Deusto

Facultad de Ingeniería
Universidad de Deusto

Ingeniaritza Fakultatea
Deustuko Unibertsitatea

Grado en Ingeniería Informática
Informatikako Ingeniaritzako Gradua

Proyecto fin de grado
Gradu amaierako proiektua

Summary

In the last 50 years artificial intelligence algorithms have demonstrated their usefulness in the modelling of engineering processes. However, these algorithms usually pose several problems that make their use difficult for non-expert users. One of them is the hyperparameters tuning (parameters that can be freely modified by users). At present, the hyperparameters tuning is either done by experts that base their decisions in their previous experiences, or they are set randomly, but does not exist well-defined methodology to determine them. Moreover, it is uncommon to find a deep statistical analysis about the improvement achieved by using meta-heuristics for hyperparameter optimization, so these hyperparameters are adjusted in non-efficient ways that make the learning process of the algorithm much harder and a less productive task.

In this project I propose the development of evolutionary algorithms as tool for the optimization of artificial intelligence models. I put special focus on bioethanol to olefins (BTO) chemical models. Moreover, I performed a statistical analysis in order to compare the performance of different evolutionary algorithms against hyperparameter random initialization. Regarding to the obtained results, I conclude that a significant improvement is obtained by applying evolutionary algorithms to the optimization of machine learning algorithms and that there are no significant differences between the crossovers. I also infer that a significant improvement is achieved in the modelling of the BTO process.

Keywords

evolutionary, meta-heuristic, hyperparameter, optimization, BTO

Acknowledgements

I am sad to write this part of the dissertation. It means many things: the bachelor is finished, many goodbyes, no more university environment, no more riding my bicycle in the way to university … and no more exams!

Just four years but many people to thank.

The University of Deusto takes an important role in the achievement of this project. It is an excellent place to learn new things not only in the technical aspect but also related with personal skills which are the really important ones. Although some things of the university can be improved (put back the bicycle parking at the entrance of the faculty! 😊) excellent people and great teaching can be found there. I would not like to omit Pázmány Péter Catholic University where I stayed for five months during my Erasmus programme. There, in Budapest, I learn many things that I will never forget.

I would also like to express gratitude to the DeustoTech Institute of Technology and with more emphasis to the Energy department, a really nice, young, promising and hard-working team that without a doubt will achieve great success in their projects. I would like to give special thanks to Yoseba Peña Landaburu, for being my tutor, to Gorka Sorrosal Yarritu, for his enormous support in machine learning and the BTO process, and to Cruz Enrique Borges Hernández who always answered politely the plenty questions that I asked (I guess that I did literally thousands of them 😁) during the development of this project. Moreover, not only Cruz responded my questions but also enlightened my when I thought that there were no possible solutions. I hope you keep on introducing students to the fascinating world of research, you do it fantastically.

I would also like to express my gratitude to the Basque Government for the support of the project via a researcher training grant (COL-2013-1-135).

Many thanks also to classmates with who I expended many hours of hard work and at the same time of happiness.

Last but not least, I would like to hug and to give $\sum_{n=1}^{\infty} thanks$ to my mother and father, Isabel and Arturo, and also to my lovely brother Aitor. Nothing of this could be possible without your unconditional support, in the good and no so good moments.

Contents

Acknowledgements **v**

1 INTRODUCTION **1**

2 BACKGROUND AND RATIONALE **3**

2.1 THE LEARNING PROBLEM 3

2.2 MACHINE LEARNING ALGORITHMS 3

2.2.1 The regression problem according to statistical learning 4

2.3 THE APPLICATION OF MACHINE LEARNING ALGORITHMS TO ENGINEERING PROBLEMS 6

2.3.1 Bioethanol to Olefins process 6

2.4 METAHEURISTICS AND HYPERPARAMETER TUNING ALGORITHMS 7

2.5 EVOLUTIONARY ALGORITHMS USED FOR HYPERPARAMETER TUNING PROBLEMS 7

3 OBJECTIVES AND SCOPE **9**

3.1 PROJECT OBJECTIVES 9

3.2 REQUIREMENTS 9

3.3 PROJECT SCOPE 9

4 METHODOLOGY **11**

4.1 PROPOSED SOLUTION TO THE LEARNING PROBLEM 11

4.1.1 Evolutionary algorithms 11

4.1.2 Artificial intelligence algorithms 15

4.1.3 Adapting evolutionary algorithms to hyperparameter tuning problem 22

4.2 DATA GATHERING 23

4.3 METHODS TO VALIDATE THE MODELS 24

4.3.1 Training and validation error 24

4.3.2 Cross-validation 25

4.3.3 Regularization 25

4.4 ERROR MEASURES FOR REGRESSION PROBLEMS 28

4.5 FRIEDMAN TEST 29

4.5.1 Post-hoc analysis 30

5 PLANNING **31**

5.1 MAIN TASKS 31
5.2 HUMAN RESOURCES PLANNING 32
5.3 WORK SCHEDULE 34
5.4 WORKING PLAN 34
5.5 WORK ENVIRONMENT 34

6 BUDGET **37**

7 DEVELOPMENT **39**

7.1 EXPERIMENTAL SET-UP 39
7.2 EXPERIMENTAL RESULTS 40
7.3 INTERPRETATIONS FROM THE EXPERIMENTS 41

8 CONCLUSIONS AND FUTURE WORK **45**

8.1 FUTURE WORK 45

9 DEFINITIONS AND ACRONYMS **47**

9.1 DEFINITIONS 47
9.2 ACRONYMS 47

Bibliography **49**

A LEAST SQUARES METHOD **55**

A.1 EXAMPLE USING LINEAR REGRESSION 55

B GRADIENT DESCENT **57**

C PUBLICATIONS **59**

D WORKING PLAN **67**

E GANTT DIAGRAM **69**

F NETWORK DIAGRAM **71**

G RESULTS DATA **73**

H SOFTWARE DOCUMENTATION **79**

H.1 CLASS DIAGRAMS 79

List of Figures

Chapter 2

2.1 Typical case of linear regression. Taken from Wikipedia 5
2.2 Schema of the project's main idea 8

Chapter 4

4.1 Function graph indicating its global maximum, minimum and local maxima and minima 14
4.2 A hard to optimize mathematical function 15
4.3 The basic structure of a perceptron 17
4.4 In the left side a recurrent multilayer neural network can be seen. In the right side a feedforward neural network 17
4.5 Two types of artificial neural network 18
4.6 Boundaries and support vectors of SVM. Images taken from Wikipedia 21
4.7 SVM used for regression problems. Modified from [63] 22
4.8 Example of C parameter avoiding overfitting. Taken from [64] 23
4.9 Over-fitting situation: After the 5th epoch the algorithm starts to over-fit 26
4.10 The VC dimension of a perceptron in a two dimensional space is three. Taken from [27] 27
4.11 Graphical representation of test error upper bound. Taken from [27] 28

Chapter A

A.1 Function graph indicating its global maximum, minimum and local maxima and minima 55

Chapter B

B.1 Path taken by gradient descent in a convex function. Taken from [40] 57

Chapter C

C.1 Grid Search Example. 60

Chapter E

E.1 Planning Gantt diagram 69

Chapter F

F.1 Planning network diagram 71

Chapter H

H.1 Individual class diagram 79
H.2 Genesis class diagram 80
H.3 Fitness class diagram 80
H.4 Evolutionary algorithm class diagram 81
H.5 Stop condition operator class diagram 82
H.6 Selection operator class diagram 82
H.7 Genetic operator class diagram 83
H.8 Reproduction operator class diagram 84
H.9 Elite operator class diagram 85

List of Tables

Chapter 5

5.1 Reduced working plan 35

Chapter 6

6.1 Human resources budget 37
6.2 Software resources budget 37

Chapter 7

7.1 MAPE results of KEEL Datasets with the different crossover algorithms. 41
7.2 Friedman test applied to KEEL results 42
7.3 Post-hoc analysis of the KEEL results 42
7.4 Friedman test applied to synthetic data results 43
7.5 Post-hoc analysis of the synthetic results 43
7.6 MAPE results of BTO process using ANN 43
7.7 MAPE results of BTO process using SVM 43

Chapter C

C.1 KEEL Dataset MAPE Results with the Different Crossover Algorithms. 63
C.2 Synthetic Dataset Results with the Different Crossover Algorithms. 63
C.3 BTO Process Modelling Dataset Results. 63
C.4 Post Hoc Analysis Done for the Different Crossover Algorithms of KEEL datasets . 64
C.5 Post Hoc Analysis Done for the Different Crossover Algorithms of syntethic datasets 64

Chapter D

D.1 Complete working plan 67

Chapter G

G.1 MAPE results of KEEL Datasets with all crossover algorithms 73
G.2 MAPE results of synthetic datasets with the all crossover algorithms. Part 1 74
G.3 MAPE results of synthetic datasets with the all crossover algorithms. Part 2 75
G.4 MAPE results of synthetic datasets with the all crossover algorithms. Part 3 76
G.5 MAPE results of synthetic datasets with the all crossover algorithms. Part 4 77

1. INTRODUCTION

In this document I describe the development of meta-heuristics for the optimization of artificial intelligence models with special emphasis in the application to industrial processes. During the thesis I follow accurately the main steps and principles of the scientific method in order to accomplish a well-defined research project.

Below I describe the parts that compose the whole thesis:

Background and rationale: In this chapter I describe the learning problem and the complexity of learning automation. As solution to the previous problem, I detail several machine learning algorithms and I discuss their application to engineering problems. Moreover, I explain the machine learning algorithms hyperparameter tuning problem and I propose a solution based on evolutionary algorithms meta-heuristics.

Objectives and scope: I defined main and secondary project objectives. Also, I specify functional and non-functional requirements. Finally, I enclose the scope of the project.

Methodology: This chapter is composed of five key sections. In the first section I deeply describe evolutionary algorithms as the solution for the hyperparameter optimization problem. Then, I describe and analyse artificial neural networks and support vector machines machine learning algorithms, putting emphasis on their hyperparameters. Then, I explain how evolutionary algorithms were adapted to fit the needs of learning algorithms. In the second part, I describe datasets used in the experiments. In the next section I explain the over-fitting problem and I analyse different approaches to deal with this problem. In the following section I consider the benefits and drawbacks of different prediction error measures. Finally, in the fifth section I specify Friedman statistical test and its use in the current project.

Planning: I break down the planning of the project by detailing all the activities and tasks that the project is made up of. I also describe the human resource plan, the working schedule and environment. Then, I also show the working plan detailing starting and finishing dates and the duration and work of each task.

Budget: I structure the total budget into human and software budgets displaying all the cost of the project.

Development: I divided this chapter in three sections. In the first one I describe the experiments set-up in high detail in order to allow to be reproduced by others. In the second part I present the obtained results in a structured way and I perform a statistical analysis. In the last part, I interpret the results and I make the conclusions.

Conclusions and future work: I summarise the main conclusions and I estimate their impact in the real world. In addition, I define the future work.

Appendices: I describe mathematically least squares (Appendix A) and gradient descent (Appendix B) convex optimization functions in order to go in deep about the main advantages and disadvantages of both of them and to explain in which situations are used. I also include the article that I wrote together with several researchers of DeustoTech Energy and was accepted

for publication in *The 26th European Modeling and Simulation Symposium* (Appendix C). Then, I include the whole working plan task identification and predecessors (Appendix D). I also give as appendices Gannt (Appendix E) and network diagrams (Appendix F). I also show, for space reasons, complete KEEL regression and synthetic datasets prediction error results (Appendix G). Finally, I give the evolutionary algorithm software documentation (Appendix H).

2. BACKGROUND AND RATIONALE

2.1. THE LEARNING PROBLEM

Several authors have defined the learning concept, McCarthy [43] defined learning as constructing or modifying representations of what is being experienced and Mitchell [46] explains that learning is improving automatically with experience. Taking into account the previous learning descriptions it can be stated that learning implies building a kind of representation or description and based on experiences gradually modify that representation so the performance can be improved in the future.

For humans learning is something natural; knowledge is acquired by practice. However transfer this ability to computers is not an easy task. Since the beginning of the 20th century, scientists have been trying to automatize the process of learning using machines. This discipline is known as machine learning. They fast realized that learning automatizing is a really complex task because it is not well understood how humans learn, making difficult to apply those steps in a computer, this is known as the *learning problem*.

Actual machine learning algorithms try to simulate a learning task by learning automatically from data or past experiences. It can be seen that they highly rely on the definition of learning that was made before because they build a representation and update it while they receive new data. However, the useful point about the automatized learning is not only the learning itself but also the predictions and decisions that they can make about future events. Machine learning algorithms, also known as learning algorithms, have been used for example in stock market [65], earthquake [53] and stroke prediction [35].

2.2. MACHINE LEARNING ALGORITHMS

Machine learning is a field of artificial intelligence that studies and builds systems that can learn from data. Learning from data can be done using two different approaches: by supervised or unsupervised training. In supervised training a set of training instances with their expected output is given to the algorithm. In mathematical terms a training instance can be described as a mapping of input variables X to output variable Y. So we want to infer a function $f : X \rightarrow Y$ from a dataset composed of pairs (x_i, y_i) where $x_i \in X$ and $y_i \in Y$. Usually $X \in \mathbb{R}^n$ and $Y \in \mathbb{R}$ (known as regression problems) or $Y \in \{class\}$ (classification problems). For example, a dataset of colon cancer tissues can be prepared where each instance has gene expression levels as X attributes and the Y value indicates if the example was taken from a tumour biopsy or not. However, in unsupervised training only input variables are given, without their expected output. In this case the model has to determine patterns among the data. A common example is to have a given disease dataset and to make subcategories of that disease or cluster them in order to have a more detailed knowledge of its variants.

2.2.1 The regression problem according to statistical learning

A regression problem is defined as a dataset composed by pairs (x_i, y_i) being $x_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$ that represents $Y = f(X, \beta) + \epsilon$ where $f(\cdot, \cdot)$ is an unknown function, $\beta \in \mathbb{R}^k$ are the parameters and ϵ is noise with usually an unknown distribution. The objective is to find a function $h(X, w)$ where $w \in \mathbb{R}^l$ is the parameter vector for $h(\cdot, \cdot)$ function such that $\int_X (f(X, \beta) - h(X, w))^2 = 0$. However, as it was said the $f(X, \beta)$ function is unknown so a problem arises about how to determine the prediction error.

A possible solution could be to find the optimal w parameters, denoted as w^*, for $h(X, w)$ so that the error when compared to the target values is minimized, in mathematical terms $w^* = \underset{w}{argmin}\ (h(X, w) - y)^2$. Nevertheless, this is not a good idea because $h(X, w^*)$ would not learn the objective function $f(X, \beta)$ but $f(X, \beta) + \epsilon$ so the noise would also be learn. This problem is known as over-fitting and I will explained it in more detail in Subsection 4.3.1. To deal with this problem I explain several techniques in Subsections 4.3.2 and 4.3.3.

Statistics approach to regression problem

As it was stated before regression tries to find a relationship between the inputs and real-valued outputs. For example, one may attempt to predict the price of a stock using regression, the predicted value would be a real value and not a discrete class.

The French mathematician Adrien-Marie Legendre and the German Johann C. Friederich Gauss were the first ones to use statistics for regression in 1805 and 1809 respectively. After them, Francis Galton was one of the first to apply regression to describe several biological phenomenon [21]. Nowadays, applications of regression analysis exist in almost every field. In economics, biology, engineering and sociology is widely used to establish relationships between different variables [34], [39], [9], [11].

Linear regression is one of the most popular types of regression where $f(\cdot)$ is a linear function. This model can be represented as $y = \beta X + \epsilon$ which is a linear combination of the $\beta \in \mathbb{R}^m$ and $X \in \mathbb{R}^{m \times m}$ values so it takes the form of:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \epsilon_i$$

where $i = 1, \cdots, n$ and x_{ij} represents the j^{th} attribute of the i^{th} input instance. The ϵ_i value is an unobserved random variable that adds noise to the linear combination. In Figure 2.1 a simple linear regression with one independent variable can be seen.

Linear regression makes several assumptions about the data. The first ones is related with linearity, this model assumes that the response variable is a linear combination of the vector parameters β and the predictor variables. The model also takes the assumption that all the response variables have the same variance in their errors which can be invalid in the case of a wide range of response variable.

Another important topic of linear regression is how to decide the slope of the hyperplane and its y-intercept. A commonly used method is the least square method which is also a convex function optimization and the squares of the vertical distance from each data point to the hyperplane is

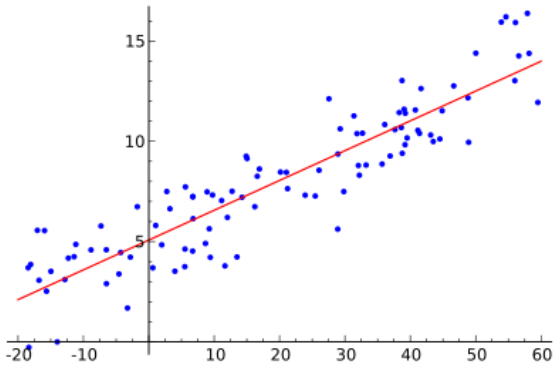


Figure 2.1: Typical case of linear regression. Taken from Wikipedia

summed. By using partial derivatives the hyperplane with minimum least square error is chosen as the function that better models the data, see Appendix A. Should be noted that in the case of backpropagation this method should not be used because the system that is obtained in that case is composed by non-linear equations, fact that makes the system quite difficult to solve analytically [68].

A more advanced type of regression is the nonlinear one. In $y_i = f(\beta, X) + \epsilon$ the $f(\cdot)$ function is nonlinear. Example of nonlinear function include exponential, trigonometric, logarithmic and power functions. It should be noted that the nonlinearity relates to the unknown parameters. Some of these functions, such as exponential or logarithmic functions, can be linearised so linear regression can be applied. For example, the exponential function $y = ae^{bx}$ can be linearised using natural logarithms by $ln(y) = ln(a) + bx$. One example of pure nonlinear regression are the rational functions which are very flexible so they can approximate a wide range of functional shapes:

$$f(\beta, x) = \frac{\sum_{j=1}^{k} \beta_j x^{j-1}}{1 + \sum_{j=1}^{m} \beta_{k+j} x^j}.$$

Machine learning approach to regression problem

Within machine learning several paradigms have been built to solve regression problems. One of them are artificial neural networks (ANN), these algorithms have been prove to be useful for pattern classification and recognition and have been widely used in business [38] and industry [69].

One of the main applications of ANN is forecasting. There are several reasons, the first one is that they are a data-driven technique that try to find relationships between the input data and the outputs. The good point about this is that usually it is easy to have data about the process to model but not to express it mathematically, so ANNs are suitable for this type of situations. However, care should be taken when making these relationships because, as it was seen, data has noise. Another key point about using ANN is that does not matter how complex the process to be modelled is and, even better, a deep theoretical knowledge of the process is not needed. One more advantage is that ANNs make few assumptions about the problem to model so they can be applied in multiple fields. In theory, taking into account the universal approximation theory

[14], [29], ANNs can approximate any continuous function to the desired accuracy, fact that makes them a really powerful tool. Moreover, they can model real-world problems better than many of the statistical models because the last ones make several assumptions that does not hold usually in practice.

However, traditional ANNs have problems related to generalization because their base their performance in the error obtained in the seen data. For that reason, support vector machines (SVM) are becoming popular as they base their decision in Structural Risk Minimization measure explained in Subsection 4.3.3 and because of its well-founded theoretical approach that forms the basis of SVM. This feature it what makes SVM a tool with a big capability to generalize. Although SVM uses hyperplanes to make decisions (see Section 4.1.2 for further information) it can also be extended to create boundary non-linear functions through the application of kernel functions.

SVM has recently been successfully applied to a wide range of applications, such as character recognition [1], face recognition [24], text categorization [33], image classification [66], spam filtering [19], signal processing [55], financial forecasting [36], bioinformatics [70] and so on. In many cases, SVM performed better than the state-of-the-art algorithms.

2.3. THE APPLICATION OF MACHINE LEARNING ALGORITHMS TO ENGINEERING PROBLEMS

Some of the many concerns of engineering are to optimize, automatize and predict the behaviour of a system. In the past these complex tasks were performed by workers using really tedious and error prone techniques. As a consequence, these processes were very expensive and often did not give optimal results. However, at the end of the eighties machine learning algorithms started to be applied in engineering making the processes and products cheaper and better.

Nowadays, machine learning algorithms are applied in multiple fields related with the industry, for example, pattern recognition [10], computer vision [57], medical diagnosis [37], speech recognition [16] or cheminformatics [32]. One usual application is to model a chemical reaction in order predict its reaction properties and to test it with new components without even needing to do new experiments. In this project I will focus on the bioethanol to olefins chemical process.

2.3.1 Bioethanol to Olefins process

It is widely known that crude oil is a limited resource so an alternative energy source should be considered. Several new processes have been researched such as biomass or biofuels, one of them is the bioethanol to olefins (BTO) process which is a catalytic transformation of bioethanol to olefins that has the biomass or other derivatives as reactants. Some of the obtained olefins are C_2-C_4 which are important products in petrochemical synthesis. The production of olefins over acid catalyst is also an important topic in the industry due to its suitability for maximizing propene.

Nevertheless, experimental trials of chemical processes have a high monetary cost becoming difficult to develop new improvements. As a solution, mathematical models of the processes were done using differential equations. Developed models could be used for optimization of the catalysis, research the effects of changing several variables of the process or to predict the behaviour of

new chemical processes without even making new tests in the laboratory. However, mathematical models of the BTO are quite complex to develop and solve [23].

Due to the previous problem different types of modelling this process have been tried. One of them are artificial neural networks and support vector machines which have had good results in the modelling of non-linear systems [48], [50]. They also have been tested in the modelling of different chemical processes [8]. In [59] a research about the application of artificial neural network to the modelling of a chemical reactor of the BTO process was done.

2.4. METAHEURISTICS AND HYPERPARAMETER TUNING ALGORITHMS

The two machine learning algorithms explained in Section 2.2 use a set of parameters that can be tuned by the user in order to make the algorithm perform better. In addition, in other areas of knowledge such as image processing, it was proved that state of the art of image classification could be improved not only by building new algorithms but also by tuning the hyperparameters of existing techniques [52], [7]. These parameters are known as hyperparameters. In the past, hyperparameters tuning was done by hand because of the low computational power available [4]. Presently, computer clusters and GPU processor allow to run more trials and to search hyperparameter space faster.

To solve this problem computationally, metaheuristics can be used. These techniques are a type of non-problem specific heuristics that guide the search process. The main goal of metaheuristics is to explore the search space in order to find near-optimal solutions. The most simple but not efficient technique is the direct search throughout the operating range. Because it is impossible to test in every possible conditions, normally it is used random samples or grid search strategies. In the first case, the operation points to test and analyse are randomly selected in the operation range, while in the second option, a grid of operating points covering the whole operation range is defined and iteratively refined over the hyperparameter space.

2.5. EVOLUTIONARY ALGORITHMS USED FOR HYPERPARAMETER TUNING PROBLEMS

Hyperparameter tuning problem can be approached as an optimization problem where the target is to minimize the objective function that, in this case, is the prediction error of the machine learning algorithm where the optimum hyperparameters should be chosen from the set $\mathbb{R}^n$.

Several techniques were addressed in the previous section but the evolutionary algorithms are a more efficient and intelligent tuning strategy. Based on biological evolution, these soft computing techniques allow a selective exploration of the operation range [28]. The main idea is shown in Figure 2.2 where it can be seen that evolutionary algorithms are used for tuning the hyperparameters of already developed machine learning algorithms that model a real-world process. Evolutionary algorithms have been used in different fields such as physics [12], politics [13] or economics [67]. In several hyperparameter tuning problems, evolutionary algorithms have been proved to perform far

better than grid search according to the accuracy-speed ratio [18], [20], [30]. The facts presented above open the possibility of improving actual machine learning techniques by the tuning of their hyperparameters.

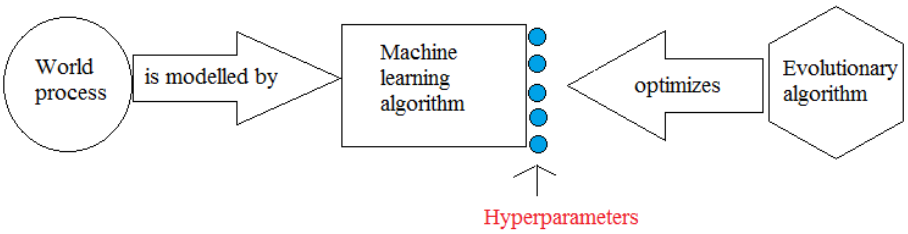


Figure 2.2: Schema of the project's main idea

3. OBJECTIVES AND SCOPE

3.1. PROJECT OBJECTIVES

Primary objectives:

- Develop evolutionary algorithms for the optimization of artificial intelligence models.
- Demonstrate statistically that hyperparameter tuning using evolutionary algorithms improves the performance of artificial intelligence models.

Secondary objective: To improve models of catalytic reactions for the generation of biofuel.

3.2. REQUIREMENTS

Below, I describe the functional and non-functional requirements of the project:

Functional requirements:

- The platform of evolutionary algorithms must facilitate the configuration of support vector machines and artificial neural networks.
- The platform must improve the BTO process modelling.
- The platform must implement uniform and non-uniform mutations.
- The platform must implement the following crossovers: arithmetic, uniform, differential, swarm, one/two point row and one/two point column.
- Users must be able to parametrize the population size and number of generations.
- Users must be able to decide to use paralleling processing or not.
- Users must be able to adjust the number of cores that work in parallel.

Non-functional requirements:

- MATLAB must be used as programming language.
- Hardware provided by DeustoTech Energy will only be available.
- The platform must operate in Windows and Linux operative systems.
- The software must use all the core processor that are available through paralleling.

3.3. PROJECT SCOPE

The project scope is defined as:

- Development of an evolutionary algorithms platform that is able to:
  - Maximize and minimize mathematical functions.
  - Tune the hyperparameters of ANNs and SVMs learning algorithms.
  - Assess which evolutionary algorithm is significantly better for a set of data.
  - Deal with any type of numerical structure such as vectors or matrices.
- The evolutionary algorithms platform will not be able to:
  - Find always the optimal result of a optimization problem.

  - Optimize learning algorithms based on Bayesian networks, clustering, tree decisions or association rules.
- The evolutionary algorithms that will be implemented are the following:
  - Genetic algorithms: Based on Darwinian species evolution. The best candidates are crossed-over and mutated in order to find a better solution.
  - Particle swarm: It simulated the behaviour of bee swarms. Each candidate solution shares information with the nearest neighbours.
- The evolutionary algorithms that will not be implemented are the following:
  - Memetic algorithms
  - Ant swarm algorithms
  - Gaussian adaptation algorithms

4. METHODOLOGY

I divided the methodology in four phases. The first one consisted on building a solution to the learning problem by using evolutionary algorithms in combination with artificial neural networks and support vector machines. In the second phase, I gathered three different types of datasets in order to test the successfulness of learning methods when modelling the data. The third phase consisted on using methods to validate that the created models performed accurately. Finally, in the last step, the objective was to demonstrate, using statistical tools, that the method that I used was better than a simple method.

4.1. PROPOSED SOLUTION TO THE LEARNING PROBLEM

In the first part of this section I explain the basic concepts of evolutionary algorithms and its application to mathematical optimization problems. After that, I describe in high detail artificial neural networks and support vector machines models. Then, I illustrate how the evolutionary algorithms and artificial intelligence models are combined in order to solve learning problems more efficiently.

4.1.1 Evolutionary algorithms

Evolutionary algorithms (EA) are a type of metaheuristic optimization algorithms based on the evolution theory described by Charles Darwin in the 19th century [15]. As these algorithms make few assumptions about the function to be optimized they perform well in all types of approximation problems. In addition, they are easy to parallel making them faster than other types of algorithms. However, they do not assure to find a global optimum solution.

The first approaches to solve optimization problems using evolutionary algorithms originated in 1960s in three different places all over the world. John Holland, in University of Michigan, started to develop genetic algorithms, Lawrence Fogel and his colleges of the University of California in San Diego set up experiments on evolutionary programming and, in Europe, Ingo Rechenberg and Hans-Paul Schwefel introduced evolution strategies. In the early nineties these three paths were unified to create what nowadays is known as evolutionary algorithms.

In the Algorithm 1 a pseudo-code of a simple EA can be seen. As it is shown, EAs use several types of operators during the execution and each of them have an important role in the whole process. They are described below:

Genesis operator: At the beginning of the algorithm several candidate solutions are randomly generated and they form the initial population. As the individuals have random values they are uniformly distributed over all the search space.

Fitness operator: It is used to measure the quality or fitness of an individual and will determine the probability of survival of each candidate. Usually a numeric value is assigned to the individual. The main problem related with this operator is that in many real applications to

Algorithm 1: Evolutionary algorithm pseudo-code

```
Use Genesis Operator to create a population of randomly generated individuals;
Apply Fitness Operator to calculate the fitness of each individual;
while A criteria is not satisfied do
    repeat
        Use Selection Operator to select the parents;
        Apply a Genetic Operator to generate offsprings;
        With Reproduction Operator select between the parents and the offsprings and add them to
        the new population;
    until A new population is created;
    Apply the Elite Operator to maintain the best individual from previous generation;
end
```

calculate the fitness demands high computational resources so it becomes in the bottle-neck of the algorithm. To solve this issue fitness approximation functions can been used.

Selection operator: Before using genetic operators parents have to be selected. The selection operators used during the project were the following:

Round-Robin: Being I random integer number between $[1, P]$ where S is the size of the population and P the population then the selected individual for $k+1$ iteration would be $P_{I_{k+1}}$ where $I_{k+1} = mod(I_k, S) + 1$. In the first iteration S is assigned a random value.

Q-Tournament: Q individuals are taken randomly from the population forming a set S, where Q is the size of the tournament. The set S is ordered by fitness value and the first N individuals are taken where N is the number of parents needed.

Genetic operator: Genetic operators are used to change the values of the individuals. Several crossover and mutation genetic operators were used. The evolutionary algorithm only uses one crossover and one mutation in each execution and in each loop is chosen randomly between the crossover and the mutation. The experiments were focused on eight types of crossovers operators:

Arithmetical crossover (AC): Being P_1 and P_2 two individuals and a random variable λ uniformly distributed between [0,1], the two offsprings are defined as:

$$C_1 = \lambda P_1 + (1-\lambda)P_2 \text{ and } C_2 = \lambda P_2 + (1-\lambda)P_1$$

where λP_x means that the vector/matrix of values of the individual P_x is multiplied by λ. Then C_x is the result of a sum of vectors/matrices.

Differential crossover (DC): Being P_1, P_2 and P_3 three individuals and a random variable λ uniformly distributed between $[0, 1]$, the crossover is defined as $C_x = P_y + \lambda(P_z – P_w)$ where $\{y, z, w\}$ are all possible permutations of the set $\{1, 2, 3\}$.

Uniform crossover (UC): Given two individuals P_1 and P_2 with L attributes, L binomial random variables λ_i with probability 0.5 are flip. C_1 is made with the features from P_1 such that λ_i are 0 and the features from P_2 such that λ_i are 1. C_2 is made switching P_1 with P_2.

One point columns crossover (1Pc): Given two individuals P_1 and P_2 with a matrix of size $m \times n$ as attribute and a randomly chosen cross point $t \in (1, n)$ we define the one point

crossover over columns as:

$$C_1 = [P_{1(1,m)\times(1,t)}, P_{2(1,m)\times(t+1,n)}] \text{ and } C_2 = [P_{2(1,m)\times(1,t)}, P_{1(1,m)\times(t+1,n)}]$$

where C_1 and C_2 are the offsprings generated by the operator.

One point rows crossover (1Pr): Given two individuals P_1 and P_2 as a matrix of size $m \times n$ as attribute and a randomly chosen cross point $t \in (1, m)$ we define the one point crossover over rows as:

$$C_1 = [P_{1(1,t)\times(1,n)}, P_{2(t+1,m)\times(1,n)}] \text{ and } C_2 = [P_{2(1,t)\times(1,n)}, P_{1(t+1,m)\times(1,n)}]$$

where C_1 and C_2 are the offsprings generated by the operator.

Two point columns/rows crossover (2Pc/2Pr): It is a modified one point crossover algorithm with two cross points that generates the corresponding resultant combinations of offsprings.

Swarm crossover (SC): For each P_i particle of the swarm P_i^e would be denoted as the best fitness obtained by particle i. Moreover, P^e will represent the best qualification obtained by the swarm. Then the crossover could be defined as $C = \lambda_1 P_i + \lambda_2(P_i^e - P_i) + \lambda_3(P^e - P_i)$ where λ_1, λ_2 and λ_3 are random real numbers uniformly distributed over the range $[0, 1]$.

The used mutation operations were the following:

Uniform mutation: Being P an individual with L variables and λ a vector with L random variables, denoted as λ_i, uniformly distributed between $[min_i, max_i]$ where min_i and max_i represent the maximum and minimum allowed values for the i attribute, the two offsprings are defined as:

$$C_1 = min[P + \lambda_i, max_i] \text{ and } C_2 = max[P - \lambda_i, min_i]$$

where max and min are defined to do not exceed the limits of the values.

Non uniform mutation: Being P an individual and min_i, max_i the inferior and superior limits of its i attribute, P_i, the two offsprings are defined as:

$$C_1 = P_i + \Delta(t, max_i - P_i) \text{ and } C_2 = P - \Delta(t, P_i - min_i)$$

where $\Delta(t, y) = yr(1 - \frac{t}{T})$ being r a random number uniformly distributed between $[0, 1]$ and T the maximum number of generations.

Reproduction operator: It selects the individuals which survive to a genetic operation. Two types of reproduction operators were used:

N-Different: The best N different individuals are selected where N is the number of individuals that will survive.

Simulated Annealing: This technique is inspired in the annealing done in metallurgy. The fitness distance from the parent to the offspring is measured, $\Delta D = f_p - f_o$ where f_p is the fitness of the parent and f_o of the offspring. If it is wanted to minimize the function

then if $\Delta D \geq 0$ the offspring will substitute the parent with probability 1. However, if $\Delta D < 0$ then the probability of substituting the parent with the offspring would be defined by $P(f_p, f_o, T)$ where $P(\cdot)$ is defined as $e^{\Delta D/T} > R(0,1)$ where T is the temperature of the algorithm, usually defined as function dependant on the current iteration number $T(t) = k/ln(1+t)$ where k is a constant and t is the current iteration number, and the bigger it is the temperature the most probability that would be to allow a transition to a worst offspring. However, the lower the temperature the transition probability tends to zero asymptotically.

Elite operator: It is used to select the best individual from the population. As from generation to generation the best individual from the previous generation is always maintained it is assured that the best individual from the last generation will be the best individual seen in all the generations.

The main purpose of evolutionary algorithms is to solve mathematical optimization problems. The simplest case is to find the global maximum or minimum of a real function with a lot of hills such as $f(x) = (\sin x)/x$ or

$$f(x) = \frac{sin(\frac{x}{10})}{\frac{x}{10}} cos\left(\frac{x}{2}\right)$$

(see the graph representation in Figure 4.2).

Theorem 1 *The global maximum and minimum of a function are defined as the largest and smallest value in the function co-domain. Let take a real valued function $f(\cdot)$ defined on a domain X, then the global maximum of that function is defined as $f(x^*)$ and its maximum point as x^* where $\forall x : (x \in X : f(x^*) \geq f(x))$. In the case of global minima x^* is defined as $\forall x : (x \in X : f(x^*) \leq f(x))$.*

A function can also have several local maxima or minima.

Theorem 2 *Local maxima point is defined as x^* if there exists some $\epsilon > 0$ such that $f(x^*) \geq f(x)$ when $|x - x^*| < \epsilon$. Similarly, the local minima is defined as $\epsilon > 0$ such that $f(x^*) \leq f(x)$.*

In Figure 4.1 global maximum, minimum and local maxima and minima can be seen graphically. The key point about this type of tests is to check that the evolutionary algorithm did not become stuck at a local maxima or minima.

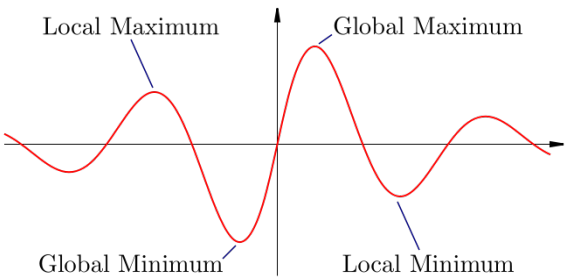


Figure 4.1: Function graph indicating its global maximum, minimum and local maxima and minima

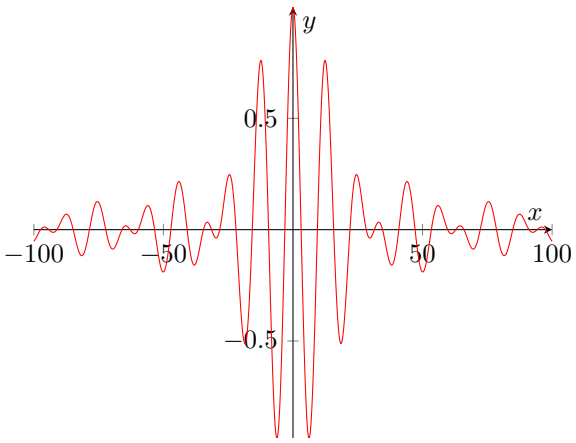


Figure 4.2: A hard to optimize mathematical function

4.1.2 Artificial intelligence algorithms

In this work, I did experimentation two well-known paradigms of artificial intelligence: artificial neural networks and support vector machines. Both of them have been widely used at the industry to model several processes.

Artificial neural network

An artificial neural network (ANN) is a computational paradigm inspired in the animals biological nervous system. ANNs started to be developed in the beginning of the 40s and although the development have continued during several years of research three key discoveries have to be highlighted. The first one took place in 1943 when Warren McCulloch and Walter Pitts described how the neurons might work and they modelled a simple ANN using electric circuits. Six years later, in 1949, Donald Hebb argued that neural pathways are strengthened each time they are used, this was one of the most important discovers in ANN. The last discovery was done by Frank Rossenbert in the late 1950s when he created the perceptron, a two layer network, which was capable of doing a naïve pattern classification by adjusting the connection weights.

The basic element of the ANN is the neuron, also known as perceptron (See Figure 4.3). A neuron can have many inputs, represented as a vector $x \in \mathbb{R}^n$ where x_i represents the i^{th} input to the neuron. Each input has a real value between 0 and 1, known as weight, so the neuron has a vector of weights $w \in \mathbb{R}^n$ where w_i is the weight for the i^{th} input. The total input is calculated as the dot product of $w \cdot x$ and the perceptron has only one output defined as $y = \varphi(w \cdot x + b)$ where $\varphi(\cdot)$ is the activation function and b is the bias which takes the same role as the y-intercept of the line. There are several commonly used activation functions:

- Heaviside step function:

$$f(y) = \begin{cases} 1, & if\ y > 0 \\ 0, & otherwise \end{cases}$$

- Linear activation function:

$$f(x) = x$$

- Sigmoid activation function:

$$f(y) = \frac{1}{1 + e^{-y}}$$

- Hyperbolic tangent function:

$$f(y) = tanh\ y = \frac{e^{y} - e^{-y}}{e^{y} + e^{-y}}$$

The perceptron uses the Rosenblatt's learning algorithm [26] rule to tune its weights. In Algorithm 2 a pseudocode can be seen for a training dataset with S instances composed of pairs (x_j, d_j) where $x_j \in \mathbb{R}^n$ is the j^{th} input vector and d_j is the desired output for that vector.

Algorithm 2: Rosenblatt's learning algorithm pseudo-code

Initialise the weights and the bias to zero or small random values;
repeat
    Calculate y for each training instance $\vec{x}$;
    Update weights: $w_i(t+1) = w_i(t) - \eta(d_j - y_j(t))x_{j,i}$ where $0 < \eta \leq 1$ is the learning rate, $w_i(t)$ is weight i at time t and $x_{j,i}$ is the i^{th} input of the j^{th} training input vector. (Note that if $d_j(t) = y_j(t)$ then $w_i(t+1) = w_i(t)$);
until *The total error* $1/S \sum_{j=1}^{S} |d_j(t) - y_j(t)|$ *is less than a specified threshold or several iterations have been completed*;

An ANN is a network composed by several connected perceptrons that simulate a biological neural network. The perceptrons of an ANN are arranged in three types of layers:

Input layer: It is the input of the network and is where input vectors are connected. The number of neurons in this layer is equal to the length of the input vector.

Output layer: It is the output of the neural network. The number of neurons that are contained in this layer is equal to the number of outputs.

Hidden layer: This layer is placed between the input and output layers and several of them can be connected. As these layers are hidden from the view of the user then they are often considered as a black-box. Adding more of these layers enables more processing capability but it also adds complexity to the model.

In Figure 4.4 a multilayer ANN with three neurons in the input layer, four neurons in the hidden layer and one output neuron can be seen. The weight structure of a multilayer neural network is represented as a vector of matrices such as $\{W^1, W^2, \cdots, W^l, \cdots W^n\}$ where n is the number of layers. Each matrix has the form of $W^l_{i,j}$ where i, j is the weight between the neuron j of layer $l-1$ and the neuron i of the layer l, in case of $l - 1 = 0$ then j represents the j^{th} input value. If $j = 0$ then the bias of the neuron is addressed.

ANNs can be classified by the way the different layers are connected. There are two types of networks: static and dynamic. In static networks the output only depends on the actual input but in the case of dynamic networks the actual output also depends on the previous output.

Examples of the previously described types of networks are feedforward and NARX respectively. Feedforward neural networks are no more than a combination of several perceptrons arranged in

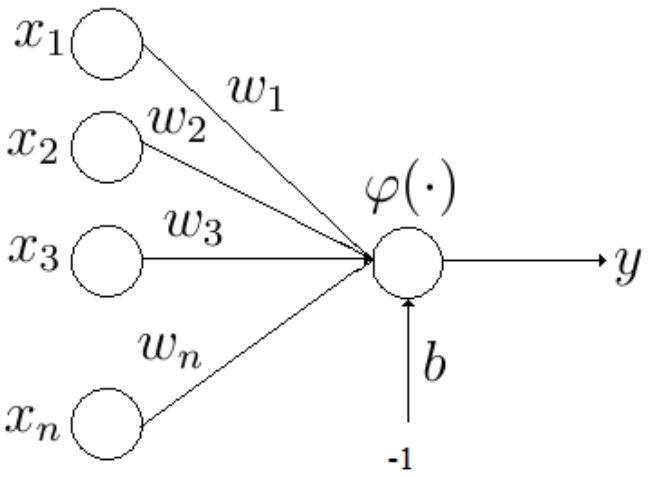


Figure 4.3: The basic structure of a perceptron

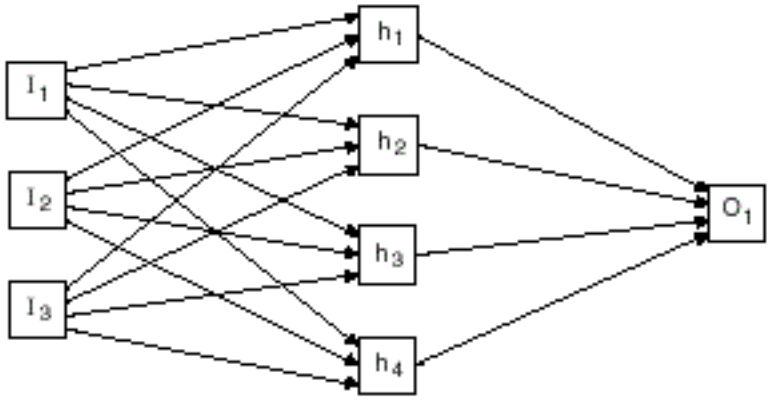


Figure 4.4: In the left side a recurrent multilayer neural network can be seen. In the right side a feedforward neural network

several layers where the outputs of the previous layers are the input of the next layers. Mathematically speaking a feedforward neural network can be described as:

$$y = \varphi\left(\sum_i W_i^m \varphi\left(\sum_j W_{ij}^{m-1} \cdots \varphi\left(\sum_n W_{kn}^1 x_n\right)\right)\right).$$

The nonlinear autoregressive exogenous model (NARX) tries to predict the next value of a time series relating the past values of the same series and the previously made predictions. Algebraically it can be stated as:

$$y_t = F(y_{t-1}, y_{t-2}, y_{t-3}, \cdots, u_t, u_{t-1}, u_{t-2}, \cdots) + \epsilon_t$$

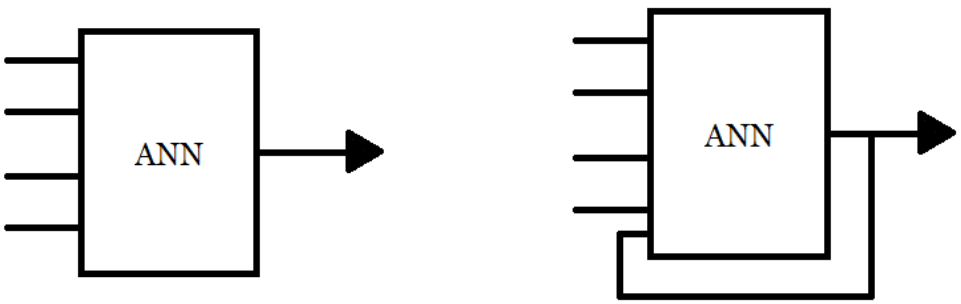
where y_i is the value predicted at time i, u_i is the value of the time series at time i and ϵ_t is added noise.

Once the ANN has been set-up it has to be trained, I used backpropagation supervised training algorithm for this project. It is based on gradient descent which is a method for finding the nearest local minimum of a function. A mathematical definition can be seen in Appendix B. In this case as the error function of a neural network is convex it will find the global minimum. Its basic mechanism can be seen in Algorithm 3.

Algorithm 3: Backpropagation pseudo-code

```
Initialize randomly network weights;
repeat
    foreach ex : Training examples do
        prediction = network(ex);
        error = prediction - targetValue;
        Compute Δw_h for all weights from hidden layer to output layer;
        Compute Δw_i for all weights from hidden layer to output layer;
        Update network weights;
    end
until All examples are correctly classified;
```

As backpropagation uses gradient descent method then the derivative of the squared error function with respect to weights should be calculated. The squared error function is defined as $E = \frac{1}{2}(t-y)^2$



(a) Static ANN

(b) Dynamic ANN

Figure 4.5: Two types of artificial neural network

and its derivative

$$\partial E/\partial w_i = \left(\frac{dE}{dy}\right)\left(\frac{dy}{dnet}\right)\left(\frac{\partial net}{\partial w_i}\right)$$

where

$$\frac{dE}{dy} = y - t, \frac{dy}{dnet} = \frac{d\varphi}{dnet}\text{(the derivative of the activation function) and}\frac{\partial net}{\partial w_i} = x_i.$$

Then if the activation function is $y = 1/1 + e^{-z}$ and its derivative $dy/dz = y(1-y)$ the slope of the error function respect to weights is $\partial E/\partial w_i = (y-t)y(1-y)x_i$. So after deciding the gradient descent learning rate the change in weight would be $\Delta w_i = -\alpha\left(\partial E/\partial w_i\right)$ where α is the learning rate.

Usually two versions of backpropagation algorithms are used:

Levenberg-Marquadt backpropagation: Weight and biases are updated according to Levenberg-Marquardt optimization. It is one of the fastest training techniques in MATLAB [42].

Bayesian regulation backpropagation: This training function also updates weight and biases according to Levenberg-Marquardt optimization but it minimizes a combination of squared errors and weights, and then determines the correct combination so as to produce a network that generalizes well [41].

As I stated above, an ANN can have several layers with different neurons and each neuron with a weight, bias and activation function. These features are the parameters of the ANN and some of them can be set by the researcher but others are only configured by the training algorithm during the learning process. The parameter that is not tuned by the training algorithm is called hyperparameter. The number of neurons and layers, activation functions and the initial weights and biases are hyperparameters. However, weights and biases are not hyperparameters because they are optimized according to the training algorithm. One more key aspect of the parameters is that they do not have an interpretation related with the problem to model, they only establish a relationship between the inputs and outputs. That is why ANNs are usually classified as black-box type learning algorithms.

The tuning of hyperparameters, such as layers, number of neurons or activation functions, has been widely researched [47], [51]. Nonetheless ANN initial weights quite often are not tuned with the above mentioned hyperparameters but are only initialized by training functions with random values. In this project and in the accepted paper C I made emphasis in the use of initial weights and biases as a technique for ANN performance improvement.

Support vector machines

Support vector machines (SVM) is a supervised learning algorithm based on statistical learning theory that is used for classification and regression problems. SVM are among the best “out-of-the-box” machine learning algorithms because they can get outstanding results without needing too much knowledge for using them. Several types of problems are solved with SVM but, in particular, text categorization, bioinformatics and image recognition are the most interesting ones nowadays [33], [25], [24].

In 1974 Valdimir V. Vapnik and Alexey J. Chervonenkis developed the statistical learning theory. Based on this work Vapnik introduced the concept of SVM in 1979. An improved version, the soft margin classifier, was introduced by Vapnik and Corinna Cortes in 1995.

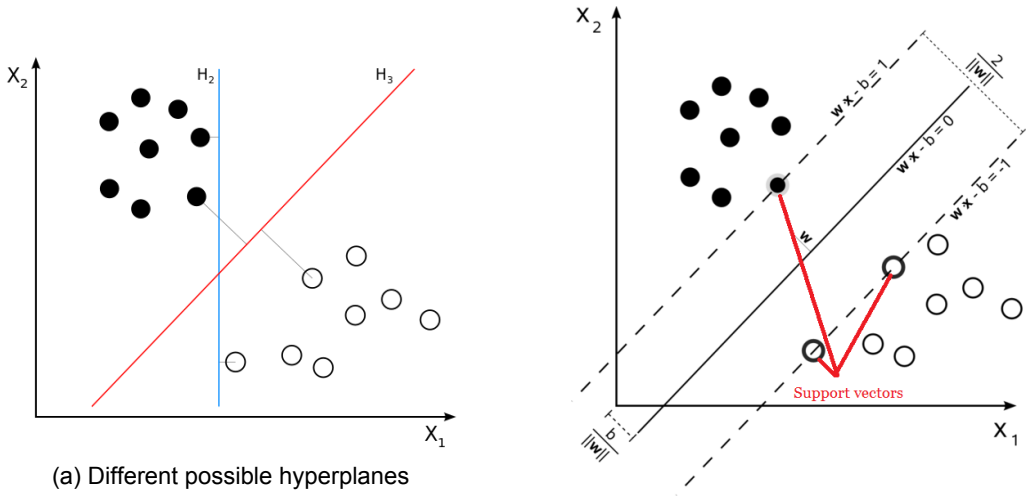
The idea behind SVM is to represent the input data in an n-dimensional space, where n is equal to the number of attributes in the problem. Each instance of the training dataset will be represented as a dot in the n-dimensional space and it will be labelled with 1 or -1. Mathematically the training data is defined as $T = \{(x_i,\ y_i) \mid x_i \in \mathbb{R}^n, y_i \in \{-1, 1\}\}_{i=1}^{N}$ where N is the number of instances in the dataset. In binary classification problems the SVM algorithm has to find a $(n-1)$-dimensional hyperplane, $w \cdot x$–$b = 0$ being $w \in \mathbb{R}^n$, that separates all the dots of class -1 from 1 without error, it is known as hard margin SVM. To separate the classes with a hyperplane the data has to be linearly separable. But, the important point about SVM, and what differentiates it from a perceptron, is that it finds the hyperplane with the maximum margin since the larger the margin the lower the generalization error will be. Maximum margin is obtained when the distance from the hyperplane and the nearest data point on each side is maximized. The points that line in the boundaries of the margin are known as support vectors because they establish the size of the margin. In the left image of Figure 4.6a can be seen that blue line correctly classifies the data but it does not has the maximum hyperplane, nonetheless the red line also classifies the data correctly and it also has the maximum margin.

In the right image of Figure 4.6b is shown that the margin is defined by the equations $w \cdot x - b = 1$ and $w \cdot x - b = -1$. No points can be between the two hyperplanes so the instances of class 1 should fulfil $w \cdot x - b >= 1$ constraint and instances of class -1 the $w \cdot x - b <= -1$ constraint. The distance between the two boundary hyperplanes is $2/\|w\|$ and $\|w\|$ should be minimized in order to maximize the distance.

Sometimes the data to be classified is not linearly separable so for that cases a kernel function could be used to map the data from a n-dimensional space to a higher dimension where is supposed to be easier to find a hyperplane that separates the classes. For example, there are two data points in the original space, $P1$ and $P2$, and a function φ that maps a point into a higher dimensional space. φ maps these two points to a higher dimensional space, $\varphi(P1)$ and $\varphi(P2)$ and then the inner product, generalization of dot product, of the transformed data points is calculated. Of course, this mapping should preserve the relevant dimensions between data points so SVM can generalize well. However, to calculate the map of every data point can be expensive to compute so it would be desirable to find a function $K(\cdot, \cdot)$, called kernel function, that $K(P1, P2) =< \varphi(P1), \varphi(P2) >$ where $< ., . >$ denotes the inner product and $P1$ and $P2$ could be any data point of the original space. This is called the kernel trick and the idea is to calculate the inner product of two points in a higher dimensional space without even knowing the φ function and only using the data points in the original space. Several kernel functions are available, for example:

- Polynomial kernel: $K(x, y) = (\alpha x^T y\ +\ c)^d$
- Radial basis function kernel: $K(x, y) = exp(-\gamma \|x - y\|^2)$
- Hyperbolic tangent kernel: $K(x, y) = tanh(\alpha x^T y + c)$

Although SVM was originally designed for binary classification multiple classification can also be done using a set of hyperplanes. The key idea is to reduce a multiclass problem into several binary classification problems, two approaches can be taken: one-against-all and one-against-one. In the first case, k SVM model are built, one per class and the i^{th} SVM is trained to distinguish between all the examples of the i^{th} class from the rest of the instances. When a new instance is wanted to



(a) Different possible hyperplanes

(b) Maximum margin SVM

Figure 4.6: Boundaries and support vectors of SVM. Images taken from Wikipedia

be classified then the k classifiers are run and the classifier which outputs the largest positive value from its output function is chosen to determinate the class of the instance [54].The one-against-one method builds one SVM for each pair of classes and classification of an instance is done by voting strategy, in which each classifier assigns the instance one class, in other words, it votes for one class, finally the class with most votes determines the class of the instance [45].

SVMs can also be applied to regression problems and they are known as support vector regression (SVR). One of them is ϵ-SVR where ϵ threshold is defined and the error function only takes into account the data points that lie out of the margin. In other words, the points that fulfil the constraints $\epsilon >= \vec{w} \cdot \vec{x} - b$ and $-\epsilon <= \vec{w} \cdot \vec{x} - b$ are not used in the error function calculation. SVM regression tries to find a continuous function such that the maximum number of data points lie within the ϵ tube. So predictions falling within ϵ distance of the true target value are not interpreted as errors. In Figure 4.7 can be seen that blue points are ignored but red points are included in the error function calculation. One more SVM algorithm for regression problem is ν-SVR which is new class of SVM introduced in [56] where they proved that ν is an upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors. It can handle both classification and regression and uses a different penalty to control the number of support vectors and training errors but the decision function is the same as ϵ-SVR. In this case ν parameter is used.

During the description of the types of SVM I presented several parameters that can be tuned when using SVM. Setting these parameters is not a straightforward activity but a really complex one because the range of values is large. There are two types of SVM parameters, the ones of the SVM itself and the kernel function parameters. Some of them are set when the algorithm is learning and others such as C, γ and ϵ are tuned by the researcher before the training phase.

Some parameters such as the normal vector w and bias are set by the training algorithm and given that the rest of the parameters are constant they will always converge to the same value for the same training points.

Sometimes mapping data to a higher dimension could make the model to overfit. For that reason in some cases is better to separate points in the original space by allowing some flexibility in the category separation. To allow that flexibility SVM uses a cost parameter, C, that controls the trade-off

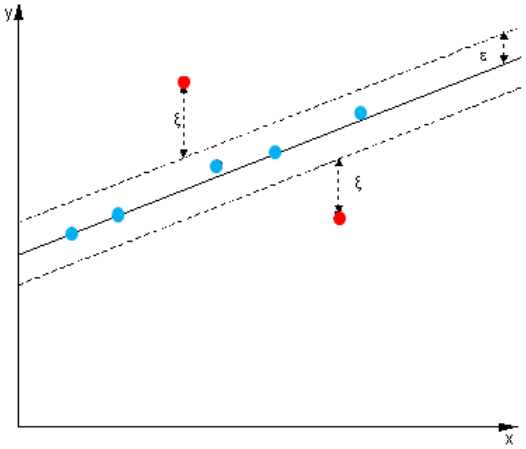


Figure 4.7: SVM used for regression problems. Modified from [63]

between training errors and margins by using a soft margin that permits some misclassifications, see Figure 4.8. The penalty error is calculated as the distance to the hyperplane of the misclassified point by error cost C. If the cost of C is increased then the penalty for misclassified points would be higher and it forces to create a more accurate model that perhaps would not generalize correctly [64].

γ and α hyperparameters take part in the kernel trick functions presented above. A bad tuning of these parameters could make the decision boundary be highly sensitive to noise in the training data. It could also provoke higher-dimensional projection to lose its non-linear power and to behave almost linearly [60].

When using SVM for regression problems an ϵ parameter was presented to define which points are taken in the error function. The bigger the ϵ value the less the error but the worst the pattern recognition will be. If a good tuning of the ϵ parameter is found then SVR will find a hyperplane that models the pattern correctly. For ν-SVR ν parameter is used to control the number of support vectors and margin errors.

4.1.3 Adapting evolutionary algorithms to hyperparameter tuning problem

After building a platform of evolutionary algorithms capable of optimization real functions I adapted it to the main purpose of this work, to tune the hyperparameters of a machine learning model. I faced three main difficulties in this process: to adapt the structure of the individuals, to create crossovers that fit with the new characteristics of the structure and to build a fitness function that reflects the successfulness of the optimized model.

Custom individual structures

I also adapted the structure of the individuals for each artificial intelligence model. In the case of SVR I only needed a vector of parameters because just C, γ and ν are optimized. However, in

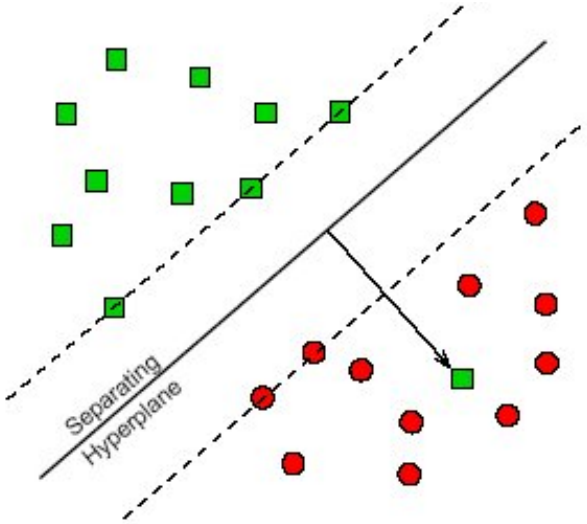


Figure 4.8: Example of C parameter avoiding overfitting. Taken from [64]

ANNs the weights are represented as several matrices. So in the case of ANN the structure of the individuals would be a vector of matrices.

Custom crossovers

As vectors are a special case of matrices then building a crossover that deals with matrices will be useful both for ANN and SVM. In the case of arithmetical, differential, uniform and swarm crossovers the algorithm was the same for ANN and SVM. However, one and two point crossovers does not make sense for SVM because the parameters are not interchangeable (for example, it has no sense to swap the values of γ and ϵ parameters). So, I only used the last algorithms for ANN hyperparameter tuning.

Custom fitness functions

Fitness functions had to be adapted to each model. The usual schema is to set the learning model with the hyperparameters of the individual to grade, train the model for a few iterations, calculate the prediction and assess the error with the target outputs of the training instances. In order to have a better prediction error measurement I used some of the techniques explained at section 4.3.

4.2. DATA GATHERING

I validated the proposed solution on three different type of dataset: the KEEL regression dataset, a synthetic dataset made by random sampling the space of neural networks and data of the concentrations over the time of several chemical species in a catalytic reaction.

The KEEL-dataset repository was created by developers of KEEL open-source software tool with the aim of providing to researches a set of benchmarks to analyse the behaviour of their learning

algorithms. There are benchmarks for classification, regression and unsupervised learning [3]. In this research, I used KEEL regression datasets.

I created the synthetic datasets were created by using feedforward Random Neural Networks (RNN). A RNN is defined as a neural network with the number of inputs, outputs, layers and neurons selected randomly within an interval. Several inputs are given to the RNN and the generated outputs are recorded obtaining as result a dataset. Using this technique infinite datasets can be obtained representing each RNN a different process to model. In order to avoid over-fitting normally distributed I added noise to the outputs of the training set, the set of data that will be used for training the learning algorithm. However, as the test set represents the real function without noise then to verify that the learning algorithm is generalizing $\int_X (f(X,\beta) - h(X,w))^2 = 0$ can be performed.

The last dataset I used, is composed by experimental results of BTO processes obtained by [22] in a laboratory scale reactor. The BTO process consists on the catalytic transformation of bioethanol into olefins, where the latter are a type of hydrocarbon with double bond carbon to carbon which are very important for the petrochemical industry. This process is carried out in fixed-bed catalytic reactor where a very active and selective catalyst exists. Nevertheless, over time, due to poisoning, the catalyst deactivates linearly or exponentially depending on the deactivation mechanism [59].

The olefin groups production rate curves at different operation conditions are the modelling objectives. One more variable to take into account when modelling is the activity of the catalyst because of its influence in the final production of olefins. Activity can be estimated in three different ways. The first one is using the activity data included in the datasets, this option is not is not a good performance estimator because this data will not be available when modelling a new process. The second approach is to use the deactivation equation

$$\frac{da}{dt} = -k'_d \cdot \theta_d(X_W) \cdot a^{n_d}$$

being $-k'_d$ and a^{n_d} deactivation kinetic parameters and $\theta_d(X_W)$ the attenuation of the coke deposition because of water, defined in the equation

$$\theta_d(X_w) = exp(-K_{dW} X_W).$$

I will use the deactivation equation to obtain the activity predictions and I will introduce them in the olefin production model. The last method consists on that the artificial intelligence model itself calculates the activity at the same time.

4.3. METHODS TO VALIDATE THE MODELS

4.3.1 Training and validation error

The main objective when using a machine learning algorithm is to learn and generalize but not to memorize. If the algorithm generalizes it is going to perform accurately in new examples. A used measure for generalization is the validation or test error which is defined as the error of the model in the unseen data and is a function of the model complexity. As the model is trained it is expected

to improve the prediction in the training and test data, or in other words, minimize the training and validation error. However, usually the dataset has errors so if the algorithm train is trained too much it will also learn the noise. If complexity increases the number of training errors will usually decrease, but the risk of over-fitting the data correspondingly increases. As consequence the performance in the test data will decrease and we will lose generalization. This phenomenon is known as over-fitting. Over-fitting can happen due to two reasons: when the number of instances of the train data is small so the model cannot generalize or when the model is too complex (the number of free parameters is large). In Figure 4.9 an example of over-fitting can be seen, it is shown that at some point the model fits too much to the training data, thus it decreases the training error but as consequence the test error increases. Fortunately, there are two ways of controlling over-fitting: cross-validation and to penalize high-complexity models.

4.3.2 Cross-validation

Cross-validation is a technique for measuring the predictive performance of a statistical model [62], [61]. It gives an insight of how well the model generalizes predicting the fit of a model to a hypothetical test set when explicit test set is not available.

In cross-validation a training set is given and it is randomly divided in k-groups with the same size. k rounds will be done, in each round, let's name them as r_x where x is an integer number between 1 and k, the r_x group will be taken and it will be used as the test data, the rest of the data is going to be the training data. The model is trained with the training data and validated with the r_k group, the validation error of each round is saved. After the k repetitions an average of k validation errors is calculated and it is taken as the generalization estimation. Minimizing the cross-validation error is a good way to select a model. The key point is that each instance of the set is used for validation only once so the results are not biased.

Cross-validation has two extreme cases: divide a set of n samples in 2 groups or n groups. The first approach is called holdout method and the second one leave-one-out-cross-validation (LOOCV). Holdout method is used when the amount of data available is very large and LOOCV when there is not enough data.

There are some key points that should be taken into account when performing this type of techniques:

- The selected model has to be used with data of the same population. For example, if a model is selected according to cross-validation error using data of female people under 20 years old then the model cannot be applied to real data from women of 60 years because the data does not belong to the same population.
- Misuses can happen if at the same time some of the data is used both to train the model and to test it. This can happen if some of the data is twinned and, as result, inaccurate model validation can happen.

4.3.3 Regularization

As it was shown, it is very easy to make a model to fit the training data just by including more degrees of freedom to it. For example, if regression is being used higher order terms can be

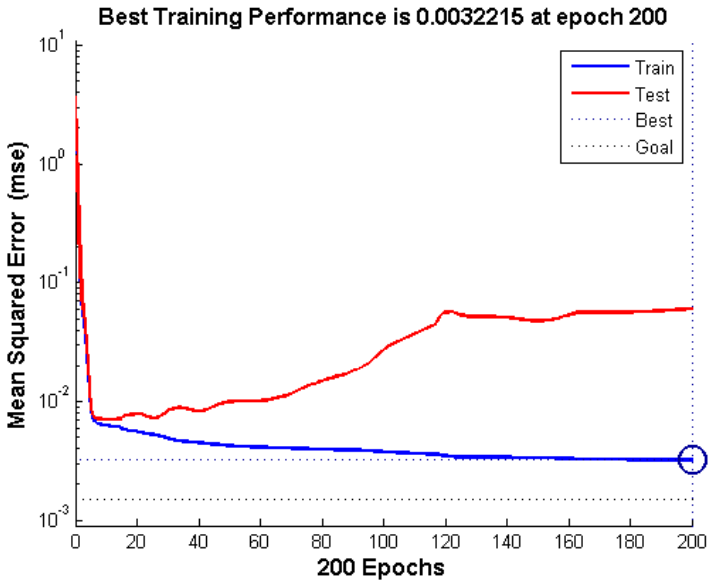


Figure 4.9: Over-fitting situation: After the 5th epoch the algorithm starts to over-fit

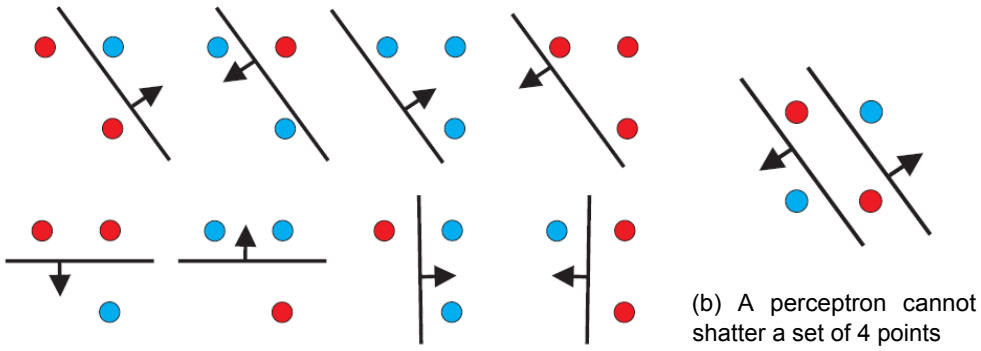
added to make the model fit better the known data but, again, over-fitting is going to happen. So a complexity-generalization dilemma is being faced.

One of the answers to this dilemma is Occam's razor principle which states that at same conditions the simplest hypothesis usually is the correct one. However, Occam's razor does not assure that the selected hypothesis is correct and, in addition, a definition of simplicity is not given. About the first issue, a more correct and complex hypothesis should be preferred against a simpler but less correct option. Regarding to the second concept, Ockham himself wrote that when two theories have the same consequences the theory with fewer assumptions should be preferred.

When learning algorithms started to be developed, several techniques for model comparison also emerged. Most of them are based on the two key points of Occam's razor: correctness and complexity. Below, some methods will be presented for comparison of models taking into account the two points of the dilemma. It is worth to say that these methods do not say anything about the quality of the model itself, it is not a comparison against null hypothesis, but is just a comparison between several models based on relative correctness and complexity.

Structural risk minimization (SRM)

VC dimension, defined by Vladimir Vapnik and Alexey Chervonenkis, is a measure of the capacity of a classification algorithm. Capacity determines how complex an algorithm can be, measured by the cardinality of the largest set of points that an algorithm can shatter. To shatter is mathematically defined as a classification model $f(\cdot)$ with a vector of parameters θ and it is said that a model can shatter a set of points if for all possible arrangements of labels of these points exists a θ that makes $f(\cdot)$ not to make any error. For example, the VC dimension of a line in a two dimensional space is three because it can shatter an arrangement of three points, see Figure 4.10a, but not an arrangement of four points, see Figure 4.10b.



(a) A percetron shattering a set of three points

(b) A perceptron cannot shatter a set of 4 points

Figure 4.10: The VC dimension of a perceptron in a two dimensional space is three. Taken from [27]

SRM is a method for model selection that provides a trade-off between the complexity of a model and its success in fitting the known data (the empirical error). It uses the VC dimension and training error to calculate a value and a model with smaller SRM value is supposed to be better. The typical steps of the procedure are shown below [58]:

1. Choose a class of functions, for example polynomials of degree *n*
2. Divide the class of functions into a hierarchy of nested subsets in order of increasing complexity. For example, polynomials of increasing degree ($p_1 \subset p_2 \subset \cdots \subset p_n$) where p_i is the set of polynomials of degree i.
3. Calculate the VC dimension of each hierarchy.
4. Perform training error, T_{error}, minimization on each subset. Thus, calculate

$$\Theta_i^* = \underset{\Theta_i}{argmin} T_{error}(\Theta_i)$$

   where Θ_i is a set of parameters for the model and Θ_i^* is the specific set of parameters that minimize the training error of the model.
5. Calculate the VC dimension of the model.
6. Select the model with lowest $J(\Theta_i^*, h)$:

$$J(\Theta, h) = Training\ error + \underbrace{\sqrt{\frac{h(log(\frac{2L}{h}) + 1) - log(\frac{\eta}{4})}{L}}}_{Confidence\ interval}$$

   where the function J is the upper bound on the test error, h is the VC dimension of the model, L is the size of the training set, and η is chosen such that $0 \leq \eta \leq 1$ so the upper bound will hold with probability $1 - \eta$.

In Figure 4.11 a graph that represents the training error, the confidence interval and the upper bound $J(\Theta, h)$ is shown.

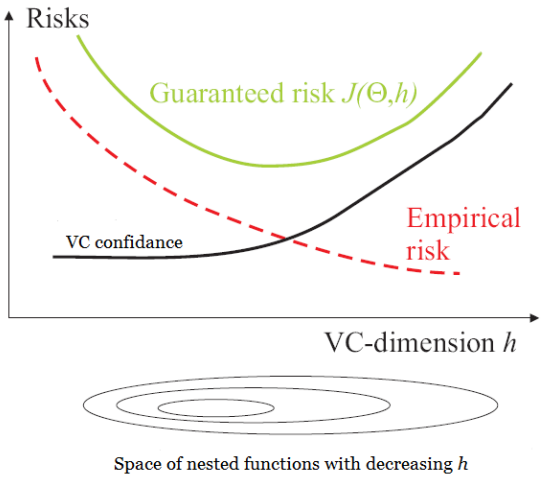


Figure 4.11: Graphical representation of test error upper bound. Taken from [27]

Akaike information criterion (AIC)

As with SRM, AIC [2] also tries to balance how good the model fits to the training data and its complexity. AIC is based on the Kullback-Leibler distance between the model and the truth. AIC is defined as: $AIC = 2k–2ln(L)$, where k is the number of free parameters of the model and L is the maximized log-likelihood. The preferred model is the one that has the minimum AIC value. As it can be seen, the more the number of parameters increases (i.e. complexity increases) then the more AIC value increases. AIC is asymptotically equivalent to LOOCV.

4.4. ERROR MEASURES FOR REGRESSION PROBLEMS

Several error measures can be found to determine the accuracy of an algorithm applied to regression problems. In this I classify section error measures and I discuss their advantages and disadvantages based on [31].

Scale-dependant measures: Using these measures several algorithms can be compared if they were applied to the same dataset. However, the main problem is that they cannot be used to compare across datasets with different scales. Commonly used measures are MSE, RMSE and MAE. It has also to be taken into account that measures such as MSE and RMSE where the errors are squared are more sensible to outliers or atypical data.

$MSE = \frac{1}{n}\sum_{i=1}^{n}(F_i - Y_i)^2$ being F_i the forecast value and Y_i the target value.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(F_i - Y_i)^2}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\|F_i - Y_i\|$$

Measures based on percentage errors: Percentage error is calculated as $p_t = 100 \cdot (Y_t - F_t)/Y_t$ and the main advantage is that they are scale independent so the measure can be used to compare the performance in different datasets. MAPE and RMSPE are most commonly used measures. It should be noted that when $Y_t = 0$ the measures are undefined and can have really asymmetrical distributions if the values are close to zero. One more disadvantage is that they do not have meaningful zero, meaning that the value of zero is arbitrary, for example in Fahrenheit and Celsius temperature scales where the value zero has different meaning. Another drawback consists on the penalty asymmetry for negative errors ($Y_t < F_t$) and positive errors ($Y_t > F_t$) where the first ones are bigger. To solve this problem symmetric measures were invented, for example sMAPE, but still some of the problems related with small values happen.

$$MAPE = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$

$$RMSPE = \sqrt{\sum_{i=1}^{n} \frac{d_t^2}{n}} \text{ being } d_t = \frac{F_t - Y_t}{Y_t}$$

$$sMAPE = \frac{1}{n} \sum_{t=1}^{n} \frac{|F_t - Y_t|}{(Y_t + F_t)/2}$$

Measures based on relative errors: A different approach is to divide each error by the error obtained using a naïve algorithm as benchmark, e_t/e_t^* where e_t^* is the error of the naïve algorithm. Usually the benchmark algorithm is the error of taking F_t as the last observation, namely random walk algorithm. Several measures that fall in this category are MRAE, MdRAE and GMRAE. The major disadvantage is that according to [31] sometimes it can have infinite variance.

$$MRAE = \frac{1}{n} \sum_{i=1}^{n} (|e_t/e_t^*|)$$

$$MdRAE = median(|e_t/e_t^*|)$$

$$GMRAE = \text{Geometric mean } (|e_t/e_t^*|)$$

4.5. FRIEDMAN TEST

Friedman test is a non-parametric statistical test where the null hypothesis H_0 states that the outcomes of each algorithm have the same distribution of probability, namely that all algorithms perform equally. It is used to detect significant differences between several algorithms [17] but it does not specify between what algorithms there are significant differences.

In this test, given a matrix M with m rows (groups) and n columns (algorithms) and being M_{ij} the element of the matrix in the i^{th} row and j^{th} column, each element of the data is ranked within their own group. If in the same groups there are tie values then the average rank of the group without those values is calculated and assigned to tied values. Next, a new matrix $R^{m \times n}$ is created were

R_{ij} is the rank of the M_{ij} element. The ranking of each algorithm is calculated as

$$r_n = \frac{1}{m}\sum_{i=1}^{m} R_{in}$$

where m is the number of rows and n the column of the algorithm to be ranked. Then, by applying more complex statistical techniques the p-value is calculated. The p-value is defined as the probability of obtaining a result at least as extreme as the one actually observed assuming that the null hypothesis is true.

If the result of the Friedman test is positive, in other words, if the p-value is inferior to the desired significance level $\alpha = 0.05$ then post-hoc analysis is needed to determine between which pair of algorithms the differences exists.

4.5.1 Post-hoc analysis

Post-hoc test are done after applying the Friedman test and are used to find between what algorithms are statistical significance. These techniques counteract the problem of multiple comparisons. This problem states that the more hypothesis are tested the most probable it is to find an algorithm that rejects the null hypothesis incorrectly. To address this issue several techniques can be used, some of them are Bonferroni, Hochberg [5] and Bergmann[6] procedures that modify the rejection criteria according to the number of comparisons in order to control the significance levels [44]. Two different type of comparisons can be done: one-against-all and all-against-all. Each comparison will create several pairwise comparisons with some corrections that state if the null hypothesis is rejected or not. In case it is rejected the algorithm with worst mean prediction error performs significantly worst than the other. The advantage of only applying one-against-all is that the statistical power is bigger and then the comparisons significance are more accurate.

5. PLANNING

In this part I explain the planning done for the project. First, I list the tasks that the project was composed of. Then, I give a detailed human resource planning. After that, I describe the working schedule and environment. Finally, I show the working plan.

5.1. MAIN TASKS

The development of the project concerns the following activities and tasks:

A.1 Organization and control

- A.1.1. Organization: In this task the planning, task assignments, project kick-off and its following phases are defined and prepared.
- A.1.2. Monitoring: Project tracking and control is done in order to fix all the possible deviations and problems that can arise.

A.2 Technical platform set-up

- A.2.1 Software set-up: Setting-up of the needed software to develop the platform is done.
- A.2.2 Computer communication set-up: Task that consists on setting-up and connecting the development computer with the high-performance computers.

A.3 Evolutionary algorithms platform design

- A.3.1 Mathematical design: Abstract design of the evolutionary algorithms applied to the optimization of functions and hyperparameters.
- A.3.2 Class diagram design: Development of UML class diagram that represents the main functioning and interaction between classes.
- A.3.3 Design documentation: Writing of the design part of the software documentation.

A.4 Evolutionary algorithms implementation

- A.4.1 Evolutionary algorithms implementation to solve function optimization problems. This task is composed of the following steps:
  - A.4.1.1 Genetic operator development: Implementation of the diverse crossovers and mutations.
  - A.4.1.2 Specific structures class development: Implementation of several classes able to deal with vector and matrix structures.
- A.4.2 Evolutionary algorithms implementation to solve learning algorithms hyperparameters optimization. This task is composed of the following steps:
  - A.4.2.1 Development of operators for the optimization of ANN hyperparameters.
  - A.4.2.2 Development of operators for the optimization of SVM hyperparameters.

A.5 Experimentation

- A.5.1 Graphical representation of academic problems results.
- A.5.2 Qualitative and temporal comparison of genetic operators: Comparison of the above mentioned evolutionary algorithms in order to determine characteristics of the quality of the results and the time necessary to find them.

- A.5.3 Evolutionary algorithms application to an ANN and SVM hyperparameter optimization problem. This experiment will be realized in the following environments:
  - A.5.3.1 BTO process chemical reactor: Hyperparameter optimization of ANNs and SVMs that model a chemical reactor of the BTO process.
  - A.5.3.2 KEEL datasets: Hyperparameter optimization of ANNs and SVMs that model KEEL regression datasets.
  - A.5.3.3 Synthetic datasets: Hyperparameter optimization of ANNs and SVMs that model synthetic datasets.
- A.5.4 Statistical analysis of the results: The obtained results will be analysed with statistical tools and conclusions will be made.

A.6 Software documentation writing

- A.6.1 Doxygen documentation: All the software platform classes and functionalities will be described using Doxygen documentation tool. Some code examples will be written.

A.7 Project closing: The project will be closed and the evolutionary algorithms platform will be delivered to the client.

5.2. HUMAN RESOURCES PLANNING

The work team will be composed of the following profiles:

- Project director (Yoseba Peña): This profile has the responsibility of doing management, coordination and monitoring activities of the project.
  Knowledge:
  - Advanced knowledge of project management.
  - Advanced knowledge of handling management software tools.
  - Wide experience in project management.

  Skills:
  - Good oral and writing communication skills.
  - Ability to solve interpersonal conflicts.
  - Good management of proposed ideas.
  - Capability to motive team members.
  - Good time management.
- Mathematician (Cruz Enrique Borges): He/she is in charge of the mathematical design of evolutionary algorithms and the strategies statistical strategies that are going to be developed.
  Knowledge:
  - Advanced knowledge of mathematics and statistics.
  - Advanced knowledge of evolutionary algorithms.
  - Good knowledge of algorithm complexity.
  - Basic knowledge of programming and software performance analysis.

  Skills:
  - Good writing communication skills.
  - Creative thinking.
- Software performance analyst (David Orive): He/she has the responsibility of analysing the performance of the algorithms when applied into a computer, find possible bottle-necks and

to determine the viability of applying evolutionary algorithms to engineering practice.
Knowledge:
  - Advanced knowledge of programming and software performance analysis.
  - Good knowledge of MATLAB programming language.
  - Good knowledge of statistics.
  - Good knowledge of algorithm complexity.
  - Basic knowledge of hardware performance.

Skills:
  - Good oral and writing communication skills.
  - Good time management.

- Engineer expert in machine learning algorithms (David Orive): This profile will apply evolutionary algorithms to improve machine learning models that are applied in industrial processes.
  Knowledge:
    - Advanced knowledge of machine learning.
    - Advanced knowledge of statistics.
    - Good knowledge aof machine learning algorithms applied to engineering.
    - Basic knowledge of software performance analysis.

  Skills:
    - Great capability to focus on the key aspects of the project.
    - Good oral and writing communication skills.
    - Good time management.
- Software developer (David Orive): He/she will implement the algorithms and the statistical analysis.
  Knowledge:
    - Advanced knowledge of sequential and object oriented programming.
    - Advanced knowledge of MATLAB programming language.
    - Advanced knowledge of algorithm complexity.
    - Advanced knowledge of Windows and Linux programming.
    - Good knowledge of software performance.

  Skills:
    - High working capacity.
    - Time management skills.
    - Great capacity to accept changes in the project.
    - Creative thinking.
- Maintenance technician (David Orive): He/she is responsible of setting-up hardware and software environment for the project.
  Knowledge:
    - Advanced knowledge of high-performance computers set-up and configuration.
    - Advanced knowledge of Windows and Linux environment configuration.
    - Advanced knowledge of computer communication protocols.

  Skills:
    - Time management skills.

5.3. WORK SCHEDULE

The project will start in September 9, 2013 and it is planned to finish in June 10, 2014. The workday will be of 6 hours starting at 8:00 and finishing at 14:00. Holidays, apart from Saturdays and Sundays, will take place on the following days:

- October 12, 2013
- October 25, 2013
- November 1, 2013
- December 6, 2013
- From December 23,2013 to January 7,2014
- March 19, 2014
- From April 17 to 25, 2014
- May 1 and 2,2014

5.4. WORKING PLAN

In Table 5.1 a reduced working plan, identification and predecessors were omitted, is shown. In Appendix D the whole working plan can be seen. Moreover, in Appendix E and F Gantt diagram and network diagram are shown respectively.

5.5. WORK ENVIRONMENT

The working place will be located at the Energy department of DeustoTech. Hardware and computer operative systems were provided by DeustoTech.

- HARDWARE
  - Dell Latitude E5500 laptop. It will be used for software development.
  - Two computers with four core and eight threads Intel Core i7 processors each one. They will be used for algorithm performance analysis.
  - One computer with two AMD Opteron 6168 of 1.9GHz processors with 12 cores each one. It will be used for complex algorithm performance analysis.
- SOFTWARE
  - Laptop operative system: Windows 8
  - High performance computers operative system: Gentoo Linux
  - Programming language: MATLAB programming language. Also, the following libraries will be used: Neural Networks Toolbox, Parallel Computing Toolbox and LibSVM.
  - Software for developing class diagrams: Visual Paradigm for UML 11.
  - Typesetting system: LaTeX using MiKTeX and TeXstudio as writing environment.
  - Software for remote control of computers: PuTTY, WinSCP and Windows desktop remote control.

Task name	Work(hours)	Duration(days)	Start	End
A.1	**129**	**166**	**mon 09/09/13**	**mon 02/06/14**
A.1.1	30	5	mon 09/09/13	fri 13/09/13
A.1.2	99	156	mon 23/09/13	mon 02/06/14
A.2	**18**	**3**	**mon 16/09/13**	**wed 18/09/13**
A.2.1	12	2	mon 16/09/13	tue 17/09/13
A.2.2	6	1	wed 18/09/13	wed 18/09/13
A.3	**69**	**13**	**thurs 19/09/13**	**mon 07/10/13**
A.3.1	27	5	thurs 19/09/13	wed 25/09/13
A.3.2	27	5	thurs 26/09/13	wed 02/10/13
A.3.3	15	3	thurs 03/10/13	mon 07/10/13
A.4	**429**	**80**	**tue 08/10/13**	**mon 17/02/14**
A.4.1	216	40	tue 08/10/13	wed 04/12/13
A.4.1.1	*108*	*20*	*tue 08/10/13*	*wed 06/11/13*
A.4.1.2	*108*	*20*	*thurs 07/11/13*	*wed 04/12/13*
A.4.2	213	40	thurs 05/12/13	mon 17/02/14
A.4.2.1	*108*	*20*	*thurs 05/12/13*	*mon 20/01/14*
A.4.2.2	*105*	*20*	*tue 21/01/14*	*mon 17/02/14*
A.5	**321**	**60**	**tue 18/02/14**	**mon 26/05/14**
A.5.1	27	5	tue 18/02/14	mon 24/02/14
A.5.2	27	5	tue 25/02/14	mon 03/03/14
A.5.3	240	45	tue 04/03/14	mon 19/05/14
A.5.3.1	*81*	*15*	*tue 04/03/14*	*tue 25/03/14*
A.5.3.2	*81*	*15*	*wed 26/03/14*	*tue 15/04/14*
A.5.3.3	*78*	*15*	*wed 16/04/14*	*mon 19/05/14*
A.5.4	27	5	tue 20/05/14	mon 26/05/14
A.6	**51**	**9**	**tue 27/05/14**	**fri 06/06/14**
A.6.1	51	9	tue 27/05/14	fri 06/06/14
A.7	**12**	**2**	**tue 09/06/14**	**wed 10/06/14**

Table 5.1: Reduced working plan

6. BUDGET

I show the human and software resources budget in Table 6.1 and 6.2 respectively. The add up of both budgets gives a total of:

TOTAL BUDGET = 87,456 + 9,259 = **96,715 euros**

Profile	Workload (hour)	Price(euros)	Cost(euros)
Project director	147	133	19,551
Mathematician	54	110	5,940
Software performance analyst	27	100	2,700
Learning algorithms engineer	243	100	24,300
Developer	585	59	34,515
Maintenance technician	18	25	450
Total	**1,074**		**87,456 euros**

Table 6.1: Human resources budget

Description	Units	Price (euros)	Cost (euros)
Matlab R2012b	3	2,000	6,000
Matlab Neural Network Toolbox	1	1,000	1,000
Matlab Parallel Computing Toolbox	1	1,000	1,000
Microsoft Office	1	270	270
Microsoft Project	1	769	769
Visual Paradigm for UML 11 (Standard)	1	220	220
		Total	**9,259 euros**

Table 6.2: Software resources budget

7. DEVELOPMENT

In this chapter I give all the information concerning the experimentation. Firstly, I describe all the details related to the experimental set-up. Then, in the second section, I present the results of the statistics analysis and in the next section I give an interpretation to these results.

7.1. EXPERIMENTAL SET-UP

To perform the experiments I used datasets explained in Section 4.2. KEEL regression datasets are publicly available on the Internet and I obtained BTO process data from experiments done in [23]. I artificially constructed synthetic datasets by taking into account the curse of dimensionality phenomenon that happens when the dimensionality of the problem increases and consequently the volume of the space rises exponentially making the data sparse in the space. In order to obtain accurate results the amount of data has also to increase exponentially according to the dimension, that is, the more number of inputs the bigger the training data has to be. For this reason the size of the synthetic dataset must be dependant on the number of inputs by the following relationship $m = 100^n$ where m is the number of instances in the training data and n the number of inputs. However, for space purposes I establish the $m = 100n$ relationship. As I stated in the data gathering section I added Gaussian distributed 0 mean and 0.01 standard deviation noise to the outputs of the training part of the dataset. I generated 100 synthetic datasets with X inputs where X is a random integer value between 1 and 6, both included.

I used Matlab programming language in combination with Parallel Computing Toolbox to allow paralleling when calculating the fitness of the individuals and Neural Network Toolbox and LibSVM libraries in order to use ANNs and SVMs respectively.

Regarding to evolutionary algorithms, I did 75 generations with 50 individuals in the population for each experiment. In all the experiments I used non-uniform mutation in company with a crossover. Related to the learning algorithms, in ANN I only used one hidden layer and I determined the number of neurons based on $h = 10n$ where h is the number of neurons in the hidden layer and n represents the number of inputs. I used sigmoid function as activation function. I performed the training of the network using Levenberg-Mardquardt backpropagation algorithm. I also used regression SVM, specifically ν-SVR, with the radial basis kernel function.

To validate that the prediction errors obtained by the learning algorithms represent the generalization capability for KEEL regression datasets I used cross-validation with 10-folds. For the BTO dataset I used 23-fold cross-validation where 23 where the number of experiments. In the case of synthetic dataset I performed holdout cross validation where the training data had noise and the test set represents the real function.

I used Nguyen-Widrow initialization method [49] as neural network random weight initialization. The main idea was to compare this initialization with a initialization based on EAs. In the case of SVM I initialized the hyperparameters using hard randomization. To calculate the prediction error of random hyperparameter initialization on an equal basis I executed the learning algorithms with random initialization n times (each time with different random weight values) being n equal to the

number of generations of the EA multiplied by the population size. So this way, it is assured that number of trials for both approaches are the same and they compete in equal terms.

The computers I used to make the experiments were the following:

- Computer A consisted on a four core and eight thread Intel Core processor with Gentoo Linux as operative system.
- Computer B was composed of a four core and eight thread Intel Core processor with Windows 8 as operative system.
- Computer C had two AMD Opteron 6168 of 1.9GHz processors with twelve cores each one using Gentoo Linux as operative system.

I calculated the validation errors by MAPE statistical error measure. The main reasons for this decision were that it is easy to interpret its meaning and the datasets had no zero values.

I used Friedman test statistical tool in order to measure if there were significant differences between the crossovers and random initialization. It has to be taken into account that the used implementation of Friedman algorithm ranked first (lower rankings) the algorithm that had the biggest error. So in this experimentation, where a lower prediction error is better then I considered better the algorithms with highest rankings.

7.2. EXPERIMENTAL RESULTS

In Table 7.1 I show the predictions errors for KEEL regression datasets using ANN, I summarised the data to the performance of best six crossovers and random weight initialization. I added the entire results in Table G.1 of Appendix G. In Table 7.2 I show the results obtained from doing Friedman test to the results of the previous table, the obtained p-value was $4.5 \cdot 10^{-9}$. Then, I give Hochberg's procedure results in Table 7.3. I also performed Bergmann procedure and it found that there are statistical significance levels between the following algorithms:

- Arithmetical crossover vs. Random
- Differential crossover vs. Random
- Uniform crossover vs. Random
- One point row crossover vs. Random
- Two point column crossover vs. Random

A note should be made about Bergmann's results, this algorithm time complexity is exponential and depending to the number of treatments that are tested it can spent a lot of time calculating the result. For that reason, I only tested the best five algorithms (arithmetic, differential, uniform, one point row and two point column) and the random initialization.

I show the MAPE prediction errors for synthetic datasets using ANN in Tables G.2, G.3, G.4 and G.5 in Appendix G. In Table 7.4 and 7.5 I show Friedman test and post-hoc analysis using Hochberg's procedure respectively. The p-value obtained by the Friedman test was $5.105 \cdot 10^{-11}$. Posteriorly, I did Bergmann's procedure with the best six crossovers (according to the error ranking), the results that I obtained for synthetic datasets with 3 or less inputs were the following:

- ArithmeticCrossover vs. Random
- DifferentialCrossover vs. Random
- UniformCrossover vs. Random

Dataset	AR	DIFF	U	2Pr	2Pc	S	Random
ANACALT.dat	1,99	1,94	2,04	2,02	2,07	1,95	2,10
abalone.dat	10,66	10,76	10,46	10,60	10,73	10,77	13,28
ailerons.dat	8,33	8,45	8,71	8,51	8,37	8,34	9,93
autoMPG6.dat	24,71	24,54	23,91	24,69	25,42	24,73	29,37
autoMPG8.dat	10,44	9,32	10,41	10,35	10,50	10,19	10,67
california.dat	134,73	152,16	133,93	148,02	122,36	152,73	264,35
compactiv.dat	14,27	15,18	14,33	14,56	14,41	14,58	15,90
concrete.dat	50,77	50,47	50,56	49,82	49,76	51,07	51,84
dee.dat	13,30	13,23	13,38	13,49	13,15	13,30	15,60
diabetes.dat	36,34	37,55	33,35	36,29	35,94	33,64	53,29
ele-1.dat	15,12	15,37	14,92	15,24	15,19	14,93	15,55
ele-2.dat	1,13	1,14	1,13	1,14	1,17	1,16	1,16
elevators.dat	92,65	81,66	92,44	91,57	92,40	93,43	94,36
forestFires.dat	7,90	8,10	8,04	8,12	8,09	8,09	8,21
friedman.dat	24,74	24,77	24,75	24,69	25,34	25,06	25,03
house.dat	2,40	2,39	2,41	2,41	2,40	2,41	2,42
laser.dat	2,44	2,48	2,50	2,46	2,53	2,64	2,85
machineCPU.dat	85,44	79,73	87,42	87,33	88,31	81,27	82,82
mortgage.dat	2,07	2,01	2,26	2,27	2,24	1,83	2,26
mv.dat	2,49	2,37	2,39	2,44	2,47	2,30	2,47
plastic.dat	1,58	1,53	1,60	1,65	1,65	1,51	1,67
quake.dat	8,54	8,79	8,39	8,40	8,42	8,48	10,36
stock.dat	8,81	8,67	8,43	8,30	8,20	8,53	11,23
tic.dat	50,27	54,17	52,68	52,02	56,02	51,67	76,20
treasury.dat	25,76	25,91	26,09	25,96	25,75	25,35	26,07
wankara.dat	2,62	2,60	2,65	2,63	2,61	2,61	2,66
wizmir.dat	20,53	21,05	20,02	20,74	20,20	20,09	22,45

Table 7.1: MAPE results of KEEL Datasets with the different crossover algorithms.

- SwarmCrossover vs. Random

And the results for synthetic datasets with inputs between 4 and 6 are:

- ArithmeticCrossover vs. DifferentialCrossover
- ArithmeticCrossover vs. Random
- DifferentialCrossover vs. UniformCrossover
- DifferentialCrossover vs. Random
- UniformCrossover vs. SwarmCrossover
- SwarmCrossover vs. Random

In Table 7.6 I show the results using swarm crossover and ANN applied to the BTO process. In Table 7.7 and I also present the results when using SVM as learning algorithm.

7.3. INTERPRETATIONS FROM THE EXPERIMENTS

Regarding to the data in Table 7.2 it can be seen that the random weight initialization is the treatment with worst error prediction and ranking. The p-value indicated by the Friedman test was $4.5 \cdot 10^{-9}$ and because I decided $\alpha = 0.01$ to be the significance level then there is a statistical significant difference between the different types of weights initialization. Then, I applied the Hochberg's

Algorithms	Friedman Ranking	Average Error Ranking
ArithmeticCrossover	6.11	24.45
SwarmCrossover	6.00	24.91
UniformCrossover	5.93	24.41
DifferentialCrossover	5.70	24.68
TwoPointRowCrossover	5.26	25.03
TwoPointColumnCrossover	5.26	24.28
OnePointRowCrossover	4.59	24.55
OnePointColumnCrossover	4.48	24.71
Random	1.67	31.63

Table 7.2: Friedman test applied to KEEL results

Null hypothesis	p-value	Significance level
AC = Random	$2.47 \cdot 10^{-9}$	0.0062
SC = Random	$6.1 \cdot 10^{-9}$	0.007
UC = Random	$1.1 \cdot 10^{-8}$	0.008
DC = Random	$6.08 \cdot 10^{-8}$	0.01
2Pr = Random	$1.43 \cdot 10^{-6}$	0.012
2Pc = Random	$1.43 \cdot 10^{-6}$	0.016
1Pr = Random	$8.65 \cdot 10^{-5}$	0.025
1Pc = Random	$1.59 \cdot 10^{-4}$	0.05

Table 7.3: Post-hoc analysis of the KEEL results

procedure post-hoc analysis by comparing random initialization against all the crossovers and it rejected all null hypothesis. So it seems that there are significant differences between the random initialization and the rest of crossover algorithms. The next step I took was to analyse if significance differences between the crossover algorithms exists. For that, I carried the Bergmann's procedure and it stated that no significant differences between different crossover algorithms exists, but it confirmed the previous hypothesis.

The Friedman test analysis for synthetic dataset also indicates that random initialization is the algorithm with worst error prediction and ranking. In relation with synthetic datasets results the p-value of $5.1 \cdot 10^{-11}$ compared with the significance level, $\alpha = 0.01$ indicates that there are also significant differences between the different approaches to initialize the weights of a ANN. The Hochberg's procedure also indicates that the null-hypothesises that stated that there are no significant differences between crossovers and random initialization are false. In order to test if differences exist between the crossovers I used again Bergmann's procedures. However, in this case before applying the Bergmann's procedure I divided the datasets in two groups according to the inputs that they had. Then according to the first group (inputs between 1 and 3) results there are significant differences between the random weight initialization and the rest of crossovers but not between the crossovers themselves. Further analysis of the second group (inputs between 4 and 6) suggest that as the number of inputs increases the uniform crossover starts to perform worst and separates from the rest of crossovers. So it seems that there could be differences between the crossovers when the dimensionality of the input space is high.

Regarding to the BTO process results, they show that there is an improvement when using EA for SVM and ANN hyperparameter optimization. Obviously, when the activity values are not predicted

Algorithms	Friedman Ranking	Average Error Ranking
DifferentialCrossover	6.38	14.99
SwarmCrossover	5.64	16.65
TwoPointRowCrossover	5.29	15.89
ArithmeticCrossover	5.23	16.46
OnePointRowCrossover	5.21	16.88
TwoPointColumnCrossover	4.99	16.93
OnePointColumnCrossover	4.86	16.54
UniformCrossover	4.78	16.84
Random	2.6	17.3

Table 7.4: Friedman test applied to synthetic data results

Null hypothesis	p-value	Significance level
DC = Random	$2.15 \cdot 10^{-22}$	0.0062
SC = Random	$5.14 \cdot 10^{-15}$	0.007
2Pr = Random	$4.52 \cdot 10^{-12}$	0.008
AC = Random	$1.33 \cdot 10^{-11}$	0.01
1Pr = Random	$1.9 \cdot 10^{-11}$	0.012
2Pc = Random	$7.98 \cdot 10^{-10}$	0.016
1Pr = Random	$6.26 \cdot 10^{-9}$	0.025
UC = Random	$1.81 \cdot 10^{-8}$	0.05

Table 7.5: Post-hoc analysis of the synthetic results

Error	Weights initialization with EA	Random weights initialization
MAPE with the activity obtained from the dataset	18.08	22.04
MAPE with the activity obtained from the activation equation	26.58	27.79

Table 7.6: MAPE results of BTO process using ANN

Error	Weights initialization with SVM	Random weights initialization
MAPE with the activity obtained from the activation equation	26.13	27.52

Table 7.7: MAPE results of BTO process using SVM

but obtained from the datasets the results are the best. However, more experiments with different ways of calculating the activity should be made.

Another aspect that should be taken into account is that to execute evolutionary algorithms, depending on the fitness function, can take some time. For example, for all KEEL regression datasets the algorithm spent five days tuning the hyperparameters. However, this drawback does not stop from using the benefits of tuning into real time applications because once the estimation

of optimal hyperparameters have been obtained they are applied into the learning algorithm and there is not need of changing them any more while the process does not change.

8. CONCLUSIONS AND FUTURE WORK

The aim of this project was to develop evolutionary algorithms for the tuning of machine learning algorithms hyperparameters. After carrying the experiments I reached several conclusions. The conclusions can be divided in two aspects: engineering and scientific. In the engineering aspect, I showed that evolutionary algorithms improve the performance of ANNs and SVMs designed for the modelling of the BTO process. Regarding to the scientific part, the first conclusion is that tuning of machine learning algorithms hyperparameters via evolutionary algorithms improves significantly the performance compared to a random initialization. Then I also demonstrated that there are no significant differences in the type of crossover used by the evolutionary algorithm internally when the number of inputs is not big enough. Moreover, I determined that optimization of ANN initial weights and biases improves their performance and should be subject of further analysis.

8.1. FUTURE WORK

The research presented in this project have raised many interesting possibilities to be performed. There are several lines of research arising from this work which should be pursued.

Firstly, research can be done in a more intelligent way by including the activation function of each neuron into the hyperparameters set. Connections between different layers and neurons can also be included allowing the possibility of disconnecting specific neuron sections if required to specialize some neurons of the network.

A second line of research, is to investigate the performance for other real-world problems. Electricity consumption forecasting by using symbolic regression is also an interesting topic of research. The main idea is to find the regression statistical function that best fits the process. This problem can be approached by using evolutionary algorithms where each individual could represent a regression function. The main problem to be solved is how to define the regression functions in order to be able to represent all the function space. In several papers trees structures have been proposed but further research has to be done.

Finally, more BTO experimentation should be done by making the learning algorithms to estimate the activity.

9. DEFINITIONS AND ACRONYMS

9.1. DEFINITIONS

The basic definitions of concepts that I used in the document are the following:

Artificial neural network: Machine learning algorithm inspired by animal's central nervous system.

BTO process: Chemical process that transforms bioethanol to olefins and has the biomass or other derivatives as reactants.

Hyperparameter: Parameters that are tuned by the user before the learning process starts.

Machine learning: It is a branch of artificial intelligence that concerns the construction and study of systems that can learn from data.

Mathematical optimization: Selection of the best element from some set of available alternatives.

Regression: Statistical process for estimating the relationships between independent and dependent variables.

Support vector machine: Machine learning algorithm based on statistical learning concepts.

9.2. ACRONYMS

The meaning of the acronyms that I used during the document are the following:

SVM: Support Vector Machine

ANN: Artificial Neural Networks

BTO: Bioethanol To Olefins

UML: Unified Modelling Language

EA: Evolutionary Algorithm

AC: Arithmetic Crossover

DC: Differential Crossover

UC: Uniform Crossover

1Pc: One Point Column Crossover

1Pr: One Point Row Crossover

2Pc: Two Point Column Crossover

2Pr: Two Point Row Crossover

SC: Swarm Crossover

Bibliography

[1] M. Abd and G. Paschos. “Effective Arabic Character Recognition Using Support Vector Machines”. In: *Innovations and Advanced Techniques in Computer and Information Sciences and Engineering*. Springer Netherlands, 2007.

[2] H. Akaike. “A new look at the statistical model identification”. In: *Automatic Control, IEEE Transactions on* 19.6 (1974), pages 716–723.

[3] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García. “KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework.” In: *Multiple-Valued Logic and Soft Computing* 17.2-3 (2011), pages 255–287.

[4] R. Bardenet, M. Brendel, B. Kégl, and M. Sebag. “Collaborative hyperparameter tuning”. In: *30th International Conference on Machine Learning, ICML 2013*. 2. 2013, pages 858–866.

[5] Y. Benjamini and Y. Hochberg. “Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing”. In: *Journal of the Royal Statistical Society Series B (Methodological)* 57.1 (1995), pages 289–300.

[6] B. Bergmann and G. Hommel. “Improvements of General Multiple Test Procedures for Redundant Systems of Hypotheses”. In: *Multiple Hypothesenprüfung / Multiple Hypotheses Testing*. Volume 70. Medizinische Informatik und Statistik. Springer Berlin Heidelberg, 1988, pages 100–115.

[7] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. “Algorithms for hyper-parameter optimization”. In: *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*. 2011.

[8] N. V. Bhat, Peter A. Minderman Jr., T. McAvoy, and N. S. Wang. “Modeling chemical process systems via neural computation”. In: *IEEE Control Systems Magazine* 10.3 (1990), pages 24–30.

[9] A. Bhattacharyya, R. Faría-González, and I. Ham. “Regression Analysis for Predicting Surface Finish and Its Application in the Determination of Optimum Machining Conditions”. In: *Journal of Manufacturing Science and Engineering* 92 (3 1970).

[10] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[11] P.M. Blau and O.D. Duncan. *The American Occupational Structure*. Wiley, 1967.

[12] D. M. Cherba, W. Punch, P. Duxbury, S. Billinge, and P. Juhas. “Conformation of an Ideal Bucky Ball Molecule by Genetic Algorithm and Geometric Constraint from Pair Distance Data: Genetic Algorithm”. In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2005.

[13] C.-I Chou, Y. ling Chu, and S.-P. Li. “Evolutionary Strategy for Political Districting Problem Using Genetic Algorithm”. In: *International Conference on Computational Science (4)*. Springer, 2007.

[14] G. Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of Control, Signals, and Systems (MCSS)* 2.4 (1989), pages 303–314.

[15] C. Darwin. *On the Origin of Species by Means of Natural Selection*. Murray, 1859.

[16] L. Deng and X. Li. “Machine learning paradigms for speech recognition: An overview”. In: *IEEE Transactions on Audio, Speech and Language Processing* 21.5 (2013), pages 1060–1089.

[17] 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”. In: *Swarm and Evolutionary Computation* 1.1 (2011), pages 3–18.

[18] S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro. “A genetic algorithm to configure support vector machines for predicting fault-prone components”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6759 LNCS (2011), pages 247–261.

[19] H. Drucker, D. Wu, and V. Vapnik. “Support Vector Machines for Spam Categorization”. In: *IEEE Transactions on Neural Networks* 10.5 (1999).

[20] F. Friedrichs and C. Igel. “Evolutionary tuning of multiple SVM parameters”. In: *Neurocomputing* 64.1-4 (2005), pages 107–117.

[21] F. Galton. “Kinship and Correlation”. In: *Statistical Science* 4 (1989), pages 81–86.

[22] A. G. Gayubo, A. Alonso, B. Valle, A. T. Aguayo, and J. Bilbao. “Selective production of olefins from bioethanol on HZSM-5 zeolite catalysts treated with NaOH”. In: *Applied Catalysis B: Environmental* 97.1–2 (2010), pages 299–306.

[23] A.G. Gayubo, A. Alonso, B. Valle, A.T. Aguayo, M. Olazar, and J. Bilbao. “Kinetic modelling for the transformation of bioethanol into olefins on a hydrothermally stable Ni-HZSM-5 catalyst considering the deactivation by coke”. In: *Chemical Engineering Journal* 167.1 (2011), pages 262–277.

[24] G. Guo, S. Z. Li, and K. Chan. “Face Recognition by Support Vector Machines”. In: *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*. FG '00. IEEE Computer Society, 2000.

[25] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. “Gene Selection for Cancer Classification Using Support Vector Machines”. In: *Mach. Learn.* (2002).

[26] S. Haykin. *Neural Networks and Learning Machines (3rd Edition)*. Prentice Hall, 2008.

[27] V. Hlavác. *Vapnik-Chervonenkis learning theory*. `http://cmp.felk.cvut.cz/~hlavac/TeachPresEn/31PattRecog/27VapnikChervonenkis.pdf`. [Online; accessed 27-June-2014].

[28] J. H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[29] K. Hornik. “Approximation Capabilities of Multilayer Feedforward Networks”. In: *Neural Netw.* 4.2 (Mar. 1991), pages 251–257.

[30] C. L. Huang and C. J. Wang. “A GA-based feature selection and parameters optimizationfor support vector machines”. In: *Expert Systems with Applications* 31.2 (2006), pages 231–240.

[31] R. J. Hyndman and A. B. Koehler. “Another look at measures of forecast accuracy”. In: *International Journal of Forecasting* 22.4 (2006), pages 679–688.

[32] S. Jamal and V. Scaria. “Cheminformatic models based on machine learning for pyruvate kinase inhibitors of Leishmania mexicana”. In: *BMC Bioinformatics* 14.1 (2013).

[33] T. Joachims. “Text Categorization with Suport Vector Machines: Learning with Many Relevant Features”. In: *Proceedings of the 10th European Conference on Machine Learning*. Springer-Verlag, 1998.

[34] J. D. Jobson. “A Multivariate Linear Regression Test for the Arbitrage Pricing Theory”. In: *The Journal of Finance* 37 (1982).

[35] A. Khosla, Y. Cao, C. C.-Y. Lin, H.-K. Chiu, J. Hu, and H. Lee. “An Integrated Machine Learning Approach to Stroke Prediction”. In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2010, pages 183–192.

[36] K. J. Kim. “Financial time series forecasting using support vector machines”. In: *Neurocomputing* 55.1–2 (2003), pages 307–319.

[37] I. Kononenko. “Machine Learning for Medical Diagnosis: History, State of the Art and Perspective”. In: *Artif. Intell. Med.* 23.1 (Aug. 2001), pages 89–109.

[38] E. Y. Li. “Artificial neural networks and their business applications.” In: *Information and Management* 27.5 (1994), pages 303–313.

[39] R. Mac Nally. “Multiple regression and inference in ecology and conservation biology: further comments on identifying important predictor variables”. In: *Biodiversity and Conservation* 11 (2002).

[40] Mathworks. *Gradient descent visualization*. `http://www.mathworks.com/matlabcentral/fileexchange/35389-gradient-descent-visualization`. [Online; accessed 24-June-2014].

[41] MATLAB. *Bayesian regulation backpropagation*. `http://www.mathworks.es/es/help/nnet/ref/trainbr.html`. [Online; accessed 15-February-2014].

[42] MATLAB. *Levenberg-Marquardt backpropagation*. `http://www.mathworks.es/es/help/nnet/ref/trainlm.html`. [Online;accesed 15-February-2014].

[43] John McCarthy. “Programs with Common Sense”. In: *Semantic Information Processing*. MIT Press, 1968, pages 403–418.

[44] Merovingian. *What are Hommel Hochberg corrections?* `http://stats.stackexchange.com/questions/71466/what-are-hommel-hochberg-corrections`. [Online; accessed 24-June-2014].

[45] J. Milgram, M. Cheriet, and R. Sabourin. “"One Against One" or "One Against All": Which One is Better for Handwriting Recognition with SVMs?” In: *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft, Oct. 2006.

[46] T. M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

[47] N. Murata, S. Yoshizawa, and S. ichi Amari. “Network information criterion-determining the number of hidden units for an artificial neural network model.” In: *IEEE Transactions on Neural Networks* 5.6 (1994), pages 865–872.

[48] K. S. Narendra and K. Parthasarathy. “Identification and control of dynamical systems using neural networks”. In: *IEEE Transactions on Neural Networks* 1.1 (1990), pages 4–27.

[49] Derrick Nguyen and Bernard Widrow. “Improving the Learning Speed of 2-layer Neural Networks by Choosing”. In: *Initial Values of the Adaptive Weights, International Joint Conference of Neural Networks*. 1990, pages 21–26.

[50] Marios M. P. and Petros A. I. *Identification and Control of Nonlinear Systems Using Neural Network Models: Design and Stability Analysis*. Technical report. ELECTRICAL ENGINEERING—SYSTEMS REP, 1991.

[51] G. Panchal, A. Ganatra, Y.P.Kosta, and D. Panchal. “Searching Most Efficient Neural Network Architecture Using Akaike's Information Criterion (AIC)”. In: *International Journal of Computer Applications* 1.5 (2010), pages 41–44.

[52] N. Pinto, D. Doukhan, J.J. DiCarlo, and D.D. Cox. “A high-throughput screening approach to discovering good forms of biologically inspired visual representation”. In: *PLoS Computational Biology* 5.11 (2009).

[53] J. Reyes, A. Morales-Esteban, and F. MartíNez-ÁLvarez. “Neural Networks to Predict Earthquakes in Chile”. In: *Appl. Soft Comput.* (), pages 1314–1328.

[54] R. Rifkin and A. Klautau. “In Defense of One-Vs-All Classification”. In: *J. Mach. Learn. Res.* 5 (2004), pages 101–141.

[55] J.L. Rojo-Álvarez, G. Camps-Valls, M. Martínez-Ramón, E. Soria-Olivas, A. Navia-Vázquez, and A.R. Figueiras-Vidal. “Support vector machines framework for linear signal processing”. In: *Signal Processing* 85.12 (2005), pages 2316–2326.

[56] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. “New Support Vector Algorithms”. In: *Neural Comput.* 12.5 (May 2000).

[57] N. Sebe, I. Cohen, A. Garg, and T. S. Huang. *Machine Learning in Computer Vision (Computational Imaging and Vision)*. Springer-Verlag New York, Inc., 2005.

[58] M. Sewell. *Structural Risk Minimization*. `http://www.svms.org/srm/srm.pdf`. [Online; accessed 27-June-2014]. 2008.

[59] G. Sorrosal. “Modelado del reactor químico del proceso BTO mediante redes neuronales artificiales”. In: *Universidad Pública del País Vasco* (2013).

[60] C. Souza. *Kernel Functions for Machine Learning Applications*. `http://crsouza.blogspot.com.es/2010/03/kernel-functions-for-machine-learning.html`. [Online; accessed 10-May-2014].

[61] M. Stone. “An Asymptotic Equivalence of Choice of Model by Cross-Validation and Akaike's Criterion”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1 (1977), pages 44–47.

[62] M. Stone. “Cross-Validatory Choice and Assessment of Statistical Predictions”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 36.2 (1974), pages 111–147.

[63] *Support Vector Machine Regression*. `http://kernelsvm.tripod.com/`. [Online; accessed 15-May-2014].

[64] *SVM - Support Vector Machines*. `http://www.dtreg.com/svm.htm`. [Online; accessed 12-May-2014].

[65] R. R. Trippi and E. Turban, editors. *Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance*. McGraw-Hill, Inc., 1992.

[66] A. Tzotsos and D. Argialas. “Support Vector Machine Classification for Object-Based Image Analysis”. In: *Object-Based Image Analysis*. Springer Berlin Heidelberg, 2008, pages 663–677.

[67] L. Waltman, N.J. van Eck, R. Dekker, and U. Kaymak. “Economic modeling using evolutionary algorithms: The effect of a binary encoding of strategies”. In: *Journal of Evolutionary Economics* 21.5 (2011), pages 737–756.

[68] *Why would someone use gradient descent for a convex function?* `http://www.quora.com/Mathematical-Optimization/Why-would-someone-use-gradient-descent-for-a-convex-function`. [Online; accessed 2-May-2014].

[69] B. Widrow, D. E. Rumelhart, and M. A. Lehr. “Neural Networks: Applications in Industry, Business and Science”. In: *Commun. ACM* 37.3 (Mar. 1994), pages 93–105.

[70] Z. R. Yang. “Biological applications of support vector machines”. In: *Briefings in Bioinformatics* 5.4 (2004), pages 328–338.

A. LEAST SQUARES METHOD

Least squares method is a common method for adjusting the parameters of regression models. The meaning of least squares comes from how the method minimizes the sum of the squares of the errors made by each point in the dataset. As it can be seen in Figure A.1 r_i indicates the error of the model for each data point. As not distinction is made from positive or negative errors then each data point is squared. Then, the total error of the model is given by the sum of squares errors $S = \sum_{i=1}^{n} r_i^2$ where r_i is the error between the target output and the prediction, $r_i = y_i - f(x_i, \vec{\beta})$.

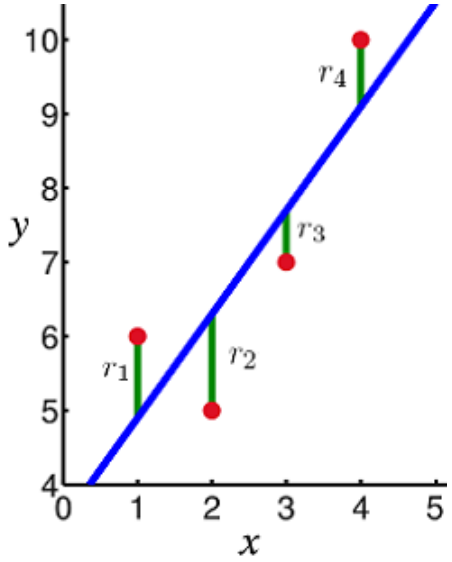


Figure A.1: Function graph indicating its global maximum, minimum and local maxima and minima

Usually, the main objective is to minimize the error of the summation so the minimum of the sum of squares is found by setting the gradient to zero. Since the model contains m parameters then the gradient contains m partial derivatives:

$$\frac{\partial S}{\partial \beta_j} = 2\sum_i r_i \frac{\partial r_i}{\partial \beta_j} = 0, j = 1, \cdots, m$$

and since $r_i = y_i - f(x_i, \vec{\beta})$ the gradient equations become:

$$-2\sum_i r_i \frac{\partial f(x_i, \vec{\beta})}{\partial \beta_j} = 0, j = 1, \cdots, m$$

A.1. EXAMPLE USING LINEAR REGRESSION

A dataset with data points of the form (x, y) wants to be modelled by the line $y = \beta_1 + \beta_2 x$. The dataset has the following point: $\{(2, 3), (3, 5), (6, 7), (7, 10)\}$. In other words, the values β_1 and β_2 that minimizes the error of the following equations wants to be found: $\beta_1 + \beta_2 2 = 3$, $\beta_1 + \beta_2 3 = 5$,

$\beta_1 + \beta_2 6 = 7$ and $\beta_1 + \beta_2 7 = 10$. The error function of this linear system can be defined as the sum of squares of the error $e = y - (\beta_1 + \beta_2 x)$ so:

$$\begin{aligned} \underset{\beta_1,\beta_2}{minimize} S(\beta_1, \beta_2) &= [3 - (\beta_1 + 2\beta_2)]^2 + [5 - (\beta_1 + 3\beta_2)]^2 + [7 - (\beta_1 + 6\beta_2)]^2 + [10 - (\beta_1 + 7\beta_2)]^2 \\ &= 4\beta_1^2 + 98\beta_2^2 + 36\beta_1\beta_2 - 50\beta_1 - 266\beta_2 + 183 \end{aligned}$$

This function can be minimized by setting its gradient to zero,

$$\nabla S(\beta_1, \beta_2) = \left[\frac{\partial S}{\partial \beta_1}, \frac{\partial S}{\partial \beta_2}\right]^T = \vec{0}$$

where $\partial S/\partial \beta_1 = 8\beta_1 + 36\beta_2 - 50$ and $\partial S/\partial \beta_2 = 36\beta_1 + 196\beta_2 - 266$. This operation results in a simple system of two linear equations with two unknowns:

$$\begin{cases} 8\beta_1 + 36\beta_2 - 50 = 0 \\ 36\beta_1 + 196\beta_2 - 266 = 0 \end{cases}$$

The result of the system is $\beta_1 = 0,823$ and $\beta_2 = 1,205$. Then the line that best fits the data is defined as: $y = 1,205x + 0,823$.

B. GRADIENT DESCENT

Gradient descent method assures to find a local minimum for any function. In the case of convex functions, such as the error function of a feedforward neural network, it is assured to find the global minimum. Gradient descent presupposes that the function is differentiable.

The algorithm work as seen in Algorithm 4. The α should be a small value greater than zero. A small value of α will make the algorithm to do small jumps and it will be stable. With good α conditions then it is guaranteed that $f(x_{k+1}) \leq f(x_k)$. In Figure B.1 a simple example of gradient descent behaviour for a convex function can be seen.

Algorithm 4: Gradient descent algorithm

The algorithm selects a random point in the function input space, x_0 ;
repeat
  Calculate $x_{k+1} = x_k - \alpha \nabla f(x_k)$;
until $x_{k+1} = x_k$;

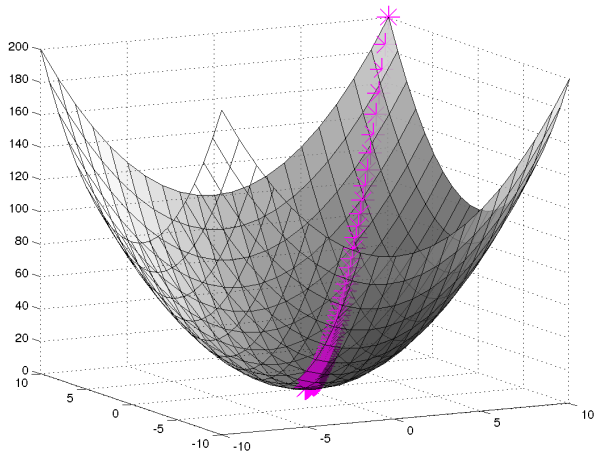


Figure B.1: Path taken by gradient descent in a convex function. Taken from [40]

C. PUBLICATIONS

Evolutionary algorithms for hyperparameter tuning on Neural Networks models

Gorka Sorrosal (a), David Orive (b), Cruz E. Borges(c), Cristina Martín(d), Ainhoa Alonso(e)

(a) (b) (c) (d) (e)DeustoTech Energy, University of Deusto, Bilbao Basque Country, Spain

(a) gsorrosal@deusto.es, (b) david.orive@opendeusto.es, (c) cruz.borges@deusto.es (d) cristina.andonegui@deusto.es (e) ainhoa.alonso@deusto.es

The 26th European Modeling and Simulation Symposium

Abstract

In this work the hyperparameter tuning problem of some soft computing techniques is studied. It is proposed an evolutionary algorithm approach to initialize the Artificial Neural Networks initial weights that improve neural models results. The obtained results have been analyzed by the Friedman test to detect significant differences between the tested algorithms and the actual random weight initialization. In this paper it is demonstrated that the tuning of neural network initial weights improves significantly their performances compared to a random initialization. In addition, different crossover algorithms have been tested in order to identify the best one for the present objective. However, the Friedman test determines that there is no significant difference between algorithms.

Keywords: Evolutionary algorithm, Artificial Neural Networks, hyperparameter tuning

1. Introduction

The hyperparameter tuning of soft computing modelling techniques, such as Artificial Neural Networks (ANN) or Support Vector Machines (SVM), is an important step in the intelligent models development which is usually done manually, requiring expert experience, and reduced to a few number of trials. Fortunately, the development of computing capacity allows the use of some methods and algorithms to automate the procediment such as Estimation of Distribution Algorithms (EDA) (Nannen and Eiben 2007), local search based algorithms (Hutter and Leyton-brown 2009) or sequential hyper-parameter optimization algorithms (Bergstra et al. 2011).

In the case of neural networks, the tuning of their hyperparameters, such as layers, number of neurons, activation functions or recurrencies, has been widely researched (Murata, Yoshizawa, and Amari 1994; Hagiwara, Toda, and Usui 1993; Panchal et al. 2010). Nonetheless ANN initial weights quite often are not tuned with the above mentioned hyperparameters but are only initialized by training functions. Frequently used training functions like Levenberg-Marquart backpropagation (Marquardt 1963; Hagan and Menhaj 1994) or Bayesian Regularization backpropagation (MacKay 1992), use random weight initializations. However, as final results are dependent of the neural network initial weights, optimization of these parameters should be considered as a technique for training functions performance improvement. In this paper is demonstrated that the tuning of initial

weights improves the performance of neural networks.

2. Hyperparameter tuning algorithms

In the past, hyperparameters tuning was done by hand because of the low computational power available (Bardenet et al. 2013). Presently, computer clusters and GPU processor allow to run more trials and to do them faster. In addition, in other areas of knowledge such as image processing, it was proved that state of the art of image classification could be improved not only by building new algorithms but also by tuning the hyperparameters of existing techniques (Pinto et al. 2009; Bergstra et al. 2011).

The most simple but not efficient technique is the direct search throughout the operating range. Because it is impossible to test in every possible conditions, normally it is used random samples or grid search strategies. In the first case, the operation points to test and analyze are randomly selected in the operation range, while in the second option, a grid of operating points covering the whole operation range is defined and iteratively refined over the hyperparameter space. In the Figure C.1 is shown an example of a grid search of two parameters done for a Support Vector Machine model, where x-axe denote modification of c hyperparamenter, y-axe modification of γ hyperparameter and z-axe represent the validation error.

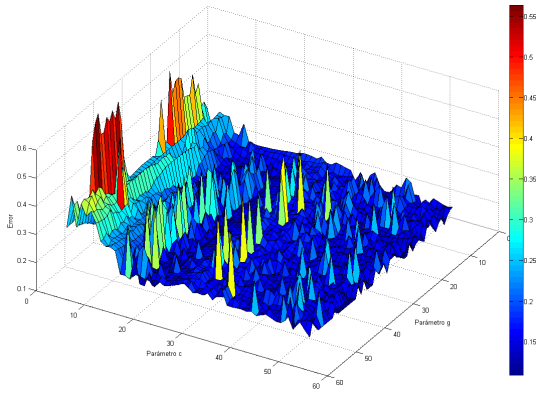


Figure C.1: Grid Search Example.

However, the evolutionary algorithms are a more efficient and intelligent tuning strategy. Based on biological evolution, these soft computing techniques allow a selective exploration of the operation range (Holland 1975). In hyperparameter tuning problems, evolutionary algorithms have been proved to perform far better than grid search according to the accuracy-speed ratio (Martino et al. 2011; Friedrichs and Igel 2005; Huang and Wang 2006). The facts presented above open the possibility of improving actual machine learning techniques.

3. Evolutionary algorithm

Evolutionary algorithms were used for tuning neural networks hyperparameters. In the Algorithm 5 a pseudocode of the used evolutionary algorithm can be seen.

In this work crossover and mutation genetic operators were used. In each sequence a crossover or mutation genetic operator is randomly used. The experiments were focused on eight types of crossovers operators:

Arithmetical crossover (AC): Being P_1 and P_2 two individuals and a random variable λ uniformly

Algorithm 5: Pseudocode of the Evolutionary Algorithm.

```
Use Genesis Operator to create a population of randomly generated individuals;
Apply Fitness Operator to calculate the fitness of each individual;
while A criteria is not satisfied do
    repeat
        Use Selection Operator to select the parents;
        Apply a Genetic Operator to generate offsprings;
        With Reproduction Operator select between the parents and the offsprings and add them to
        the new population;
    until A new population is created;
    Apply the Elite Operator to maintain the best individual;
end
```

distributed between [0,1], the two offsprings are defined as:

$$C_1 = \lambda P_1 + (1-\lambda)P_2 \text{ and } C_2 = \lambda P_2 + (1-\lambda)P_1$$

where λP_x means that the vector/matrix of values of the individual P_x is multiplied by λ. Then C_x is the result of a sum of vectors/matrices.

Differential crossover (DC): Being P_1, P_2 and P_3 three individuals and a random variable λ uniformly distributed between $[0, 1]$, the crossover is defined as $C_x = P_y + \lambda(P_z – P_w)$ where $\{y, z, w\}$ are all possible permutations of the set $\{1, 2, 3\}$.

Uniform crossover (UC): Given two individuals P_1 and P_2 with L attributes, L binomial random variables λ_i with probability 0.5 are flip. C_1 is made with the features from P_1 such that λ_i are 0 and the features from P_2 such that λ_i are 1. C_2 is made switching P_1 with P_2.

One point columns crossover (1Pc): Given two individuals P_1 and P_2 with a matrix of size $m \times n$ as attribute and a randomly chosen cross point $t \in (1, n)$ we define the one point crossover over columns as:

$$C_1 = [P_{1(1,m)\times(1,t)}, P_{2(1,m)\times(t+1,n)}] \text{ and } C_2 = [P_{2(1,m)\times(1,t)}, P_{1(1,m)\times(t+1,n)}]$$

where C_1 and C_2 are the offsprings generated by the operator.

One point rows crossover (1Pr): Given two individuals P_1 and P_2 as a matrix of size $m \times n$ as attribute and a randomly chosen cross point $t \in (1, m)$ we define the one point crossover over rows as:

$$C_1 = [P_{1(1,t)\times(1,n)}, P_{2(t+1,m)\times(1,n)}] \text{ and } C_2 = [P_{2(1,t)\times(1,n)}, P_{1(t+1,m)\times(1,n)}]$$

where C_1 and C_2 are the offsprings generated by the operator.

Two point columns/rows crossover (2Pc/2Pr): It is a modified one point crossover algorithm with two cross points that generates the corresponding resultant combinations of offsprings.

Swarm crossover (SC): For each P_i particle of the swarm P_i^e would be denoted as the best fitness obtained by particle i. Moreover, P^e will represent the best qualification obtained by the

swarm. Then the crossover could be defined as $C = \lambda_1 P_i + \lambda_2(P_i^e - P_i) + \lambda_3(P^e - P_i)$ where λ_1, λ_2 and λ_3 are random real numbers uniformly distributed over the range $[0, 1]$.

During the experiments the rest of operators were kept constant, using Round Robin as selection operator, n-different as reproduction operator and best candidate as elite operator.

4. Experimental setup

The proposed method has been validated on three different type of dataset: the KEEL regression dataset, a synthetic dataset made by random sampling the space of neural networks and data of the concentrations over the time of several chemical species in a catalytic reaction.

The KEEL-dataset repository was created with the aim of providing to researches a set of benchmarks to analyze the behavior of their learning algorithms. There are benchmarks for classification, regression and unsupervised learning (Alcalá-Fdez et al. 2011). In this research, KEEL regression datasets were used. The synthetic datasets were created by using feedforward Random Neural Networks (RNN). A RNN is defined as a neural network with the number of inputs, outputs, layers and neurons selected randomly within an interval. Several inputs are given to the RNN and the generated outputs are recorded as a dataset. Using this technique infinite datasets can be obtained representing each RNN a different process to model. In order to avoid over-fitting normally distributed noise is added to the outputs of the training set.

The last dataset used, has been one composed by the Bioethanol To Olefins (BTO) process experimental results obtained by (Gayubo et al. 2010) in a laboratory scale reactor. The BTO process consists on the catalytic transformation of bioethanol, as substitute raw material to oil, into olefins, commodities for the petrochemical industry. The modeling objective is the olefins production rate curves at different operation conditions. These datasets have been divided in two parts: the training set and the validation set. In order to determine the performance of the developed models, a cross validation technique named Leave-One-Out Cross Validation (LOOCV) has been used. In this technique the validation part changes iteratively, moving through the whole dataset. Training each time with the training part and validating the models with the part that has been excluded from the training. The validation errors have been measured by statistical calculations such as the Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) defined by equations:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(F_i - Y_i)^2}$$

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

5. Experimental results

In Table C.1 the MAPE errors of the predictions obtained for KEEL regression datasets can be seen, where the results of the neural network without hyperparameter tuning tend to be worst. In Table C.2 the mean errors of each crossover tested with 100 different synthetic datasets using RMSE measure are given. The MAPE and RMSE measures of the BTO process using a NARX structure neural network are shown in Table C.3. These results have been analyzed by the Friedman test, which is a non-parametric statistical test that is used to detect differences between several

algorithms (Derrac et al. 2011). In Friedman test the null hypothesis H_0 states that the outcomes of each algorithm have the same distribution of probability, namely that all algorithms perform equally. If the result of the Friedman test is positive (namely, the p-value is inferior to the desired significance level $\alpha = 0.05$) post hoc analysis is needed to determine between which pair of algorithm the differences exists.

Dataset	AR	DIFF	U	2Pr	2Pc	S	Random
ANACALT.dat	1,99	1,94	2,04	2,02	2,07	1,95	2,10
abalone.dat	10,66	10,76	10,46	10,60	10,73	10,77	13,28
ailerons.dat	8,33	8,45	8,71	8,51	8,37	8,34	9,93
autoMPG6.dat	24,71	24,54	23,91	24,69	25,42	24,73	29,37
autoMPG8.dat	10,44	9,32	10,41	10,35	10,50	10,19	10,67
california.dat	134,73	152,16	133,93	148,02	122,36	152,73	264,35
compactiv.dat	14,27	15,18	14,33	14,56	14,41	14,58	15,90
concrete.dat	50,77	50,47	50,56	49,82	49,76	51,07	51,84
dee.dat	13,30	13,23	13,38	13,49	13,15	13,30	15,60
diabetes.dat	36,34	37,55	33,35	36,29	35,94	33,64	53,29
ele-1.dat	15,12	15,37	14,92	15,24	15,19	14,93	15,55
ele-2.dat	1,13	1,14	1,13	1,14	1,17	1,16	1,16
elevators.dat	92,65	81,66	92,44	91,57	92,40	93,43	94,36
forestFires.dat	7,90	8,10	8,04	8,12	8,09	8,09	8,21
friedman.dat	24,74	24,77	24,75	24,69	25,34	25,06	25,03
house.dat	2,40	2,39	2,41	2,41	2,40	2,41	2,42
laser.dat	2,44	2,48	2,50	2,46	2,53	2,64	2,85
machineCPU.dat	85,44	79,73	87,42	87,33	88,31	81,27	82,82
mortgage.dat	2,07	2,01	2,26	2,27	2,24	1,83	2,26
mv.dat	2,49	2,37	2,39	2,44	2,47	2,30	2,47
plastic.dat	1,58	1,53	1,60	1,65	1,65	1,51	1,67
quake.dat	8,54	8,79	8,39	8,40	8,42	8,48	10,36
stock.dat	8,81	8,67	8,43	8,30	8,20	8,53	11,23
tic.dat	50,27	54,17	52,68	52,02	56,02	51,67	76,20
treasury.dat	25,76	25,91	26,09	25,96	25,75	25,35	26,07
wankara.dat	2,62	2,60	2,65	2,63	2,61	2,61	2,66
wizmir.dat	20,53	21,05	20,02	20,74	20,20	20,09	22,45

Table C.1: KEEL Dataset MAPE Results with the Different Crossover Algorithms.

Error	AR	DIFF	U	1Pr	1Pc	2Pr	2Pc	S
RMSE	13,6601	9,8318	15,1253	16,462	17,8816	17,5901	42,2664	11,6117

Table C.2: Synthetic Dataset Results with the Different Crossover Algorithms.

Error	Weights initialization with EA	Random weights initialization
RMSE	0.0299	0.0344
MAPE(%)	18.0857	22.042

Table C.3: BTO Process Modelling Dataset Results.

In this test, given a matrix M with m rows (groups) and n columns (algorithms) and being M_{ij} the element of the matrix in the i^{th} row and j^{th} column, each element of the data is ranked within their own group. If in the same groups there are tie values then the average rank of the group without

Null hypothesis	p-value	Significance level
AC = R	$2.47 \cdot 10^{-9}$	0.0062
SC = R	$6.1 \cdot 10^{-9}$	0.007
UC = R	$1.1 \cdot 10^{-8}$	0.008
DC = R	$6.08 \cdot 10^{-8}$	0.01
2Pr = R	$1.43 \cdot 10^{-6}$	0.012
2Pc = R	$1.43 \cdot 10^{-6}$	0.016
1Pr = R	$8.65 \cdot 10^{-5}$	0.025
1Pc = R	$1.59 \cdot 10^{-4}$	0.05

Table C.4: Post Hoc Analysis Done for the Different Crossover Algorithms of KEEL datasets

Null hypothesis	p-value	Significance level
DC = R	$2.15 \cdot 10^{-22}$	0.0062
SC = R	$5.14 \cdot 10^{-15}$	0.007
2Pr = R	$4.52 \cdot 10^{-12}$	0.008
AC = R	$1.33 \cdot 10^{-11}$	0.01
1Pr = R	$1.9 \cdot 10^{-11}$	0.012
2Pc = R	$7.98 \cdot 10^{-10}$	0.016
1Pr = R	$6.26 \cdot 10^{-9}$	0.025
UC = R	$1.81 \cdot 10^{-8}$	0.05

Table C.5: Post Hoc Analysis Done for the Different Crossover Algorithms of syntethic datasets

those values is calculated and assigned to tied values. Next, a new matrix $R_{m \times n}$ is created were R_{ij} is the rank of the M_{ij} element. The ranking of each algorithm is calculated as:

$$r_n = \frac{1}{m} \sum_{i=1}^{m} R_{in}$$

Where r_n denotes the ranking of the n algorithm.

The p-value of the Friedman test carried on over the KEEL-dataset is $4.5 \cdot 10^{-9}$ so we can conclude that there are statistical differences between the different proposed algorithm.

The results of post hoc analysis over the KEEL and synthetic datasets can be seen in Table 4 and Table 5, where the Hochberg's procedure rejects those hypotheses that have a p-value equal or minor than 0.05. As can be seen, it seems that there are only significant differences between the random initialization and the rest of the algorithms, but not between them. In order to test that last hypothesis the Friedman Test Analysis without the random initialization algorithm was redone. In this case the Bergmann's procedure (Bergmann and Hommel 1988) has been carried on. The results state that there are no significant differences between the different crossing algorithms.

6. Conclusions and future works

In this paper it is demonstrated that the tuning of neural network initial weights improves significantly their performances compared to a random initialization. However, there are no significant differences in the type of crossover used by the evolutionary algorithm internally.

As a future work it is necessary to improve the neural networks hyperparameter tuning, in order to research in a more intelligent selection of each neuron activation function and connections between

different layers and neurons, disconnecting specific neuron sections if required.

Acknowledgments

This work has been carried out with the financial support of the Basque Government via the PI2013-40 project and a researchers training grant (COL-2013-1-135).

References

Alcalá-Fdez, J., A. Fernandez, J. Luengo, J. Derrac, S. Garcia, L. Sánchez, and F. Herrera. 2011. KEEL Data-Mining Software Tool : Data Set Repository , Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing 17 (2-3): 255–287.

Bardenet, R., M. Brendel, B. Kégl, and M. Sebag. 2013. Collaborative Hyperparameter Tuning. In International Conference on Machine Learning. Vol. 28. Bergmann, B., and G. Hommel. 1988. Improvements of General Multiple Test Procedures for Reduntant Systems of Hypotheses. Medizinische Informatik Und Statistik 70: 100–115.

Bergstra, James, R. Bardenet, Y. Bengio, and B. Kégl. 2011. Algorithms for Hyper-Parameter Optimization. Advances in Neural Information Processing Systems (NIPS) 24: 1–9.

Derrac, Joaquín, Salvador García, Daniel Molina, and Francisco Herrera. 2011. A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms. Swarm and Evolutionary Computation 1 (1) (March): 3–18.

Friedrichs, Frauke, and Christian Igel. 2005. Evolutionary Tuning of Multiple SVM Parameters. Neurocomputing 64 (March): 107–117.

Gayubo, Ana G., Ainhoa Alonso, Beatriz Valle, Andrés. T. Aguayo, and Javier Bilbao. 2010. Selective Production of Olefins from Bioethanol on HZSM-5 Zeolite Catalysts Treated with NaOH. Applied Catalysis B: Environmental 97 (1-2) (June 9): 299–306.

Hagan, Martin T, and Mohammad B Menhaj. 1994. Training Feedforward Networks with the Marquardt Algorithm. IEEE Transactions on Neural Networks 5 (6): 989–993.

Hagiwara, Katsuyuki, Naohiro Toda, and Shiro Usui. 1993. On the Problem of Applying AIC to Determine the Structure of a Layered Feed-Forward Neural Network. In International Joint Conference on Neural Networks, 2263–2266. Holland, J.H. 1975. Adaptation in Natural and Artificial Systems. Edited by University of Michigan Press.

Huang, Cheng-Lung, and Chieh-Jen Wang. 2006. A GA-Based Feature Selection and Parameters Optimizationfor Support Vector Machines. Expert Systems with Applications 31 (2) (August): 231–240.

Hutter, Frank, and Kevin Leyton-brown. 2009. ParamILS : An Automatic Algorithm Configuration Framework. Journal of Artificial Intelligence Research 36: 267–306.

MacKay, David J. C. 1992. Bayesian Interpolation. Neural Computation 4 (3) (May): 415–447.

C. PUBLICATIONS

Marquardt, Donald W. 1963. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. Journal of the Society for Industrial and Applied Mathematics 11 (2): 431–441.

Martino, Sergio Di, Filomena Ferrucci, Carmine Gravino, and Federica Sarro. 2011. A Genetic Algorithm to Configure Support Vector Machines for Predicting Fault-Prone Components. Lecture Notes in Computer Science: 247–261.

Murata, Noboru, Shuji Yoshizawa, and Shun-ichi Amari. 1994. Network Information Criterion-Determining the Number of Hidden Units for an Artificial Neural Network Model. IEEE Transactions on Neural Networks 5 (6): 865–872.

Nannen, Volker, and A E Eiben. 2007. Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters. In International Joint Conference on Artificial Intelligence, 975–980.

Panchal, Gaurang, Amit Ganatra, Y.P. Kosta, and Devyani Panchal. 2010. Searching Most Efficient Neural Network Architecture Using Akaike 's Information Criterion (AIC). International Journal of Computer Applications 1 (5): 41–44.

Pinto, Nicolas, David Doukhan, James J DiCarlo, and David D Cox. 2009. A High-Throughput Screening Approach to Discovering Good Forms of Biologically Inspired Visual Representation. PLoS Computational Biology 5 (11) (November): 1–12.

D. WORKING PLAN

Identification	Task name	Work(hours)	Duration(days)	Start	End	Predecessors	Human resource
1	**A.1**	**129**	**166**	**mon 09/09/13**	**mon 02/06/14**		Project director
2	A.1.1	30	5	mon 09/09/13	fri 13/09/13		
3	A.1.2	99	156	mon 23/09/13	mon 02/06/14	2	
37	**A.2**	**18**	**3**	**mon 16/09/13**	**wed 18/09/13**		
38	A.2.1	12	2	mon 16/09/13	tue 17/09/13	2	Maintenance technician
39	A.2.2	6	1	wed 18/09/13	wed 18/09/13	38	Maintenance technician
40	**A.3**	**69**	**13**	**thurs 19/09/13**	**mon 07/10/13**		
41	A.3.1	27	5	thurs 19/09/13	wed 25/09/13	2	Mathematician
42	A.3.2	27	5	thurs 26/09/13	wed 02/10/13	41	Developer
43	A.3.3	15	3	thurs 03/10/13	mon 07/10/13	42	Developer
44	**A.4**	**429**	**80**	**tue 08/10/13**	**mon 17/02/14**		
45	A.4.1	216	40	tue 08/10/13	wed 04/12/13		
46	*A.4.1.1*	*108*	*20*	*tue 08/10/13*	*wed 06/11/13*	*39;43*	Developer
47	*A.4.1.2*	*108*	*20*	*thurs 07/11/13*	*wed 04/12/13*	*46*	Developer
48	A.4.2	213	40	thurs 05/12/13	mon 17/02/14		
49	*A.4.2.1*	*108*	*20*	*thurs 05/12/13*	*mon 20/01/14*	*47*	Developer
50	*A.4.2.2*	*105*	*20*	*tue 21/01/14*	*mon 17/02/14*	*47*	Developer
51	**A.5**	**321**	**60**	**tue 18/02/14**	**mon 26/05/14**		
52	A.5.1	27	5	tue 18/02/14	mon 24/02/14	50	Mathematician
53	A.5.2	27	5	tue 25/02/14	mon 03/03/14	52	Software performance analyst
54	A.5.3	240	45	tue 04/03/14	mon 19/05/14		
55	*A.5.3.1*	*81*	*15*	*tue 04/03/14*	*tue 25/03/14*	*53*	Engineer expert in MLA
56	*A.5.3.2*	*81*	*15*	*wed 26/03/14*	*tue 15/04/14*	*53*	Engineer expert in MLA
57	*A.5.3.3*	*78*	*15*	*wed 16/04/14*	*mon 19/05/14*	*53*	Engineer expert in MLA
58	A.5.4	27	5	tue 20/05/14	mon 26/05/14	55;56;57	
59	**A.6**	**51**	**9**	**tue 27/05/14**	**fri 06/06/14**		
60	A.6.1	51	9	tue 27/05/14	fri 06/06/14	58	Developer
61	**A.7**	**12**	**2**	**tue 09/06/14**	**wed 10/06/14**	3;60	Project director

Table D.1: Complete working plan

E. GANTT DIAGRAM

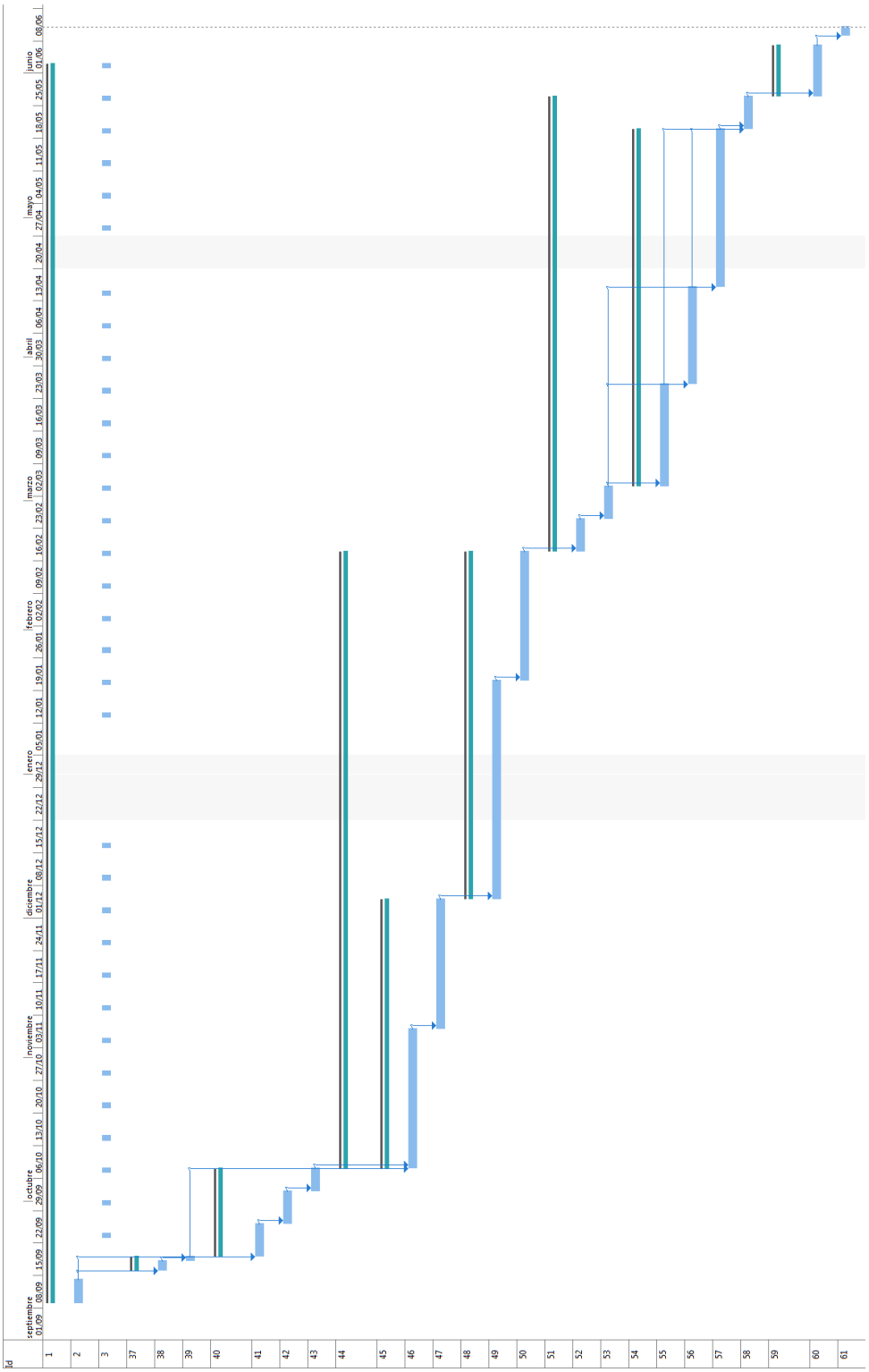


Figure E.1: Planning Gantt diagram

F. NETWORK DIAGRAM

The critical path is composed of the following tasks: A.1.1, A.3.1, A.3.2, A.3.3, A.4.1.1, A.4.1.2, A.4.2.1, A.4.2.2, A.5.1, A.5.2, A.5.3.1, A.5.3.2, A.5.3.3, A.5.4, A.6.1 and A.7.

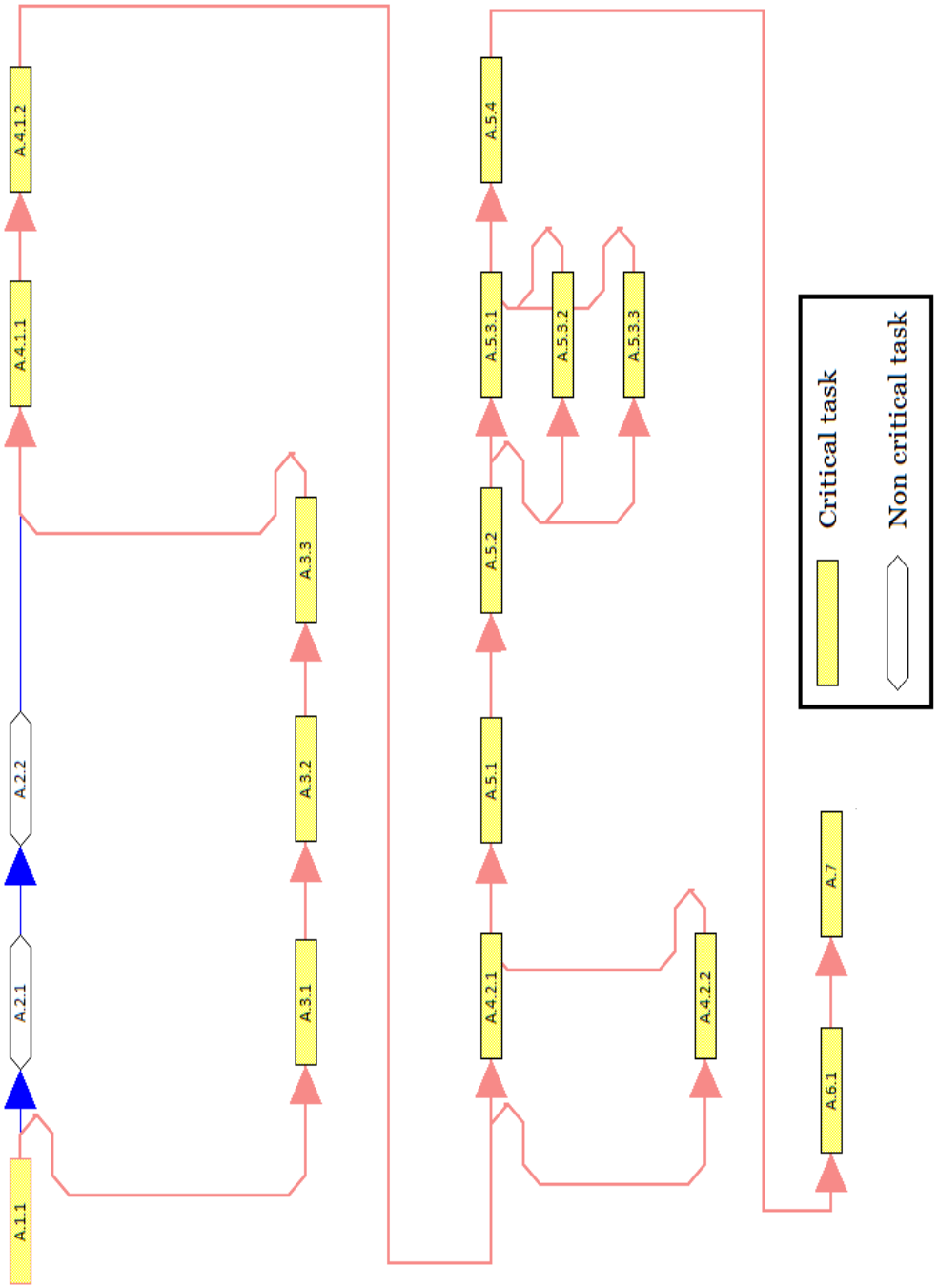


Figure F.1: Planning network diagram

G. RESULTS DATA

Dataset	AR	DIFF	U	1Pc	1Pr	2Pr	2Pc	S	Random
ANACALT.dat	1.99	1.94	2.04	2.06	2.06	2.02	2.07	1.95	2.10
abalone.dat	10.66	10.76	10.46	10.87	10.52	10.60	10.73	10.77	13.28
ailerons.dat	8.33	8.45	8.71	8.59	8.38	8.51	8.37	8.34	9.93
autoMPG6.dat	24.71	24.54	23.91	25.01	25.14	24.69	25.42	24.73	29.37
autoMPG8.dat	10.44	9.32	10.41	10.74	10.83	10.35	10.50	10.19	10.67
california.dat	134.73	152.16	133.93	131.68	131.14	148.02	122.36	152.73	264.35
compactiv.dat	14.27	15.18	14.33	14.38	14.81	14.56	14.41	14.58	15.90
concrete.dat	50.77	50.47	50.56	51.38	49.23	49.82	49.76	51.07	51.84
dee.dat	13.30	13.23	13.38	13.40	14.05	13.49	13.15	13.30	15.60
diabetes.dat	36.34	37.55	33.35	32.81	37.11	36.29	35.94	33.64	53.29
ele-1.dat	15.12	15.37	14.92	15.05	15.24	15.24	15.19	14.93	15.55
ele-2.dat	1.13	1.14	1.13	1.14	1.17	1.14	1.17	1.16	1.16
elevators.dat	92.65	81.66	92.44	94.12	93.22	91.57	92.40	93.43	94.36
forestFires.dat	7.90	8.10	8.04	8.06	8.14	8.12	8.09	8.09	8.21
friedman.dat	24.74	24.77	24.75	24.75	25.10	24.69	25.34	25.06	25.03
house.dat	2.40	2.39	2.41	2.41	2.40	2.41	2.40	2.41	2.42
laser.dat	2.44	2.48	2.50	2.51	2.48	2.46	2.53	2.64	2.85
machineCPU.dat	85.44	79.73	87.42	87.20	87.87	87.33	88.31	81.27	82.82
mortgage.dat	2.07	2.01	2.26	2.25	2.26	2.27	2.24	1.83	2.26
mv.dat	2.49	2.37	2.39	2.48	2.46	2.44	2.47	2.30	2.47
plastic.dat	1.58	1.53	1.60	1.65	1.63	1.65	1.65	1.51	1.67
quake.dat	8.54	8.79	8.39	8.30	8.54	8.40	8.42	8.48	10.36
stock.dat	8.81	8.67	8.43	8.64	8.42	8.30	8.20	8.53	11.23
tic.dat	50.27	54.17	52.68	54.67	57.73	52.02	56.02	51.67	76.20
treasury.dat	25.76	25.91	26.09	25.76	25.88	25.96	25.75	25.35	26.07
wankara.dat	2.62	2.60	2.65	2.64	2.63	2.63	2.61	2.61	2.66
wizmir.dat	20.53	21.05	20.02	20.30	18.71	20.74	20.20	20.09	22.45

Table G.1: MAPE results of KEEL Datasets with all crossover algorithms

Dataset	AR	DIFF	U	1Pc	1Pr	2Pr	2Pc	S	Random	No. inputs
Dataset1	148,76	134,93	131,41	120,61	152,38	120,15	140,14	162,02	125,54	6
Dataset2	25,13	20,98	39,10	39,76	39,94	37,11	36,87	28,14	34,04	5
Dataset3	3,50	3,90	3,53	3,50	3,45	3,22	3,31	3,80	3,37	4
Dataset4	2,23	2,19	2,14	2,21	2,25	2,20	2,10	2,26	2,37	6
Dataset5	2,48	1,98	2,85	2,80	2,88	2,85	2,57	2,00	2,80	6
Dataset6	0,36	0,43	0,33	0,54	0,53	0,40	0,52	0,41	1,49	1
Dataset7	0,55	0,49	0,52	0,49	0,50	0,52	0,50	0,54	0,64	2
Dataset8	0,11	0,09	0,09	0,06	0,07	0,06	0,09	0,08	0,31	1
Dataset9	64,02	77,18	77,48	76,22	72,71	73,03	87,52	75,20	69,01	6
Dataset10	36,57	33,62	38,46	38,48	40,26	38,11	36,67	38,09	42,72	4
Dataset11	3,93	3,76	3,96	3,88	3,88	3,76	3,58	3,94	3,93	4
Dataset12	0,20	0,20	0,27	0,23	0,20	0,26	0,24	0,10	0,39	1
Dataset13	0,30	0,35	0,30	0,28	0,29	0,31	0,32	0,32	0,71	2
Dataset14	12,30	9,29	12,27	12,05	12,39	11,80	12,40	12,72	12,56	4
Dataset15	8,22	7,18	8,24	8,56	8,80	8,98	8,55	7,64	9,25	5
Dataset16	0,44	0,53	0,40	0,47	0,40	0,47	0,49	0,49	0,76	2
Dataset17	5,74	2,78	5,63	6,64	7,18	7,05	6,46	3,43	4,40	4
Dataset18	1,30	1,06	1,19	1,25	1,29	1,36	1,19	1,45	1,62	2
Dataset19	13,39	14,45	14,94	13,87	14,31	14,51	14,27	13,43	17,58	5
Dataset20	0,32	0,35	0,32	0,34	0,40	0,34	0,45	0,33	0,79	1
Dataset21	0,26	0,25	0,27	0,26	0,25	0,24	0,28	0,27	0,52	2
Dataset22	13,61	14,41	16,54	16,94	14,39	16,27	16,26	11,05	14,84	5
Dataset23	23,55	24,00	26,09	25,26	21,73	26,24	24,72	25,16	29,19	5
Dataset24	13,65	9,48	12,73	12,30	12,51	12,33	13,72	12,11	14,76	3
Dataset25	0,05	0,06	0,04	0,04	0,06	0,06	0,06	0,06	0,35	1

Table G.2: MAPE results of synthetic datasets with the all crossover algorithms. Part 1

Dataset	AR	DIFF	U	1Pc	1Pr	2Pr	2Pc	S	Random	No. inputs
Dataset26	2,30	2,38	2,36	2,25	2,12	2,31	2,28	2,43	3,01	4
Dataset27	1,33	1,25	1,32	1,43	1,29	1,41	1,13	1,35	1,52	3
Dataset28	0,14	0,18	0,21	0,18	0,19	0,21	0,17	0,20	0,31	1
Dataset29	65,29	58,26	64,81	65,14	63,17	76,20	67,91	64,70	71,95	5
Dataset30	0,14	0,16	0,21	0,17	0,22	0,13	0,17	0,21	0,42	1
Dataset31	2,43	1,97	2,31	2,28	2,55	2,15	2,30	1,79	2,60	3
Dataset32	0,28	0,10	0,30	0,29	0,28	0,27	0,28	0,27	0,41	3
Dataset33	1,07	1,01	1,05	0,84	0,88	0,98	1,04	0,95	2,36	2
Dataset34	0,26	0,28	0,37	0,39	0,39	0,33	0,35	0,40	1,08	1
Dataset35	0,33	0,36	0,31	0,32	0,33	0,33	0,28	0,32	0,26	1
Dataset36	2,36	2,15	2,31	2,45	2,43	2,36	2,49	1,94	2,71	4
Dataset37	0,57	0,53	0,59	0,56	0,52	0,53	0,54	0,55	0,98	2
Dataset38	0,41	0,31	0,30	0,39	0,34	0,31	0,35	0,41	0,71	1
Dataset39	13,14	19,34	19,04	17,73	16,96	19,74	14,45	16,92	25,87	4
Dataset40	35,62	41,19	40,46	41,29	35,36	38,01	41,58	37,25	41,34	6
Dataset41	53,16	46,95	47,18	43,31	48,38	51,69	54,79	55,41	49,70	6
Dataset42	22,85	22,73	24,63	23,73	24,53	23,68	23,54	20,34	24,29	6
Dataset43	1,63	0,92	1,74	1,75	1,78	1,72	1,84	1,16	1,94	3
Dataset44	43,88	47,73	49,36	44,21	46,54	44,86	53,17	41,96	50,11	5
Dataset45	41,64	35,97	42,01	41,71	43,14	41,16	40,37	37,29	40,19	6
Dataset46	9,48	8,44	9,25	9,20	9,78	8,83	9,28	9,17	10,38	6
Dataset47	90,56	83,05	88,89	93,27	87,95	86,63	90,32	90,30	85,34	6
Dataset48	17,23	18,54	15,52	12,37	15,17	17,20	15,92	15,06	19,00	3
Dataset49	0,84	0,77	0,81	0,80	0,70	0,75	0,74	0,82	1,15	2
Dataset50	3,91	3,75	4,01	3,62	3,62	3,76	3,80	3,61	4,30	4

Table G.3: MAPE results of synthetic datasets with the all crossover algorithms. Part 2

Dataset	AR	DIFF	U	1Pc	1Pr	2Pr	2Pc	S	Random	No. inputs
Dataset51	1,78	0,95	1,59	1,77	1,74	1,80	1,71	1,41	1,88	3
Dataset52	1,35	1,25	0,98	1,31	1,26	1,19	1,43	1,30	2,32	1
Dataset53	1,36	1,01	2,46	2,64	2,90	2,85	2,86	2,09	3,70	2
Dataset54	11,60	8,70	11,80	10,42	11,37	11,79	11,04	11,21	10,99	4
Dataset55	10,45	7,07	12,14	12,65	13,17	11,61	12,92	9,25	12,67	4
Dataset56	14,37	9,14	15,10	11,04	11,48	13,66	13,33	14,28	12,37	3
Dataset57	3,12	2,09	1,89	2,69	3,04	2,76	3,09	2,73	2,61	2
Dataset58	51,44	49,52	46,95	55,93	53,04	43,02	40,72	44,62	51,88	4
Dataset59	5,49	3,48	4,96	3,57	4,70	4,49	4,47	3,09	4,52	3
Dataset60	3,57	4,41	3,34	3,87	3,47	4,07	3,97	4,05	4,69	3
Dataset61	15,77	15,44	14,63	20,34	18,72	18,98	19,43	17,54	17,76	4
Dataset62	5,12	4,48	5,79	5,60	5,51	5,84	5,04	5,29	10,14	3
Dataset63	0,11	0,13	0,07	0,08	0,14	0,11	0,11	0,12	0,31	1
Dataset64	2,66	2,28	3,06	3,08	3,22	3,06	3,00	2,39	3,13	6
Dataset65	0,43	0,50	0,61	0,45	0,45	0,54	0,64	0,53	2,14	1
Dataset66	30,16	23,94	31,48	29,38	31,07	26,59	32,21	25,74	29,56	3
Dataset67	6,97	6,59	8,95	8,85	8,93	9,58	8,55	6,25	8,78	5
Dataset68	117,87	111,85	106,34	137,73	94,53	109,60	91,78	119,28	119,66	5
Dataset69	124,79	82,99	138,93	124,60	126,65	120,25	130,18	107,99	119,12	6
Dataset70	9,70	10,16	13,29	13,05	13,42	13,24	12,88	9,55	13,83	6
Dataset71	0,59	0,60	0,44	0,60	0,55	0,45	0,56	0,51	0,64	2
Dataset72	17,40	17,63	16,86	14,64	16,89	15,87	18,34	18,82	16,76	3
Dataset73	28,28	27,35	27,21	26,50	25,16	25,76	27,41	26,98	29,82	4
Dataset74	0,17	0,15	0,18	0,17	0,16	0,17	0,16	0,17	0,32	1
Dataset75	11,68	13,14	12,60	10,35	13,90	12,39	12,03	10,04	13,80	2

Table G.4: MAPE results of synthetic datasets with the all crossover algorithms. Part 3

Dataset	AR	DIFF	U	1Pc	1Pr	2Pr	2Pc	S	Random	No. inputs
Dataset76	0,11	0,12	0,08	0,08	0,09	0,10	0,08	0,11	0,53	1
Dataset77	9,61	7,12	11,54	11,10	11,16	11,62	11,34	8,30	8,65	6
Dataset78	0,87	0,65	0,84	0,73	0,44	0,47	0,67	0,72	2,12	1
Dataset79	0,10	0,11	0,07	0,09	0,10	0,10	0,10	0,07	0,35	1
Dataset80	27,21	25,65	21,77	27,80	27,94	27,72	24,29	27,29	24,98	5
Dataset81	148,83	118,12	146,05	139,87	115,42	99,58	146,80	178,39	152,45	6
Dataset82	29,23	18,09	38,19	40,11	40,20	33,39	39,83	33,41	35,72	5
Dataset83	3,61	3,13	3,43	3,46	3,72	3,51	3,41	3,51	4,10	4
Dataset84	2,15	2,17	2,07	2,07	2,23	2,14	2,10	2,20	2,40	6
Dataset85	2,44	1,88	2,68	2,77	2,79	2,64	2,80	1,95	2,64	6
Dataset86	0,37	0,52	0,55	0,40	0,50	0,32	0,41	0,41	1,51	1
Dataset87	0,50	0,51	0,49	0,50	0,52	0,53	0,51	0,53	0,75	2
Dataset88	0,07	0,09	0,07	0,08	0,06	0,07	0,07	0,11	0,40	1
Dataset89	77,56	71,01	67,77	76,22	79,47	56,41	74,56	74,53	74,77	6
Dataset90	30,16	39,94	39,55	36,69	37,09	36,60	39,29	38,86	40,10	4
Dataset91	3,64	4,00	3,82	3,76	3,83	3,71	3,56	3,97	3,89	4
Dataset92	0,26	0,18	0,28	0,14	0,19	0,19	0,29	0,21	0,36	1
Dataset93	0,35	0,29	0,27	0,34	0,34	0,30	0,27	0,31	0,75	2
Dataset94	12,24	11,76	11,93	12,31	10,86	11,25	11,24	9,78	11,57	4
Dataset95	7,84	7,55	9,19	8,40	8,76	8,97	8,72	7,47	8,91	5
Dataset96	0,49	0,47	0,52	0,46	0,46	0,49	0,47	0,47	0,81	2
Dataset97	4,58	2,82	6,72	6,19	6,26	6,29	7,08	3,35	4,34	4
Dataset98	1,35	1,33	1,04	1,33	0,92	1,27	1,07	1,12	1,49	2
Dataset99	14,50	14,60	14,73	14,63	12,81	14,96	13,47	13,42	16,57	5
Dataset100	0,41	0,31	0,38	0,39	0,36	0,31	0,46	0,32	0,69	1

Table G.5: MAPE results of synthetic datasets with the all crossover algorithms. Part 4

H. SOFTWARE DOCUMENTATION

The aim of this software documentation is to provide detailed information about all the classes so the user can use the software easily and even implement his/her own classes by inheriting the existing ones. Moreover, I give some simple examples in order to show how the platform of evolutionary algorithms should be used. The whole documentation can be seen in the following web page: `http://energia.deusto.es:3000/davidpfg/`

H.1. CLASS DIAGRAMS

In the section class diagrams I give all the evolutionary algorithms operators so they can be understood more intuitively.

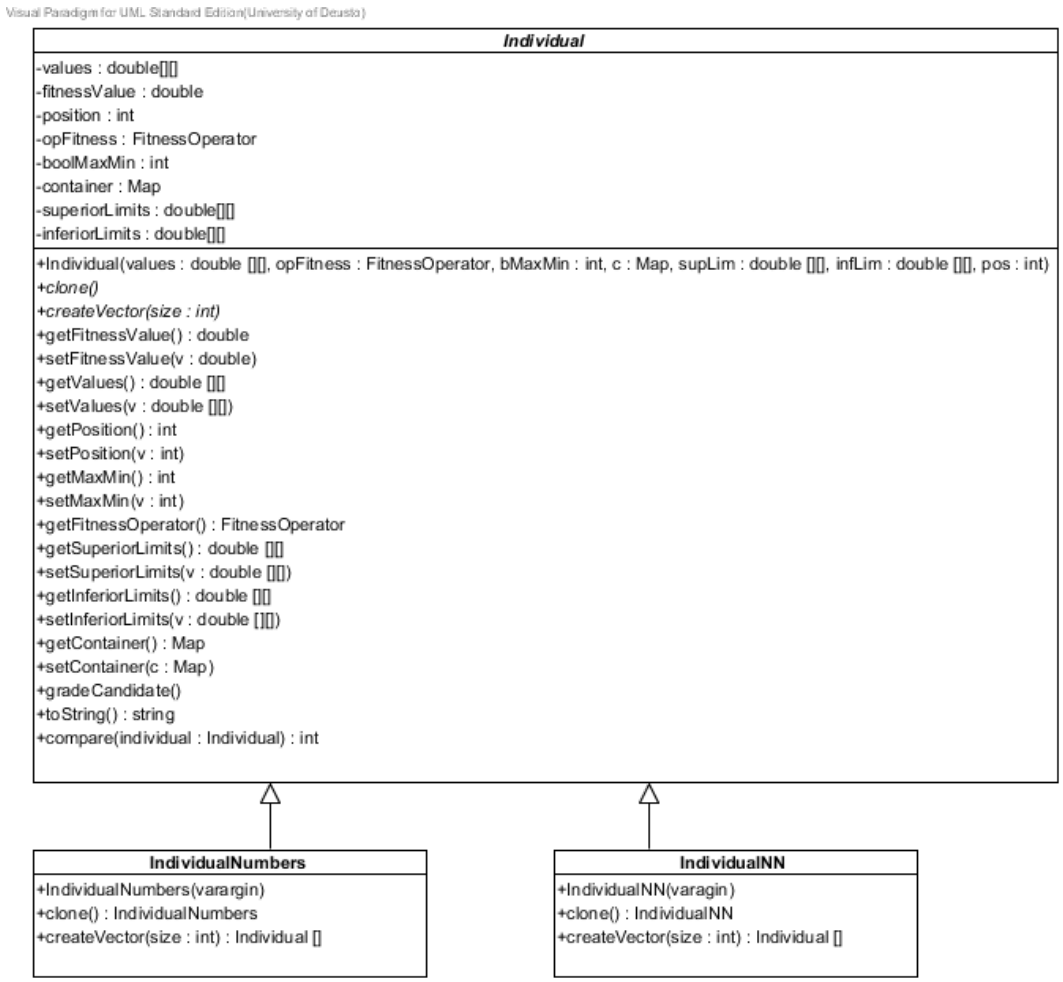


Figure H.1: Individual class diagram

H. SOFTWARE DOCUMENTATION

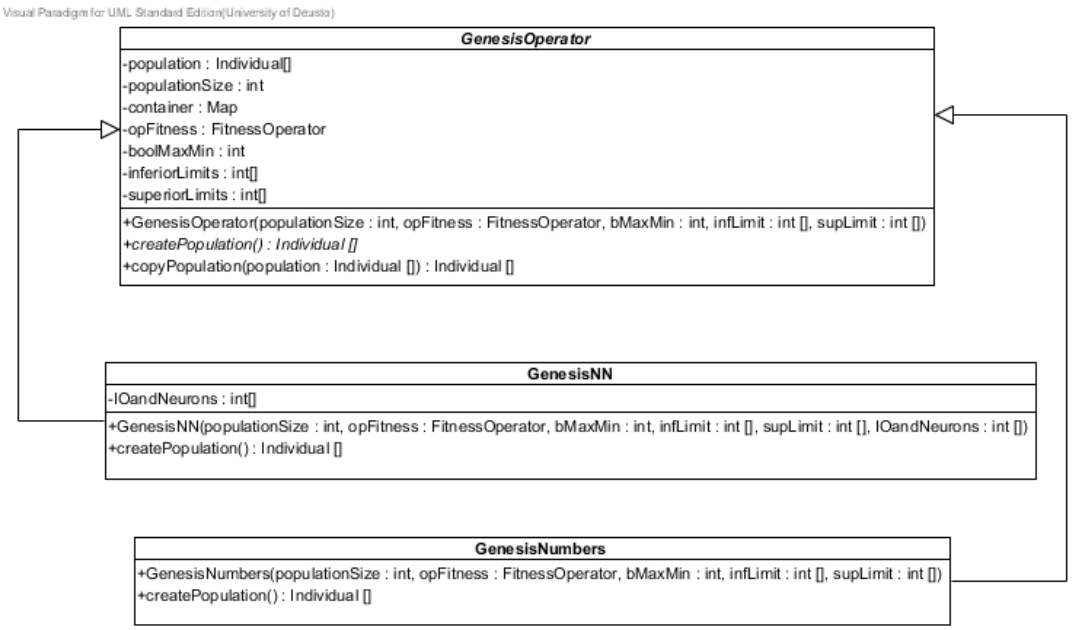


Figure H.2: Genesis class diagram

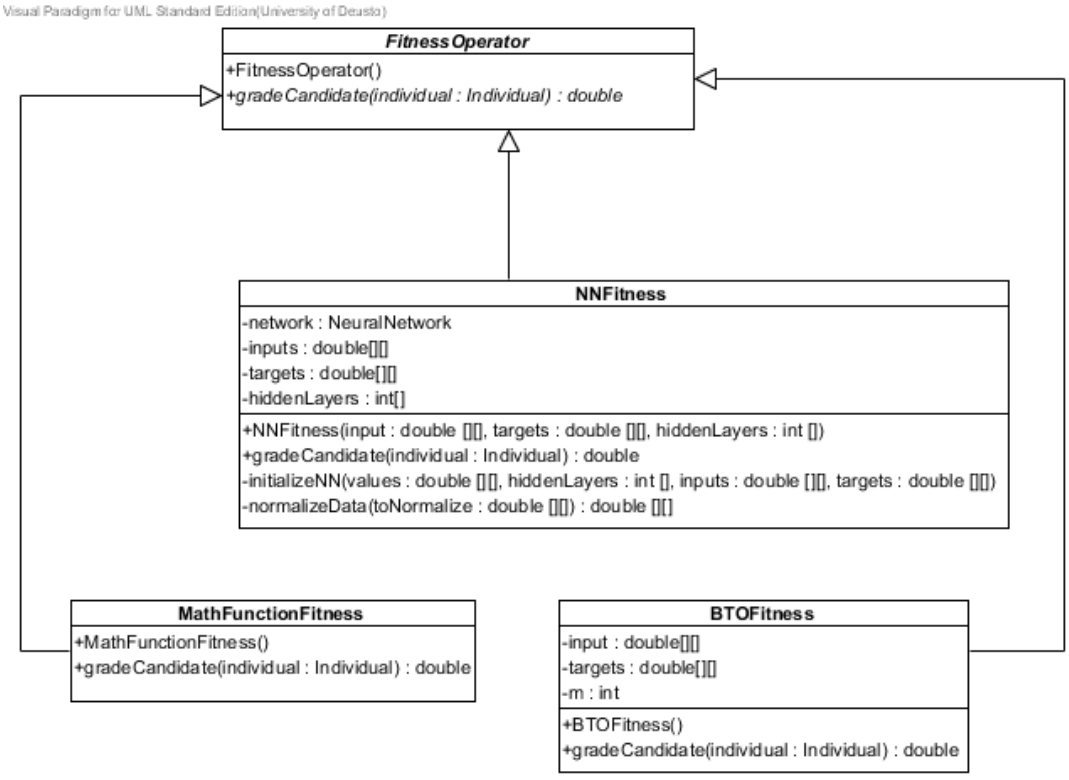


Figure H.3: Fitness class diagram

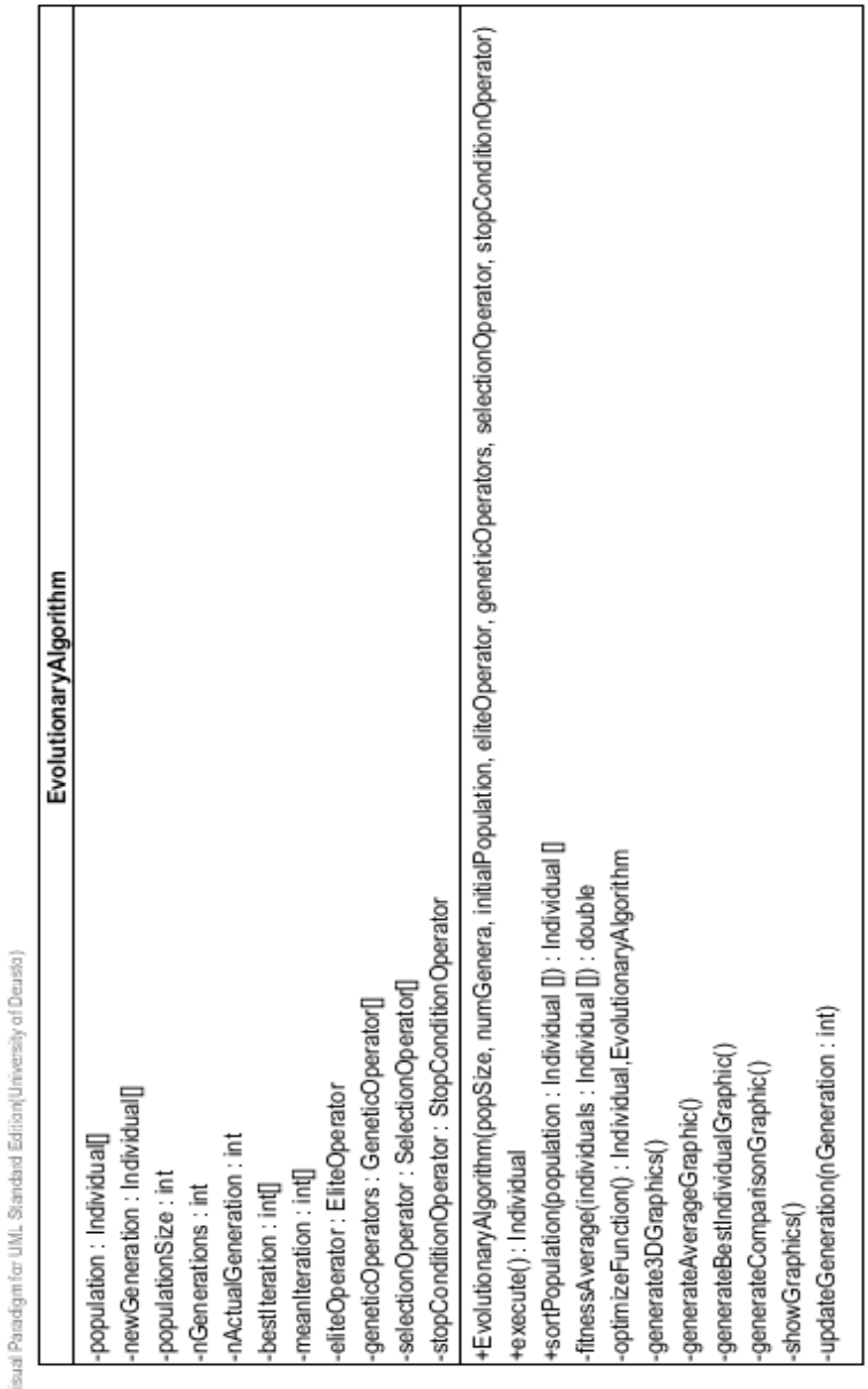


Figure H.4: Evolutionary algorithm class diagram

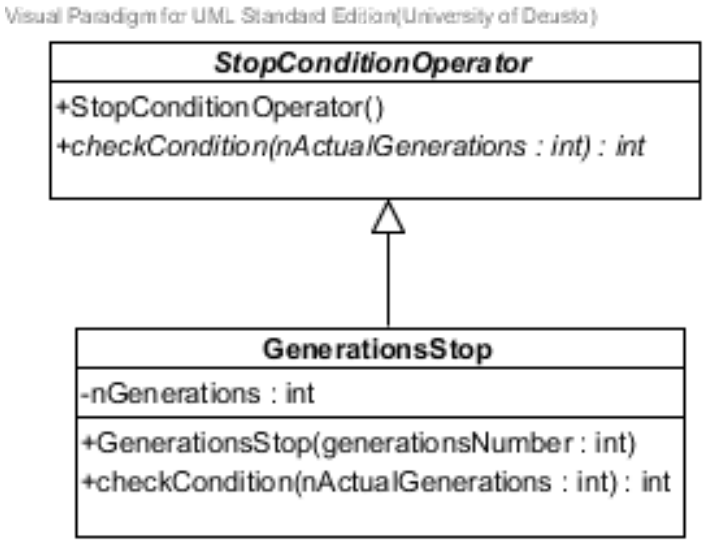


Figure H.5: Stop condition operator class diagram

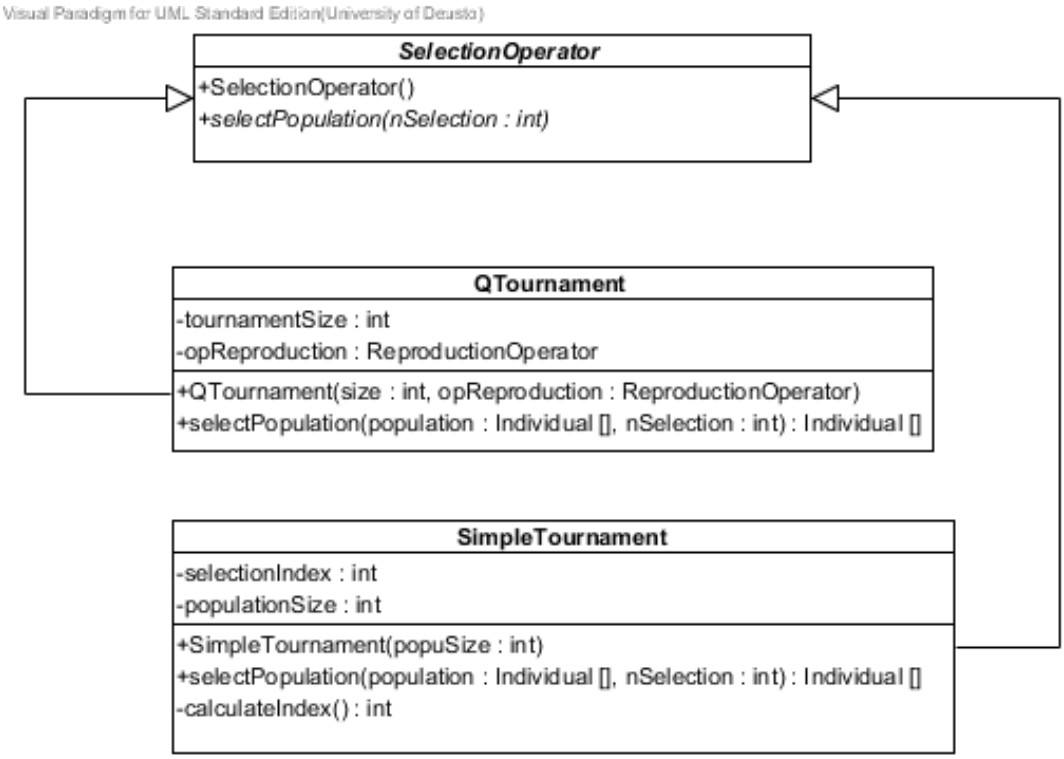


Figure H.6: Selection operator class diagram

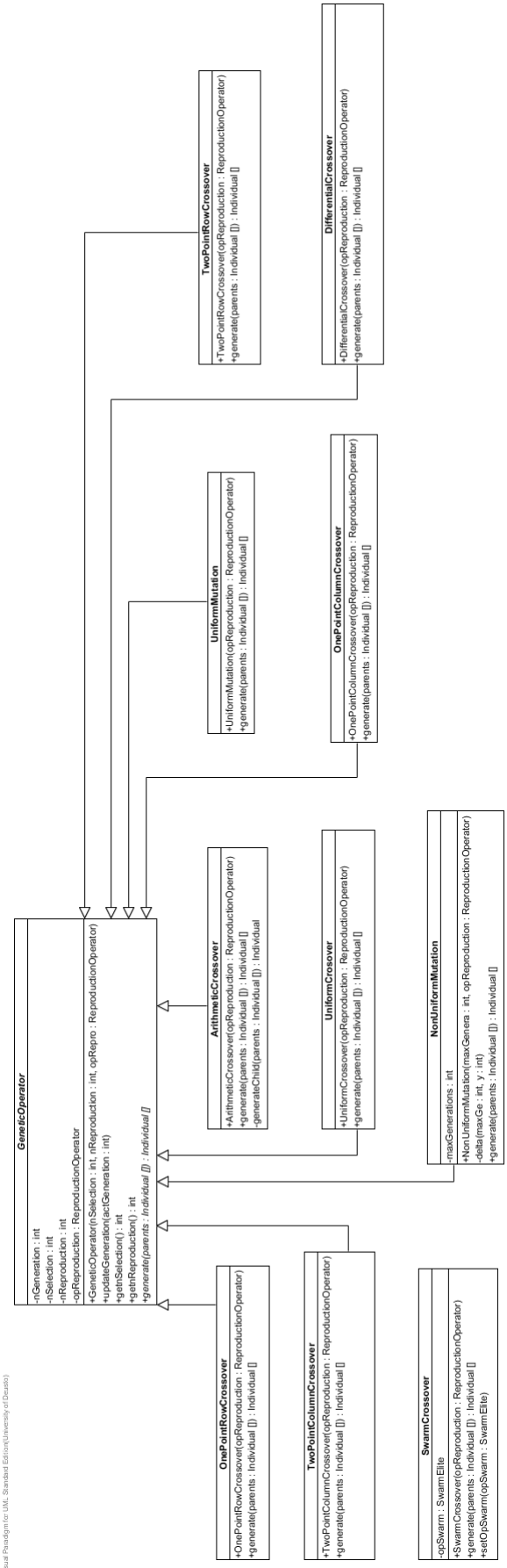


Figure H.7: Genetic operator class diagram

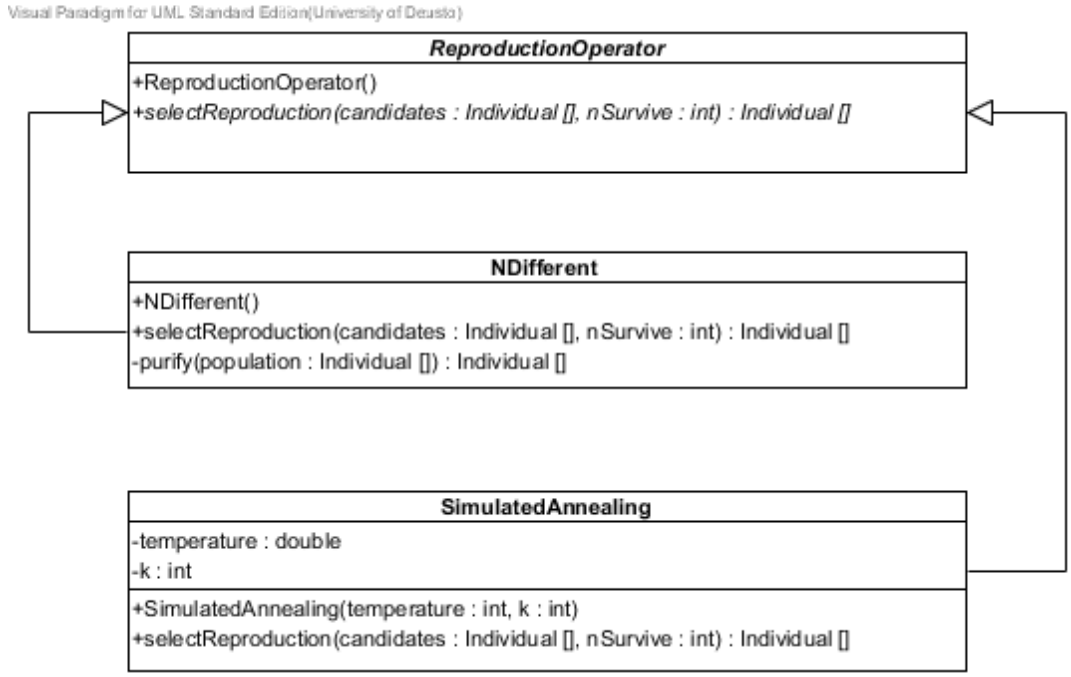


Figure H.8: Reproduction operator class diagram

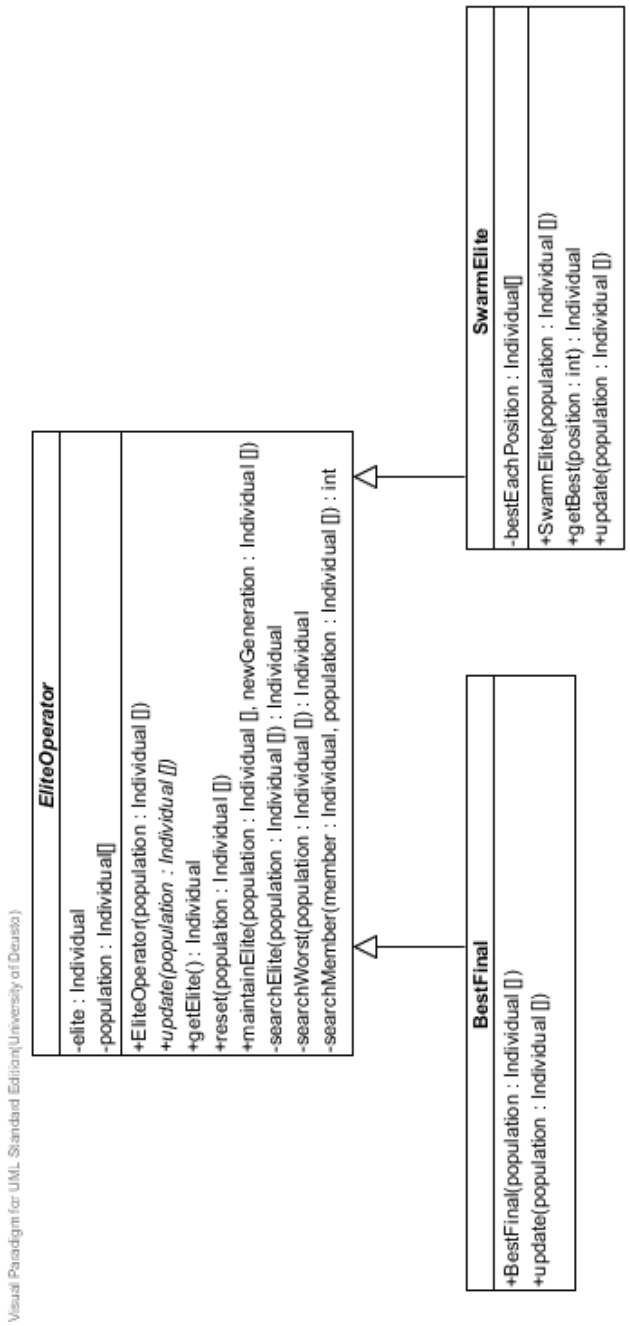


Figure H.9: Elite operator class diagram