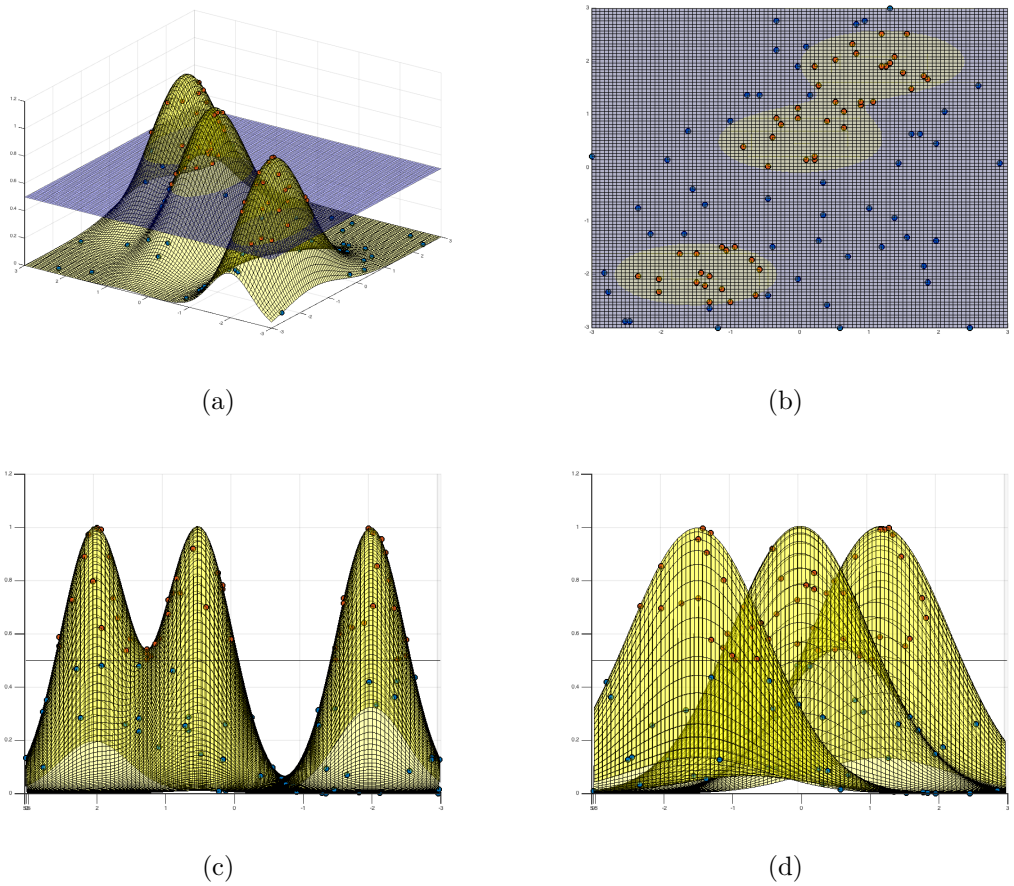
- A first line of future research consists of the application of the techniques developed in this thesis for the case of classification problems, taken into account the specific problematic of this field.

- For both classification and Regression problems, the evolutionary multi-parametric SVR developed in this thesis could be further improved. The basic idea is, not only consider a different γ for each dimension of the feature space, but also, consider a different vector of γ for each support vector of the problem. This would require a kind of re-evaluation of the SVR after a first step of training with a single-parameter Gaussian kernel. Figure 6.1 shows a problem where we have just three support vector and the Gaussians of each support vector are presented. We can see that these Gaussians have different widths but have exactly the same shape which correspond to the expression:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\sum_{m=1}^{M} \gamma_m (x_{im} - x_{jm})^2\right) \tag{6.1}$$

$$g(\mathbf{x}) = sign\left(\sum_{n=1}^{N} y_n \alpha_n K(\mathbf{x}_n, \mathbf{x}) + b\right) \tag{6.2}$$

  We propose for future works to define different Kernel parameters for each support vector, thus there would be infinite amount of shapes which can adapt better to



(a) (b)

(c) (d)

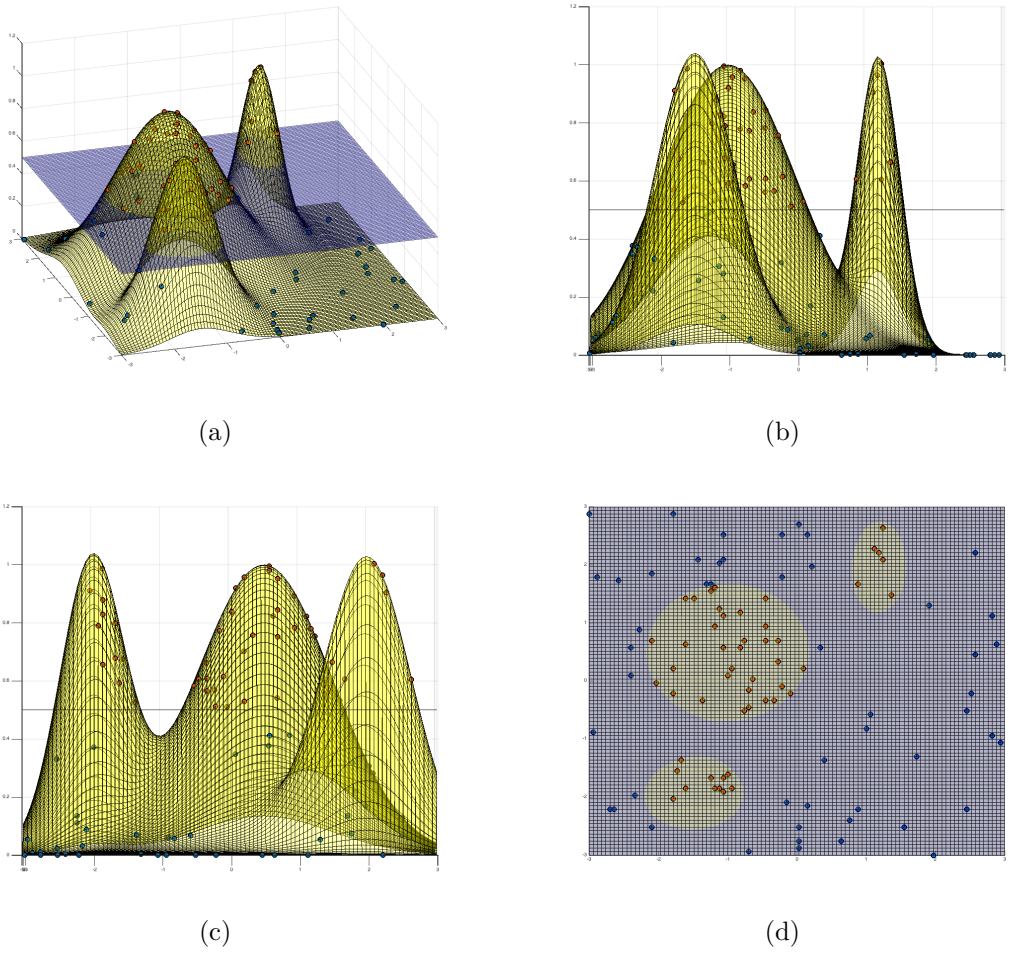
Figure 6.1: Example with three different support vector machines with the same multi-parametric Kernel.

complex problems. Figure 6.2 shows an example with Kernel tuned for each support vector.

$$K_n(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\sum_{m=1}^{M} \gamma_{nm}(x_{im} - x_{jm})^2\right) \tag{6.3}$$

$$g(\mathbf{x}) = sign(\sum_{n=1}^{N} y_n \alpha_n K_n(\mathbf{x}_n, \mathbf{x}) + b) \tag{6.4}$$

- Regarding new research lines in GMDH networks, this is a field not much explored in terms of its application to regression problems. One of the main advantages of this network is its extremely fast training, which makes it perfect to hybridize with evolutionary algorithms. In the last few years, these hybrid approach have been mainly constructed using alternative fast-training approaches, such as Extreme



(a) (b)

(c) (d)

Figure 6.2: Example with three different multi-parametric Kernels for each support vector.

Learning Machines (ELMs) [111], but these networks have different problems of variance control when they are evaluated to obtain the fitness values of evolutionary algorithms. The use of GMDH type networks can help control this problem, obtaining good performance hybrid approaches to difficult regression problems.

- The further development of GMDH networks implies to tackle an issue of this approach, the so-called *multicollinearity* of the network, which produces very high error values in specific cases. A deeper study of this phenomenon could lead to new or alternative implementations of the GMDH, either evolutionary-based approaches, such as the one proposed in this Thesis, or based on different computational frameworks.

Appendix A

List of publications

This section is a compilation of the scientific publications produced as a result of this research work, apart from those other studies conducted during the training process.

A.1 Papers related to the research work performed in this PhD.

Papers in international journals

1. J. Gascón-Moreno, S. Salcedo-Sanz, B. Saavedra-Moreno, L. Carro-Calvo, J.A. Portilla-Figueras, "An Evolutionary-based Hyper-Heuristic Approach for Optimal Construction of Group Method of Data Handling Networks", *Information Sciences*, vol.247, pp.94-108, 2013, (JCR: 4.038).

2. J. Gascón-Moreno, E.G. Ortiz-García, S. Salcedo-Sanz, L. Carro-Calvo, B. Saavedra-Moreno, J.A. Portilla-Figueras, "Evolutionary optimization of multi-parametric kernel ϵ -SVMr for forecasting problems", *Soft Computing*, vol. 17, pp.213-221, 2013, (JCR: 1.271).

3. B. Saavedra-Moreno, S. Salcedo-Sanz, L. Carro-Calvo, J. Gascón-Moreno, S. Jiménez-Fernández, L. Prieto, "Very fast training neural-computation techniques for real measure-correlate-predict wind operations in wind farms", *Journal of Wind Engineering and Industrial Aerodynamics*, vol 116, pp.49-60, 2013, (JCR: 1.414).

4. J. Gascón-Moreno, S. Salcedo-Sanz, E. G. Ortiz-García, F.J. Acevedo-Rodriguez, J.A. Portilla-Figures. "New validation methods for improving standard and multi-parametric Support Vector regression training time" (ISSN:0957-4174). *Expert Systems with Applications.* vol 9, num 39, pp. 8220-8227, 2012. (JCR: 2.24)

5. E.G. Ortiz-García, S. Salcedo-Sanz, A. M. Pérez-Bellido, J. Gascón-Moreno, J.A. Portilla-Figueras and L. Prieto, "Short-term wind speed prediction in wind farms based on banks of support vector machines", *Wind Energy*, vol.14, pp.193–207, 2011. (JCR: 3.07)

International conferences

1. J. Gascón-Moreno, S. Salcedo-Sanz, E.G. Ortiz-García, L. Carro-Calvo, B. Saavedra-Moreno, J.A. Portilla-Figueras, "A binary-encoded tabu-list genetic algorithm for fast Support Vector Regression hyper-parameters tuning", *International Conference on Intelligent Systems Design and Applications (ISDA)*, 2011.

2. J. Gascón-Moreno, S. Salcedo-Sanz, E.G. Ortiz-García, A. Paniagua-Tineo, B. Saavedra-Moreno, J.A. Portilla-Figueras, "Multi-parametric Gaussian Kernel Function Optimization for ϵ-SVMr Using a Genetic Algorithm", *International Work Conference on Artificial Neural Networks*, IWANN 2011, Part II, LNCS 6693, pp. 113-120, 2011.

3. A. Paniagua-Tineo, S. Salcedo-Sanz, E.G. Ortiz-García, J. Gascón-Moreno, B. Saavedra-Moreno, J.A. Portilla-Figueras, "On the performance of μ -GA Extreme Learning Machines in Regression Problems", *International Work Conference on Artificial Neural Networks*, IWANN 2011, Part II, LNCS 6693, pp. 153-160, 2011.

4. B. Saavedra-Moreno, S. Salcedo-Sanz, J.A. Paniagua-Tineo, J. Gascón-Moreno, A. Portilla-Figueras, "Optimal Evolutionary Wind Turbine Placement in Wind Farms Considering New Models of Shape, Orography and Wind Speed Simulation", *International Work Conference on Artificial Neural Networks*, IWANN 2011, Part I, LNCS 6691, pp. 25-32, 2011.

A.2 Other publications achieved during the training process

International conferences

1. E. G. Ortiz-García, S. Salcedo-Sanz, A. M. Pérez-Bellido, J. Gascón-Moreno, J.A. Portilla-Figueras. "Support Vector Regression Algorithms in the Forecasting of Daily Maximums of Tropospheric Ozone Concentration in Madrid", *International Conference Hybrid Artificial Intelligent Systems*, vol. 6077, pp. pp 304-311, 2010.

2. E.G. Ortíz-García, J. Gascón-Moreno, S. Salcedo-Sanz, A.M. Pérez-Bellido, J.A. Portilla-Figueras and L. Carro-Calvo. "A Novel Estimation of the Regularization Parameter for *epsilon*-SVM", *Intelligent Data Engineering and Automated Learning (IDEAL), 10th International Conference, Burgos*, pp. 34-41, 2009.

Appendix B

Technology transfer to productive sector

The research process of this Thesis has been accompanied by software development works for the productive sector. The main contribution of this Thesis to the productive sector is concentrated in three different software tools. The first one is a wind forecasting tool developed for Etulos Solute, S.L. This program use different machine learning algorithms to estimate the wind speed and direction for a given time horizon (24, 12 or 6 hours), based on meteorological data such as pressure and temperature. The second contribution is a real Measure-Correlate-Predict operation software for data reconstruction, which has been applied in order to fill wind data generated by turbines in a wind farm of Iberdrola S.A. Finally, a software to classify a company into bankrupt or healthy, given certain finance information.

B.1 Wind Prediction based on pressure data

Accurate wind speed prediction is an important problem that energy companies face every day when they have to estimate this wind farms production. Sometimes they need short-term forecastings, due to the fact that wind resource is not always available, and it is necessary to forecast the production that is going to be thrown into the energy system. For example, in USA hourly energy prediction is required in every wind farm. Regarding long-term forecasting, it is used in order to search the best place to install a wind farm and the distribution of the wind turbines into the selected area.

Usually, algorithms for long-term forecasting are based on historical wind measures, such us [112] and [113]. However, there are some atmospheric variables such as pressure, temperature and radiation which can help explain certain wind conditions. These variables can be used to make a wind long-term reconstruction in those areas where we

do not have wind historical data. We have developed a software which can make wind reconstruction and long-term prediction based on synaptic pressure data, given a certain coordinates of the area we want to know about.

B.1.1 Theoretical Background

Atmospheric pressure is the main source of wind generation due to pressure gradient in the high layers of the atmosphere. The simplest model of wind generation by pressure gradient is obtained by equalizing pressure gradient and the Coriolis force in the atmosphere. It is called the *geostrophic wind* approximation. The geostrophic wind derivation is as follows:

The Coriolis force by unit of volume is:

$$\begin{aligned} f_{cx} &= \rho \cdot f \cdot v \\ f_{cy} &= -\rho \cdot f \cdot u \end{aligned} \tag{B.1}$$

where f_{cx} and f_{cy} are the zonal and meridional components of the force, and f stands for the Coriolis parameter. On the other hand, the force due to pressure gradient can be expressed as follows:

$$\begin{aligned} f_{px} &= -\frac{\partial p}{\partial x} \\ f_{py} &= -\frac{\partial p}{\partial y} \end{aligned} \tag{B.2}$$

Making equal Equations (B.1) and (B.3) we obtain the expression for the geostrophic wind:

$$\begin{aligned} u_g &= -\frac{1}{f \cdot \rho}\frac{\partial p}{\partial x} \\ v_g &= \frac{1}{f \cdot \rho}\frac{\partial p}{\partial y} \end{aligned} \tag{B.3}$$

Geostrophic wind is a simple, but effective model of wind speed in high levels of atmosphere, out of the Planetary Boundary Level (about 1000m of height). It, however, differs from the measured wind measured at the ground, due to turbulence and other effects.

B.1.2 Data source

The meteorological data were provided by National Oceanic and Atmospheric Administration (NOAA) or European Centre for Medium-Range Weather Forecast (ECMWF). These two organizations have freely available a great amount of meteorological data, including wind speed and pressure. ECMWF provides us with scripts to access to the data in several different languages (Python, Perl, Ruby, CSharp, etc), all the source code and information about the API are available at [114]. We can download the data with two different format *netcdf* and *grib*, but also there is available a library to work with these two formats in [115]. Pressure is given by a grid of points with a certain resolution, ECMWF gives the possibility of data in different resolutions: 2.5º, 1.5º and 0.75º. On the other hand, NOAA has available all the data on a ftp server in [116], the data is in netcdf format. NOAA only has an unique possible resolution of 2.5º for free access. The data is available in sampling steps of 6 hours, 12 hours and 24 hours.

In order to get the wind estimator the system need wind historical data from an area as close as possible to the region of the study. The program parses the pressure data from the region around point and generate a file for the dates that we have previously read it from the wind file. As a result, we get the full training set with pressure and wind data (inputs/output). Once the model has been created the program only needs a new grid of pressure to make a wind speed estimation.

B.1.3 Learning algorithms

Regarding the training process, two different machine learning algorithms are implemented in this software, the user has the possibility to choose which of them to apply. The first one is a clustering method presented in [117]. This article describes an evolutionary algorithm for clustering in a search space of atmospheric pressures in a grid. This is the first work which incorporates an evolutionary algorithm to the extraction of these synoptic pressure patterns. The objective is to obtain N groups of synoptic pressure situations which produce the most similar wind vectors (wind speed and direction) in a given point (usually a wind farm project).

The second algorithm is the well-known Support Vector Regression, already described in Section 2.1. In this case, we treat the wind forecasting as a regression problem and we get a real value output. In addition, SVR has the advantage of add news meteorological information (just appending them to the input vector) and see whether the new data is useless or not.

B.1.4 Results

In this section, it is presented a performance demonstration of this software over one location that there are available wind data. This data is from Colorado: Latitude North 37.7078º, longitude West 104.8112º and elevation of 1886 meters. The data have been recorded from 2/1/2006 to 9/8/2010 in time steps of 10 minutes, daily average will be obtained from these data.

Table B.1 shows the forecasting made with Clustering evolutionary algorithm for daily average wind. The table contains the result for different number of classes, and also shows the results with two different data sources (NOAA and ECMWF). In terms of number of classes, it seems that with a few number of classes (less than 24) the error is lower than with a large number of classes. Also we see that the results with data from NOAA are better than the one obtained with data from the European agency. Table B.2 shows the results for SVR algorithm. In this case, the error is lower than the clustering method and also the results are better with NOAA data pressure than ECMWF. Figures B.1 and B.2 represent the wind prediction for Colorado with the real values. We can see how close is our two estimation methods graphically. Furthermore, Figures B.3 and B.4 present in other way the wind forecasting with the Weibull distribution, we see that for B.4 there is not big differences between the real and estimated distribution. However Figure B.3 shows that the Weibull distribution obtained with 10 classes clustering, is quite different than the original one. Note however, that in terms of error (1.9347 m/s) it is very similar to the estimation made by SVM (1.9076 m/s). This means that the 10 classes have values in those areas where more wind values are concentrated.

B.2 Real Measure-Correlate-Predict Operation Software

Usually, a typical problem collecting data is the existence of gaps, due to errors in the measurements or because over certain circumstances the measures could not be carried out. This is a typical problem in wind farms using data from in-situ measuring towers (see [118]). Data reconstruction also known as Measure-Correlate-Predict (MCP) is usually carried out by different statistical methods: traditional ones, such as linear MCP techniques have been applied in different studies of wind farm site assessment [119] or wind speed reconstruction [120]. But also some machine learning methods such as neural networks [121], Support Vector Machines [122] or Bayesian Networks [123].

However most of the articles do not cover Real MCP Operations (RMCPO). A RMCPO problem consists in tackling the raw reconstruction or prediction problem by

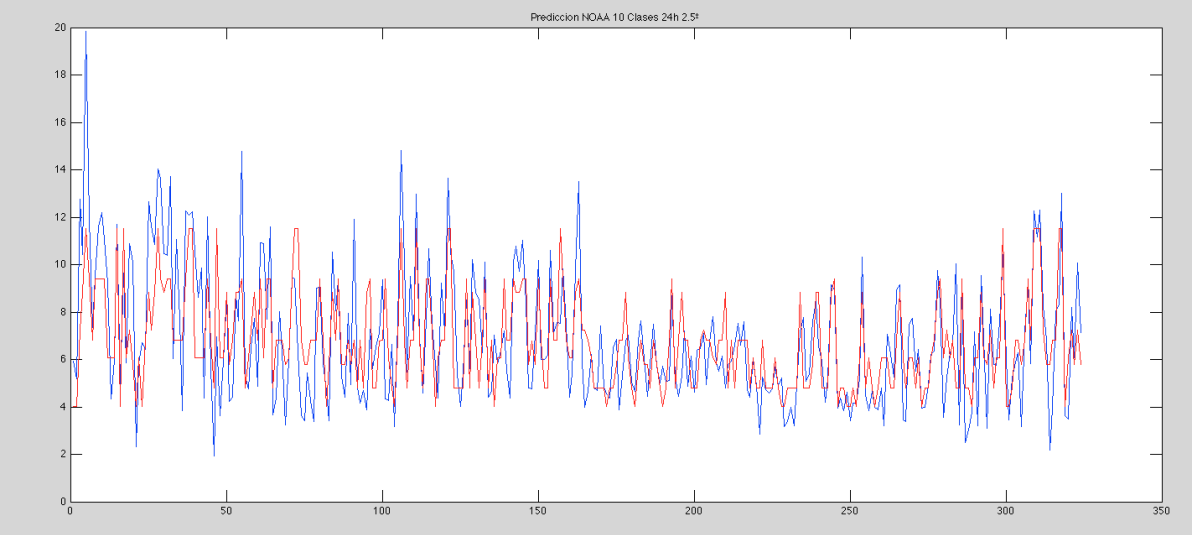


Figure B.1: Comparison between real wind value (blue line) and estimation with 10 classes clustering (red line).

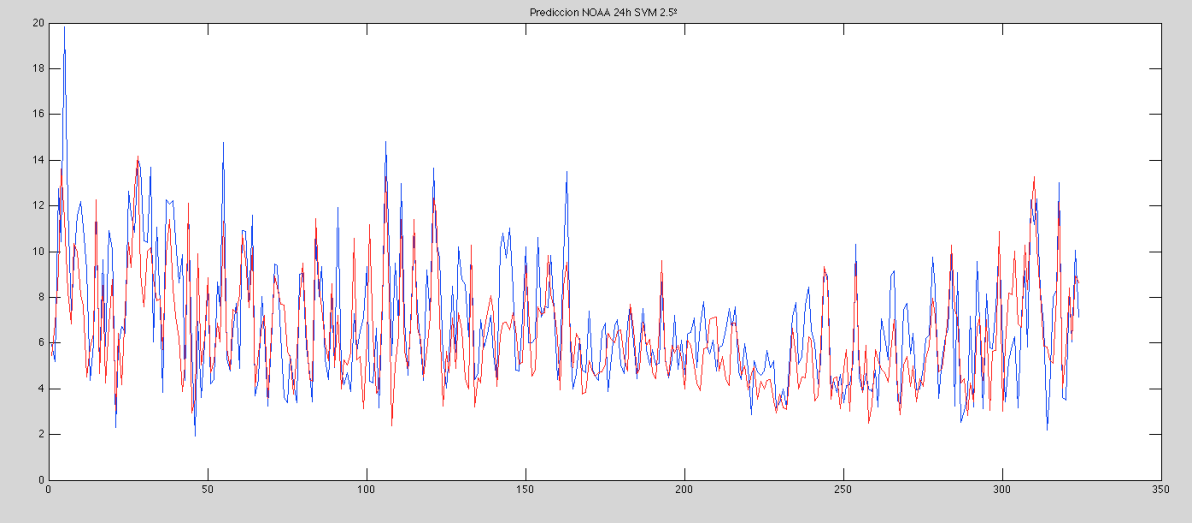


Figure B.2: Comparison between real wind value (blue line) and estimation with SVR algorithm (red line).

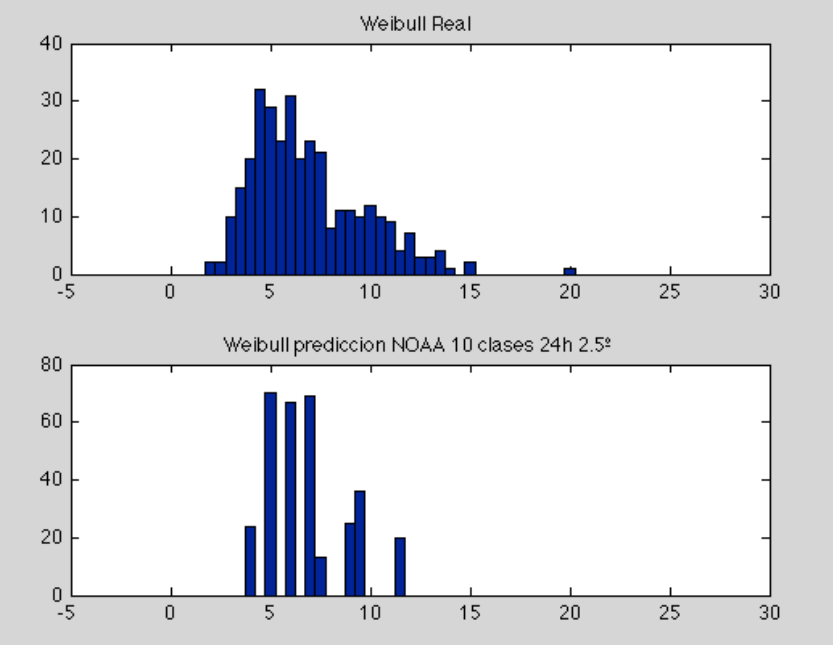


Figure B.3: Weibull distribution for real data and estimation results with 10 classes clustering.

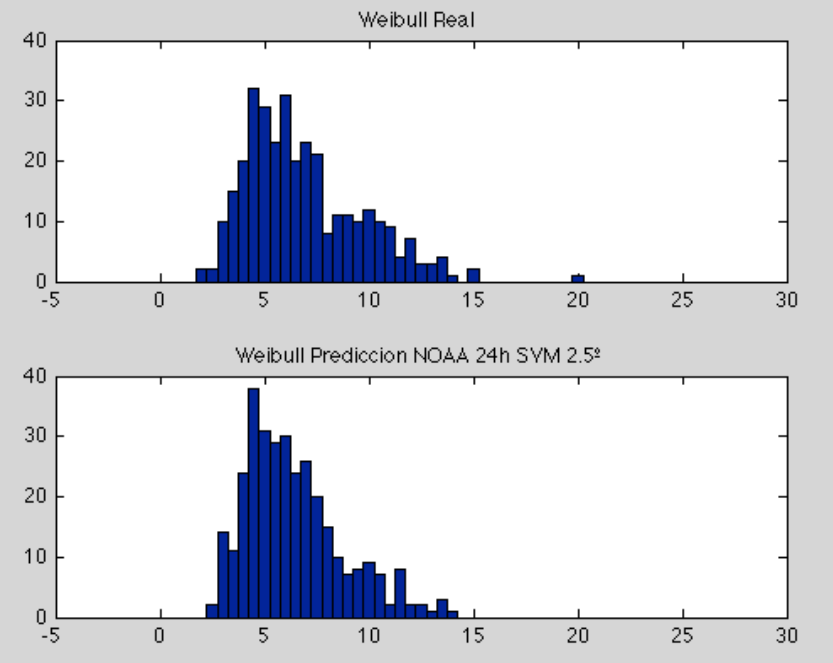


Figure B.4: Weibull distribution for real data and estimation results with SVR algorithm.

Table B.1: Clustering evolutionary algorithm forecasting results for different number of classes. Pressure data are from NOAA and ECMWF both with 2.5º of resolution.

	NOAA (2.5º)		ECMWF (2.5º)	
Class	RMSE(m/s)	Bias	RMSE(m/s)	Bias
4	2.0434	-0.0145	2.3310	-0.0426
6	2.0022	-0.2407	2.2053	-0.2798
8	2.0162	-0.1271	2.1646	-0.2325
10	1.9347	-0.1097	2.1618	-0.1156
12	2.0685	-0.2651	2.3316	-0.1570
14	2.0996	-0.0427	2.1822	-0.0391
16	2.0224	-0.0151	2.2117	-0.1692
18	2.0705	0.0500	2.1605	-0.0720
20	2.1296	-0.0387	2.3174	-0.0401
22	2.0830	0.0774	2.2201	0.0232
24	2.3309	0.1866	2.2622	-0.1553
26	2.1429	-0.1733	2,3847	-0.0388
28	2.1367	0.0366	2.2410	0.0563
30	2.1139	-0.0714	2,2647	0.0307
32	2.0161	0.1282	2.4446	0.0750
34	2.0526	0.0984	2.2562	-0.0055
36	2.2160	-0.0443	2.3576	0.0341
38	2.0643	-0.0697	2.2979	0.0746
40	2.3606	0.0673	2.7552	-0.0211
42	2.3312	-0.0880	2.3600	-0.1635
44	2.2205	-0.0172	2.4053	0.0087
46	2.3818	0.0332	2.3213	-0.1085
48	2.0942	-0.1337	2.2329	-0.0180
50	2.4222	-0.0125	2.2577	0.0035

Table B.2: SVR algorithm forecasting Colorado wind results. Pressure data are from NOAA and ECMWF both with 2.5º of resolution.

NOAA (2.5º)		ECWMF (2.5º)	
RMSE(m/s)	Bias	RMSE(m/s)	Bias
1.9076	-0.4623	1.9582	-0.3602

training prediction models from the existing data and estimates the missing values. With large set of data a lot of different models have to be created to fill all the gaps. Thus,

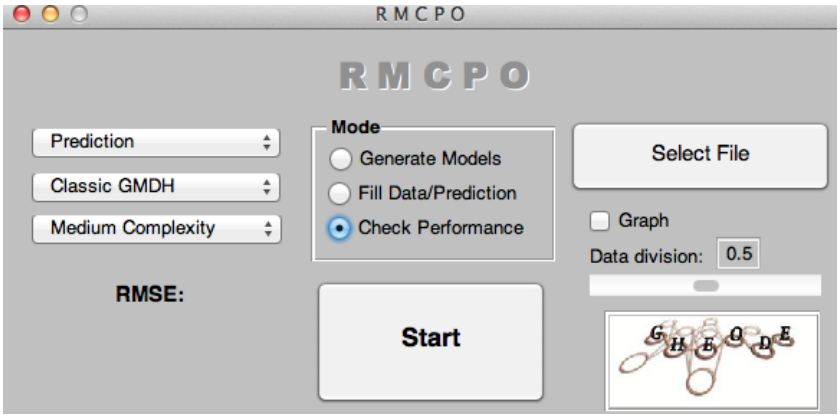


Figure B.5: RMCPO User Interface with MATLAB.

very fast training approaches must be used in RMCPO problems, and that is why linear regression approaches are usually applied in industry for RMCPO problems.

Our RMCPO software has been developed in order to perform the gap filling task in a straightforward way. The machine learning method behind this software is the Group of Method of Data Handling, described in Section 2.2 where we saw that GMDH is a very fast method with a training time much lower than other usual methods such as SVM, neural network, etc, that makes GMDH an excellent option for RMCPO processes. We also implement HH-GMDH algorithm described in Chapter 5 which training process takes a bit longer but in general gets better results than the basic GMDH. As a benchmark it is also implemented the Multiple Linear Regression (MLR) with Least Squares Error as parameters estimator (2.2.6).

B.2.1 User Interface

The program has been developed with MATLAB. The User Interface has been designed to be very simple in order to help those people that are not very familiar with computer science and machine learning. Most of the bottoms and combo boxes are self-descriptive. This software has two operating ways, first one and principal is filling incomplete data sets (RMCPO). But also the program has the possibility of making estimations over a new data inputs, with a model previously trained and saved. Figure B.5 shows this user interface.

B.2.2 RMCPO

This mode receives a data set with gaps of correlated measures and the software fills these gaps with the information available. Let $\mathbf{D}$ be a $N \times M$ matrix, where N is the

number of samples and M is the number of inputs for a given sample. The RMCPO problem consists in, given $\mathbf{D}$, build a set of $K(\mathbf{D})$ regression models, for covering all the possible combinations of available data and missing values to be reconstructed. All the regressions models have at least two inputs features.

$$D = \begin{bmatrix} 5.1 & X & 3.8 & 2.9 \\ 6.0 & 7.2 & X & X \\ X & X & 1.3 & 1.2 \\ 9.4 & 6.2 & 4.8 & 7.9 \\ X & 9.6 & X & 8.5 \\ 6.2 & 2.9 & 4.0 & X \\ 4.1 & 7.2 & 4.3 & 6.8 \\ 4.5 & 5.2 & 6.1 & X \\ 2.1 & 3.6 & 5.1 & 3.8 \\ 8.6 & 9.2 & 10.0 & X \end{bmatrix} \tag{B.4}$$

Matrix B.4 is an example of a data set with gaps, and represents wind speeds of four different wind turbines. If we would like to fill wind speed of turbine #2 in the first row, we should create a model with three inputs from turbines #1, #3, and #4. So the training set needs sample with the four inputs, that is rows 4, 7, and 9. However, if we need to get the value of turbine #3 in the second row, we need inputs from turbine #1 and #2, the training set needs values from turbines #1, #2 and #3 to create the model, we can use rows 4, 6, 7, 8, 9 and 10.

The total number of models to cover all the possibilities depends on the input feature space M and it equals:

$$K(\mathbf{D}) = \sum_{i=0}^{M-3} \frac{M!}{i!(M-i)!}(M-i) \tag{B.5}$$

Next steps summarize how to use the program for RMCPO:

- Select mode **Generate Models**, choose the data file that we want to refill and press **start**. The process will take a while till all the models are created.
- Once we have all the models, change the mode to **Fill Data/Prediction** and press **start**. This way, it is created a file called **FileName_reconst.dat**. This file has all the gaps filled.

It is also possible to test the performance of the software and different techniques (GMDH and MLR). For this task, we pass a full data set without any gap, this way the program

generate automatically a new file with random gaps, generate all model to refill the file and finally check the error with the original file.

B.2.2.1 File Format

Data files extension uses *.dat*, where each input data is separated by coma and the output is the last value of the row. See the next example:

$$\begin{bmatrix} X_{11}, X_{12}, X_{13}, \ldots, X_{1N}, Y_1 \\ X_{21}, X_{22}, X_{23}, \ldots, X_{2N}, Y_2 \\ \vdots \\ X_{M1}, X_{M2}, X_{M3}, \ldots, X_{MN}, Y_M \end{bmatrix} \tag{B.6}$$

B.2.2.2 RMCPO performance comparison over a real problem

The performance of this software has been measured in real data from a wind farm in Spain. Figure B.6 shows the location of the eight measuring stations considered, within the same wind farm, situated in Guadalajara, Spain. The data set gathers hourly wind samples from November 2008 to November 2010. We have carried out and experiment in order to focus on the specific reconstruction. We have available a total of 239 complete temporal series, of different length, with a total of 4391 hours of average wind speed data for the eight towers. The distribution length of the considered series is shown in Figure B.7. The performance of GMDH in a RMCPO, consisting in the reconstruction of the missing values in the set of eight considered towers.

First, the problem of wind speed reconstruction using the complete data from neighbor towers is tackled. In this problem, the wind value of different neighbor towers is used as input in the regression techniques, in order to estimate the wind at the same time in an objective tower. Since we have available all the data in each tower, we can evaluate the accuracy of each considered method. This evaluation is carried out in terms of different well-known statistical evaluation indices in each tower: such as root mean square error (RMSE) and the mean bias error (MBE), which checks whether the model overestimates or underestimates the wind speed, the Coefficient of Determination (R^2), which provides information about the percentage of the variance that the model is able to explain, and the Index of Agreement (IoA), which gives information about how close the predicted wind speed values are to the observed ones. These performance indices have been used profusely before in different studies, including in environmental applications [124–126]. We also compare the algorithms in terms of the final computation time for training each method. We have carried out wind speed reconstruction and prediction for all the towers

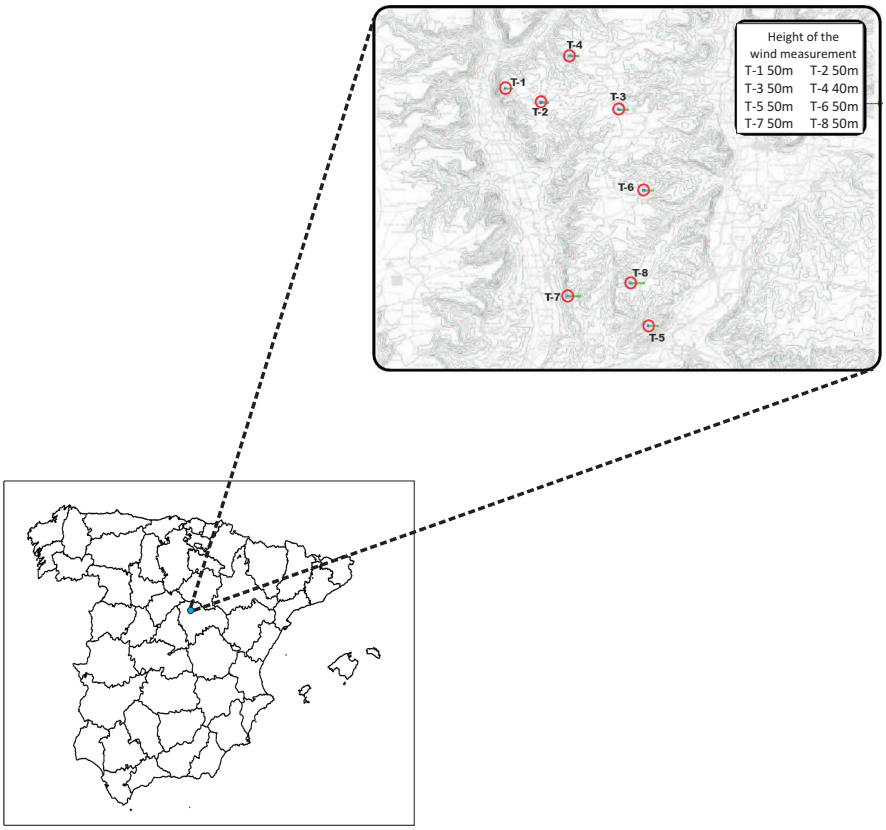


Figure B.6: Situation of the wind measuring towers in Spain and within the wind farm.

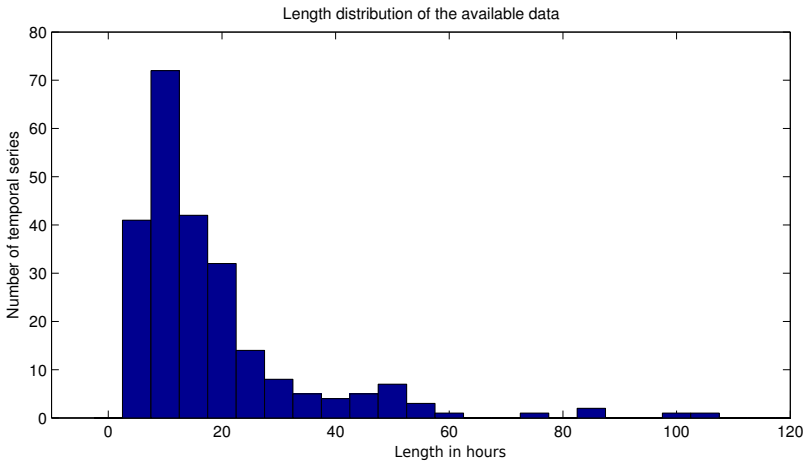


Figure B.7: Distribution of the wind speed time series (complete samples) in terms of their duration.

by using a different number of neighbor towers: the 3 nearest towers to the target one (called 3T case), the 5 nearest (called 5T case) and all the available towers (called 7T case). Table B.3 shows a summary of towers used to reconstruct/predict each target tower. This table can be seen together with Figure B.6, where the current disposition of the towers is shown. The table details which towers are used in the 3T case (using the 3 nearest towers to do the reconstruction/prediction) and in the 5T case (using the 5 nearest towers to do the reconstruction/prediction). Note that in the 7T case all the towers but the target one are used in the reconstruction/prediction process.

Table B.3: Towers used to reconstruct/predict the target tower, by using a different number of neighbor towers (the 3 nearest (3T case) or the 5 nearest (5T case)). Note that in the 7T case all the towers but the target one are used in the reconstruction/prediction process.

Target tower	Nearest neighbor towers (3T case)	Nearest neighbor towers (5T case)
1	2, 3, 4	2, 3, 4, 6, 7
2	1, 3, 4	1, 3, 4, 6, 7
3	1, 2, 4	1, 2, 4, 6, 8
4	1, 2, 3	1, 2, 3, 6, 8
5	6, 7, 8	2, 3, 6, 7, 8
6	2, 3, 8	2, 3, 5, 7, 8
7	5, 6, 8	2, 3, 5, 6, 8
8	5, 6, 7	2, 3, 5, 6, 7

Table B.4 shows the performance of the GMDH, and Tables B.5, B.6 and B.7 show the alternative methods for comparison (MLR, MLP and SVR), in terms of the different evaluation indices considered. Note that the performance of GMDH is quite acceptable, with evaluation indices comparable to the obtained by the compared approaches. The SVR seems to be the most accurate method among all tested in general, the MLR (reference method) is the one which provides the poorer results. Regarding the differences between the wind speed reconstruction using 3, 5 or 7 towers, it is interesting to see how the reconstruction using 3 reference towers (3T case) provides poorer results than the reconstruction using 5 (5T case) or 7 (7T case). However, the reconstruction using 5 towers (5T case) is, in many cases, better than the one using 7 towers. This indicates that the information provided by 5 towers is enough to obtain a good quality reconstruction of the wind speed, whereas the information of the 3 nearest towers is not enough to provide the best possible reconstruction.

Table B.4: Wind speed reconstruction results obtained by the GMDH network.

GMDH reconstruction 3T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.955	0.019	0.977	0.950
2	0.952	0.001	0.987	0.676
3	0.946	0.002	0.9823	0.738
4	0.938	0.058	0.983	0.761
5	0.965	-0.001	0.985	0.818
6	0.781	-0.009	0.976	0.847
7	0.951	0.001	0.969	1.263
8	0.972	-0.006	0.986	0.789
GMDH reconstruction 5T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.928	0.010	0.978	0.928
2	0.950	-0.003	0.990	0.599
3	0.974	0.003	0.985	0.690
4	0.941	0.047	0.984	0.730
5	0.969	-0.009	0.987	0.773
6	0.787	-0.018	0.975	0.868
7	0.987	0.013	0.970	1.256
8	0.985	-0.010	0.987	0.770
GMDH reconstruction 7T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.928	0.010	0.978	0.928
2	0.949	-0.003	0.990	0.585
3	0.973	0.003	0.985	0.690
4	0.941	0.047	0.984	0.730
5	0.969	-0.009	0.987	0.773
6	0.796	-0.030	0.975	0.871
7	0.926	-0.041	0.971	1.248
8	0.985	0.005	0.987	0.760

Table B.5: Wind speed reconstruction results obtained by the MLR method (Reference).

MLR reconstruction 3T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.937	-0.084	0.976	0.966
2	0.961	0.080	0.989	0.611
3	0.944	0.038	0.983	0.712
4	0.959	-0.059	0.984	0.747
5	0.968	0.021	0.987	0.770
6	0.830	-0.053	0.977	0.835
7	0.822	-0.190	0.964	1.312
8	0.975	0.071	0.986	0.780
MLR reconstruction 5T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.962	-0.059	0.978	0.936
2	0.981	0.087	0.990	0.599
3	0.953	0.032	0.9841	0.702
4	0.949	-0.045	0.984	0.727
5	0.963	0.016	0.986	0.766
6	0.824	-0.059	0.978	0.830
7	0.806	-0.202	0.966	1.280
8	0.959	0.057	0.987	0.764
MLR reconstruction 7T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.963	-0.052	0.978	0.943
2	0.983	0.057	0.990	0.590
3	0.954	0.031	0.984	0.700
4	0.948	-0.048	0.984	0.727
5	0.963	0.019	0.987	0.767
6	0.826	-0.058	0.977	0.837
7	0.812	-0.169	0.966	1.271
8	0.962	0.041	0.987	0.755

Table B.6: Wind speed reconstruction results (average values of 30 runs) obtained by the MLP.

MLP reconstruction 3T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.958	-0.080	0.977	0.950
2	0.968	0.084	0.989	0.616
3	0.938	-0.010	0.985	0.677
4	0.911	-0.097	0.981	0.783
5	0.942	0.014	0.986	0.762
6	0.836	-0.066	0.977	0.846
7	0.834	-0.169	0.966	1.292
8	0.992	0.069	0.987	0.755
MLP reconstruction 5T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.965	-0.056	0.978	0.930
2	0.982	0.086	0.989	0.605
3	0.937	-0.003	0.985	0.669
4	0.933	-0.039	0.984	0.724
5	0.946	0.006	0.987	0.760
6	0.840	-0.052	0.976	0.852
7	0.819	-0.238	0.967	1.269
8	0.970	0.043	0.987	0.751
MLP reconstruction 7T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.968	-0.059	0.978	0.935
2	0.981	0.056	0.990	0.590
3	0.952	0.012	0.986	0.668
4	0.931	-0.046	0.983	0.748
5	0.945	0.018	0.987	0.766
6	0.853	-0.073	0.975	0.878
7	0.844	-0.130	0.969	1.232
8	0.982	0.041	0.988	0.729

Table B.7: Wind speed reconstruction results obtained by the SVR.

SVR reconstruction 3T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.952	-0.141	0.977	0.961
2	0.972	0.088	0.989	0.616
3	0.953	0.038	0.985	0.672
4	0.931	-0.011	0.983	0.747
5	0.965	0.081	0.987	0.772
6	0.833	-0.015	0.977	0.844
7	0.832	-0.129	0.967	1.275
8	0.984	-0.036	0.988	0.734
SVR reconstruction 5T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.979	-0.106	0.978	0.930
2	0.978	0.069	0.990	0.599
3	0.955	0.039	0.986	0.665
4	0.944	-0.019	0.984	0.726
5	0.965	0.077	0.987	0.768
6	0.807	-0.043	0.977	0.844
7	0.789	-0.211	0.967	1.252
8	0.981	-0.036	0.988	0.738
SVR reconstruction 7T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.984	-0.099	0.978	0.941
2	0.976	0.058	0.990	0.590
3	0.958	0.039	0.986	0.661
4	0.940	-0.029	0.984	0.727
5	0.965	0.078	0.987	0.767
6	0.817	-0.048	0.977	0.845
7	0.831	-0.121	0.970	1.214
8	0.979	-0.036	0.989	0.704

Table B.8 shows the training time in the reconstruction problem, in the case of data from 7 towers (7T case). This table shows one of the main advantages of using GMDH networks in wind speed reconstruction. This algorithm takes about 1 second per training model, but it is still less than the MLP, and of course than the SVR approach, which is the algorithm that employs more time in a single model training. On the other hand, MLR approach is the fastest one (about 30 ms per training).

Table B.8: Computation time of wind speed reconstruction (using data from 7 towers) in each considered tower, in the example with complete wind speed samples (in seconds).

Tower	GMDH	MLP	SVR	MLR (Reference)
T1	1.078	20.001	584.5	0.028
T2	1.149	21.315	617.1	0.023
T3	0.788	20.975	586.5	0.029
T4	0.891	22.850	601.3	0.030
T5	0.982	21.153	596.7	0.027
T6	1.034	22.179	589.9	0.027
T7	1.101	20.336	598.1	0.031
T8	0.972	22.891	619.2	0.024

B.2.3 Forecasting

The Software has also a mode to make predictions with a model previously trained. We summarize in the next steps how this forecasting mode works:

- Build the estimation model: we set the mode **General Models** and select the file with the training data. After press **start** the program initiate the training process and after a few time we get a model save in a folder with the name selected mode $GMDH\_model.mat$, $HHGMDH\_model.mat$ or $MLR\_model.mat$.
- Once created the model, we use it to get predictions from a file with new samples to estimate. This case we have to select mode **Fill Data/Prediction** and press start. The results are saved in $GMDH\_pred.dat$, $HHGMDH\_pred.dat$ or $MLR\_pred.dat$

Like for RMCPO mode, in prediction, there is also the possibility of checking the performance with a complete data set, the program will make a partition in two different subsets one for training and one for the test. After a training process we get a model which performance is measured with the test set. The program allows choosing the size of the partition, and also creating graphs with the estimations and the real outputs (Figure B.8).

B.2.3.1 Results

The performance of the forecast module will be tested over the same data set than the RMCPO, predicting the wind speed in each of the towers from the measures of the other

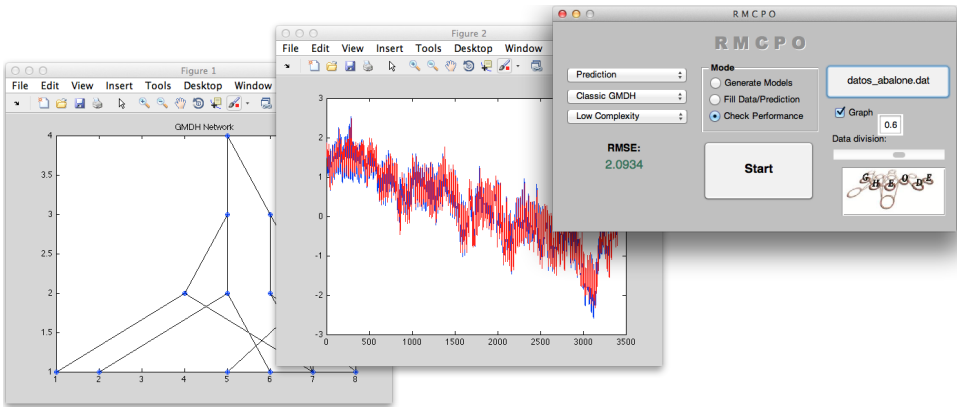


Figure B.8: Checking performance in prediction mode.

towers. In order to face this problem with complete samples the initial data have been split in train, validation and test sets. Figure B.9 shows how the 239 time series have been split. The distribution in seasons of the samples in the train, validation and test sets considered is shown in Figure B.10. Tables B.9 to B.12 show the results obtained by the different techniques considered in the prediction problem, with different number of towers to train the predictors (3T, 5T and 7T cases). Note that the results obtained in the prediction problem are slightly worse in terms of accuracy (in all evaluation indices) for the compared algorithms, as expected. However, again the results obtained by the GMDH are quite competitive to the other tested algorithms, and improve the results of the reference method (MLR). We can also observe in this problem the same effect regarding the number of reference towers used to do the prediction: the results with 3 towers are worse than with 5 and 7 towers in the prediction, but the results using 5 towers are, in average, slightly better than the results obtained using 7 towers in the prediction. Table B.13 shows the training time obtained by the compared algorithms in each tower, for the case of predictions using data from 7 towers. As in the previous case, the training time of the GMDH is less than the other neural approaches compared.

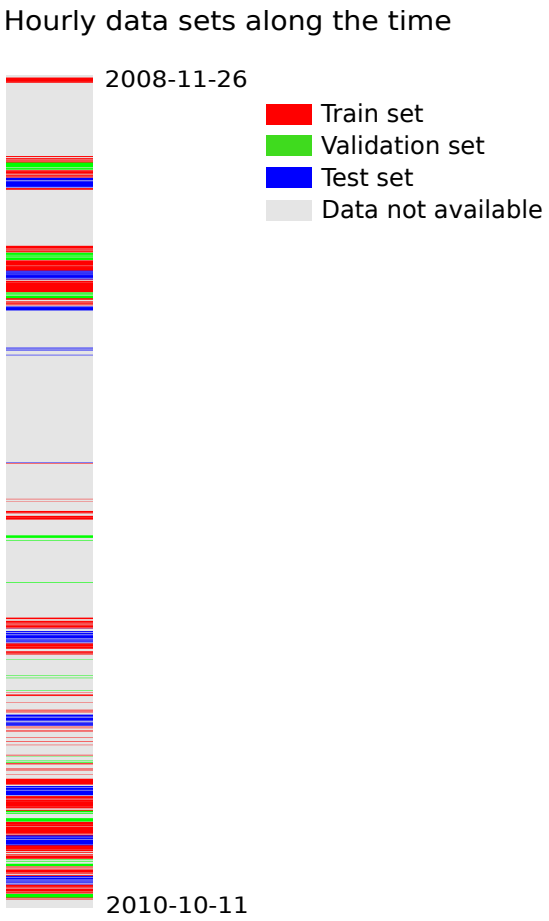


Figure B.9: Distribution of the sets for training, validation and test.

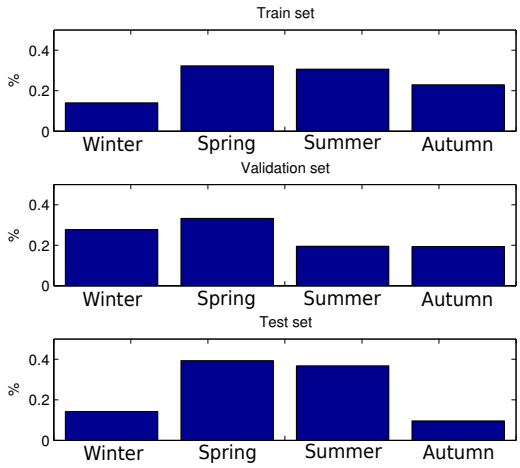


Figure B.10: Balanced distribution over the time of the sets.

Table B.9: Wind speed prediction results obtained by the GMDH network.

GMDH prediction 3T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.836	0.004	0.945	1.409
2	0.875	-0.023	0.957	1.182
3	0.842	-0.012	0.939	1.305
4	0.833	0.022	0.946	1.282
5	0.884	-0.032	0.947	1.470
6	0.702	-0.054	0.934	1.346
7	0.843	-0.030	0.930	1.823
8	0.867	-0.004	0.951	1.406
GMDH prediction 5T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.775	0.016	0.943	1.400
2	0.846	-0.022	0.956	1.181
3	0.867	-0.003	0.943	1.272
4	0.833	0.023	0.946	1.285
5	0.897	-0.043	0.952	1.410
6	0.717	-0.031	0.939	1.305
7	0.899	-0.015	0.933	1.804
8	0.920	-0.015	0.955	1.361
GMDH prediction 7T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.777	0.017	0.944	1.395
2	0.868	-0.029	0.957	1.185
3	0.867	-0.003	0.943	1.272
4	0.841	-0.036	0.944	1.310
5	0.926	-0.075	0.949	1.459
6	0.717	-0.031	0.939	1.305
7	0.934	-0.074	0.936	1.785
8	0.948	-0.044	0.954	1.387

Table B.10: Wind speed prediction results obtained by the MLR method (Reference).

MLR prediction 3T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.853	-0.081	0.945	1.408
2	0.876	0.063	0.958	1.175
3	0.832	0.037	0.939	1.305
4	0.869	-0.035	0.947	1.282
5	0.861	0.057	0.949	1.436
6	0.746	-0.053	0.941	1.297
7	0.733	-0.154	0.924	1.834
8	0.877	0.106	0.952	1.394
MLR prediction 5T				
Tower	R^2	MBE (m/s	IoA	RMSE (m/s)
1	0.885	-0.051	0.949	1.377
2	0.890	0.068	0.958	1.170
3	0.840	0.048	0.941	1.291
4	0.863	-0.021	0.948	1.273
5	0.849	0.040	0.950	1.413
6	0.743	-0.060	0.941	1.295
7	0.714	-0.174	0.927	1.789
8	0.856	0.081	0.953	1.370
MLR prediction 7T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.885	-0.047	0.948	1.381
2	0.891	0.045	0.959	1.171
3	0.838	0.040	0.941	1.284
4	0.860	-0.026	0.948	1.270
5	0.850	0.048	0.949	1.422
6	0.740	-0.052	0.942	1.281
7	0.726	-0.132	0.927	1.795
8	0.857	0.077	0.954	1.362

Table B.11: Wind speed prediction results (average values of 30 runs) obtained by the MLP.

MLP prediction 3T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.833	-0.081	0.945	1.401
2	0.877	0.072	0.957	1.188
3	0.802	-0.003	0.942	1.265
4	0.821	-0.046	0.944	1.295
5	0.834	0.060	0.949	1.416
6	0.755	-0.028	0.939	1.314
7	0.750	-0.147	0.926	1.819
8	0.886	0.100	0.954	1.371
MLP prediction 5T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.850	-0.075	0.948	1.374
2	0.881	0.055	0.958	1.180
3	0.817	0.058	0.944	1.253
4	0.842	-0.018	0.946	1.278
5	0.851	0.004	0.951	1.402
6	0.763	-0.057	0.941	1.302
7	0.738	-0.182	0.930	1.768
8	0.873	0.089	0.955	1.349
MLP prediction 7T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.845	-0.060	0.947	1.379
2	0.881	0.048	0.958	1.179
3	0.832	0.065	0.945	1.246
4	0.844	-0.003	0.947	1.272
5	0.835	0.079	0.949	1.425
6	0.758	-0.069	0.940	1.312
7	0.752	-0.074	0.931	1.760
8	0.893	0.080	0.956	1.350

Table B.12: Wind speed prediction results obtained by the SVR.

SVM prediction 3T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.865	-0.052	0.948	1.378
2	0.886	0.024	0.959	1.168
3	0.799	0.041	0.942	1.263
4	0.804	-0.002	0.946	1.272
5	0.884	0.031	0.951	1.411
6	0.769	0.005	0.943	1.279
7	0.785	-0.120	0.932	1.769
8	0.898	0.105	0.956	1.349
SVM prediction 5T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.850	-0.054	0.948	1.377
2	0.862	0.057	0.958	1.173
3	0.792	0.051	0.941	1.270
4	0.867	-0.067	0.947	1.279
5	0.887	0.018	0.951	1.417
6	0.764	0.009	0.942	1.295
7	0.764	-0.247	0.931	1.771
8	0.885	0.098	0.955	1.353
SVM prediction 7T				
Tower	R^2	MBE (m/s)	IoA	RMSE (m/s)
1	0.852	-0.065	0.945	1.407
2	0.881	0.053	0.958	1.175
3	0.854	-0.021	0.943	1.272
4	0.858	-0.091	0.946	1.289
5	0.884	0.017	0.950	1.437
6	0.775	0.006	0.942	1.296
7	0.730	-0.162	0.926	1.809
8	0.904	0.122	0.953	1.390

Table B.13: Computation time of wind speed prediction (using data from 7 towers) in each considered tower, in the example with complete wind speed samples (in seconds).

Tower	GMDH	MLP	SVR	MLR (Reference)
T1	0.992	17.254	501.4	0.024
T2	1.091	17.973	507.9	0.025
T3	0.699	18.597	505.1	0.023
T4	0.783	17.366	499.3	0.027
T5	0.647	17.426	517.9	0.025
T6	0.546	18.007	499.6	0.024
T7	0.981	17.884	534.4	0.027
T8	1.127	17.677	525.9	0.028

B.3 Company bankrupt classification software

The third software development from this Thesis which produced technology transfer to the productive sector was a program to classify the bankrupt of companies. It was called *NeuCompBankrupt*, and consists of a graphical user interface which allows the user to launch different machine learning tools to classify a company into bankrupt or healthy. Figure B.11 shows a screen shot of the NeuCompBankrupt user interface.

Figure B.11: Image from the graphical user interface of software tool *NeuCompBankrupt*.

The tool allows training different machine learning classifiers, such as SVMs and GMDH networks, defined in this Thesis in Chapter 2. Note that in this case these

techniques are defined for classification problems. These machine learning tools are fed with several economic parameters of the company in order to classify it as bankrupt or healthy. This inputs are: ROA, ROIC, ROCE, cash ratio, guarantee ratio, debt to equity ratio, debt to capital ratio, sales, interest coverage ratio, distress criterion, working capital as a percentage of total assets, debt to income ratio, economic threshold level and financial independence.

The behavior is very similar to the previous software user interface. By pressing the button "model generation" and setting the algorithm to be used, the tool trains the selected approach, and generates a file *Modelo_AlgoritmoUti.mat*, which includes the definition of the trained classifier. In order to carry out the final bankrupt prediction, we have to select the option "prediction" in the tool's menu. Once the file with the inputs has been selected, a new file will be generated with the result of the classification for the company (bankrupt or healthy).

Bibliography

[1] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Adaptive Computation and Machine Learning series, The MIT Press, 1st ed., 2012.

[2] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. Springer, 2013 ed., 2013.

[3] K. Murphy, *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning series, The MIT Press, 1st ed.

[4] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. 0172-7397, New York: Springer, 2nd ed., 2009.

[5] Y. Abu-Mostafa, M. Magdon-Ismail, and H. Lin, *Learning From Data*. AMLBook, 1st ed., 2012.

[6] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors", *Nature*, no. 323, pp. 533–536, 1986.

[7] J. Abbot and J. Marohasy, "Input selection and optimization for monthly rainfall forecasting in queensland, australia, using artificial neural networks", *Atmospheric Research*, pp. 166–178, 2014.

[8] H. Abbass, "An evolutionary artificial neural networks approach for breast cancer diagnosis", *Aritificial Intelligence in Medicine*, vol. 25, pp. 265–281, 2002.

[9] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs computer: Benchmarking machine learning algorithms for traffic sign recognition", *Neural Networks*, vol. 32, pp. 323–332, August 2012.

[10] O. Mengshoel and J. Shumann, "Software health management with bayesian networks", in *International Workshop on Software Health Management*, NASA Ames Research Center, 2011.

[11] R. Cowell, R. Verrall, and Y. Yoon, "Modeling operational risk with bayessian networks", *The Journal of Risk and Insurance*, vol. 74, pp. 795–827, 2007.

[12] R. McEliece, D. MacKay, and C. Jung-Fu, "Turbo decoding as an instane of pearl's "belief propagation" algorithm", *Selected Areas in Communications, IEEE Journal*, vol. 16, pp. 140–152, February 1998.

[13] H. Lin, L. Wen-yong, M. Li, and X. Jian-min, "Public transport network optimization based on a multi-objective optimization problems evolutionary algorithm", pp. 4408–4412, 2009.

[14] P. García-Díaz, S. Salcedo-Sanz, A. Portilla-Figueras, and S. Jiménez-Fernández, "Mobile network deployment under electromagnetic pollution control criterion: An evolutionary algorithm approach", *Expert Systems with Applications*, vol. 40, no. 1, pp. 365–376, 2013.

[15] B. Saavedra-Moreno, S. Salcedo-Sanz, A. Paniagua-Tineo, J. Gascón-Moreno, and J. A. Portilla-Figueras, "Optimal evolutionary wind turbine placement in wind farms considering new models of shape, orography and wind speed simulation", *Advances in Computational Intelligence: 11th International Work-Conference on Artificial Neural Networks, IWANN*, pp. 25–32, 2011.

[16] V. Vapnik, *The Nature of Statistical Learning Theory*, vol. 2. New York: Springer-Verlag, 1995.

[17] V. Vapnik, *Statistical Learning Theory*. Wiley, October 1998.

[18] C. Bishop, *Pattern Recognition and Machine Learning*. 1613-9011, New York: Springer-Verlag, 1 ed., 2006.

[19] A. Smola and B. Schölkopf, "A tutorial on support vector regression", *Statistics and Computing*, vol. 14, pp. 199–222, 2003.

[20] Y. Abu-Mostafa, M. Magdon-Ismail, and H. Lin, *Learning From Data, e-Chapter 8: "Support Vector Machines"*. Amlbook.com, March 2012.

[21] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York: Cambridge University Press, 2004.

[22] R. Cottle, "William karush and the kkt theorem", *Documenta Mathematica*, pp. 255–269, 2012.

[23] S. Wolf, "An introduction to duality in convex optimization", *Network Architectures and Services*, 2011.

[24] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines", *Advances in kernel methods -support vector learning*, 1998.

[25] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers", *COLT'92 Proceedings of the fifth anual workshop on Computational learning theory*, pp. 144–152, 1992.

[26] C. Chang and L. Chih-Jen, "LIBSVM: A library for support vector machines", *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, May 2011.

[27] R. Fan, P. Chen, and T. Joachims, "Working set selection using second order information for training svm", *Journal of Machine Learning Research*, vol. 6, pp. 1889–1918, 2005.

[28] A. Ivakhnenko, "The group of method of data handling, a rival of the method of stochastic approximation", *Soviet Automatic Control c/c of Automatica*, vol. 1, pp. 43–55, 1968.

[29] V. Volterra, *Theory of Functionals and of Integrals and Integro-Differential Equations*. New York.: Dover Publications, 1959.

[30] N. Nikolaev and H. Iba, "Polynomial harmonic GMDH learning networks for time series modeling", *Neural Networks*, vol. 16, no. 10, pp. 1527 – 1540, 2003.

[31] H. Madala and A. Ivakhnenko, *Inductive Learning Algorithms for Complex System Modeling*. CRC Press, 1994.

[32] G. Kim, "Bordering method, encyclopedia of mathematics". http://www.encyclopediaofmath.org/index.php?title=Bordering_method&oldid=12141. Accessed: 2016-05-15.

[33] V. Shelekhova, "Harmonic algorithm GMDH for large data volume", *Systems Analysis Modelling Simulation*, vol. 20, pp. 117–1126, 1995.

[34] A. Ivakhnenko, "Heuristic self-organization in problems of engineering cybernetics", *Automatica*, vol. 6, no. 6, pp. 207–219, 1970.

[35] V. Stepashko, "GMDH algorithms as basis of modeling process automation after experimental data", *Soviet Journal of Automation and Information Sciences*, vol. 21, no. 4, pp. 43–53, 1988.

[36] V. Dimitrov, "A stocastic GMDH algorithm with successive introduction of pairs of variables", *Soviet Automatic Control c/c of Automatica*, vol. 3, no. 5, pp. 61–63, 1970.

[37] O. Shelud'ko and S. Patereu, "A probabilistic GMDH algorithm with sequential discrimination of input features", *Soviet Automatic control c/c of Automatica*, vol. 6, no. 3, pp. 29–33, 1973.

[38] R. Rifkin and R. Lippert, "Notes on regularized least squares", *Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory,Center for Biological and Computational Learning (CBCL), MIT-CSAIL-TR-2007-025 CBCL-268*, 2007.

[39] T. Nishikawa and S. Shimizu, "The characteristics of a biased estimator applied to the adaptive GMDH", *Mathematical and Computer Modelling*, vol. 17, no. 1, pp. 37 – 48, 1993.

[40] I. Rechenberg, *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, Technical University of Berlin, 1971.

[41] H. Beyer and H. Schwefel, "Evolution strategies: A comprehensive introduction", *Natural Computing*, vol. 1, pp. 3–52, 2002.

[42] L. Fogel, A. Owens, and M. Walsh, *Artificial Intelligence through Simulated Evolution*. John Wiley, 1966.

[43] J. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. 978-0262581110, A Bradford Book, reprint edition ed., 1992.

[44] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, *Handbook of Metaheuristics*, vol. 57 of *0884-8289*, ch. Hyper-Heuristics: An Emerging Direction in Modern Search Technology, pp. 457–474. Springer US, 2003.

[45] J. Potvin, *Handbook of Metaheuristics*, vol. 146 of *0884-8289*, ch. A classification of Hyper-Heuristic Approaches, pp. 449–468. Springer US, 2010.

[46] W. Crowston, F. Glover, G. Thomson, and J. Trawick, "Probabilistic learning combinations of local job shop scheduling rules", *Technical Report ONR Research Memorandum. Carnegie Mellon University*, vol. 117, pp. 46–93, 1963.

[47] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit", in *Third International Conference PATAT 2000* (W. E. E. Burke, ed.), vol. 2079 of *0302-9743*, pp. 176–190, Springer Berlin Heidelberg, 2001.

[48] G. Ochoa, J. Vazquez-Rodriguez, S. Petrovic, and E. E. Burke, "Dispatching rules for production scheduling: A hyper-heuristic landscape analysis", *Evolutionary Computation, IEEE Congress*, pp. 1873–1880, 2009.

[49] P. Garrido and C. Castro, "A flexible and adaptive hyper-heuristic approach for (dynamic) capacited vehicle routing problems", *Fundamenta Imormaticae - Emergent Computing*, vol. 119, pp. 29–60, 2012.

[50] G. Kendall and M. Mohamad, "Channel assignment in cellular communication using a great deluge hyper-heuristic", in *Proc. of the IEEE International Conference on Network*, pp. 790–795, 2004.

[51] E. Burke, M. Gendreau, M. Hyde, G. Kendall, Ochoa, G. Özcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art", *Journal of the Operational Research Society*, pp. 1695–1724, 2013.

[52] B. Patrick, *Probability and Measure*. Wiley-Interscience, 3rd ed., 1995.

[53] J. McDonald, *Handbook of Biological Statistics*. Baltimore, Maryland: Sparky House Publishing, 3rd ed., 2014.

[54] MATLAB, "paired t-test". https://es.mathworks.com/help/stats/ttest.html. Accessed: 2016-05-15.

[55] MATLAB, "One-sample kolmogorov-smirnov test". https://es.mathworks.com/help/stats/kstest.html. Accessed: 2016-05-15.

[56] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms", *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3 – 18, 2011.

[57] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance", *Journal of the American Statistical Association*, vol. 132, pp. 674–701, 1937.

[58] R. Iman and J. Davenport, "Approximations of the critical region of the friedman statistic", *Communications in Statistics*, vol. 9, pp. 571–595, 1980.

[59] W. Daniel, *Applied Nonparametric Statistics*. Duxbury Thomson Learning, 2 ed., 2000.

[60] B. Holland and M. Copenhaver, "An improved sequentially rejective bonferroni test procedure", *Biometrics*, vol. 43, no. 2, pp. 417–423, 1987.

[61] M. Akay, "Support vector machines combined with feature selection for breast cancer diagnosis", *Expert Systems with applications*, vol. 36, no. 2, pp. 3240–3247, 2009.

[62] X. Shao and V. Cherkassky, "Multi-resolution support vector machine", *Proceedings of international joint conference on neural networks*, vol. 2, pp. 1065–1070, 1999.

[63] A. Shamsheyeva and A. Sowmya, "The anisotropic gaussian kernel for SVM classification of HRCT images of the lung", *Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004. Proceedings of the 2004*, pp. 439 – 444, 2005.

[64] T. Phienthrakul and B. Kijsirikul, "Evolutionary strategies for multiscale radial basis function kernels in support vector machines.", *Proceedings of the 2005 conference on genetic and evolutionary computation, GECCO 05*, pp. 905–911, 2005.

[65] C. Rieger and B. Zwicknagl, "Deterministic error analysis of support vector regression and related regularized kernel methods", *Machine Learning Research*, 2009.

[66] D. Zheng, J. Wang, and Y. Zhao, "Non-flat function estimation with a multi-scale support vector regression", *Neurocomputing*, vol. 70, pp. 420–429, 2006.

[67] Z. Yang, J. Guo, W. Xu, X. Nie, J. Wnag, and J. Lei, "Multi-scale support vector machine for regression estimation", *Lecture Notes Computer Science*, vol. 3971, pp. 1030–1037, 2006.

[68] Y. Zhao and J. Sun, "Multikernel semiparametric linear programming support vector regression", *Expert Systems with Applications*, vol. 38, no. 3, pp. 1611 – 1618, 2011.

[69] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee, "Choosing multiple parameters for support vector machines", *Machine Learning*, vol. 46, no. 1-3, pp. 131–159, 2002.

[70] A. A. Villa, M. Fauvel, J. Chanussot, P. Gamba, and J. Benediktsson, "Gradient optimization for multiple kernels parameters in support vector machines classification", *Geoscience and Remote Sensing Symposium (IGARSS), IEEE International*, pp. 224–227.

[71] A. Gijsberts, G. Metta, and L. Rothkrantz, "Evolutionary optimization of least-squares support vector machines", *Special Issue in Annals of Information systems*, vol. 8, pp. 227–297, 2010.

[72] P. Pai, W. Hong, and Y. Lee, "Determining parameters of support vector machines by genetic algorithms—applications to reliability prediction", *Int J Oper Res*, vol. 2, pp. 1–7, 2004.

[73] D. Brodic, "Optimization of the anisotropic gaussian kernel for text segmentation and parameter extraction", *6th IFIP TC 1/WG 2.2 International Conference, TCS 2010, Held as Part of WCC 2010, Brisbane, Australia, September 20-23, 2010. Proceedings*, pp. 140–152, 2010.

[74] S. Abe, "Training of support vector machines with mahalanobis kernels", in *Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005* (W. Duch, J. Kacprzyk, E. Oja, and S. Zadrożny, eds.), vol. 3697 of *Lecture Notes in Computer Science*, pp. 571–576, Springer Berlin Heidelberg, 2005.

[75] M. Fauvel, A. Villa, J. Chanussot, and J. Benediktsson, "Mahalanobis kernel for the classification of hyperspectral images", *Geoscience and Remote Sensing Symposium (IGARSS), IEEE International*, pp. 3724–3727, July 2010.

[76] F. Friedrichs and C. Igel, "Evolutionary tuning of multiple SVM parameters", *In Proc. of the 12th European Symposium on Artificial Neural Networks (ESANN)*, pp. 519–524, 2004.

[77] S. Salcedo-Sanz, E. Ortiz-García, A. Pérez-Bellido, A. Portilla-Figueras, and L. Prieto, "Short term wind speed prediction based on evolutionary support vector regression algorithms", *Expert Systems with Applications*, vol. 38, no. 4, pp. 4052 – 4057, 2011.

[78] E. Ortiz-García, S. Salcedo-Sanz, A. Pérez-Bellido, and A. Portilla-Figueras, "Improving the training time of support vector regression algorithms through novel hyper-parameters search space reductions", *Neurocomputing*, vol. 72, pp. 3683 – 3691, 2009.

[79] K. Smets, B. Verdonk, and E. Jordan, "Evolution of performance measures for SVR hyperparameter selection", *Neural Networks. IJCNN 2007. International Joint Conference on,*, pp. 637–642, 2007.

[80] C. Wu, G. Tzeng, and R. Lin, "A novel hybrid genetic algorithm for kernel function and parameter optimization in support vector regression", *Expert Systems with Applications*, vol. 36, no. 3, Part 1, pp. 4725–4735, 2009.

[81] S. Hou and Y. Li, "Short-term fault prediction based on support vector machines with parameter optimization by evolution strategy", *Expert Systems with Applications*, vol. 36, no. 10, pp. 12383–12391, 2009.

[82] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*. 1619-7127, Springer-Verlag, 1 ed., 2003.

[83] A. Asuncion and D. Newman, "UCI machine learning repository". http://www.ics.uci.edu/mlearn/MLRepository.html, 2007. Accessed: 2016-05-15.

[84] "Statlib datasets archives". ftp://rcom.univie.ac.at/mirrors/lib.stat.cmu.edu/datasets/.index.html, 2005. Accessed: 2016-05-15.

[85] K. Bramstedt, J. Gleason, D. Loyola, W. Thomas, A. Bracher, M. Weber, and J. Burrows, "Comparison of total ozone from the satellite instruments gome and TOMS with measurements from the dobson network 1996-2000", *Atmospheric Chemistry and Physics*, pp. 1409–1419, 2003.

[86] A. Silva, "A quarter century of TOMS total column ozone measurements over Brazil, journal = Physics, volume = 69, number = 12, pages = 1447-1458, year = 2007",

[87] R. McPeters, P. Bhartia, A. Krueger, and J. Herman, "Nimbus 7 total ozone mapping spectrometer (toms) data products user's guide", *NASA Reference Publication*, 1996.

[88] P. Bhartia, C. Wellemeyer, S. Taylor, N. Nath, and A. Gopalan, "The version 8 SBUV ozone profile algorithm", *In Proc. of the Quadrennial Ozone Symposium, Kos, Greece*, 2004.

[89] R. Stolarski, A. Douglass, S. Steenrod, and S. Pawson, "Trends in stratospheric ozone: Lessons learned from a 3d chemical transport model", *Journal of Atmospheric Science*, vol. 63, no. 3, pp. 1028–1041, 2006.

[90] K. Wolter and M. Timlin, "Trends in stratospheric ozone: Lessons learned from a 3d chemical transport model", *Journal of Atmospheric Science*, vol. 63, pp. 315–324, 1998.

[91] S. Oh, W. Pedrycz, and H. Park, "The design of self-organizing polynomial neural networks", *Information Sciences*, vol. 141, pp. 237–258, 2002.

[92] C. Kondo and T. Kondo, "Revised GMDH-type neural network algorithm self-selecting optimum neural network architecture", *Artificial Life and Robotics*, vol. 14, no. 4, pp. 519–523, 2009.

[93] S. Oh, W. Pedrycz, and H. Park, "Genetically optimized fuzzy polynomial neural networks", *IEEE Transactions on Fuzzy Systems*, vol. 14, no. 1, pp. 125–144, 2006.

[94] S. Roh, W. Pedrycz, and S. Oh, "Genetic optimization of fuzzy polynomial neural networks", *IEEE Transactions on Industrial Electronics*, vol. 54, p. 2219, 2238 2007.

[95] T. Ohtani, H. Ichihashi, T. Miyoshi, and K. Nagasaka, "Orthogonal and successive projection methods for the learning of neurofuzzy GMDH", *Information Sciences*, vol. 110, no. 1-2, pp. 5–24, 1998.

[96] B. Zhu, C. He, P. Liatsis, and X. Li, "A GMDH-based fuzzy modeling approach for constructing ts model", *Fuzzy Sets and Systems*, vol. 189, pp. 19–29, 2012.

[97] H. Park, B. Park, H. Kim, and S. Oh, "Self-organizing polynomial neural networks based on genetically optimized multi-layer perceptron archictecture", *International Journal of Control, Automation and Systems*, vol. 2, no. 4, pp. 423–434, 2004.

[98] F. Kalantary, H. Ardalan, and N. Nariman-Zadeh, "An investigation on the Su-NSPT correlation using GMDH type neural networks and genetic algorithms", *Engineering Geology*, vol. 104, no. 1-2, pp. 144–155, 2009.

[99] N. Nariman-Zadeh, A. Darvizeh, and G. Ahman-Zadeh, "Hybrid genetic design of GMDH-type neural networks using singular value decomposition for modeling and predicting of the explosive cutting process", *Proceedings of the Institution of Mechanical Engineers*, vol. 217, pp. 779–790, 2003.

[100] D. Kim and G. Park, "GMDH-type neural network modeling in evolutionary optimization", *Lecture Notes Computer Science*, vol. 3533, pp. 563–570, 2005.

[101] R. Madandoust, R. Ghavidel, and N. Nariman-zadeh, "Evolutionary design of generalized GMDH-type neural network for prediction of concrete compressive strength using UPV", *Computational Materials Science*, vol. 49, no. 3, pp. 556–567, 2010.

[102] N. Nikolaev and H. Iba, "Accelerated genetic programming of polynomials", *Genetic Programming and Evolvable Machines*, vol. 2, no. 3, pp. 231–257, 2001.

[103] M. Hiassat, M. Abbod, and N. Mort, *Using genetic programming to improve the GMDH in time series prediction*. Chapman and Hall, CRC press, 2003.

[104] M. Abbod and K. Deshpande, "Using intelligent optimization methods to the group method of data handling in time series prediction", *Lecture Notes Computer Science*, vol. 5103, pp. 16–25, 2008.

[105] G. Onwubolu, "Design of hybrid differential evolution and group method of data handling networks for modeling and prediction", *Information Sciences*, vol. 178, pp. 3616–3634, 2008.

[106] T. Kondo, "The learning algorithm of the GMDH neural network and its application to medical image recognition", pp. 1109–114, 1998.

[107] H. Tamura and T. Kondo, "Heuristic free group method of data handling algorithm for generating optimal partial polynomials with application to air pollution prediction", *International journal of Systems Science*, vol. 11, no. 9, pp. 1095–1111, 1980.

[108] E. Ozcan, B. Bilgin, and E. Korkmaz, "A comprehensive analysis of hyper- heuristics", *Intelligent Data Analysis*, vol. 12, no. 1, pp. 3–23, 2008.

[109] L. Han, P. Cowling, and G. Kendall, "An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem", *Proceedings of Congress on Evolutionary Computation (CEC2002)*, pp. 1185–1190, 2002.

[110] MATLAB, "Neural network toolbox matlab". https://es.mathworks.com/help/nnet/index.html. Accessed: 2016-05-15.

[111] G. Huang, Q. Zhu, and C. Siew, "Extreme learning machine: Theory and applications", *Neurocomputing*, vol. 70, pp. 489–501, 2006.

[112] J. Torres, A. García, M. De Blas, and A. De Francisco, "Forecast of hourly average wind speed with ARMA models in Navarre (Spain)", *Solar energy*, vol. 79, pp. 65–77, 2005.

[113] M. Khashei, M. Bijariand, and G. Ardali, "Improvement of auto-regressive integrated moving average models using fuzzy logic and artificial neural networks (ANNs)", *Neurocomputing*, vol. 72, no. 4-6, pp. 956–967, 2009.

[114] ECMWF, "Accessing ecmwf data servers in batch". https://software.ecmwf.int/wiki/display/WEBAPI/Accessing+ECMWF+data+servers+in+batch. Accessed: 2016-05-15.

[115] ECMWF, "Grib-api". https://software.ecmwf.int/wiki/display/GRIB. Accessed: 2016-05-15.

[116] NOAA, "Earth system research laboratory, PSD climate and weather data". Accessed: 2016-05-15.

[117] L. Carro-Calvo, S. Salcedo-Sanz, N. Kirchner-Bossi, A. Portilla-Figures, L. Prieto, R. Garcia-Herrera, and E. Hernandez-Martín, "Extraction of synoptic pressure patterns for long-term wind speed estimation in wind farms using evolutionary computing", *Energy*, vol. 36, pp. 1571–1581, 2011.

[118] B. Saavedra-Moreno, S. Salcedo-Sanz, L. Carro-Calvo, J. Gascón-Moreno, S. Jiménez-Fernández, and L. Prieto, "Very fast training neural-computation techniques for real measure-correlate-predict wind operations in wind farms", *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 116, pp. 49 – 60, 2013.

[119] E. Sreevalsan, S. Das, R. Sasikumar, and M. Ramesh, "Wind farm site assessment using Measure-Correlate-Predict (MCP) analysis", *Wind Engineering*, vol. 31, no. 2, pp. 111–116, 2007.

[120] A. Rogers, J. Rogers, and J. Manwell, "Comparison of the performance of four measure-correlate-predict algorithms", *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 93, no. 3, pp. 243–264, 2005.

[121] S. Kalogirou, "Artificial neural networks in renewable energy systems applications: a review", *Renewable and Sustainable Energy Reviews*, vol. 5, no. 4, pp. 372–401, 2001.

[122] M. Mohandes, T. Halawani, S. Rehman, and A. Hussain, "Support vector machines for wind speed prediction", *Renewable Energy*, vol. 29, no. 6, pp. 939–947, 2004.

[123] J. Carta, S. Velázquez, and J. Matías, "Use of bayesian networks classifiers for long-term mean wind turbine energy output estimation at a potential wind energy conversion site", *Energy Conversion and Management*, vol. 52, no. 2, pp. 1137–1149, 2011.

[124] K. Moustris, I. Ziomas, and A. Paliatsos, "3-days-ahead forecasting of regional pollution index for the pollutants NO_2, CO, SO_2 and O_3 using artificial neural networks in athens", *Soil Pollution Journal*, vol. 209, pp. 29–43, 2010.

[125] C. Wilmott, S. Ackleson, and R. Davis, "Statistics for the evaluation and comparison of models", *Journal of Geophysics Research*, vol. 90, pp. 8995–9005, 1985.

[126] A. Dutot, J. Rynkiewicz, F. Steiner, and J. Rude, "A 24-h forecast of ozone peaks and exceedance levels using neural classifiers and weather predictions", *Environmental Modelling and Software*, vol. 22, pp. 1261–1269, 2007.