Surrogate-Assisted Evolutionary Multi-Objective Full Model Selection

By:

Alejandro Rosales Pérez

A dissertation submitted in partial fulfillment
of the requirements for the degree of:

DOCTOR OF SCIENCE IN COMPUTER SCIENCE

at

Instituto Nacional de Astrofísica, Óptica y Electrónica

January, 2016
Tonantzintla, Puebla

Supervised by:

Dr. Carlos A. Coello Coello, CINVESTAV-IPN
Dr. Jesus A. González Bernal, INAOE
Dr. Carlos A. Reyes García, INAOE

Abstract

Classification problems have become a popular task in pattern recognition. This is, perhaps, because they can be used in a number of problems, such as text categorization, handwriting recognition, etc. This has resulted in a large number of methods. Some of theses methods, called pre-processing, aim at preparing the data to be used and others, called learning algorithms, aim at learning a model that maps from the input data into a category. Additionally, most of them have a set of adjustable parameters, called hyper-parameters, that directly impact the performance of the learned models. Hence, when a classification model is constructed, one has to choose among the set of methods and to configure the corresponding hyper-parameters, which can result in a decision with a high number of degrees of freedom. The latter could be a shortcoming when non-expert machine learning users have to face such a problem.

This thesis deals with the problem of full model selection, which is defined as the problem of finding a combination of pre-processing methods and learning algorithms together with the hyper-parameters that best fit to a dataset. Traditionally full model selection has been approached as a single objective optimization problem and only considered up to two main types of pre-processing. Here, we face this in a broader sense by considering four types of pre-processing and as a multi-objective optimization problem. The multi-objective formulation allows accounting two or more criteria in the optimization stage looking for those solutions with a good trade-off. Here, we used two criteria widely adopted in machine learning, which are the error rate and model complexity.

Evolutionary Algorithms have gained popularity for dealing with multi-objective optimization problems. In recent years, they have been successfully applied to solving different supervised/machine learning problems. However, a drawback of them is that they require a relatively high number of objective functions evaluations in order to get a reasonably good approximation towards the optimal solutions. In this thesis we explore the use of surrogate assisted optimization to deal with the problem of multi-objective full model selection.

We have proposed three methods for handling the problem of model selection

for non-expert users. The first method considers different model types and makes use of the VC Dimension theory to estimate in a general and straightforward fashion the model complexity. This approach is called MOTMS: Multi-Objective Model Type Selection and shows the effectiveness of the VC Dimension for the aim that we look for.

The second approach formulates the full model selection problem as a nested multi-objective optimization one and is called EN-MOMS-PbE: Evolutionary Nested Multi-Objective full Model Selection with Pareto-based Ensembles. Thus, the optimization component has to deal with two optimization levels, in the upper level the model structure is optimized while in the lower level the hyper-parameters for a given model structure are optimized. This second approach also proposes seven strategies for dealing with the trade-off solutions. An experimental comparison is performed under them and we found that a solution based on evolutionary ensemble performs better than the others. EN-MOMS-PbE shows a competitive performance when compared to state of the art model selection methods.

Finally, in order to improve the efficiency of our proposal, the third approach explores the idea of using surrogate-assisted optimization for reducing the number of fitness evaluations required by the evolutionary algorithm. The surrogates are incorporated in the lower optimization of EN-MOMS-PbE, because most of the evaluations are performed at this stage. This third approach is called SEN-MOMS-PbE: Surrogate Evolutionary Nested Multi-Objective full Model Selection with Pareto-based Ensembles. The experimental evaluation shows that SEN-MOMS-PbE is able to significantly reduce the number of evaluations while preserving almost the same performance of EN-MOMS-PbE.

Resumen

Los problemas de clasificación se han vuelto una tarea muy popular en reconocimiento de patrones. Quizá esto es debido a que se encuentran en una gran variedad de aplicaciones, tales como reconocimiento de texto, aplicaciones médicas, etc. Esto ha dado como resultado una gran cantidad de métodos que se han propuesto a lo largo de los años. Algunos de estos métodos, conocidos como de pre-procesamiento, tienen como objetivo preparar los datos para ser usados en una etapa posterior; otros, conocidos como algoritmos de aprendizaje, tienen como fin aprender un modelo que permita mapear un dato de entrada a una determinada categoría. Además, la mayoría de ellos tienen una serie de parámetros ajustables, llamados hiper-parámetros, que pueden afectar drásticamente el desempeño de los modelos. Todo esto resulta en una toma de decisiones con muchos grados de libertad, lo cual puede resultar problemático para usuarios no expertos

Esta tesis trata con el problema de la selección de modelo completo en tareas de clasificación. El problema de selección de modelo completo es definido como la tarea de encontrar una combinación de métodos de pre-procesamiento y algoritmos de aprendizaje que mejor se ajusten a los datos. Tradicionalmente, el problema de selección de modelo completo ha sido enfocado como uno de optimización de un criterio y a lo más, dos tipos principales de pre-procesamiento han sido considerados. En este trabajo, lo abordamos en un sentido más amplio, al considerar cuatro tipos principales de pre-procesamiento y al formularlo como un problema de optimización multi-objetivo. Esto último permite considerar dos o más criterios durante la etapa de optimización. Sin embargo, hemos considerado dos aspectos ampliamente usados en el área de aprendizaje máquina, los cuales son el error del modelo y la complejidad de éste.

Los algoritmo evolutivos se han convertido en técnicas populares para tratar con problemas de optimización multi-objetivo. En los últimos años, éstos se han aplicado exitosamente a una gran variedad de problemas en las áreas de aprendizaje supervisado y aprendizaje de máquina. No obstante, un inconveniente de éstos es que requieren de un gran número de evaluaciones de las funciones a optimizar a fin de

obtener una aproximación razonablemente buena a las soluciones óptimas. Por esta razón, en esta tesis se explora el uso de la optimización asistida por subrogados en el problema de la selección de modelo completo.

Se han propuesto tres métodos para abordar el problema de selección de modelo completo de clasificación para usuarios no expertos. El primero hace uso de la teoría de la dimensión VC para estimar de una forma genérica y sencilla la complejidad de los modelos. Este enfoque es llamado MOMTS, por las siglas en inglés de *Multi-Objective Model Type Selection*. MOMTS mostró que la dimensión VC es una herramienta eficaz.

El segundo enfoque formula el problema de selección de modelo completo como uno de optimización multi-objetivo anidado y es llamado EN-MOMS-PbE, por las siglas en inglés de *Evolutionary Nested Multi-Objective full Model Selection with Pareto-based Ensembles*. Debido al enfoque anidado, la optimización tiene dos niveles; en el nivel superior trabaja con la estructura del modelo, mientras que en el inferior se optimizan los hiper-parámetros de la estructura dada. EN-MOMS-PbE también propone siete maneras differentes de usar la información contenida en las soluciones compromiso para construir un modelo. El estudio experimental reveló que la estrategia basada en un conjunto (*ensemble*) de estrategias evolutivas se desempeña mejor que las otras. EN-MOMS-PbE también mostró un desempeño competitivo al ser comparado con métodos del estado del arte.

Finalmente, y a fin de mejorar la eficiencia de los métodos propuestos, en el tercer enfoque se explora la idea de la optimización asistida por subrogados para reducir el número de evaluaciones requeridas por el algoritmo evolutivo. Los subrogados son incorporados en el nivel inferior de la optimización, ya que es aquí donde se produce el mayor número de evaluaciones de las funciones objetivo. Este tercer enfoque es llamado SEN-MOMS-PbE, por las siglas en inglés de *Surrogate Evolutionary Nested Multi-Objective full Model Selection with Pareto-based Ensembles*. La evaluación experimental muestra que SEN-MOMS-PbE es capaz de reducir de una manera significantiva el número de evaluaciones, mientras que conserva casi el mismo desempeño que EN-MOMS-PbE

Acknowledgments

First and foremost, I would like to thank sincerely to my mom, Alba Pérez Palacios, who with her endless love has supported and encouraged me in every project in my life. I would also want to thank to my family, my sister Claudia Viridiana and her two children: Daniela Alejandra and Santiago, and my godparents: Gloria María and Rubén Dario. All of them are an important part of my life and without them, my success would not be possible.

I would like to show my gratitude to all people who were involved in this research topic: Dr. Jesús A. González Bernal, Dr. Carlos A. Reyes García, and Prof. Carlos A. Coello Coello. Their advices had been enriching to bring this project to fruition. Thank you for giving me incentive along these four years both in good and bad moments.

My sincere thanks to the committee members: Dr. Efrén Mezura Montes, Dr. Hugo Jair Escalante Balderas, Dra. Alicia Morales Reyes, Dr. Manuel Montes y Gómez, and Dr. Felipe Orihuela Espina. They spent part of their time in reviewing this manuscript and their comments have been beneficial for improving it.

I would also like to thank Prof. Francisco Herrera and Dr. Salvador García, from the University of Granada. They supervised my work during my research stay at Spain. Paco, Salva, together with all members from SCI^{2}S group were very nice with me and their hospitality made my stay very pleasant.

My gratefulness to all my friends from INAOE: María Alejandra, Adrián Pastor, Alejandro Torres, Rigoberto, Marisol, Miguel Ángel, Miguel, Adrián Leal, Adrián León, Mauricio, among others. I will always remember all of you, with whom I have lived great moments. Thank you for your friendship.

My special gratitude to the computer science department at INAOE, without forgetting all researchers and administrative staff, thanks for your support during these four years.

Finally, I acknowledge CONACyT for the scholarship No. 329013, which has supported me along these four years.

Contents

List of Figures

List of Tables

Part I

Introduction

Chapter 1

Introduction

Begin at the beginning,– the King said,
gravely,– and go on till you come to the end;
then stop

Lewis Carroll

Classification is a mainstream in pattern recognition. Perhaps its popularity relies on the fact that it can be used in a wide range of applications, such as in medical diagnosis, image recognition, and text recognition, among others. During several years, a number of pattern recognition/machine learning methods have been proposed. These methods encompass data pre-processing and learning algorithms. Due to this umbrella of algorithmic options, in the design of a classification model the user has to face the problem of choosing one in order to achieve a peak performance in the application at hand. Moreover, these machine learning algorithms often have a set of adjustable parameters, called hyper-parameters[1], that need to be fit before the construction of the classification model. It has been shown that the classification performance can be improved by configuring the hyper-parameters of the existing techniques (Bardenet et al., 2012; Coates et al., 2011; Pinto et al., 2009). These can become a shortcoming when the user is not an expert in machine learning or is not acquainted with those methods.

During the last few years, the interest for developing methods leading toward an automated machine learning algorithm has significantly increased. Here, the idea is to perform the task of choosing the algorithm and/or its hyper-parameters for a given dataset; this is sometimes called *model selection*. These methods can be categorized in three main groups:

[1]The term hyper-parameters is used to distinguish them from the model parameters. For example, in an artificial neural network the number of neurons and learning rate are the hyper-parameters, while the parameters are the weights that are learned during the training phase.

- **methods based on meta-learning**: they attempt to learn a function that maps from a set of meta-features[2] to a model. The central problem of this approach is to define the set of meta-features to describe the dataset;

- **methods based on optimization**: as the name suggests, the idea here is to formulate the model selection problem as an optimization one and to explore the hyper-parameters space looking for those that best optimize a given criterion. In this approach, however, the evaluation of the criterion could be computationally expensive and could require a relatively high number of evaluations during the search. Nevertheless, there could be some scenarios that require the model selection to be performed in a few trials; and

- **hybrid methods**: they consider the model selection problem both as a learning and optimization one. This is a promising approach, which aims at reducing the computational cost of the optimization based methods.

In spite of the considerable amount of research available for solving this problem, there are still gaps that motivate the continuous development of new methods. For instance, most of the existing studies only focus on a single learning algorithm and methods for data pre-processing are usually not taken into consideration. Thus, selecting a combination of data pre-processing methods, learning algorithms and hyper-parameters is a problem with a large number of degrees of freedom and is referred in the literature as the *full model selection* problem.

Machine learning problems can be naturally approached as multi-objective optimization problems, where the typical idea is to find a trade-off between the model accuracy and the model complexity; this, however, has not been fully exploited in the full model selection problem. Evolutionary Algorithms (EAs) have become popular optimization techniques for solving multi-objective problems; these are usually called Multi-Objective EAs (MOEAs). In recent years, MOEAs have been successfully applied in problems from domains of data-mining and supervised learning (Jin and Sendhoff, 2008; Mukhopadhyay et al., 2014a,b), such as feature selection (Wang and Huang, 2009; Wang et al., 2015), association rules mining (Martín et al., 2014a,b; Minaei-Bidgoli et al., 2013), clustering (Handl and Knowles, 2007; Mukhopadhyay et al., 2015), and classification (Antonelli et al., 2014; Rosales-Pérez et al., 2014, 2015). MOEAs, however, usually require performing a large number of evaluation steps of the objective function to get a reasonable approximation to the so-called Pareto front.

[2]The characteristics that describe the properties of the dataset are called meta-features.

Motivated by the aforementioned gaps, this work raises from the necessity of developing efficient and effective methods in the field of model selection, where the term efficient refers to minimize as much as possible the number of function evaluations to get an effective full model selection. In this regard, our work explores the multi-objective formulation of the full model selection problem and the use of MOEAs, which have shown to be powerful algorithms for solving multi-objective problems.

1.1 Working Hypothesis

Based on the above, the main hypothesis for this research is stated as follows:

> A novel method based on surrogate[3] evolutionary computation can be proposed for solving the full model selection problem as a multi-objective one, looking for solutions that provide a good trade-off among the criteria, with a reduced number of function evaluations. As a result of this, it is expected that the trade-off full models would be highly effective.

1.2 Goal

The main goal of this research is to advance in the state of the art with respect to the design of effective and efficient multi-objective full model selection methods through surrogate-assisted evolutionary computation. Here, we focus on classification problems, but the ideas can be mapped to other learning problems. To achieve this, the following particular goals are defined:

- to gain knowledge in the full model selection problem and, more particularly, in the application of evolutionary computation in this kind of problems;
- to advance the knowledge in the full model selection problem by developing methods, based on multi-objective evolutionary algorithms, for classification problems;
- to advance the knowledge of decision making in trade-off solutions by developing strategies that allow exploiting the information from them, with the aim of constructing more effective full classification models;

[3] A surrogate model is an approximation of the objective function to be optimized, which is cheaper to evaluate than the original one.

- to design, implement, and evaluate a novel approach based on surrogate-assisted evolutionary computation for efficiently solving the problem of full model selection with the aim of producing trade-off models with a reduced set of function evaluations.

1.3 Contributions

This thesis has contributed with the following:

- A general framework for the full model selection problem, which helps choose a model for a classification task considering two criteria: the model performance and its complexity.
- A new method based on multi-objective evolutionary computation for handling the problem of model type selection. The method is called MOMTS.
- A new method, called EN-MOMS-PbE, for handling in a hierarchical fashion the full model selection problem.
- The use of the VC dimension[4] as a tool to experimentally determine the model complexity to any model type and to control the overfitting.
- Several strategies to handle the trade-off solutions, which enable the construction of model ensembles.

As a result of these, the following lists the papers directly derived from this thesis or those where ideas from this work have been used:

- **Journal**
  - **Rosales-Pérez, A.**, Gonzalez, J. A., Coello Coello, C. A., Escalante, H. J., Reyes-Garcia, C. A., 2014. Multi-objective model type selection. Neurocomputing 146, 83 — 94.
  - **Rosales-Pérez, A.**, Gonzalez, J. A., Coello Coello, C. A., Escalante, H. J., Reyes-Garcia, C. A., 2015. Surrogate-assisted multi-objective model selection for support vector machines. Neurocomputing 150, Part A, 163 — 172.

[4]The VC dimension is a measure capacity/complexity of a classification model.

- **Congress**
  - **Rosales-Pérez, A.**, Escalante, H. J., Gonzalez, J. A., Reyes-Garcia, C. A., Coello Coello, C. A., 2013. Bias and Variance Optimization for SVMs Model Selection. In Proceedings of the Twenty-Sixth International FLAIRS Conference.
  - **Rosales-Pérez, A.**, Escalante, H. J., Gonzalez, J. A., Reyes-Garcia, C. A., Coello Coello, C. A., 2013. Bias and Variance Multi-objective Optimization for Support Vector Machines Model Selection. In Proceedings of the 6th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA 2013), pp. 108 – 116.
  - **Rosales-Pérez, A.**, Coello Coello, C. A., Gonzalez, J. A., Reyes-Garcia, C. A., Escalante, H. J., 2013. A hybrid surrogate-based approach for evolutionary multi-objective optimization. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation (IEEE-CEC 2013), pp. 2548 – 2555.
  - **Rosales-Pérez, A.**, Escalante, H. J. Coello Coello, C. A., Gonzalez, J. A., Reyes-Garcia, C. A., 2014. An evolutionary multi-objective approach for prototype generation. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation (IEEE-CEC 2014), pp. 1100 – 1107.
  - **Rosales-Pérez, A.**, Gonzalez, J. A., Coello Coello, C. A., Reyes-Garcia, C. A., Escalante, H. J., 2014. Evolutionary Multi-Objective Approach for Prototype Generation and Feature Selection. In Proceedings of the 2014 Iberoamerican Congress on Pattern Recognition (CIARP 2014), pp. 424 – 431.

1.4 Document Outline

This thesis is structured in three parts besides the introductory one. These are listed below.

- In Part II, there are three chapters that describe the background concepts, required to make this document as self-contained as possible. This part encompasses the following chapters:
  - In chapter 2, we describe basic concepts related to the topic of multi-objective optimization.
  - In chapter 3, the general concepts of evolutionary computation are described, with emphasis on multi-objective evolutionary algorithms.

  - In chapter 4, the concepts related to supervised learning/machine learning are described.

- Part III presents the contributions of this thesis, and is organized in the following chapters:

  - In chapter 5 is presented MOMTS: Multi-Objective Model Selection, a method for handling the model type selection problem as a multi-objective one. It uses the VC dimension as a measure of the model complexity.
  - In chapter 6, we describe the formulation of the multi-objective full model selection problem as a hierarchical optimization problem. We also present several strategies for handling the trade-off solutions.
  - In chapter 7, the hybridization with the surrogate models that aims at reducing the number of fitness function evaluations is described.

- Part IV outlines the general conclusions of this thesis.

Part II

Theoretical Background Review

Chapter 2

Multi-Objective Optimization

In mathematics you don't understand things.
You just get used to them.

John Von Neumann

The optimization term refers to finding a feasible solution that causes the minimum (or the maximum) possible value in one or more objective functions. Optimization is a task common to many engineering and scientific disciplines. For instance, in an engineering application, one could wish to get the minimum possible cost of production and/or to maximize the obtained benefit for a given product. Optimization problems that involve optimizing a single objective function are known as ***single-objective optimization problems*** **(SOPs)**, whereas those that involve two or more objectives are known as ***multi-criteria*** or ***multi-objective optimization problems*** **(MOPs)**.

In single-objective optimization problems, it is possible to determine whether a solution is better than another one by comparing the attained value of each of these in the objective function. In this kind of problems, one normally wants to find the best solution for the problem at hand.

On the other hand, in multi-objective optimization problems, the objectives are usually in conflict. In such cases, there does not exist a single solution that would simultaneously be the best for all objectives. Hence, in this kind of problems, one usually wants to find a set of solutions that satisfy the best trade-off among the objectives.

The main focus of this chapter is to describe multi-objective optimization problems. Next, we present some basic definitions related to an optimization problem and after that, we present the multi-objective optimization problem.

2.1 Basic Concepts

A general optimization problem usually consists of the following elements:

- **decision variables**, which represent the set of n parameters of the problem whose values are modified during the optimization process in order to solve the problem. These are typically denoted by an n-dimensional vector, $\mathbf{x} = [x_1, \ldots, x_n]$;
- **objective functions**, which are the evaluation criteria to be used so as to estimate how good a solution is. In SOPs, there is a single objective function; whilst in MOPs there are two or more objective functions;
- **constraints**, these are restrictions imposed by the particular characteristics of the environment or available resources. These constraints must be satisfied to consider an acceptable solution.

Those solutions that satisfy the constraints are called *feasible*. Therefore, in the presence of constraints, the entire decision variable space does not need to be feasible. The set of all feasible solutions is known as the *feasible region*.

2.2 The Multi-Objective Optimization Problem

A general MOP can be formally stated as follows (Coello Coello et al., 2007; Deb, 2001; Miettinen, 1999) [1]:

$$\begin{aligned} &\text{minimize } \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_l(\mathbf{x})]^T \\ &\text{subject to } \mathbf{x} \in \mathcal{X} \end{aligned} \tag{2.1}$$

where $\mathbf{x} = [x_1, \ldots, x_n]^T \in \mathbb{R}^n$ is a vector of decision variables, $f_i(\mathbf{x})$, $i = 1, \ldots, l$, are the $l-$objective functions, and $\mathcal{X}$ is the set of feasible solutions.

When evaluating the objectives in a MOP, a projection occurs from the decision variables vector to a vector with the corresponding values in the objective functions. Thus, in a MOP there exist two different spaces: the decision variables space and the objective space. This is illustrated in Figure 2.1.

Solving a MOP involves finding a solution that simultaneously minimizes all objectives. Nevertheless, the objectives are normally in conflict, making that, in those

[1] Without loss of generality, we only assume minimization problems. Maximization problems can be converted to minimization ones using the duality principle (Deb, 2012; Rao, 1984; Ravindran et al., 2006); i.e., multiply by -1 the objective functions to maximize them.

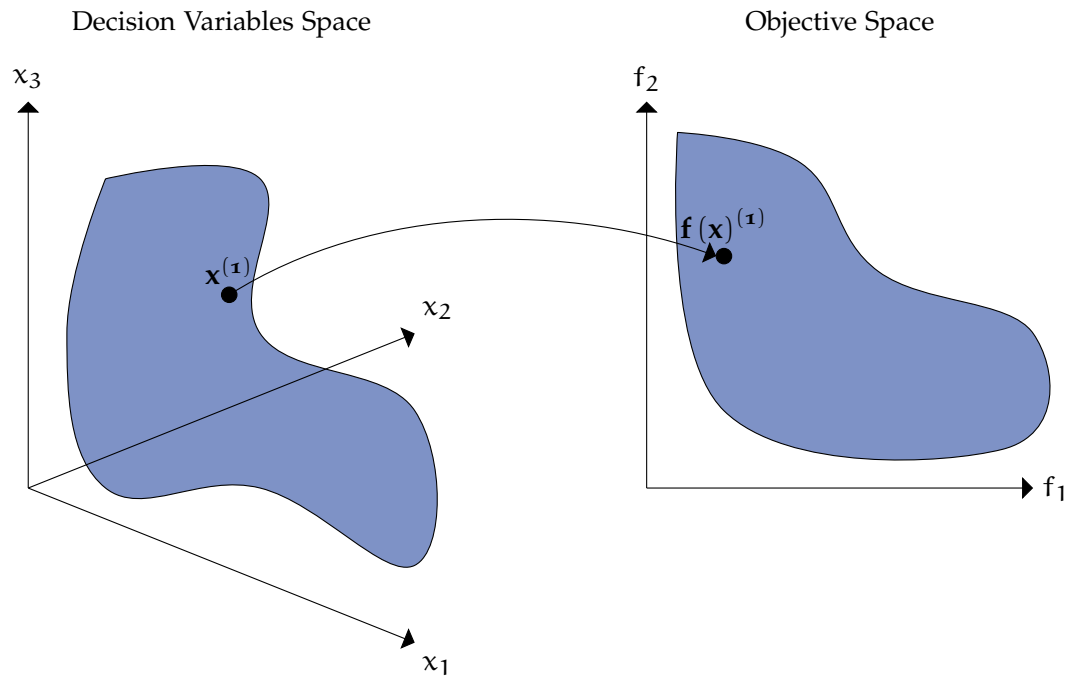


Figure 2.1: Two spaces in a MOP. Each feasible point in the decision variables space is projected to a corresponding one in objective space.

cases, there is not a single solution that would be the best for all objectives. Thus, the notion of optimum in MOPs differs from that in SOPs, in which the feasible set is totally ordered according to the objective function, $f(\mathbf{x})$, and the goal is to find a solution that gets the minimum value in such function. In MOPs, the feasible set usually is no longer totally ordered, but partially ordered; hence, the notion of optimum refers to finding a set of solutions that satisfy a good trade-off among the objectives. The notion of optimum most commonly adopted is the one proposed by Francis Ysidro Edgeworth (Edgeworth, 1881) and later generalized by Vilfrido Pareto (Pareto, 1964). Some authors call this notion the *Edgeworth-Pareto optimum*, but the most commonly accepted term is *Pareto optimum* (Coello Coello et al., 2007). Next, Pareto optimality definitions are presented.

2.2.1 Dominance and Pareto Optimality

Most modern multi-objective optimization algorithms use the concept of Pareto dominance to determine whether a solution is better than another. Roughly speaking, the idea is to compare two solutions in order to determine whether one of these dominates the other one or not. More formally, Pareto dominance is defined as follows:

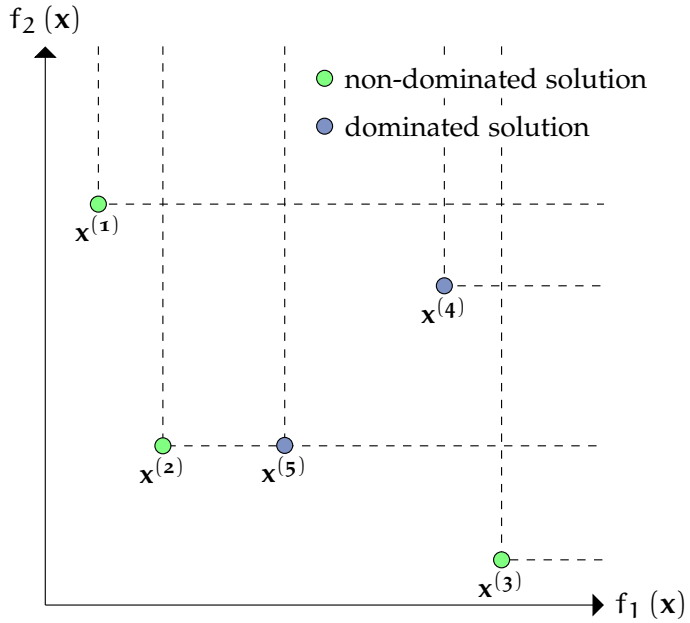


Figure 2.2: Pareto dominance relation for a bi-objective problem.

Definition 2.1 (Pareto Dominance) *It is said that a solution* $\mathbf{x}^{(1)}$ *dominates a solution* $\mathbf{x}^{(2)}$ *(denoted by* $\mathbf{x}^{(1)} \preceq \mathbf{x}^{(2)}$*) iff* $\mathbf{x}^{(1)}$ *is not worse than* $\mathbf{x}^{(2)}$ *for any objective and there exists at least one objective for which* $\mathbf{x}^{(1)}$ *is better than* $\mathbf{x}^{(2)}$*, i.e.,*

$$\forall_{i\in\{1,\ldots,l\}} : f_i\left(\mathbf{x}^{(1)}\right) \leqslant f_i\left(\mathbf{x}^{(2)}\right) \wedge \exists_{i\in\{1,\ldots,l\}} : f_i\left(\mathbf{x}^{(1)}\right) < f_i\left(\mathbf{x}^{(2)}\right) \tag{2.2}$$

Figure 2.2 illustrates the Pareto dominance relation for a bi-objective optimization problem. For instance, in this figure, we note that solution $\mathbf{x}^{(2)}$ dominates solution $\mathbf{x}^{(4)}$, since the value attained by $\mathbf{x}^{(2)}$ both in $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ is lower than that obtained by $\mathbf{x}^{(4)}$. The solution $\mathbf{x}^{(2)}$ also dominates solution $\mathbf{x}^{(5)}$ because the behavior of $\mathbf{x}^{(2)}$ in $f_2(\mathbf{x})$ is not worse than the one obtained by $\mathbf{x}^{(5)}$ in $f_2(\mathbf{x})$ (in fact, both have the same performance in that objective function), but $\mathbf{x}^{(2)}$ is better than $\mathbf{x}^{(5)}$ in $f_1(\mathbf{x})$. Regarding $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, we note that neither $\mathbf{x}^{(1)}$ dominates $\mathbf{x}^{(2)}$ nor $\mathbf{x}^{(2)}$ dominates $\mathbf{x}^{(1)}$, since $\mathbf{x}^{(1)}$ is better than $\mathbf{x}^{(2)}$ in $f_1(\mathbf{x})$, but it is worse in $f_2(\mathbf{x})$. Therefore, when comparing any two solutions, $\mathbf{x}$ and $\mathbf{y}$, the outcome of Pareto dominance is one out of three possibilities: $\mathbf{x} \preceq \mathbf{y}$, $\mathbf{y} \preceq \mathbf{x}$, or $\mathbf{x} \npreceq \mathbf{y} \wedge \mathbf{y} \npreceq \mathbf{x}$.

Some important properties of the Pareto dominance relation are the following:

- **it is not reflexive**, since any solution $\mathbf{x}$ cannot dominate itself;
- **it is not symmetric**, because if $\mathbf{x} \preceq \mathbf{y}$, this does not imply that $\mathbf{y} \preceq \mathbf{x}$;

- **it is transitive**, because if $\mathbf{x} \preceq \mathbf{y}$ and $\mathbf{y} \preceq \mathbf{z}$, this implies that $\mathbf{x} \preceq \mathbf{z}$.

When comparing the solution $\mathbf{x}^{(1)}$ with respect to the others shown in Figure 2.2, we can note that none of these dominates it. A solution that is not dominated by any other solution is called ***non-dominated***. Furthermore, when the comparison is performed with respect to the entire feasible decision space, the non-dominated solution is called ***Pareto optimal***.

Definition 2.2 (Pareto optimal solution) *A solution $\mathbf{x}^*$ is Pareto optimal iff there does not exist any other solution in $\mathcal{X}$, the entire feasible decision space, that dominates $\mathbf{x}^*$, i.e.,*

$$\nexists \mathbf{x}' \in \mathcal{X} : \mathbf{x}' \preceq \mathbf{x}^* \tag{2.3}$$

Continuing with the example shown in Figure 2.2, when performing all possible pairwise comparisons among the solutions using Definition 2.1, we can notice that this definition does not produce a single solution, but a set of trade-off solutions. In the example at hand, there is not any other solution that dominates $\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$, and $\mathbf{x}^{(3)}$. The set of solutions that are non-dominated solutions is called ***non-dominated set***. Similarly, if these solutions are non-dominated with respect to the entire feasible decision space (i.e., Pareto optimal solutions), the resulting non-dominated set is referred to as ***Pareto optimal set***.

Definition 2.3 (Pareto optimal set) *The set of solutions that are not dominated by any member of the entire feasible decision space is the Pareto optimal set $\mathcal{P}^*$, i.e.,*

$$\mathcal{P}^* = \{\mathbf{x} : \mathbf{x} \in \mathcal{X} \text{ is Pareto optimal}\} \tag{2.4}$$

As we previously stated, there exists a mapping between decision variables space and objective space. The image of the Pareto optimal set is called ***Pareto front***.

Definition 2.4 (Pareto front) *The Pareto front, $\mathcal{PF}^*$, is the image of the Pareto optimal set in the objective space, i.e.,*

$$\mathcal{PF}^* = \{\mathbf{f}(\mathbf{x}) : \mathbf{x} \in \mathcal{P}^*\} \tag{2.5}$$

Thus, when solving a MOP, we are interested in finding solutions that satisfy the best possible trade-off among the objectives. We are also interested in obtaining a well-distributed set of solutions along the Pareto front since that will provide with more options to choose from.

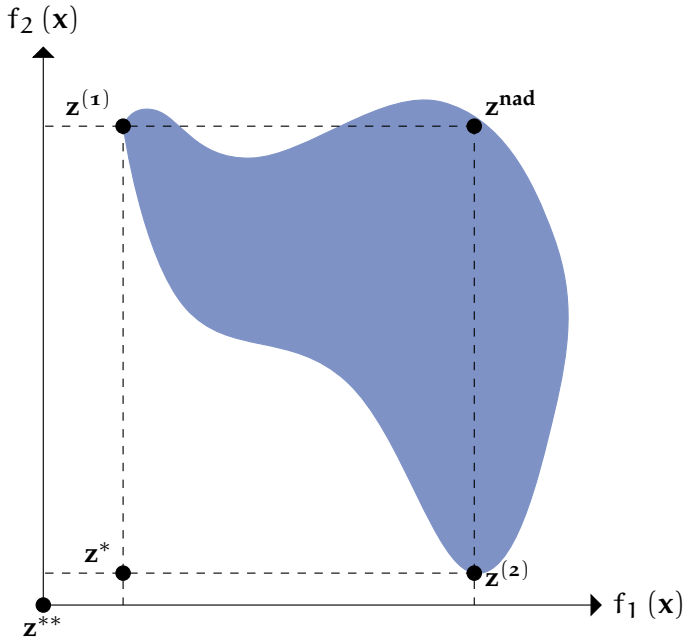


Figure 2.3: Ideal objective vector $\mathbf{z}^*$, utopian objective vector $\mathbf{z}^{**}$, and nadir objective vector $\mathbf{z}^{nad}$

2.2.2 Special Solutions

There exist some special solutions, which are used for some multi-objective optimization algorithms. These solutions are the ideal objective vector, utopian objective vector, and nadir objective vector, which are shown in Figure 2.3 and revised next.

Ideal Objective Vector

For a MOP with conflicting objectives, there does not exist a single solution that wold be optimal for all objectives. A vector constructed with the optimal values of each objective is called ***ideal objective vector***.

Definition 2.5 (Ideal objective vector) *An ideal objective vector $\mathbf{z}^*$ is the one whose components are obtained by minimizing each objective function (subject to the constraints) individually, i.e.,*

$$\mathbf{z}^* = [\min f_1(\mathbf{x}), \ldots, \min f_l(\mathbf{x})] \tag{2.6}$$

In general, the ideal objective vector corresponds to a non-existing feasible solution, since the solutions that minimize each objective function do not need to be the same. If the ideal objective vector corresponds to a feasible solution, this would be the solution

of the MOP and the Pareto optimal set would be reduced to this point. In this case, the objectives are not in conflict.

Utopian Objective Vector

The ideal objective vector denotes the lower bound of all objective functions. Some algorithms could require a solution that is strictly better than any Pareto optimal solution, this is the ***utopian objective vector***.

Definition 2.6 (Utopian objective vector) *An utopian objective vector* $\mathbf{z}^{**}$ *is the one whose components are better that those of the ideal objective vector, i.e.,*

$$\mathbf{z}^{**} = [z_1^* - \epsilon_1, \ldots, z_l^* - \epsilon_l] \mid \epsilon_{i \in \{1,\ldots,l\}} > 0 \tag{2.7}$$

Like the ideal objective vector, the utopian vector also represents a non-existing feasible solution.

Nadir Objective Vector

Unlike the ideal objective vector that represents the lower bound of each objective, the ***nadir objective vector***, $\mathbf{z}^{\mathbf{nad}}$, represents the upper bound of each objective in the entire Pareto optimal set.

The nadir objective vector corresponds to either an existing feasible solution or a non-existing one. Computing the nadir vector is not straightforward. Nadir vector can be used to normalize the objective vector with respect to the entire Pareto optimal set.

2.3 Decision Making

The solution of a MOP is the Pareto optimal set. This set is usually formed by a large (or even infinite) number of Pareto optimal solutions. Mathematically, each of these is an equally acceptable solution of the MOP. Nonetheless, in many situations is desirable to obtain a single solution from this. Selecting one out of the set of Pareto optimal solutions calls for information that is not typically contained in the objective functions. The ***decision maker*** **(DM)** is the responsible of choosing the most preferred one.

The DM's preferences can be incorporated in order to induce a total order among the solutions in the Pareto optimal set. There are several approaches for doing so. A taxonomy of the techniques based on the stage of the search at which the DM's preferences are incorporated was proposed by Hwang and Masud (1979) and Miettinen (1999). This taxonomy is the following:

- **no-preference methods**, which do not assume any information about the importance of the objectives. A heuristic is used to find a single optimal solution;
- **a priori methods**, in which the DM's preferences are expressed before the optimization process. The difficulty is, however, that the DM does not necessarily know beforehand what is possible to attain in the problem;
- **a posteriori methods**, which could also be called *methods for generating Pareto optimal solutions*. Here, the optimization process is first performed to generate the Pareto optimal set. After that, it is presented to the DM, who selects the most preferred solution among the alternatives. This approach, however, has the inconvenience that DM must deal with a large set of alternatives;
- **interactive methods**, in which the DM works together with an interactive optimization process. At each step, partial preference information is supplied by the DM to the optimizer, which, in turn, generates better alternatives according to the information received. This approach, however, requires the availability of the DM, who is involved in providing information about the direction search from time to time during the optimization process.

Next, we revise some methods according to the previous taxonomy.

2.3.1 A Priori Methods

In a priori methods, the DM must specify his/her preferences before the search process. In this category, we can find the value function method, and goal programming, among others.

Value Function Method

In this method, the DM must provide a mathematical function $U : \mathbb{R}^l \rightarrow \mathbb{R}$ that represents his/her preferences with respect to all l-objectives. This function should provide a complete ordering in objective space. Hence, the task is to maximize the value function as follows:

$$\begin{aligned} &\text{maximize } U(\mathbf{f}(\mathbf{x})) \\ &\text{subject to } x \in \mathcal{X} \end{aligned} \tag{2.8}$$

This method seems to be very simple; in fact, it could be an excellent method if the DM has an explicit mathematical formulation for the value function that fully represents the DM's preferences. Nevertheless, as one can figure out, the obtained

solution strongly depends on the chosen value function, which should be globally applicable to the entire search space. Furthermore, a function that imposes a total ordering in the set of feasible solutions might not exist.

Lexicographic Ordering

In this method, the DM arranges the objective functions according to their importance, from best to worst. After that, the most important objective function is minimized subject to the original constraints. If the problem has a unique solution, this is the solution of the whole multi-objective problem. Otherwise, it proceeds to minimize the second most important objective function subject to the original constraints and an additional one to guarantee that the most important objective function preserves its optimal value. Let the objective functions be arranged according to the lexicographic order from the most important $f_1(\mathbf{x})$ to the least important $f_l(\mathbf{x})$, we write the lexicographic problem as:

$$\begin{aligned} &\text{lex minimize } f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_l(\mathbf{x}) \\ &\text{subject to } x \in \mathcal{X} \end{aligned} \tag{2.9}$$

In this method, the DM must face the issue of putting the objective functions into an absolute order of importance. It is very likely that least important objective functions would not be taken into account. If the most important objective further has a unique solution, the other objectives do not have any influence on the solution. Notwithstanding these inconveniences, this method is usually robust.

2.3.2 A Posteriori Methods

In these methods, the search is first performed to generate the Pareto front and a decision is taken after that. In this category, we can find the weighted sum method, ϵ-constraint method, among others.

Weighted Sum Method

This method converts a MOP into a SOP by multiplying each objective with a weight. Hence, the problem to solve using this method is expressed as follows:

$$\begin{aligned} &\min f(\mathbf{x}) = \sum_{i=1}^{l} \omega_i f_i(\mathbf{x}) \\ &\text{subject to } \mathbf{x} \in \mathcal{X} \end{aligned} \tag{2.10}$$

where $\omega_i \geqslant 0$ is the weight factor that represents the DM's preference of each objective.

Under the weighted sum method, it is usually assumed that the objectives are normalized, as well as it is assumed that the weight vector is also normalized, i.e., $\sum_{i=1}^{l} \omega_i = 1$.

This method has several disadvantages, such as the difficulty to generate Pareto optimal solutions in problems in which the Pareto optimal front is non-convex. Setting the weights is another shortcoming, since the mapping from a weight vector to a Pareto optimal solution is usually unknown, i.e., a uniformly distributed weight vector does not necessarily produce a uniformly distributed set of Pareto optimal solutions. Moreover, two different weight vectors do not necessarily lead to different Pareto optimal solutions. In spite of these drawbacks, it is probably the simplest approach to generate different Pareto optimal solutions. It can further be used as an a priori method.

ϵ-Constraint Method

In the ϵ-constraint method, one objective is selected to be optimized, whilst the rest are converted into constraints by setting an upper bound to each of them. The problem to be solved is as follows:

$$\begin{aligned} &\min f_k(\mathbf{x}) \\ &\text{subject to } f_j(\mathbf{x}) \leqslant \epsilon_j, \forall_{j \in \{1,\ldots,l\} \wedge j \neq k} \\ &\mathbf{x} \in \mathcal{X} \end{aligned} \tag{2.11}$$

where ϵ_j is the upper bound for the j^{th} objective function, which is varied by the optimizer in order to find different Pareto optimal solutions.

An advantage of this is its ability on finding Pareto optimal solutions, even when the objective space is non-convex. However, the solution largely depends on the ϵ vector, which should be chosen such that it lies between the minimum and maximum values of each objective function. Furthermore, this method could be more laborious due to the fact that the number of constraints is increased. As the weighted sum method, the ϵ-constraint method can also be used as an a priori method.

Method of Weighted Metrics

Instead of using a weighted sum of the objectives, in this method, weighted metrics such as l_p and l_∞ are used. Under this approach, the task is as follows:

$$\begin{aligned} &\text{minimize } l_p = \left(\sum_{i=1}^{l} \omega_i \left| f_i(\mathbf{x}) - z_i^* \right|^p \right)^{1/p} \\ &\text{subject to } \mathbf{x} \in \mathcal{X} \end{aligned} \tag{2.12}$$

where $\mathbf{z}^* = \left[z_1^*, \ldots, z_l^* \right]$ is the ideal solution, p is a parameter that can take any value from the range 1 to ∞. For a large value of p, the above problem is equivalent to minimizing the largest deviation $\left| f_i(\mathbf{x}) - z_i^* \right|$, this is known as the *weighted Tchebycheff* problem.

The resulting Pareto optimal solution obtained greatly depends on the chosen l_p metric. One should note that since the objectives could be in different orders of magnitude, it is suitable to normalize the objective vectors. Moreover, since this method requires the ideal solution, all l-objectives need to be optimized independently. However, there exists a theorem for the weighted Tchebycheff metric that guarantees finding each Pareto optimal point when the reference point is the utopian vector (Miettinen, 1999).

2.3.3 Interactive Methods

The main aspect of interactive methods is that during the optimization process, the DM is involved in providing some kind of information, such as a weight vector, a reference point, or another in order to guide the search. Since the DM is involved in the optimization, simplicity is lost. Among the methods that are included into this category, we can find:

- interactive surrogate worth trade-off method,
- step method,
- reference point method,
- Guess method,
- etc.

A detailed description of these and others is beyond the purpose of this document, but interested readers can refer to (Miettinen, 1999).

2.3.4 Comments

The methods described above aim at solving a MOP by means of converting it into a SOP. Therefore, classical single-objective optimization algorithms can be easily used to solve the resulting SOP. However, from these methods, we can observe some difficulties in finding multiple Pareto optimal solutions. For instance, only one Pareto optimal solution is expected to be found in a single simulation. Moreover, all Pareto optimal solutions could not be found by some methods in non-convex problems, or it could be required some knowledge about the problem for setting the weight or the ϵ.

In spite of the shortcomings of these methods, they have some strengths. For instance, the main one is the proof of convergence to a Pareto optimal set (interested readers are referred to (Miettinen, 1999) for details of this). Moreover, these methods are in general very simple and easy to implement.

2.4 Summary

In this chapter, we described the basics of multi-objective optimization. We highlighted that multi-objective optimization problems differ from the single-objective ones. In the case of having conflicting objectives, there does not exist a single solution that would be the best for all objectives, i.e., there is not a solution that simultaneously minimizes all objectives. For dealing with such cases, Pareto optimality provides a framework to determine the solutions that satisfy the best trade-off among the objectives, which was also described in the present chapter.

This chapter also introduced some classical methods which take into consideration the decision maker's preferences. Most of these methods convert the MOP into a SOP, for which a single-objective optimization algorithm can be used. There exist proofs of convergence for some of these methods, such that there is a guarantee that the solution of this single-objective problem corresponds to a Pareto optimal solution. However, they usually should be executed for a certain number of times in order to find different Pareto optimal solutions.

In the next chapter, we will present evolutionary algorithms, which have become popular techniques to solve MOPs. We will first give an overview of these algorithms and after that, we will explain how can they be used as an alternative for finding an approximation of the Pareto optimal set.

Chapter 3

Multi-Objective Evolutionary Algorithms

God created a number of possibilities in case some of his prototypes failed – that is the meaning of evolution.

Graham Greene

Evolutionary Algorithms (EAs) abstract the natural evolutionary principles into algorithms that can be used for searching optimal solutions to a problem. Unlike mathematical programming techniques; such as direct-search methods or gradient-based methods, which work with a single solution that is updated by following some deterministic rules, EAs maintain a population of solutions (also called individuals), which is evolved by using evolutionary operators such as selection, recombination, and mutation. Each individual in the population has a fitness value, which measures how good a solution is for the problem. A selection operator focuses on individuals with high fitness, in order to give preference to the fittest individuals to reproduce. Recombination and mutation operators aim at creating new individuals (solutions) by means of exchanging the information from the parents and perturbing the children. The children are then evaluated to know the corresponding fitness value. Finally, a survival step decides who survives in the population and this process is repeated until a stopping criterion is met.

Although simplistic from a biological point of view, these algorithms are complex enough to provide robust and powerful adaptive search mechanisms. In particular, due to their population-based search nature, they may be attractive for solving multi-objective optimization problems. The growing interest in using evolutionary algorithms for such problems has originated the so-called Evolutionary Multi-Objective Optimization area. This chapter is devoted to review some of the most popular evolutionary algorithms designed for multi-objective optimization. We first introduce the

main paradigms of this family of algorithms and after that, their extension for dealing with multiple conflicting objectives.

3.1 Evolutionary Algorithms: Paradigms

Evolutionary algorithms have become popular optimization techniques for solving a number of real world problems. Since the 1960s, a number of EAs have been proposed. Among these, the classical EAs include evolution strategies (ES) (Rechenberg, 1965, 1973; Schwefel, 1977), evolutionary programming (EP) (Fogel et al., 1966), and genetic algorithms (GA) (Holland, 1975). These classical EAs have inspired the development of other EAs, such as genetic programming (GP) (Koza, 1992), differential evolution (DE) (Storn and Price, 1997; Price et al., 2006), among others. In this section, we will focus on the three main paradigms, ES, EP, and GA, but a more comprehensive review can be found in (De Jong, 2006; Eiben and Smith, 2003; Yu and Gen, 2010).

3.1.1 Evolution Strategy

Evolution strategy (ES) was created in the mid-1960s by Ingo Rechenberg (1965) and was later developed by Hans-Paul Schwefel (1977). ES uses mutation mechanism as its main evolutionary operator. In its original version, ES considered a single parent which produced a single child, this is the so-called $(1+1)$-ES. The child becomes the parent for the next generation if it is better than the current parent; i.e., if the child has a better fitness value than the one obtained by the current parent; otherwise, the child is discarded. A new individual is generated in the following way:

$$\mathbf{x}' = \mathbf{x} + \mathcal{N}(0, \sigma) \tag{3.1}$$

where $\mathbf{x}$ is the current individual (parent), $\mathbf{x}'$ is the mutated individual, and $\mathcal{N}(0, \sigma)$ is a normally distributed random number vector with zero mean and standard deviation σ.

The standard deviation σ might be the same for all variables or it might be different for each variable. An evolution strategy does not only evolve the problem's variables, but it is also able to evolve its parameters; this is known as "self-adaptation". There are also other types of ES like $(\mu+\lambda)$-ES, (μ,λ)-ES, in which it is possible to perform a recombination, but it is normally a secondary operator (i.e., mutation is still the main operator). Algorithm 1 describes the $(\mu+\lambda)$-ES.

Algorithm 1 $(\mu + \lambda)$-ES

Require: $f(x)$: fitness function,
μ: parent population size,
λ: offspring population size,
A stopping criterion

Ensure: An optimized solution

1: Create μ individuals to form the initial population **P**
2: Evaluate the individuals in **P**
3: **while** a stopping condition is not satisfied **do**
4: Apply a recombination operator (crossover) over the individuals in **P** to create $\mathbf{Q}'$ with λ children
5: Apply a mutation operator (Equation (3.1)) over the individuals in $\mathbf{Q}'$ to create **Q** with λ children
6: Evaluate the individuals in **Q**
7: Choose the best μ individuals from $\mathbf{P} \cup \mathbf{Q}$ to form the population **P** for the next generation
8: **end while**

3.1.2 Evolutionary Programming

Evolutionary Programming (EP) was proposed by Fogel et al. (1966) in the mid-1960s. It was originally developed to simulate evolution as a learning process aiming to generate artificial intelligence. In the classical formulation, the predictors were evolved in the form of finite state machines. Later on, Fogel (1992) extended the initial work on EP. Nowadays, there are several variants of EP for optimizing real-valued parameters, which have become the *standard* EP (Deb, 2001; Eiben and Smith, 2003). Algorithm 2 shows the standard EP algorithm. This implementation of EP can be regarded as a $(\mu + \mu)$-ES without recombination.

3.1.3 Genetic Algorithms

Genetic algorithms (GAs) were first introduced by Holland (1975) in the early 1960 and were originally called *reproductive plans*. In the original formulation conceived by Holland (1975), a GA had a binary-string representation, proportional selection, and the recombination as the main evolutionary operator. Several changes to this formulation have been developed, including different kinds of representations, selection, recombination (crossover), mutation, and elitism operators. Nowadays, GAs are the

Algorithm 2 Evolutionary Programming

Require: $f(x)$: fitness function,
μ: population size,
A stopping criterion
Ensure: An optimized solution
1: Create μ individuals to form the initial population **P**
2: Evaluate the individuals in **P**
3: **while** a stopping condition is not satisfied **do**
4: Apply a mutation operator over the individuals in **P** to create **Q** with μ children
5: Evaluate the individuals in **Q**
6: Choose the best μ individuals from $\mathbf{P} \cup \mathbf{Q}$ to form the population **P** for the next generation
7: **end while**

most popular kind of evolutionary algorithms. Algorithm 3 describes a general GA.

3.2 Multi-Objective Evolutionary Algorithms

As we stated in chapter 2, in multi-objective optimization problems (MOPs) there does not generally exist a single solution that minimizes all objectives simultaneously, but a set of solutions that correspond to the best possible trade-offs among the objectives. When solving a MOP, one usually not only wants to find this set of trade-off solutions, but also that these solutions are well-distributed along the Pareto front. Evolutionary algorithms (EAs) are well suited for approximating the Pareto front of a MOP because they work with a population of solutions rather than with a single solution at a time, as normally happens with mathematical programming techniques. This enables approximating several members of the Pareto optimal set simultaneously in a single run of the algorithm. Furthermore, they are less susceptible than mathematical programming techniques to the shape and continuity of the Pareto front.

Since the seminal work of Schaffer (1985) in the mid-1980s, a considerable number of multi-objective evolutionary algorithms (MOEAs) have been proposed. For example: the Multi-Objective Genetic Algorithm (MOGA) (Fonseca and Fleming, 1993), Niched Pareto Genetic Algorithm (NPGA) (Horn et al., 1994), Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler and Thiele, 1999), and its improved version SPEA2 (Zitzler et al., 2001), Pareto Archived Evolution Strategy (PAES) (Knowles and Corne, 2000), Non-dominated Sorting Genetic Algorithm (NSGA) (Srinivas and Deb, 1994) and

Algorithm 3 Genetic Algorithm

Require: $f(x)$: fitness function,
N: population size,
A stopping criterion
Ensure: An optimized solution
1: Create N individuals to form the initial population **P**
2: Evaluate the individuals in **P**
3: **while** a stopping condition is not satisfied **do**
4: Select parents from **P**
5: Apply recombination to the selected parents via a crossover operator to create an offspring population $\mathbf{Q}'$
6: Apply mutation to the individuals in $\mathbf{Q}'$ to create **Q**
7: Evaluate the individuals in **Q**
8: Select individuals for the next generation
9: **end while**

NSGA-II (Deb et al., 2002), and Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) (Zhang and Li, 2007), among others. Next, we revise some of these popular MOEAs.

3.2.1 Pareto-Based MOEAs

These MOEAs are based on a common idea: the use of a non-dominated sorting procedure. They consider the non-dominated solutions as suitable candidates to be kept. Some of the most popular Pareto-based MOEAs are SPEA2, PAES, and NSGA-II, which are briefly described below.

Strength Pareto Evolutionary Algorithm (SPEA)

The Strength Pareto Evolutionary Algorithm (SPEA) was proposed by Zitzler and Thiele (1999). SPEA uses an external archive to keep the non-dominated solutions found during the search. At each generation, the newly found non-dominated solutions are compared with the existing ones in the external archive and the resulting non-dominated solutions are preserved. For each individual in this external set, a *strength* is computed. In SPEA, the fitness of each member is computed according to the strengths of all external non-dominated solutions that dominate it. SPEA not only preserves the non-dominated solutions in the external archive, but such non-dominated solutions

also participate in the genetic operations along with the current population, so that they steer the population towards potentially good search regions.

A revised version of this algorithm, called SPEA2, was proposed by Zitzler et al. (2001). Algorithm 4 shows the pseudo-code for SPEA2. SPEA2 has three main differences with respect to its predecessor: (1) it incorporates a fine-grained fitness assignment strategy which takes into account for each individual the number of individuals that dominate it and the number of individuals dominated by it; (2) it uses a nearest neighbor density estimation technique which guides the search more efficiently, and (3) it has an enhanced archive truncation method that guarantees the preservation of boundary solutions.

Algorithm 4 SPEA2

Require: $\mathbf{f}(\mathbf{x})$: fitness functions,
N: population size,
g: number of generations
Ensure: A set of non-dominated solutions
1: Initialize population $\mathbf{P_t}$
2: Create an empty external archive $\mathtt{EP} \rightarrow \emptyset$
3: **while** stopping criterion is not satisfied **do**
4: Compute the fitness of each individual in $\mathbf{P_t}$ and EP
5: Copy all non-dominated individuals in $\mathbf{P_t}$ and EP to $\mathbf{P_{t+1}}$
6: **if** $|\mathbf{P_{t+1}}|$ is greater than the archive size **then**
7: Use the truncation operator to remove elements
8: **else if** $|\mathbf{P_{t+1}}|$ is less than the archive size **then**
9: Use dominated individuals in $\mathbf{P_t}$ to fill EP
10: **end if**
11: Perform binary tournament selection with replacement to fill the mating pool
12: Apply evolutionary operators to the mating pool and set $\mathbf{P_{t+1}}$ to the resulting population
13: **end while**

Pareto Archived Evolution Strategy (PAES)

The Pareto Archived Evolution Strategy was introduced by Knowles and Corne (2000). PAES may represent the simplest possible (but not trivial) algorithm capable of generating diverse solutions in the Pareto optimal set. The algorithm is identified as being a (1+1) evolution strategy, using local search from a population of one but using a

reference archive of previously found solutions in order to identify the approximate dominance ranking of the current solution vector. A description of PAES is presented in Algorithm 5. An initial individual is first created, who serves as a parent to create new solutions. The next step is to create an offspring from the parent individual, which is achieved by applying a mutation operator to the parent. After that, a series of comparisons are performed in order to determine whether the child individual should be added to an external archive or not, and what solution should be the parent for the next generation. PAES stores the non-dominated solutions found so far during the search in an external archive.

Algorithm 5 PAES

Require: $\mathbf{f}(\mathbf{x})$: fitness functions,
N: population size,
g: number of generations
Ensure: A set of non-dominated solutions
1: Create an empty external archive $EP \rightarrow \emptyset$
2: Create an initial individual p_t
3: Add p_t to external archive $EP : EP = EP \cup p_0$
4: **while** stopping criterion is not satisfied **do**
5: Mutate p_t to create c_t
6: **if** $p_t \preceq c_t$ **then**
7: Discard c_t
8: **else if** $c_t \preceq p_t$ **then**
9: Replace p_t with c_t, and add c_t to archive $EP : EP = EP \cup c_t$
10: **else if** $\exists a \in EP \mid a \preceq c_t$ **then**
11: Discard c_t
12: **else**
13: Apply test (p_t, c_t, EP) to determine who becomes the new current solution and whether to add c_t to EP
14: **end if**
15: **end while**

One of the main characteristics of PAES is a novel approach to maintain diversity, which consists of a crowding procedure that divides objective space in a recursive manner. Each solution is placed into a certain grid location, based on the values of its objectives. A map of such a grid is maintained, indicating the number of solutions that reside in each grid location. Figure 3.1 shows an example of the adaptive grid used by

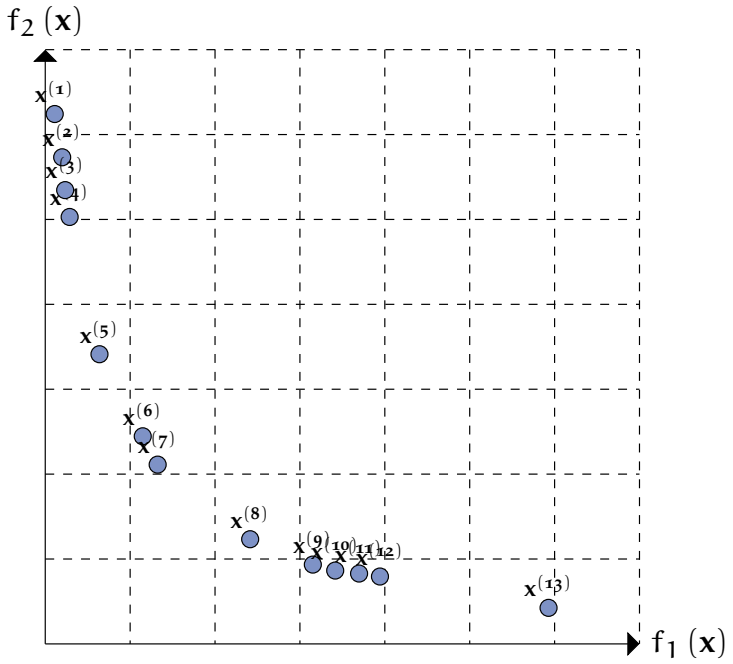


Figure 3.1: Example of the adaptive grid used by PAES

PAES.

Non-dominated Sorting Genetic Algorithm (NSGA)

Goldberg's ranking scheme based on the non-dominated sorting concept (Goldberg, 1989) was implemented in a more straightforward fashion by Srinivas and Deb (1994). The non-dominated sorting genetic algorithm (NSGA) sorts the population in different non-dominated layers. This allows classifying the population into a number of mutual equivalent classes (or non-dominated sets). In this classification, the first set is formed by the non-dominated solutions in the current population. The second set is formed by the non-dominated solutions of the current population, but excluding those belonging to the first set. The third set is composed by the non-dominated solutions of the remaining unranked individuals of the current population, and so on.

An improved version called NSGA-II was proposed by Deb et al. (2002). The pseudo-code of the NSGA-II is shown in Algorithm 6. As it is shown in Figure 3.2, NSGA-II ranks and sorts each individual according to its non-domination level, applies evolutionary operators to create an offspring population, and then combines the parents and offspring population and performs a partition into fronts on this combined population. The NSGA-II considers a crowding distance in the selection operator

in order to maintain diversity. Finally, the population for the next generation is selected from the combined population considering the ranked fronts and the crowding distance.

Algorithm 6 NSGA-II

Require: $\mathbf{f}(\mathbf{x})$: fitness functions,
N: population size,
g: number of generations
Ensure: A set of non-dominated solutions
1: Initialize population $\mathbf{P_t}$
2: Evaluate objective functions
3: Assign rank based on Pareto dominance
4: **while** a stopping criterion is not satisfied **do**
5: Generate child population $\mathbf{Q_t}$
6: Binary tournament selection
7: Evolutionary operations
8: **for** each parent and child in population **do**
9: Assign rank based on Pareto dominance
10: Generate set of non-dominated vectors
11: Add solutions to next generation starting from the first front until individuals found determine crowding distance between points on each front
12: **end for**
13: Apply elitism over the lowest front and those outside a crowding distance
14: Create next generation
15: **end while**

3.2.2 MOEA Based on Decomposition

The multi-objective evolutionary algorithm based on decomposition (MOEA/D) (Zhang and Li, 2007) is one of the most recent MOEAs reported in the state of the art. It is based on the idea of decomposing a MOP into a number of scalar objective optimization problems, also called subproblems, through a weighted aggregation of the objectives. MOEA/D minimizes all these subproblems iteratively in a single run. A neighborhood relation based on the distance of the aggregation weights vectors is defined among the subproblems. The optimal solutions to two neighboring subproblems should be very similar. Each subproblem has its best solution found so far in the population and is optimized in MOEA/D by using information from its neighbors. The definition of the

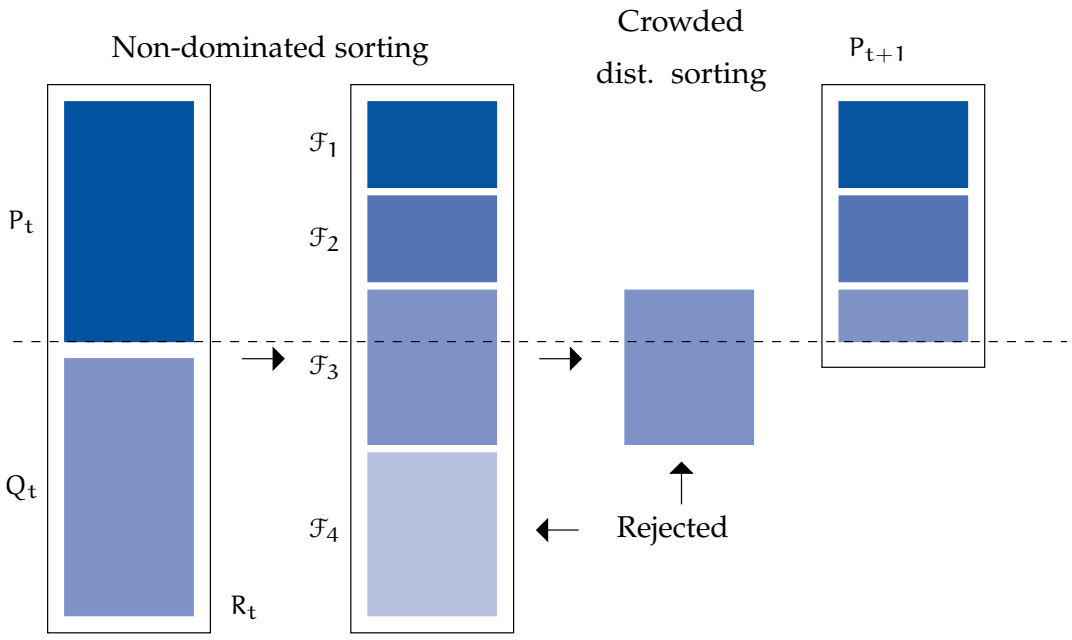


Figure 3.2: Diagram that shows the way in which the NSGA-II works. P_t and Q_t are, respectively, the parent and offspring populations at generation t. $\mathcal{F}_1$ is the best ranked set of solutions, $\mathcal{F}_2$ the second best, and so on.

weights plays an important role in MOEA/D so as to generate a good approximation of the Pareto front.

A description of MOEA/D is presented in Algorithm 7. MOEA/D starts by creating an empty external population (EP) (step 1), which is used to store the non-dominated solutions found so far during the search. In MOEA/D, the T closest weight vectors in $\{\lambda^1, \ldots, \lambda^N\}$ to a weight vector λ^i constitute the neighborhood of λ^i. Thus, for each vector λ^i, the Euclidean distance between this and the others is computed, and its closest T weight vectors are determined, where T defines the neighborhood size. The indexes of such T closest weight vectors are assigned to $B(i)$ (step 2). Next, the initial population of N individuals is randomly created (step 3). The individuals of the initial population are evaluated using the fitness functions. For each objective, the lowest value attained by the individuals in the initial population is used to initialize a reference vector $\mathbf{z}$ (step 4). Then, a new solution $\mathbf{y}$ is generated. To do this, the parents are randomly selected from the neighborhood, to which evolutionary operators (such as crossover and mutation) are applied to create $\mathbf{y}$ (step 7). In case $\mathbf{y}$ violates any constraint, the next step consists of applying some repair heuristic in order to make $\mathbf{y}$ a feasible solution (step 8). Next, the reference vector $\mathbf{z}$ is updated in case that an objective with a lower value is found (step 9). After that, the neighboring solutions are

updated by considering all the neighbors of the i^{th} subproblem and replacing $\mathbf{x}^j$ by $\mathbf{y}'$ if $\mathbf{y}'$ performs better than $\mathbf{x}^j$ (step 10). The external population EP initialized in step 1 is updated by the new generated solution if and only if this solution is non-dominated with respect to those that are in EP. Moreover, if the new solution dominates any of those stored in EP, such solutions are removed from EP (step 11). Steps 7 to 11 are repeated until a stopping criterion is met.

Algorithm 7 MOEA/D

Require: A stopping criterion,
N: number of subproblems considered in MOEA/D,
A uniform spread of N weight vectors: $\lambda^1, \ldots, \lambda^N$,
T: the number of weight vectors in the neighborhood of each weight vector
Ensure: EP: an external population

1: Initialize $EP \rightarrow \emptyset$
2: Compute the Euclidean distance between any two weight vectors and then work out the T closest weight vectors to each weight vector. For each $i = 1, \ldots, N$, set $B(i) = \{i_1, \ldots, i_T\}$, where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the closest weight vectors to λ^i.
3: Generate an initial population $\mathbf{x}^1, \ldots, \mathbf{x}^N$
4: Initialize $\mathbf{z} = [z_1, \ldots, z_m]$ by setting $z_j = \min_{1 \leqslant i \leqslant N} f_j\left(\mathbf{x}^i\right)$
5: **while** stopping criterion is not satisfied **do**
6: **for** $i = 1$ **to** N **do**
7: Randomly select two indexes k, l from $B(i)$, and then generate a new solution $\mathbf{y}$ from $\mathbf{x}^k$ and $\mathbf{x}^l$ by using genetic operators.
8: Apply a repair/improvement heuristic on $\mathbf{y}$ to produce $\mathbf{y}'$.
9: Update $\mathbf{z}$, for each $j = 1, \ldots, m$ if $z_j > f_j(\mathbf{y})$, then set $z_j = f_j(\mathbf{y})$
10: Update of Neighboring Solutions: For each index $j \in B(i)$, if $g^{te}\left(\mathbf{y}'\lambda^j, \mathbf{z}\right) \leqslant g\left(\mathbf{x}^j\lambda^j, \mathbf{z}\right)$, then set $\mathbf{x}^j = \mathbf{y}'$, $FV^j = F(\mathbf{y}')$
11: Update of EP: Add $F(\mathbf{y}')$ to EP if it is non-dominated with respect to the vectors stored in EP, and remove from EP the vectors dominated by $F(\mathbf{y}')$.
12: **end for**
13: **end while**

One of the key issues in MOEA/D is the method used for decomposing the MOP into a number of scalar objective problems. In this regard, there are several methods that allow constructing the aggregate functions, such as the weighted sum approach, the Tchebycheff approach, the penalty boundary intersection, etc. A review of these methods can be found in (Miettinen, 1999).

3.3 Performance Measures

As we previously stated, when solving a MOP, one wants (i) to find a set of non-dominated solutions that best approximate the Pareto optimal set, i.e., minimize the distance between the generated set and the Pareto set; and (ii) that the generated solutions are distributed along the Pareto front, i.e., maximize the diversity among the solutions in the generated set. In order to assess the effectiveness of a MOEA, several performance measures have been proposed. Here, we revised some of them.

3.3.1 Hypervolume Metric

The Hypervolume (HV) performance measure was proposed by Zitzler and Thiele (1999). This measure calculates the volume (in objective space) covered by the members of the generated non-dominated front. In a mathematical sense, for each solution in the resulting non-dominated set, a hyper-cube is constructed with a reference point $\mathbf{z}'$ and the solution has a diagonal corner with the reference point, i.e.,

$$HV = volume\left(\cup_{i=1}^{|\mathcal{P}|} v_i\right) \tag{3.2}$$

The reference point can simply be constructed with a vector of worst objective functions values. Figure 3.3 shows an example of the HV of a bi-objective problem. The hypervolume is shown in the shaded region. An algorithm with a larger value of HV is preferred.

3.3.2 Generational Distance and Averaged Hausdorff Distance

The generational distance (GD) measures the average distance between the true Pareto front, $\mathcal{PF}^*$, and the approximation, $\mathcal{PF}_{app}$ obtained by a MOEA. It is computed as:

$$GD = \frac{1}{|\mathcal{PF}_{app}|}\left(\sum_{i=1}^{|\mathcal{PF}_{app}|} d_i^p\right)^{(1/p)} \tag{3.3}$$

where d_i is the distance to the i^{th} member of $\mathcal{PF}_{app}$ and the closest one in $\mathcal{PF}^*$.

A variant of this performance measure is the *inverted generational distance*, in which instead of computing the distance between each member of the approximation and the closest one of the true Pareto front, the distance, $\hat{d}$, between each member of the true

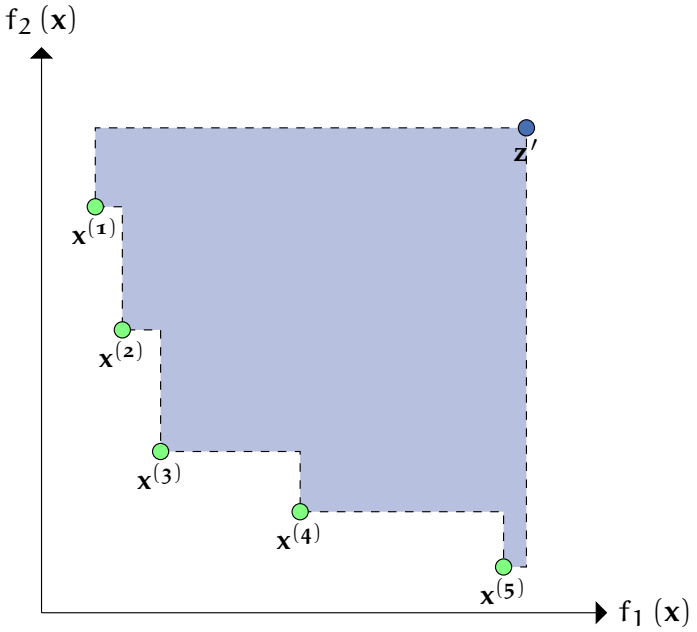


Figure 3.3: The hypervolume enclosed by the non-dominated solutions

Pareto front and the closest one in the approximation set is computed, i.e.,

$$\mathrm{IGD} = \frac{1}{|\mathcal{PF}^*|}\left(\sum_{i=1}^{|\mathcal{PF}^*|} \hat{d}_i^{\,p}\right)^{(1/p)} \tag{3.4}$$

Schütze et al. (2012) have pointed out some drawbacks of these two measures and they proposed alternative versions of them. These are the following:

$$\begin{aligned} \mathrm{GD}_p &= \left(\frac{1}{|\mathcal{PF}_{app}|}\sum_{i=1}^{|\mathcal{PF}_{app}|} d_i^{\,p}\right)^{(1/p)} \\ \mathrm{IGD}_p &= \left(\frac{1}{|\mathcal{PF}^*|}\sum_{i=1}^{|\mathcal{PF}^*|} \hat{d}_i^{\,p}\right)^{(1/p)} \end{aligned} \tag{3.5}$$

From these alternative versions, they proposed the averaged Hausdorff distance, (Δ_p), as a performance measure for assessing MOEAs. The averaged Hausdorff distance is computed as:

$$\Delta_p = \max\left(\mathrm{GD}_p, \mathrm{IGD}_p\right) \tag{3.6}$$

The averaged Hausdorff distance combines both generational distance and inverted generational distance, taking the maximum value among these.

3.3.3 Two-Set Coverage

It was proposed by Zitzler et al. (2000) and compares two sets. Let $\mathcal{A}$ and $\mathcal{B}$ be two sets that contain solutions vectors for the MOP, the two-set coverage measure, $\mathcal{C}(\mathcal{A}, \mathcal{B})$, computes the portion of solutions in $\mathcal{B}$ which are weakly dominated by those in $\mathcal{A}$, i.e.,

$$\mathcal{C}(\mathcal{A}, \mathcal{B}) = \frac{1}{|B|} |\{b \in \mathcal{B} | \exists a \in \mathcal{A} : a \preceq b\}| \tag{3.7}$$

The value of $\mathcal{C}(\mathcal{A}, \mathcal{B})$ ranges from 0 to 1, where 1 means that all elements in $\mathcal{B}$ are weakly dominated by $\mathcal{A}$, and a 0 value means that none of the members of $\mathcal{B}$ are weakly dominated by $\mathcal{A}$. We should recall from subsection 2.2.1 that the non-dominance relation is not symmetric, thus it is convenient to compute the two-set coverage in both directions, i.e., compute both $\mathcal{C}(\mathcal{A}, \mathcal{B})$ and $\mathcal{C}(\mathcal{B}, \mathcal{A})$.

3.4 Summary

In this chapter, evolutionary algorithms were described. First, the three main approaches in evolutionary computation were presented, which are evolutionary programming, evolution strategies, and genetic algorithms. These have inspired the creation of other evolutionary algorithms, such as differential evolution.

Proposals to extend these classic evolutionary algorithms for handling multi-objective optimization problems were also presented. Among these extensions, we found PAES (Knowles and Corne, 2000), NSGA-II (Deb et al., 2002), and MOEA/D (Zhang and Li, 2007). These MOEAs allow to generate an approximation to the true Pareto front in a single run. Some performance measures have been proposed to assess the performances of MOEAs when solving MOPs. Among them, the generational distance, averaged Hausdorff distance, and two-set coverage are described in this chapter.

Chapter 4

Supervised Learning

Whoso neglects learning in his youth, loses the past and is dead for the future

Euripides

Supervised learning is the task of inferring a function from a labeled dataset, i.e., a set of samples with known outputs. This function is a model that is used to predict the response of future data points from the same problem by mapping a data point from the feature space to an output space. Hence, the goal of supervised learning is to construct a model that generates a reasonable prediction for new data points.

Classification is a common tasks in supervised learning, in which the output is a categorical value that represents a class label. Its popularity relies on its application in a wide range of fields, such as medicine, astronomy, text recognition, etc. As a result, a number of learning algorithms to construct a classification model have been proposed, including decisions trees, artificial neural networks, or the k-nearest neighbors rule, among others. In spite of the variety of algorithms currently available, to date there is not a universal "best" one; this is sometimes referred to as the **No Free Lunch Theorem** (Wolpert, 1996).

Learning is done from data. The data, however, could contain irrelevant/redundant information, which could affect the quality of the learned model. Data pre-processing could help to alleviate this issue. Data pre-processing embraces several techniques for data sampling, data transformation, feature selection, etc. In this chapter we focus on describing some concepts related to supervised learning and metrics commonly used to assess the performance of the constructed models.

4.1 Classification

Classification is the task of estimating the output value of a data sample, where the output is characterized for being a categorical value. For constructing a classification model, we usually require a learning algorithm and a training dataset. A dataset used for training consists of a set of data samples, where each sample, $\mathbf{x} = \{x_1, x_2, \ldots, x_d\}$, is described by a set of d attributes. It also has a target attribute $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$ with k classes, which is called the class attribute. This dataset records a set of samples, also called instances, which represent the examples of the task to be learned.

Given a dataset, the objective of the learning algorithm is to produce a function to map a sample from the attribute space to a class label, i.e., $f(\mathbf{x}) : \mathbf{x} \rightarrow c \in \mathcal{C}$, where $\mathbf{x}$ is the sample and $\mathcal{C}$ is the set of class labels. This function can be used to predict the class labels of the future samples. This function is also called classification model or simply classifier.

There exists a number of learning algorithms that can be used for constructing the classification model. We describe some of the most popular approaches next.

4.1.1 Instance-Based Learners

Instance-based learners (IBL) are a kind of algorithms belonging to the lazy learning family. This means that, instead of performing a training phase in order to make an abstraction from data, IBL represents each training sample as a point in a multi-dimensional feature space, these points are stored in memory, and new samples are classified based on the labels assigned to their closest samples kept in memory.

The fundamental assumption on such algorithms is that similar instances will have similar classifications. The issue is how to define the similarity between a pair of instances. For doing so, a similarity function is used, which typically computes distance between instances. IBL has also a classification function which specifies how instance similarities yield a final classification. Examples of instance-based learners are the nearest neighbor algorithm, k^* algorithm, among others.

Nearest neighbor algorithm (Cover and Hart, 1967) is an instance-based learner, which uses a specific distance function to determine the single most similar instance from the training set. The class label of the most similar instance is given as the classification for the new instance. A generalization of the nearest neighbor rule is the k-nearest neighbor algorithm. The k-nearest neighbors of the new instance are found and the new instance's classification is based on the predominant class label among them.

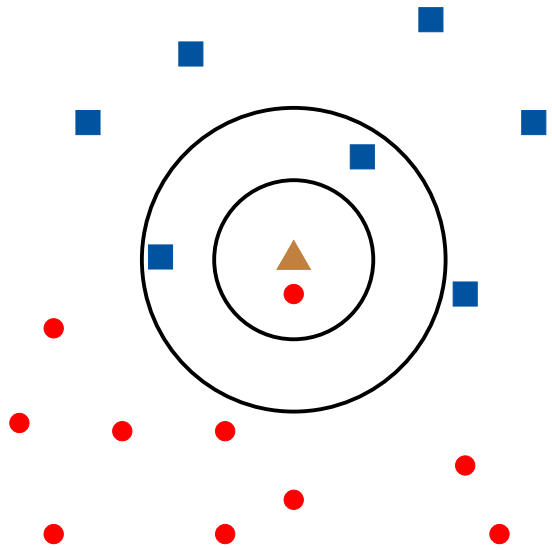


Figure 4.1: Example of instance-based learning. An unknown sample, marked with a triangle, is classified as a circle class based on the nearest neighbor, or as a square class based on k-nn, with $k = 3$.

Aha et al. (1991) describe three instance-based learners, called IB1, IB2, and IB3. IB1 is an implementation of a nearest neighbor algorithm using the Euclidean distance for real-value attributes and a Boolean function for nominal attributes. IB1 differs from the nearest neighbor algorithm in that IB1 normalizes the attributes to a common scale, it processes instances incrementally, and assumes that the missing values are maximally different to the present value. The IB2 algorithm is similar to IB1, except that IB2 reduces storage requirements by saving only the misclassified instances. IB3 extends the IB2 algorithm by maintaining a record of the saved instances, in order to summarize its classification performance, and by employing a significance test to determine which instances are good classifiers and which ones are believed to be noisy.

Cleary and Trigg (1995) proposed the $k^\star$ algorithm, which is essentially similar to other instance-based algorithms, in the sense that the classification of a new instance is based on the class labels of the most similar instances in the training set, but with the difference that $k^\star$ uses an entropy measure as the similarity function, which is inspired from information theory. $k^\star$ is a method that is generally very accurate (Amthauer, 2008).

Figure 4.1 shows an example of instance-based learners. Given an unknown sample, represented in this case by the brown triangle sample, this can be categorized as a red circle if the nearest neighbor rule is used. On the other hand, it can be categorized as a blue square class if the k-nearest neighbor rule is used, with a k value equal to 3.

4.1.2 Decision Trees

Decision trees are approaches that allow constructing a model following a divide and conquer strategy. Roughly speaking, a decision tree is a graphical representation in which each internal node is associated with a decision and the terminal nodes are generally associated with a class label. Each internal node is associated to test an attribute in order to decide what path should be taken. The path between two nodes is represented by a link, which contains the value of the decision. Therefore, a decision tree classifies an example by propagating it along a path from the root node down to a leaf node which contains the classification for this example.

The construction of decision trees generally involves a splitting process in order to choose an attribute at each internal node to make a decision. At the beginning, the most important attribute is chosen for being used in the root. At each node, the dataset is split and the outcome is used to construct a new decision tree. One of the main issues when constructing a decision tree is to determine what attribute should be chosen next, which is approached by selecting, at each level, the most discriminative one. A good attribute should be able of separating (as much as possible) the samples among the different classes, i.e., a good attribute is the one that decreases the impurity[1] in a set of samples as much as possible. There are several measures to determine the impurity, such as entropy-based, as it is used by the ID3 and C4.5 algorithms (Quinlan, 1986, 1993), Gini index, χ^2, or G-square, as they are used in CART (Breiman et al., 1984).

Random forest is another algorithm that belongs to decision tree family. A decision tree is said to be unstable because small changes in the dataset used to construct the decision tree lead to significantly different classifiers (Duda et al., 2001). This is exploited by random forest, which creates several partitions of the training dataset with replacement and fits a tree per each new dataset. After training, a new sample is classified by propagating on each individual tree in the forest and a vote is made over these individual predictions.

Figure 4.2 shows an example of a decision tree. In the root node is located the most discriminative feature, according to the adopted criterion. Links represent the path to be taken, based on the value of the feature. This is constructed in a recursive manner. The terminal nodes have the class labels and one of them is reached when an instance is classified.

[1] We say that a set of samples is pure when all samples belong to the same class. A set of samples is impure when it has more than one class.

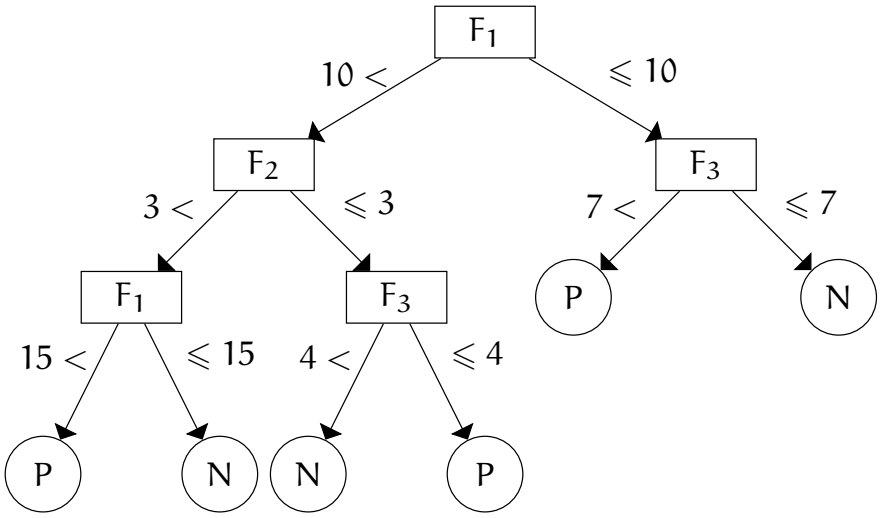


Figure 4.2: Example of a decision tree. The internal nodes represent the attribute in which a decision is taken, links represent the path to be taken based on the value of a feature and the terminal nodes represent the class label.

4.1.3 Neural Networks

An artificial neuron is a computational model inspired in the natural neurons. The first mathematical model of a single artificial neuron is credited to McCulloch and Pitts (1943). This neuron consists of a weighted sum of its inputs, followed by a non-linear function called the activation function. Formally,

$$y_k = \begin{cases} 1 & \text{if } \sum_j \omega_{kj} I_j - \mu_k \geqslant 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

The activation of the McCulloch-Pitts neuron can be generalized in the following form:

$$y_j = f_j \left(\sum_i \omega_{ji} \mathbf{I}_i \right) \tag{4.2}$$

where f_j is a (non-)linear activation function. Figure 4.3 shows an example of the McCulloch-Pitts neuron with 3 inputs and 1 output.

Rosenblatt (1958) studied the capabilities of a group of neurons in a single layer and he called this structure the "perceptron". Rosenblatt (1962) also proposed a learning rule for learning the weights in the Perceptron to be used in classification problems. However, Rosenblatt's perceptron was only able to discriminate linearly separable data. A modification to the percepton was introduced with the Multi-Layer Perceptron

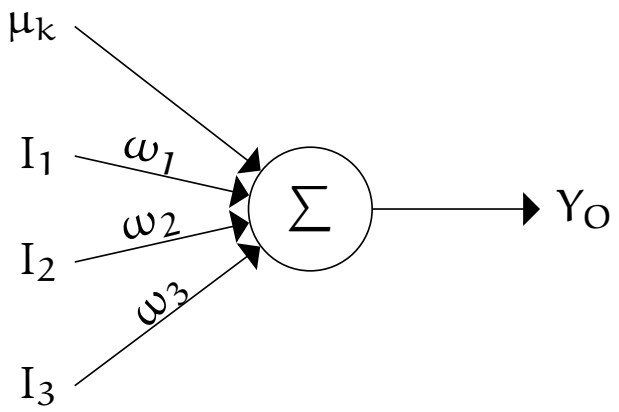


Figure 4.3: McCulloch-Pitts Neuron.

(MLP), which is able to distinguish data that are not linearly separable. A MLP is a kind of Artificial Neural Network (ANN). Figure 4.4 shows an example of a MLP with 3 layers.

The output of a MLP is computed by a single forward pass through the network, i.e.,

$$\begin{aligned} y_i^H &= f_H\left(\sum_j \omega_{ij}^{IH}\mathbf{I}\right) \\ Y_{O_i} &= f_O\left(\sum_j \omega_{ij}^{HO}\mathbf{y}^H\right) \end{aligned} \tag{4.3}$$

where f^H and f^O are, respectively, the activation functions in the hidden layer and the output layer, ω^{IH} is the weight matrix of the link from input to hidden layers, and ω^{HO} is the weights matrix from hidden to output layers.

As we can see from Equation (4.3), the output is a function that depends on the inputs and the weights values. Hence, the weights are fitted during the training phase of a MLP. The backpropagation algorithm (Werbos, 1974) is one method used for this end. The backpropagation consists of the following steps:

1. Assignment of initial weights
2. Forward pass, which simply computes the output value for the training samples using Equation (4.3).
3. Backward propagation, which propagates the error from the output layer to the input layer.

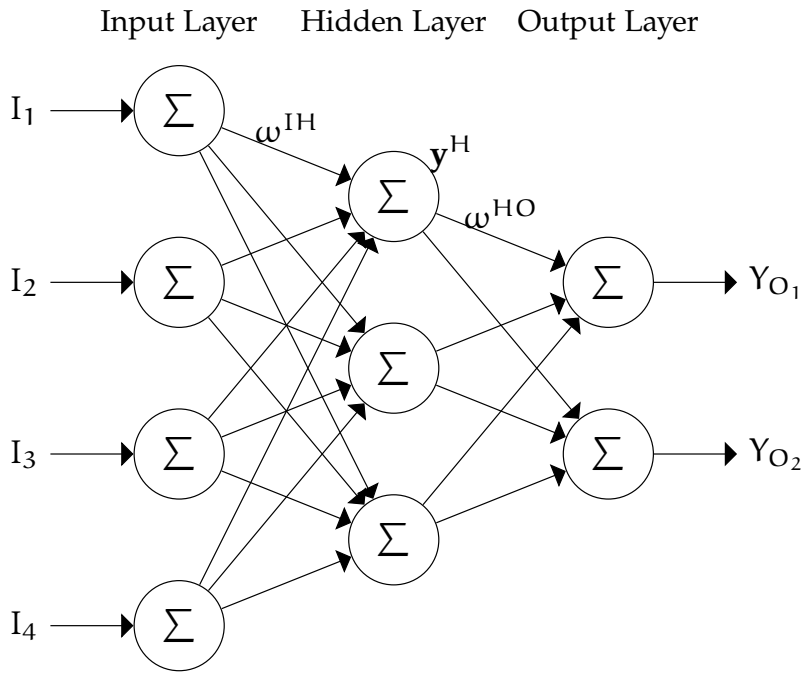


Figure 4.4: Example of a Multi-Layer Perceptron.

(a) Compute the error in the output layer,

$$\delta_O = -(y^* - Y_O)\, f'_O \tag{4.4}$$

where y^* is the desired output and f'_O is the derivative of the activation function.

(b) Compute the error in the hidden layer,

$$\delta_H = \sum_j \omega_{ij}^{HO} \delta_O f'_H \tag{4.5}$$

(c) Adjust the weights

Once the weights are adjusted in the learning phase, they are used for predicting the output of new samples by means of the propagation step.

4.1.4 Support Vector Machines

Support vector machines (SVMs) are supervised learning algorithms proposed by Vapnik (1995) that can be used both classification and regression. SVM were originally proposed for linearly separable classification problems. For a two-class classification problem, SVM finds the hyper-plane that maximizes the margin separation between

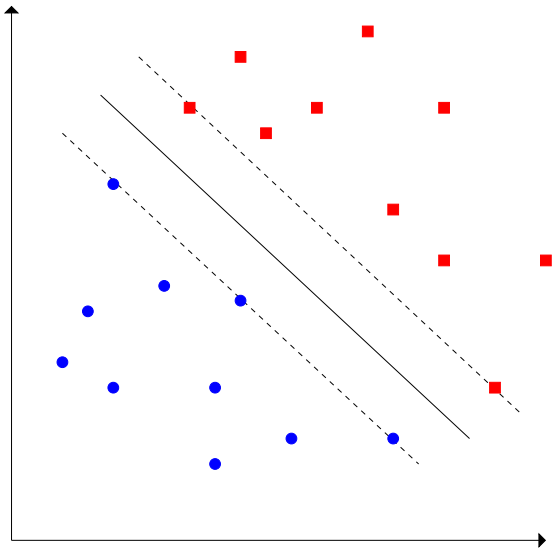


Figure 4.5: Example of a linear classifier with optimum separation hyper-plane

two classes. A special characteristic of SVMs is that the solution to the classification problem is represented by the support vectors that determine the maximum margin hyper-plane. The optimum separation hyper-plane is the linear classifier with the maximum margin for the given training set. Figure 4.5 shows an example of the optimum separation hyper-plane.

SVMs can also be used to non-linearly separable classification problems. In such cases, the data is mapped to a higher dimensional feature space using a kernel function, where the classes can be linearly separable. This is usually known as the *kernel trick*.

There are several libraries that implement the SVM, among them, we can list WEKA toolbox (Hall et al., 2009), LibSVM (Chang and Lin, 2011), LibLinear (Fan et al., 2008), etc.

4.2 Data Pre-Processing

Data pre-processing is another pattern recognition task. As the name suggests, the main idea is to prepare the data, by means of some type of processing, in order to be used for another process, such as the learning of a classification model. Examples of data pre-processing methods include data cleaning, data transformation, data sampling, and feature selection. We describe some pre-processing techniques next.

4.2.1 Noise Filtering

Datasets are not usually perfect and they frequently contain corrupt data. These meaningless data are noisy data. Noise has two main sources: implicit errors introduced by measurement tools, such as different types of sensors; and random errors introduced by batch processes or experts when data are gathered, such as in a document digitalization process (Zhu and Wu, 2004). Hence, these kinds of data can negatively affect the performance of a classification algorithm, introducing new properties in the problem domain. For instance, the boundaries of the classes and the overlapping between them can be affected as a consequence of noise.

Two types of noise can be distinguished in a given dataset (Zhu and Wu, 2004):

1. **Class noise**: this occurs when a sample is incorrectly labeled. Two types of class noise can be distinguished:
   - *Contradictory samples*: this happens when in a dataset there are two or more duplicated samples, but with different class labels.
   - *Missclassification*: this refers to a sample that is labeled with a class different to the real one.
2. **Attribute noise**: this refers to corrupted attribute values for one or more attributes.

Most of the works found in the literature are only focused on class noise (Frenay and Verleysen, 2014; García et al., 2014; Zhu and Wu, 2004). There exist methods for detecting and removing noisy points from a dataset. Some of them are described next.

- **Edited Nearest Neighbor (ENN)** (Wilson, 1972): ENN removes noisy points and those close to the decision boundary following a simple but effective rule. This rule consists of removing an instance X if it does not agree with the majority of its k nearest neighbors. This yields to smoother decision boundaries.
- **Relative Neighborhood Graph Edition (RNGE)** (Sánchez et al., 1997): RNGE constructs a proximity graph and an instance X is removed if it is incorrectly classified by its neighbors in the graph.
- **Iterative Partitioning Filter (IPF)** (Khoshgoftaar and Rebours, 2007): This method removes noisy instances in multiple iterations until a stopping criterion is reached. The stopping criterion activates if for a number of consecutive iterations, the number of identified noisy examples in each of these iterations is less than a

percentage training dataset size. IPF splits the training dataset into *NP* equal sized subsets, and for each subset it builds a classification model. These models are combined into an ensemble, which is used to predict the training dataset and those that are missclassified are removed and the process is repeated.

- **Ensemble Filter (EF)** (Brodley and Friedl, 1999): EF splits the training dataset into k equal sized subsets using k-fold cross validation (see subsection 4.3.2). Using these subsets, it constructs M classification models and performs the cross validation over them. Then, they vote the class label of each sample in the training dataset and those that are not correctly classified are removed.

4.2.2 Data Sampling

Sampling allows to work with a small or a large amount of data in order to construct models more quickly, while still producing accurate predictions. It can also help to deal with unbalanced datasets by means of duplicate samples (oversampling) or removing some of them (undersampling). Some approaches for data sampling are described in the following:

- **Random Sampling**: It consists of randomly choosing a set of samples from the training dataset to be either removed or duplicated. A slight variant is the stratified version, where the idea is to preserve the proportions of the classes.

- **Synthetic Minority Over-sampling Technique (SMOTE)**: SMOTE creates artificial samples based on the feature space similarities between existing minority samples. For a sample from the minority class, SMOTE takes its k nearest neighbors. A new sample is created by randomly selecting one out of the k neighbors, then multiplying the corresponding feature vector difference with a random number in the range $[0, 1]$.

- **Instance selection (IS)**: IS aims at selecting a relevant instances subset from the entire training set, while preserving the performance of the whole dataset. There is a variety of IS techniques, among which we can find the Edited Nearest Neighbor (ENN) (Wilson, 1972), Relative Neighborhood Graph Edition (RNGE) (Sánchez et al., 1997), the Decremental Reduction Optimization Procedure (DROP) (Wilson and Martinez, 2000), Fast Condensed Nearest Neighbor (FCNN) (Angiulli, 2007), etc. In DROP, an instance X is removed if its associates can be correctly classified without such an instance X. FCNN starts with an empty set of instances, $\mathcal{S}$, and it runs over all instances in the training set. An

instance is added to the set S if it is wrongly classified by its k nearest neighbors when S is used as the training set. A comprehensive review and taxonomy of these techniques can be found in (García et al., 2012, 2014)

4.2.3 Data Transformation

Data transformation aims at transforming or consolidating data into an appropriated format in order to improve the efficiency of the mining process, and the patterns found may be easier to understand (García et al., 2014). Data transformation encompasses normalization, discretization, data projection, among others. These are described next.

- **Normalization**: Some learning algorithms can be susceptible to the way in which features are scaled. Hence, it is desirable to have all features expressed in a common range. Normalization aims at scaling numerical values of a numerical feature to a given *min-max* range. Let $\left[X'_{min}, X'_{max}\right]$ be the range in which features will be scaled, the normalized value of a sample X is computed as:

$$X' = \frac{X'_{max} - X'_{min}}{X_{max} - X_{min}} (X - X_{min}) + X'_{min} \tag{4.6}$$

- **Standardization**: Another type of feature scaling is the standardization, also known as the Z-score. In this technique, the main idea is that all features have a mean zero and unit standard deviation. Therefore, the standardized value is computed as:

$$x'_i = \frac{x_i - \mu_i}{\sigma_i} \tag{4.7}$$

  where x_i is the value sample in the i^{th} feature, μ_i and σ_i are the mean and standard deviation of the i^{th} feature, respectively.

  Both normalization and standardization could help to improve some models. For instance, in the *k*-NN algorithm all features will have the same weight; for neural networks, it can improve the convergence speed during the optimization of the gradient descend

- **Discretization**: Some learning algorithms have been developed to work with nominal features. However, the classification problem can involve some real-valued or continuous features, which can narrow the applicability of such algorithms. Discretization transforms the continuous values to a set of discrete ones. This can be accomplished in a supervised or unsupervised manner.

In unsupervised discretization, the class label is not taken into account. Representative unsupervised discretizers are equal width (Wong and Chiu, 1987) and equal frequency (Wong and Chiu, 1987). Equal width intervals involves sorting the continuous values and dividing the range into k equally sized bins, i.e.,

$$\delta = \frac{X_{max} - X_{min}}{k} \tag{4.8}$$

and construct the boundaries, or thresholds, as $X_{min} + i \times \delta$ for $i\{1, \ldots, k-1\}$. This method is applied to each feature independently.

On the other hand, supervised discretizers make use of the information for the class labels. They are more sophisticated and are able to automatically determine the number of bins. Some supervised discretizers are the Class-Attribute Interdependence Maximization (CAIM) (Kurgan and Cios, 2004), Chi-merge (Kerber, 1992), and Minimum Description Length Principle (MDLP) (Fayyad and Irani, 1993), among others. A comprehensive review and experimental comparison among them is performed by García et al. (2013).

- **Nominal to binary**: Some datasets can be composed by nominal features. This can be a shortcoming for those learning algorithms that cannot correctly handle them, such as SVMs or ANNs. Nominal to binary transformation aims at avoiding the aforementioned shortcoming by mapping each nominal feature to a set of newly generated ones. Let k be the number of different values of a nominal attribute. These values are mapped to a new set of k features, and only one of these will have a value of 1 while the rest will have a value of 0, which is determined by the nominal value.

4.2.4 Feature Selection

Feature selection is one of the tasks of pattern recognition. The main idea is to select a subset of features from the original one that would be relevant to describe the dataset. Feature selection could help to improve the classification performance by removing the irrelevant or redundant features that could be present in the dataset. There are three main approaches for feature selection: filter, wrapper, and embed methods. For the purpose of this document, only the former two are explained.

Filter Feature Selection

Filter approaches generally use properties of the dataset in order to evaluate the merit of each feature or a subset of features through an independent measure. This approach

tends to be computational efficient since the evaluation criterion does not involve any learning algorithm.

Some filter approaches for feature selection include:

- Information gain, which evaluates the merit of each feature using information theory. The information gain of a feature is determined as:

$$I(Class, Feature) = H(Class) - H(Class|Feature) \tag{4.9}$$

where

$$H(Class) = -\sum p(Class)\log(p(Class))$$
$$H(Class|Feature) = -\sum\sum p(Feature, Class)\log(p(Class|Feature))$$

A characteristic of this method is that it works with nominal features, in case of numerical features, these can be converted to nominal ones by applying a discretization method. The features are ranked according to their information gain value and a threshold can be defined in such a way that those features whose value is above the threshold are selected.

- The χ^2 statistic can also be used for feature selection (Liu and Setiono, 1995). The main idea is to rank each feature based on its χ^2 value, which can be computed as follows:

$$\chi^2 = \sum_i \sum_j \frac{(A_{ij} - Eij)^2}{E_{ij}} \tag{4.10}$$

with

$$E_{ij} = \frac{\sum_j A_{ij} \times \sum_i A_{ij}}{N} \tag{4.11}$$

where j and i run, respectively, over all possible classes and intervals, A_{ij} is the number of samples in the i^{th} interval for the j^{th} class, E_{ij} is the expected frequency, and N is the total number of samples.

Analogously to information gain, numerical features should be first discretized. Those features whose χ^2 value is above a predefined threshold are selected.

- RELIEF (Kira and Rendell, 1992) ranks each feature according to its merit. RELIEF uses an instance-based approach for weighting each feature. This is achieved by

repeatedly sampling an instance and considering the value of the given feature for the nearest instance of the same and from other class. Therefore, a feature would receive a high weight if it differentiates between the sample and the one with the other class and if it has a similar value to the one of the same class. The weight of each feature is updated using the following expression:

$$\omega_{F_i} = \omega_{F_i} - \mathrm{diff}(x_i, h_i)^2 + \mathrm{diff}(x_i, m_i)^2 \tag{4.12}$$

where x_i is the i^{th} value of the sampled instance, h_i and m_i are, respectively, the i^{th} value of the nearest sample from the same class and from the other class, and $\mathtt{diff}$ is a function that computes the difference between two instances in a given feature.

The weight of each feature reflects its ability to distinguish among the values. Features are ranked by weight and those that are above a threshold are selected.

- Correlation subset feature selection (Hall, 1998) evaluates the merit of a subset of features, instead of a single one. This method considers that a good subset of features should have a high correlation with the class and low intercorrelation among them. The merit of a subset of features is estimated as follows:

$$M_{subset} = \frac{k c_{fc}}{\sqrt{k + k(k-1)\, c_{ff}}} \tag{4.13}$$

where k is the number of features in the subset, c_{fc} is the mean value of the correlation between a feature and the class, and c_{ff} is the mean value of the intercorrelation feature to feature.

This method requires a search strategy, such as forward aggregation, backward elimination, or heuristic search. The goal is to choose the subset that maximizes the merit computed with Equation (4.13).

Some properties of the filter approach include (García et al., 2014):

- the criterion to evaluate the features is usually cheaper than an estimation through the classification performance. Thus, filter methods are usually faster;
- since they do not use the learning algorithm to perform selection, a learning bias is not introduced, making that the selected features can be used to construct classification models using different learning algorithms;
- due to the low complexity of the evaluation criterion, they can deal with large sized data.

Wrapper Feature Selection

In contrast to the filter approach, wrapper methods use the learning algorithm as a black-box to evaluate the goodness of a subset of features. In general, the potential subset of features is used to construct a classification model, whose performance is evaluated and it serves as an indicator of how good such subset is. Therefore, this approach is more computationally expensive than filter methods.

A search strategy is required to explore the space of feature subsets. Some examples of search strategies include branch and bound, genetic algorithms and particle swarm optimization, among others.

4.3 Performance Assessment

Once a model is constructed, one important question is how to assess its predictive performance on unknown samples. The ability to correctly classify these unknown samples is called *generalization capability*. One usually wants to find a model with a good generalization capability. Therefore, the evaluation is crucial, since it can tell us how good a particular model or classifier is for a particular problem. In this section, we describe some evaluation methods used to assess the expected performance of a model.

4.3.1 Measurements of Model Performance

Before explaining the techniques for assessing the model performance, some measurements are introduced.

Figure 4.6 shows an example of a confusion matrix for a binary classification problem. From it, several scores or measurements can be computed. The list of scores in supervised classification may be large, including standard scores and those designed for specific classification problems. Here, we revisit the best well-known scores.

Among the existing scores, some of the most popular are the following:

- **Accuracy** measures the portion of samples that are correctly classified, i.e.,

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4.14}$$

  Accuracy ranges from 0 to 1, in which 0 means all samples are incorrectly classified and 1 that all samples are correctly classified.

- The **error rate** is the complement of accuracy and is computed as:

$$\text{Err} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4.15}$$

		Predicted Class $\hat{C}^+$	Predicted Class $\hat{C}^-$
Correct Class	C^+	TP	FN
	C^-	FP	TN

Figure 4.6: Confusion matrix for a binary classification problem. It shows the positive samples that are correctly classified (TP), the positive samples that are incorrectly classified (FN), the negative samples that are incorrectly classified (FP), and the negative samples that are correctly classified (TN)

- The **true positive rate** is the portion of positive samples that are classified as positive, and is computed as:

$$TPR = \frac{TP}{TP + FN} \tag{4.16}$$

- Analogously the **false positive rate** is the portion of negative samples that are classified as positive, and is computed as:

$$FPR = \frac{FP}{TN + FP} \tag{4.17}$$

- **Average accuracy** is an alternative measure to the overall accuracy when the classes proportion of the dataset is not well balanced. The average accuracy computes the mean of the accuracy per class, i.e.,

$$\overline{Acc} = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) \tag{4.18}$$

  Average accuracy measures the balanced performance of a classification model between the classes.

- The Receiver Operating Characteristic (ROC) curve is a graphical representation between true positive rate and false positive rate. When the output of a classification model is the class label, the ROC curve corresponds to a single point. In classification models where the output is the probability of belonging to a class label, the ROC curve is constructed by varying the discriminative threshold in order to obtain multiple points. Figure 4.7 shows an example of a ROC curve. The **Area under ROC curve** (AUC) summarizes the information about the classification performance in the ROC curve and is also used as a measure of the classification model performance.

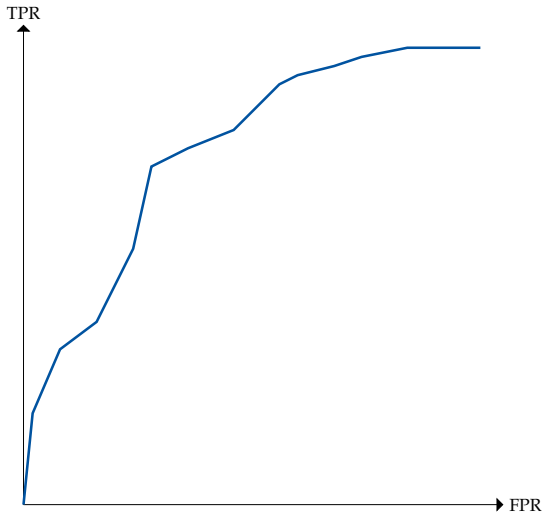


Figure 4.7: Example of a Receiver Operating Characteristic (ROC) Curve

- The **Kappa statistic** measures the degree of agreement between two categorical variables. In classification, the categorical variables correspond to the predicted class and the desired class. The Kappa statistic can be computed as:

$$k = \frac{Acc - P_e}{1 - P_e} \tag{4.19}$$

with

$$P_e = \frac{1}{N^2}\Big[(TP + FN)(TP + FP) + (TN + FP)(TN + FN)\Big] \tag{4.20}$$

where N is the number of samples, i.e., $N = TP + TN + FP + FN$.

This measure ranges from -1 (total disagreement) through 0 (random classification) to 1 (perfect agreement).

The scores have been described for the binary classification case; some of them, however, can be extended to multi-class classification problems. In the case of the area under the ROC curve (AUC), Hand and Till (2001) describe a simple generalization for multi-class problems. These measurements could serve as indicators of the performance of a particular model. Next, some evaluation strategies are presented.

4.3.2 Mechanisms for Validation

Replication is probably one of the most common approaches to evaluate the performance of a classification model. There are different types of replication mechanism

for validation. The simplest one is to split the dataset into two disjoint subsets called training set, which is used to fit the model, and validation set, which is used for computing some measurement, such as the error, to evaluate the model. This approach is usually known as *hold-out*. This approach is useful for large-sized datasets.

Nonetheless, the amount of available data could not be large enough. Therefore, it could be useful to perform multiple splits of the dataset and averaging the measurements obtained in each split. One way of doing this is to split the dataset into K disjoints sets of roughly equal size. At each iteration, the i^{th} subset is used as a test-set and the rest as training-set. At the end, the results obtained in each iteration are averaged. This is known as *K-fold cross validation*.

When the proportion of the classes is also preserved in the subsets, this is called *stratified K-fold cross validation*. Moreover, when the value of K is equal to the dataset size, this is known as *Leave-One-Out Cross Validation*.

In K-fold cross validation, the dataset is split in a random fashion, which could result in variation of the results. This variation could be reduced by performing Q times the K-fold cross validation. This is referred to as $Q \times K$*-fold cross validation*. At the end, the results obtained in the Q runs are averaged.

4.3.3 The VC Dimension

Vapnik and Chervonenkis defined the VC dimension (Vapnik, 1995) as a measure of the capacity/complexity of a learning algorithm. The VC Dimension is defined through the notion of "shattering", which is described as follows: if we have a set of n samples that can be separated by a set of indicator functions F (functions that map a sample to its corresponding binary label) in all 2^n possible ways, we say that the set of samples is shattered by the set of functions F. The VC dimension can be formally defined as (Cherkassky and Mülier, 2007):

> A set of functions F has a VC dimension h if there are h samples that can be shattered by the set of functions F, but there are not $h+1$ samples that can be shattered by the set of functions F.

Figure 4.8 shows an example of the VC dimension for a linear function in a 2-dimensional space. We say that this function has a VC dimension equal to 3, since it can distinguish the examples among all its 8 possible configurations. In general, a linear function has a VC dimension of $d+1$, where d is the dimensionality of the problem.

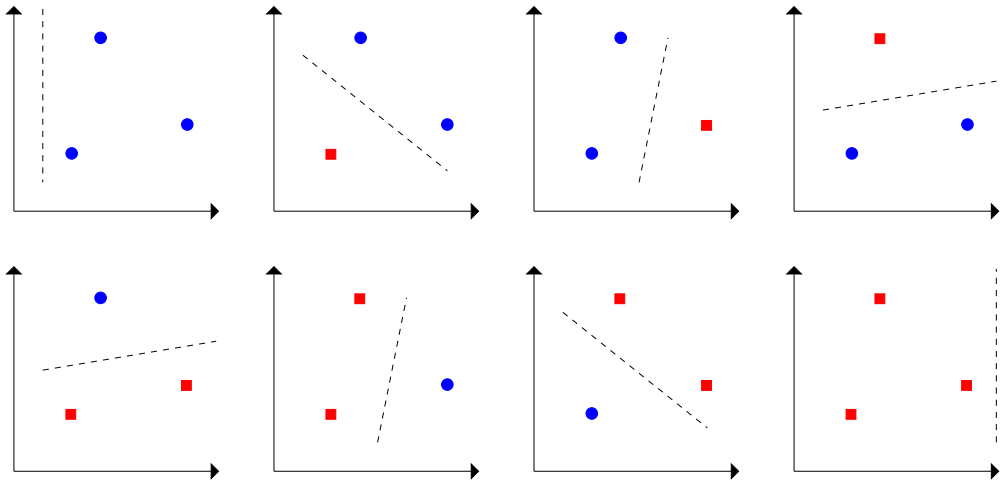


Figure 4.8: VC dimension of a linear function. A linear function in a 2-dimensional space has a VC dimension equal to 3, since it is capable of separating 3 points in its 8 possible configurations.

Notwithstanding that the VC dimension can be seen as a measure of the model complexity (Guyon, 2009), exact analytic estimates of this are only known for a few classes of functions (linear models), whereas for many others it is unknown. To overcome this, Vapnik et al. (1994) proposed a method to experimentally estimate the effective VC dimension of a model. This approach is based on the best fitting between an analytic formula and measurements of the maximum deviation between the error rates on two independent data sets of varying sizes. Conceptually, this approach can be applied to any learning algorithm (Cherkassky and Mülier, 2007).

The maximum deviation, $\xi(n)$, of the error rates between two independent labeled data sets is defined as:

$$\xi(n) = \max_{\omega} \left(\left| \operatorname{err}\left(\mathbf{Z}_n^1\right) - \operatorname{err}\left(\mathbf{Z}_n^2\right) \right| \right) \tag{4.21}$$

where $\mathbf{Z}_n^1$ and $\mathbf{Z}_n^2$ are two independent labeled data sets of size n, $\operatorname{err}(\mathbf{Z}_n)$ is the error rate on the data set $\mathbf{Z}_n$, and ω is the set of parameters of a binary classifier.

As it is stated by Vapnik et al. (1994), $\xi(n)$ is bounded as follows:

$$\xi(n) \leqslant \Phi(n/h) \tag{4.22}$$

where

$$\Phi(\tau) = \begin{cases} 1 & \text{if } \tau < 0.5 \\ a\frac{\log(2\tau)+1}{\tau-k}\left(1+\sqrt{1+\frac{b(\tau-k)}{\log(2\tau)+1}}\right) & \text{if } \tau \geqslant 0.5 \end{cases} \tag{4.23}$$

where $\tau = n/h$, and the values of the parameters $a = 0.16$ and $b = 1.2$ were empirically determined. The value of $k = 0.14928$ is determined such that $\Phi(0.5) = 1$.

Since the bound in Equation (4.22) is tight, it can be assumed that

$$\xi(n) \approx \Phi(n/h) \tag{4.24}$$

The VC dimension h can be estimated from Equation (4.23) and Equation (4.24). The maximum deviation $\xi(n)$ can be estimated by simultaneously minimizing the error rate on one labeled set and maximizing the error rate in the other one. This can be accomplished through the following procedure (Cherkassky and Mülier, 2007; Vapnik et al., 1994):

1. Generate a random labeled set $\mathbf{Z}_{2n}$ of size $2n$.
2. Split the set $\mathbf{Z}_{2n}$ into two sets of size n: $\mathbf{Z}_n^1$ and $\mathbf{Z}_n^2$.
3. Flip the labels of the set $\mathbf{Z}_n^1$, to form $\overline{\mathbf{Z}}_n^1$.
4. Merge the two sets: $\overline{\mathbf{Z}} = \overline{\mathbf{Z}}_n^1 \cup \mathbf{Z}_n^2$, and train the binary classifier with the set $\overline{\mathbf{Z}}$.
5. Evaluate $\mathbf{Z}_n^1$ and $\mathbf{Z}_n^2$ with the trained classifier. Measure the difference of the error rates between the two sets: $\xi(n) = \mid \text{err}\left(\mathbf{Z}_n^1\right) - \text{err}\left(\mathbf{Z}_n^2\right) \mid$.

This procedure gives an estimate of $\xi(n)$ from which an estimate of h can be obtained. In order to reduce the variability in the estimation, this procedure is repeated for different data sets varying the samples sizes $n_1, \ldots, n_k$. Moreover, to reduce the variability due to the random samples, the procedure is repeated several times (m_j) for each sample set of size n_i. The average value for each experiment is taken for each n_i: $\overline{\xi}(n_1), \ldots, \overline{\xi}(n_k)$. The effective VC dimension can be estimated by finding the parameter h^* that best fits $\xi(n)$ with the theoretical formula $\Phi(n/h)$, as follows:

$$h^* = \underset{h}{\text{argmin}} \sum_{i=1}^{k} \left[\overline{\xi}(n_i) - \Phi(n_i/h)\right]^2 \tag{4.25}$$

4.4 Summary

This chapter described some concepts related to pattern recognition, such as classification, pre-processing, and evaluation techniques. Some classification paradigms were presented in a general fashion, such as instance-based learning, decision trees, and neural networks. Paradigms for pre-processing were also presented. Pre-processing methods encompass noisy filtering, data sampling, data transformation and feature selection. Finally, strategies as well as some measurements for evaluating the model performance were presented.

Part III

Contributions

Chapter 5

Multi-Objective Model Type Selection

Essentially, all models are wrong, but some are useful.

George E. P. Box

Today, there is a number of learning algorithms reported in the state-of-the-art, and these could have a set of hyper-parameters that can be tuned, resulting in a excessively large number of possible models overall. Each of these models could achieve different performance using the same dataset. Selecting the appropriate hyper-parameters is a constant problem in machine learning, and this is referred to as *model selection*.

This chapter describes a proposed approach for dealing with the model type selection problem, which in this study is interpreted as the problem of choosing both the learning algorithm and its hyper-parameters for a given classification task. Firstly, some previous works related to the model selection problem are described in section 5.1. The proposed method, called **MOMTS**: ***M**ulti-**O**bjective **M**odel **T**ype **S**election*, is presented in section 5.2. Next, section 5.3 details the experiments performed and some final remarks are given in section 5.4.

5.1 Related Works

In the literature, there are several studies that address the model selection problem. Among these, some of them have approached it as an optimization problem, differing in the search technique adopted, including gradient-based methods (Ayat et al., 2005; Bengio, 2000; Cawley and Talbot, 2007), grid-search (Chang and Lin, 2011), or bio-inspired meta-heuristics such as evolutionary algorithms (Chatelain et al., 2010; Friedrichs and Igel, 2005; Gorissen et al., 2009; Li et al., 2011; Lorena and de Carvalho, 2008), artificial immune systems (Aydin et al., 2011) or particle swarm optimizers (Bao

et al., 2013; Escalante et al., 2009; Lin et al., 2008), etc.

Grid search requires to discretize the search space, which is attained by the variation of each hyper-parameter with a step size through a wide range of values and the performance of each combination is typically assessed through a k-fold cross-validation technique. Such cross-validation makes grid search a computationally expensive method which is suitable only when few hyper-parameters need to be set. The way in which the search space is discretized is another crucial issue in grid search.

Gradient-based methods are highly efficient and have been successfully applied to hyper-parameter optimization for SVMs. Notwithstanding, they still have some drawbacks. For instance, the objective function has to be differentiable with respect to the hyper-parameters (Bao et al., 2013; Friedrichs and Igel, 2005; Suttorp and Igel, 2006). Moreover, the effectiveness of these methods highly depends on the initial point chosen for the search, which causes that they can be susceptible to getting trapped in a local optimal solution.

Several studies have adopted evolutionary algorithms to alleviate the above mentioned shortcomings, since they are more robust to local optimal solutions than gradient-based methods, and they can be computationally cheaper than grid search methods. Furthermore, other advantages over gradient-based methods are that evolutionary algorithms are derivative-free and they can be parallelized.

Another major issue in model selection is the criterion used for this purpose. In this direction, we can differentiate the works that consider a single-objective criterion and those that consider multiple criteria. The single-objective criterion approaches are generally based on an estimation of the generalization error through the well-known K fold cross validation (Bao et al., 2013; Escalante et al., 2009; Lorena and de Carvalho, 2008; Sun et al., 2012). Attention has also been paid on considering multiple criteria. These works typically consider the model performance and some criterion for penalizing the model complexity (Aydin et al., 2011; Suttorp and Igel, 2006). Others have considered either to minimize the sensitivity and specificity (Chatelain et al., 2010; Li et al., 2011), or different estimates of the model performance (Gorissen et al., 2010, 2009; Pilát and Neruda, 2013). Some studies, alternatively, have approached multiple criteria by means of simplifying the objectives in a weighted linear combination of these (Sánchez-Monedero et al., 2011) instead of optimizing simultaneously each objective.

Despite these efforts, most of the existing studies consider a single model type (i.e., the learning algorithm is fixed *a priori* and the model selection task consists of choosing its hyper-parameters), which could not be the most suitable solution for a particular

problem. To the best of the author's knowledge, nowadays the works that address both the learning algorithm and the hyper-parameters selection are scarce (e.g. (Escalante et al., 2009; Gorissen et al., 2009; Sun et al., 2012)), and most of them tackle the problem as a single-objective one. Notwithstanding, several studies (Chatelain et al., 2010; Gorissen et al., 2010; Jin and Sendhoff, 2008) had shown that considering the complexity can help reduce the overfitting in the learned models.

This chapter introduces **MOMTS**: Multi-Objective Model Type Selection, a multi-objective approach that addresses both the problem of choosing a learning algorithm and its hyper-parameters during the model selection process. The error on training samples and the model complexity are considered as the objectives to be optimized. Unlike previous works in which the model complexity estimation depends on the learning algorithm (e.g., the number of support vectors in support vector machines), MOMTS estimates it through the VC-dimension (Vapnik, 1995).

The main contribution of this chapter is to propose a general model selection framework, whose formulation makes it applicable to any learning algorithm. Additional contributions are the following:

- a multi-objective approach for tackling the model type selection problem (i.e., model type plus its hyper-parameters),
- the use of the VC-dimension in the model type selection formulation for estimating the model complexity to any model type, and
- since the outcome of the multi-objective optimization process is a set of solutions (models), that satisfy an optimal trade-off between the objectives from which a model should be chosen, the strategies proposed for constructing a final classification model from the set are an additional contribution.

The performance of the proposed approach is assessed on several binary classification benchmark datasets widely used in the literature. The experimental results and comparisons show that MOMTS is able to select highly effective classification models.

5.2 Multi-Objective Approach for Model Selection

This section describes the proposed approach MOMTS: Multi-Objective Model Type Selection. In MOMTS, the training error and the model complexity are considered as the objectives to be minimized. The general approach of MOMTS is shown in Figure 5.1. The process starts by creating an initial population, for which in our problem, each individual in the initial population represents a potential model for a classification task.

After that, the criteria to be optimized are computed. Next, the models are evolved through by applying some evolutionary operators to create an offspring population, which represents new potential models for the given classification task. Thereafter, the models that satisfy the best trade-off between the two objectives to be optimized are stored in an external archive. This process is repeated until a stopping criterion is reached. At the end of the search, a final classification model is constructed, which is used for predicting the class labels of unknown samples.

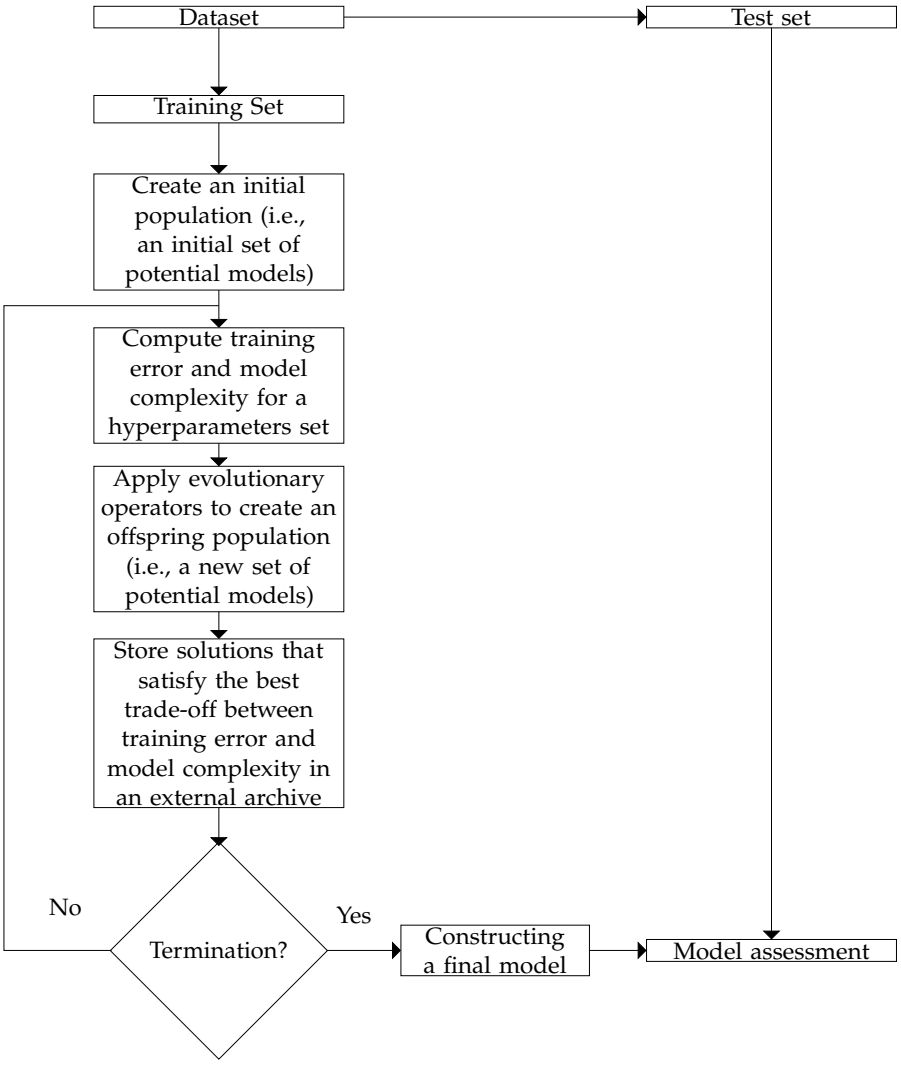


Figure 5.1: The general approach for the multi-objective model selection

Here, five different model types are considered: support vector machines (SVMs), neural networks (NNs), random forest (RF), j48 and random trees (RTs). All of these are available in the WEKA toolbox (Hall et al., 2009), and LibSVM (Chang and Lin, 2011) for the SVM. Table 5.1 shows the learning algorithms considered in our study. It

Table 5.1: Description of the learning methods considered in our study.

Learn. Alg.	Hyper-parameters	Description
J48	Confidence: A confidence threshold for pruning. K: A minimum number of instances per leaf.	It constructs a pruned or unpruned C4.5 decision tree.
NNs	Neurons: Number of neurons in the hidden layer lr: Learning Rate for the backpropagation algorithm. Momentum: Momentum Rate for the backpropagation algorithm. Epochs: Number of epochs to train through. Seed: The value used to seed the random number generator.	It constructs a multi-layer perceptron using the backpropagation algorithm.
RF	Trees: Number of trees to build. K: Number of features to consider. Depth: The maximum depth of the trees. Seed: The value used to seed the random number generator.	It constructs a forest of random trees.
RT	K: Number of features to randomly investigate. Depth: The maximum depth of the tree. Seed: Seed for random number generator.	It constructs a tree that considers K randomly chosen attributes at each node. It does not perform pruning.
SVMs	Kernel: The kernel type to be used. d: The degree of a polynomial kernel. γ: Gamma value of an RBF kernel. B: A bias value in polynomial kernel. C: The complexity constant. Seed: Seed for random number generator.	It constructs a support vector classifier.

also shows for each method the corresponding hyper-parameters.

In MOMS, the Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) (see subsection 3.2.2) is used as the search algorithm. In the rest of the section we explain our proposal in detail.

5.2.1 Representation and Initialization

Evolutionary Algorithms work with a population of solutions. In our proposed approach, each solution, also called individual, represents a potential model for the classification task. As previously stated, the task of the model selection proposal is to choose among a pool of learning algorithms and their corresponding hyper-parameters. To accomplish this, each model (the learning algorithm plus its hyper-parameters) should be encoded in a D-dimensional vector. In this study, each solution is encoded

in a 7-dimensional vector as follows:

$$\mathbf{x}^i = \left[x^i_m, x^i_{hp_1}, \ldots, x^i_{hp_{D-1}}\right] \tag{5.1}$$

where x^i_m controls the learning algorithm, and $\left[x^i_{hp_1}, \ldots, x^i_{hp_{D-1}}\right]$ represents the hyper-parameters for the learning algorithm.

Since the hyper-parameters are numerical values, and in order to have the hyper-parameters values as accurate as possible, a real encoding for the individuals is used. One should note that there are some discrete variables, such as x^i_m, which represent a learning algorithm. In the evolutionary operator, these types of variables are internally treated as real-value ones and they take the nearest allowable discrete value.

From Table 5.1, we can observe that different learning algorithms require different hyper-parameters. For example, in J48 two hyper-parameters are considered, whilst in SVMs there are six hyper-parameters. Thus, x^i_m and the six hyper-parameters are the seven variables in our representation. The configuration given by an individual and the training set are used to fit a model.

The seven variables constitute the search space for our problem. An initial population is created, but instead of creating the initial population randomly, we used the Latin Hypercube sampling technique (Mckay et al., 2000) for this purpose, aiming to have a representative population of the search space. A Latin hyper-cube is constructed such that each of the optimization variables is divided into N equal levels, where N is the population size and there is only one point (individual) at each level. To determine the location of each point, the Latin hyper-cube sampling technique maximizes the minimum distance between pairs of points, with the aim to spread them out as much as possible within the search space. In this manner, we ensure to have a representative initial population from the entire search space.

Once the initial population is created, it is used for producing an offspring population by applying the evolutionary operators until a stopping criterion is satisfied, and a set of non-dominated solutions is obtained.

5.2.2 Evolutionary Operators

As evolutionary operators, we used a differential evolution operator (Price et al., 2005), and polynomial mutation (Deb, 2001). In the differential evolution operator, each

element y_j in the new solution $\mathbf{y} = [y_1, \ldots, y_n]$ is generated as follows:

$$y_j = \begin{cases} x_j = x_j^i + F \times \left(x_j^k - x_j^l\right) & \text{with probability } CR, \\ x_j^i & \text{with probability } 1 - CR \end{cases} \tag{5.2}$$

where CR and F are two control parameters.

Polynomial-based mutation generates the new solution, $\mathbf{y} = [y_1, \ldots, y_n]$ as follows:

$$y_j = \begin{cases} \bar{y_j} + \Delta_j \times (U_b - L_b) & \text{with probability } pm \\ \bar{y_j} & \text{with probability } 1 - pm, \end{cases} \tag{5.3}$$

where pm is the probability of mutation, U_b and L_b are the upper and lower bounds of the hyper-parameters, respectively, Δ_j is a polynomial distribution for random numbers generation in the following way:

$$\Delta_j = \begin{cases} (2 \times \text{rand})^{\frac{1}{\eta+1}} - 1 & \text{if rand } < 0.5 \\ 1 - [2 \times (1 - \text{rand})]^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases} \tag{5.4}$$

where "rand" is a uniform random number in $[0, 1]$, and η is the distribution index for the mutation operator.

5.2.3 Fitness Functions

In the proposed approach, the model selection problem is tackled as a multi-objective optimization problem, and a MOEA is used for that task. Since the search is based on a population of solutions, it is required to have a way to measure how good a model performs in order to choose the best one. The fitness function is in charge of this, and its definition is a crucial issue in model selection. One could try to estimate the effectiveness of the model based on the error on the training samples, also known as empirical error, and the optimization problem would try to minimize that error. Nonetheless, this would be an optimistic estimation of the model performance, and could lead to models with a high complexity, causing the problem known as over-fitting; in other words, the model has a good performance on the training samples, but not on unseen samples (see Cherkassky and Mülier (2007); Duda et al. (2001); Hastie et al. (2009) for more information about this problem). To overcome this shortcoming, the model complexity should also be controlled. Taking this into account, here we propose not only to minimize the error on the training data, but also to minimize the model complexity.

The VC-dimension is a measure of the capacity of the model, which is related to its complexity, and it is used in the present study. These two goals, the error rate on the training set, f_1, and model complexity, f_2, integrate the fitness function and are defined as follows

$$\begin{aligned} f_1 &= \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}\left(c_i, \hat{c}_i\right) \\ f_2 &= \underset{h}{\operatorname{argmin}}\sum_{i=1}^{k}\left[\overline{\xi}\left(n_i\right) - \Phi\left(n_i/h\right)\right]^2 \end{aligned} \tag{5.5}$$

where N is the number of samples in the training set, c_i is the class label, $\hat{c}_i$ is the class predicted by the model, $\mathcal{L}\left(c_i, \hat{c}_i\right)$ is a loss function, $\xi\left(n_i\right)$ is the experimental maximum deviation error rate of two observed independent labeled datasets, and $\Phi\left(n_i/h\right)$ is the expectation of the largest deviation error between two sets (see subsection 4.3.3 for details about complexity estimation). We used 0/1 loss function because it is well suited for classification tasks. The 0/1 loss function is defined as:

$$\mathcal{L}\left(c_i, \hat{c}_i\right) = \begin{cases} 1 & \text{if } \hat{c}_i \neq c_i \\ 0 & \text{if } \hat{c}_i = c_i \end{cases} \tag{5.6}$$

In consequence, the goal of performing this optimization is to simultaneously minimize the training error and model complexity. The outcome of this optimization step is a set of potential models that satisfies the best trade-off between the objectives, from which a model should be chosen. The next section explains how we approach this issue.

5.2.4 Constructing a Final Model

Once the evolutionary search is completed, a set of non-dominated solutions is obtained. Mathematically, all of them are equally acceptable solutions of the multi-objective optimization problem and, in our case, each of them represents a potential model for a given classification task. Therefore, it is desirable to select one model to be used to predict new samples from such set. In model selection for classification tasks, one usually wants to choose the model with the highest possible generalization capability. Nevertheless, it could not be clear which classification model from the non-dominated set is the "best" one. We studied three strategies for constructing a final classification model, which are explained in the rest of this section.

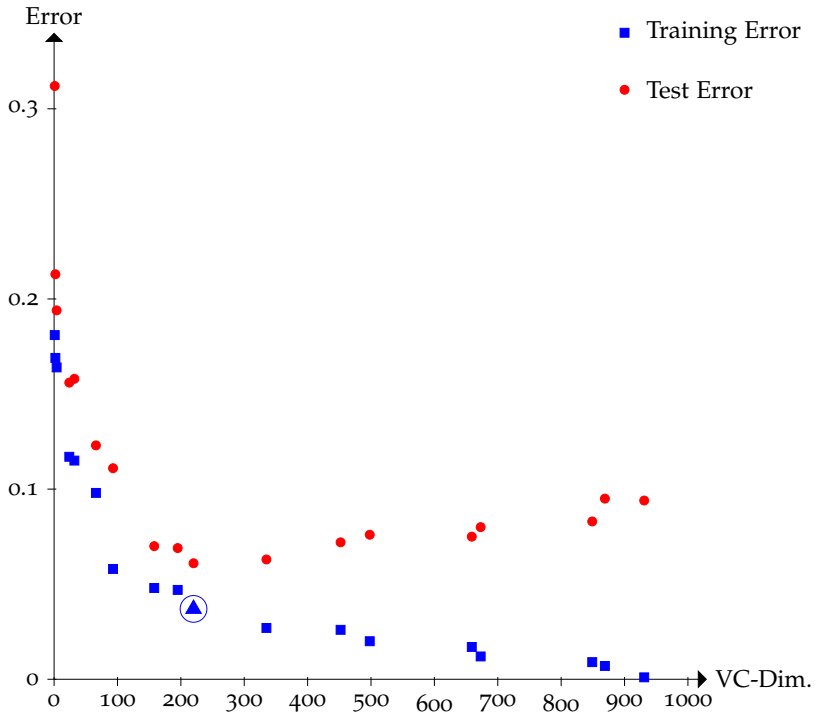


Figure 5.2: Behavior of non-dominated solutions on training samples and test samples

Choosing a Single Model

As previously stated, for the problem at hand, the solution with the best possible generalization ability is sought. In order to identify such solution, the performance of the non-dominated solutions on unseen samples is studied. We noticed that the best solutions are in the knee of the curve, while solutions with low complexity and high complexity lead to models with a poor generalization performance. Both problems are known in machine learning as under-fitting and over-fitting, respectively. Figure 5.2 depicts an example of this behavior for a particular case. It also shows the trade-off between the training error and the model complexity, such that by increasing the model complexity, the training error is reduced.

We empirically observed that in most cases, the solution with a good generalization performance is the one nearest to a reference point z^*, which is defined as:

$$z_j^* = \min_{1 \leqslant i \leqslant L} f_j\left(x^i\right) \text{ for } j = \{1, 2\} \tag{5.7}$$

where L is the cardinality of the non-dominated set.

As it is shown in Figure 5.2, the objectives are measured in different scales. For avoiding that one objective has a higher impact than the other one in the distance

computation, both objectives are firstly normalized in the range 0 to 1. Subsequently, the Euclidean distance is computed on the normalized objective vector. In the end, the closest solution is chosen,[1] which is used to predict future samples of the problem. One should note that since the objectives are normalized, the reference vector $\mathbf{z}^*$ would correspond to the $(0, 0)$ point. Figure 5.2 shows with a triangle the solution that would be chosen with this strategy.

Ensemble of the Whole Non-Dominated Front

Ensembles of classifiers are based on the idea of combining the predicted outputs from different individual classification models. They have been successfully used for improving the performance of individual models. One should have to remember that the output of the MOEA is a set of non-dominated solutions. Based on this, one might ask why not to construct an ensemble with the potential models (solutions) in the non-dominated front instead of choosing a single model. Now the problem is to determine which models should be used in the ensemble. In the absence of knowledge about the preferences, all non-dominated solutions are equally good. With these ideas in mind, all of them are used for constructing an ensemble.

In order to determine what models will be included within an ensemble, it is also required to define how individual predictions in each individual model should be combined. This is approached following a linear weighting combination. Therefore, the final prediction given by the classification model is given as follows:

$$\hat{c} = \sum_{i=1}^{L} \omega_i \hat{c}_i \tag{5.8}$$

where L is the number of single classification models, and is equal to the cardinality of the non-dominated set, $\hat{c}_i$ is the prediction given by the i^{th} single model, and ω_i is the weight associated to that model. The weight vector $\Omega = [\omega_1, \ldots, \omega_L]$ is a normalized vector, whose values depend on the distances between the reference point (defined by Equation (5.7)) and the potential solution. The normalized weight vector is computed as follows:

$$\omega_j = \frac{\hat{d}_j}{\sum_{i=1}^{L} \hat{d}_i} \tag{5.9}$$

where $\hat{d}_j$ is the inverse of the Euclidean distance between the j^{th} solution and the reference point $\mathbf{z}^*$.

[1] One should note that this is a suggestion. If the model, however, does not satisfy the performance requirements, any other can be chosen from the non-dominated set without performing a new search.

Ensemble of Some Solutions in the Non-Dominated Front

It is known from machine learning that for constructing an ensemble, two conditions have to be satisfied: the individual models should be accurate (i.e., the performance should be better than a random guessing), and they have to be diverse (i.e., single models should incur in different errors on new samples) among them (Dietterich, 2000). This issue is explored in the third strategy for the final model construction. Therefore, for constructing an ensemble in this third strategy, a subset of potential models in the non-dominated front needs to be chosen, such that they would be accurate and diverse among them. One should recall that the models were optimized, and the ones that satisfy the best trade-off are obtained as a result. Thus, it can be assumed that the models in the resulting non-dominated set are accurate. By making this, the problem is reduced to choosing a subset of models that are as diverse as possible among them, which are used in the ensemble. In order to determine such subset, a forward aggregation approach is used. In the forward aggregation approach, we start by adding the solution closest to the reference point, z^*. After that, a second model that maximize diversity is added, followed by a third model and so on. This process is repeated while diversity among models is not deteriorated.

Under the adopted approach, a diversity measure is required. There are a number of diversity measures reported in the literature, and a review of these can be found in (Kuncheva and Whitaker, 2003). In this study, one based on entropy is used, but any other can also be adopted. This measure is defined as follows:

$$E = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{L-\lceil\frac{L}{2}\rceil}\min\{l(\mathbf{s}_i), L-l(\mathbf{s}_i)\} \tag{5.10}$$

where N is the number of samples, L is the number of individual models, and $l(\mathbf{s}_i)$ is the number of models that correctly predict the sample $\mathbf{s}_i$. This measure ranges between 0 and 1, where 0 indicates no difference and 1 the highest possible diversity.

Finally, the prediction given by the ensemble is based on the weighted linear combination of the predictions of the individual models, as is shown in Equation (5.8).

5.2.5 Remarks

One should note that under the proposed approach the expert's knowledge is not exploited. This could be a key issue in order to improve the performance of the models. In the agnostic learning vs. prior knowledge challenge (Guyon et al., 2008) it was shown that, even when prior knowledge outperforms agnostic learning for most of the problems, the agnostic learning models are also very powerful. This is because

they are able to quickly achieve high performances, which were close to the best achievable ones. Furthermore, there could be some situations in which this knowledge is not available. For these reasons, we bet in favor of not using expert's knowledge. Notwithstanding, if such knowledge is available, this could be integrated in several manners to the proposed approach. For example, based on the characteristics of the problem at hand, an expert could suggest than a particular learning algorithm would be more suitable than the others. This suggestion could be used for fixing *a priori* the learning algorithm. Thus, the search would be performed under its hyper-parameters, reducing the search space. The expert's knowledge can also be used for choosing a single solution from the non-dominated front. Another way in which it could be used would be in the ensemble construction, through the assignment of weights of each classifier. For the experimental evaluation, it is assumed the expert's knowledge is not available. The next section describes the experiments and results obtained by our proposal.

5.3 Experiments and Results

This section describes the experiments performed as well as the results obtained by our proposal using a benchmark test suite. First, a comparative study between the three proposed strategies for constructing a final classification model from the resulting non-dominated front is presented. After that, the statistical tests to validate the obtained results when compared to other approaches reported in the specialized literature are presented.

5.3.1 Experimental Settings

In order to evaluate the feasibility of our proposal in the model selection problem, we used the IDA benchmark[2] datasets introduced by Rätsch et al. (2001). This benchmark is well-suited for this purpose and it has been widely used in several related studies (e.g. (Bao et al., 2013; Cawley and Talbot, 2010, 2007; Escalante et al., 2009; Rätsch et al., 2001)). Table 5.2 describes the suite of thirteen benchmark datasets, which are diverse in the number of samples and features. They correspond to binary classification problems[3], and have been previously pre-processed in (Rätsch et al., 2001), in which the samples with missing values have been removed and all features have been standardized, i.e., all features have mean zero and a standard deviation of one.

[2] Available at http://www.raetschlab.org/Members/raetsch/benchmark

[3] Without loss of generality, the experiments are performed on binary classification problems. Multi-class classification problems can be approached with multiple binary classifiers.

Table 5.2: Details of the datasets used in our experiments. The table shows the number of features for each dataset and the number of instances for training and testing for each replication of each dataset.

ID	Dataset	Feat.	Training Samples	Testing Samples	Replications
1	Banana	2	400	4900	100
2	Breast Cancer	9	200	77	100
3	Diabetes	8	468	300	100
4	Flare Solar	9	666	400	100
5	German	20	700	300	100
6	Heart	13	170	100	100
7	Image	20	1300	1010	20
8	Ringnorm	20	400	7000	100
9	Splice	60	1000	2175	20
10	Thyroid	5	140	75	100
11	Titanic	3	150	2051	100
12	Twonorm	20	400	7000	100
13	Waveform	21	400	4600	100

The typical experimental protocol used with this benchmark was introduced by (Rätsch et al., 2001), and is sometimes called the median protocol. The median protocol consists of performing the model selection on the first five partitions. After that, the median values of the hyper-parameters resulting from those partitions are taken, which are used to estimate the error rate for each partition. However, this protocol can introduce an optimistic bias into the performance estimation (Cawley and Talbot, 2010). In order to overcome this bias in the performance evaluation, the model selection process is performed independently for each partition of each dataset; this protocol is known as the internal protocol. The use of the internal protocol leads to a total of 1140 MOMTS executions.

The configuration parameters is the following. For the differential evolution crossover, the values of $F = 0.5$, $CR = 0.7$ are fixed. With respect to the mutation operator, the mutation probability pm was fixed to 0.1 and index distribution to 20. These parameters were experimentally tuned by evaluating the performance under each configuration of $pm = \{0.1, 0.2, 0.3\}$, $CR = \{0.5, 0.6, 0.7, 0.8, 0.9\}$, and $F = \{0.3, 0.4, 0.5, 0.6, 0.7\}$ on the first five partitions of the splice dataset, one of the largest both in number of training samples and features. The stopping criterion is defined as

performing 1,000 fitness functions evaluations. To achieve this, the population size is set to 20, and the number of generations to 50. Moreover, the VC-dimension for each model is estimated experimentally. Thus, it is required to train and to test a number of times each model; this number is fixed to 10. Next, we present the results reached by MOMTS, comparing the proposed strategies for a final model construction and with other evolutionary and non-evolutionary approaches for model selection.

5.3.2 Experimental Results and Discussion

This section presents the results obtained by MOMTS, **M**ulti-**O**bjective **M**odel **T**ype **S**election, so as to demonstrate its feasibility for the model selection problem. Table 5.3 shows the average error rates and standard deviations on the test sets attained for the three proposed strategies for constructing a final model, i.e., choosing a single model (MOMTS-S1), ensemble of the whole non-dominated front (MOMTS-S2), and the ensemble of some solutions in the non-dominated front (MOMTS-S3). As a baseline, we report the results obtained by using random forest (RF) with its default hyper-parameters, which is a standard learning algorithm based on an ensemble of decision trees.

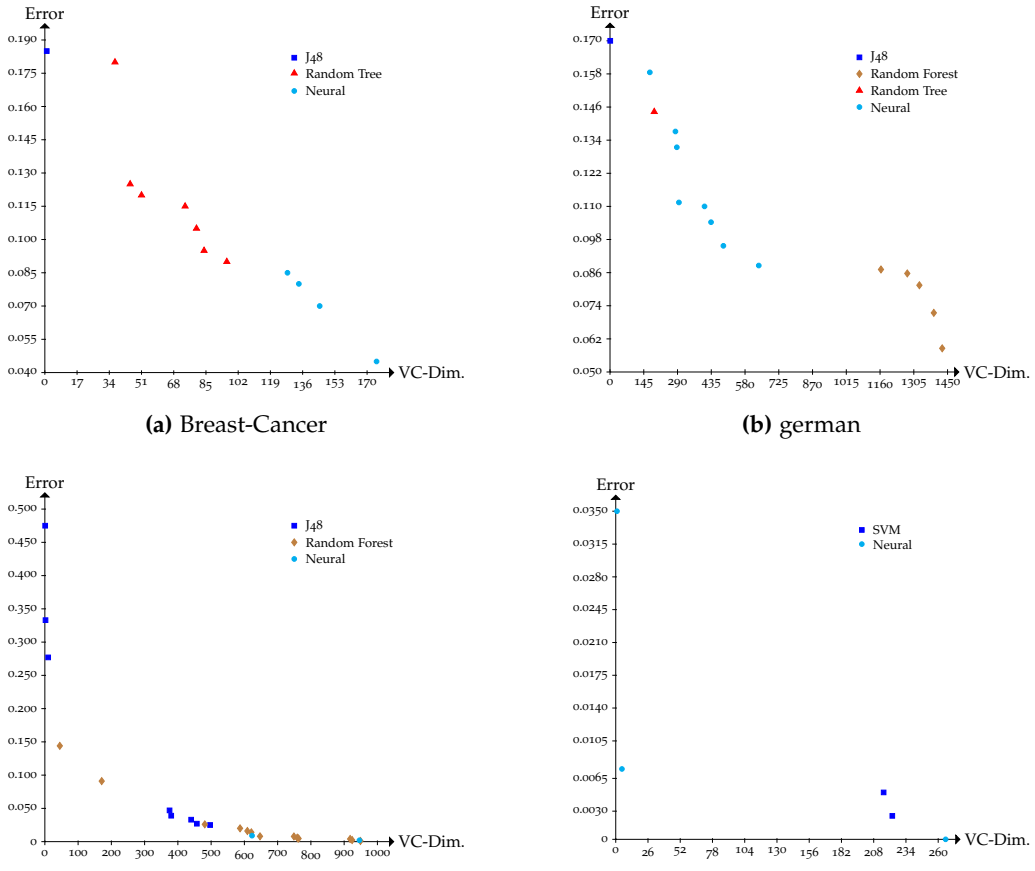
The results reported by MOMTS are compared with those reported by Cawley and Talbot (Cawley and Talbot, 2007), who used Bayesian regularization at the second level of inference, adding a regularization term in the model selection criterion. Furthermore, in order to make a fair comparison, experiments with approaches that consider different learning algorithms and their hyper-parameters during the model selection process are also performed. For that sake, PSMS (Escalante et al., 2009) and SUMO (Gorissen et al., 2009) are adopted, which are two evolutionary approaches that were proposed for model selection. PSMS is a single-objective approach based on a particle swarm optimizer that minimizes the balanced error rate estimated through $k-\texttt{fold}$ cross validation. SUMO adopts a genetic algorithm as a search algorithm and the fitness function can be defined as minimizing some measure obtained via some evaluation strategy; here, the measure was fixed to be the error rate and the evaluation strategy to be the $k-\texttt{fold}$ cross validation. In both cases, the number of particles/individuals was set to 20, and the number of iterations/generations to 50, resulting in 1,000 fitness function evaluations. This is the same number of fitness function evaluations set for MOMTS. The reference results used the same 100 partitions (20 in case of the image and splice datasets) for training and testing, and also used the same experimental protocol (i.e., the internal protocol), making the results directly comparable.

Figure 5.3 shows the non-dominated fronts generated by MOMTS for some datasets

Table 5.3: Results obtained by the proposed approach, and those obtained by random forest (RF), LS-SVM using Bayesian regularization, PSMS, and SUMO. The reported results are the average and standard error on test sets over 100 or 20 replications of each dataset. The best result for each dataset is shown in **boldface**.

Dataset	RF	LS-SVM-BR	PSMS	SUMO	MOMTS-S1	MOMTS-S2	MOMTS-S3
Banana	13.14 ± 0.069	10.59 ± 0.050	11.08 ± 0.083	10.88 ± 0.074	14.34 ± 0.105	$\mathbf{10.48 \pm 0.046}$	12.91 ± 0.160
Breast Cancer	27.94 ± 0.412	27.08 ± 0.494	33.01 ± 0.658	26.27 ± 0.448	29.89 ± 0.736	$\mathbf{25.61 \pm 0.593}$	27.82 ± 0.676
Diabetes	25.83 ± 0.212	23.14 ± 0.166	27.06 ± 0.259	23.49 ± 0.177	28.34 ± 0.318	$\mathbf{23.08 \pm 0.174}$	25.66 ± 0.214
Flare Solar	36.44 ± 0.173	$\mathbf{34.07 \pm 0.171}$	34.81 ± 0.173	38.47 ± 0.573	34.90 ± 0.224	34.59 ± 0.189	34.52 ± 0.214
German	25.16 ± 0.240	$\mathbf{23.59 \pm 0.216}$	30.10 ± 0.720	23.83 ± 0.213	28.30 ± 0.274	23.67 ± 0.224	25.89 ± 0.218
Heart	20.26 ± 0.382	$\mathbf{16.19 \pm 0.348}$	20.69 ± 0.634	17.67 ± 0.355	23.14 ± 0.542	16.48 ± 0.241	18.75 ± 0.351
Image	$\mathbf{2.09 \pm 0.095}$	2.90 ± 0.154	2.90 ± 0.112	2.45 ± 0.126	3.79 ± 0.226	2.24 ± 0.123	3.03 ± 0.246
Ringnorm	9.57 ± 0.095	$\mathbf{1.61 \pm 0.015}$	7.98 ± 0.660	1.72 ± 0.071	2.66 ± 0.079	2.49 ± 0.074	3.02 ± 0.164
Splice	8.50 ± 0.210	10.91 ± 0.154	14.63 ± 0.324	10.94 ± 0.146	7.43 ± 0.373	$\mathbf{4.84 \pm 0.156}$	6.71 ± 0.269
Thyroid	5.89 ± 0.273	4.63 ± 0.218	4.32 ± 0.235	4.85 ± 0.224	6.48 ± 0.350	$\mathbf{4.00 \pm 0.194}$	6.11 ± 0.347
Titanic	22.51 ± 0.124	22.59 ± 0.120	24.18 ± 0.193	34.99 ± 0.523	26.53 ± 0.127	$\mathbf{22.08 \pm 0.085}$	22.22 ± 0.100
Twonorm	9.02 ± 0.074	2.84 ± 0.021	3.09 ± 0.127	$\mathbf{2.55 \pm 0.022}$	5.21 ± 0.555	3.73 ± 0.179	5.70 ± 0.679
Waveform	13.75 ± 0.071	$\mathbf{9.78 \pm 0.044}$	12.80 ± 0.325	$\mathbf{9.78 \pm 0.060}$	11.34 ± 0.180	9.93 ± 0.043	10.95 ± 0.256

in a particular trial. It is expected that these non-dominated fronts are an approximation to the true Pareto front. We can observe that different solutions are distributed along the non-dominated front. We can also note that the non-dominated front is formed by solutions that represent different learning algorithms. Each corresponds to models with different levels of complexity. Although, in some cases, a learning algorithm is represented by more than one solution, these correspond to different configurations of its hyper-parameters, which could lead to diverse models. Thus, in the resulting non-dominated front there are models, which are learned by different learning algorithms with a different hyper-parameters configuration[4].



(a) Breast-Cancer **(b)** german

(c) splice **(d)** twonorm

Figure 5.3: Non-dominated fronts generated from a particular trial of MOMTS. The solutions in the non-dominated front represent different learning algorithms with different hyper-parameter configurations.

[4]The full list of the models generated by MOMTS for each partition of each dataset is available at http://ccc.inaoep.mx/~arosalesp/Resources/Models.zip

Comparison of Strategies for Constructing the Final Model

The results of the proposed strategies for constructing a final model are shown in the last three columns of Table 5.3. The results of the ensemble approaches outperformed those obtained when a single model is chosen in most cases. The single model was better than the ensemble of some solutions in the non-dominated front in 2 out of 13 datasets (ringnorm and twonorm datasets). This seems reasonable insomuch as it is well-known that using an ensemble of models helps to improve the predictions. Between the two ensembles approaches, the one based on the whole non-dominated front showed better results in 12 out of 13 datasets.

An ANOVA statistical test is applied so as to determine if the difference between the proposed strategies is significant at the considered level of $\alpha = 0.05$, and Tukey's test is used as a post-hoc test. The results obtained by this test are shown in Table 5.4. In this table, we can note that the analysis of variance showed a statistical significance difference for most datasets, except for the flare solar one, to which the post-hoc test is not applied. According to the pairwise comparisons, we can also note that MOMTS-S3 performs significantly better than MOMTS-S1 in 6 out of 13 datasets (banana, diabetes, german, heart, image, and titanic). On the other hand, MOMTS-S2 showed to be significantly better than MOMTS-S1 in 10 out of 13 benchmark datasets (banana, breast cancer, diabetes, german, heart, image, splice, thyroid, titanic, and waveform), and also significantly outperformed MOMTS-S3 in 10 out of 13 datasets (banana, diabetes, german, heart, image, ringnorm, splice, thyroid, twonorm, and waveform). It seems clear that MOMTS-S2 is the best of the three approaches. However, for assessing the statistical difference among them over the different datasets, Demšar (2006) recommends the Friedman's test for comparing multiple classifiers over multiple datasets. This test is performed with a level of $\alpha = 0.05$, and the Nemenyi test as the post hoc test. According to these tests, the ensemble of the whole front approach is statistically superior to the others.

Comparison with Other Model Selection Approaches

Table 5.3 also shows the performance of random forest (RF), LS-SVM with Bayesian Regularization (LS-SVM-BR), which uses a radial basis function kernel, as well as the results obtained with the application of PSMS and SUMO in the benchmark datasets. Due to the fact that the best results of our proposal were reached with MOMTS-S2, this approach is used for comparison.

First, we compare with RF, which is used as a baseline to evaluate the benefits of performing model selection. From the reported results in Table 5.3, we can note that

Table 5.4: Reported results by ANOVA and Tukey HSD tests, for performing all possible pairwise comparisons among the 3 variants of MOMTS. The reported results are the p-value for ANOVA and the adjusted p-value (APV) for Tukey HSD test. Cases whose p-value is below $\alpha = 0.05$ are marked with an asterisk (*).

		APV Tukey HSD		
Dataset	p ANOVA	MOMTS-S1 vs. MOMTS-S2	MOMTS-S1 vs. MOMTS-S3	MOMTS-S2 vs. MOMTS-S3
Banana	< 0.0001*	< 0.0001*	< 0.0001*	< 0.0001*
Breast Cancer	< 0.0001*	< 0.0001*	0.076	0.0534
Diabetes	< 0.0001*	< 0.0001*	< 0.0001*	< 0.0001*
Flare Solar	0.4252	—	—	—
German	< 0.0001*	< 0.0001*	< 0.0001*	< 0.0001*
Heart	< 0.0001*	< 0.0001*	< 0.0001*	0.0002*
Image	< 0.0001*	< 0.0001*	0.0302*	0.0233*
Ringnorm	0.0008*	0.5397	0.0657	0.0031*
Splice	< 0.0001*	< 0.0001*	0.1736	< 0.0001*
Thyroid	< 0.0001*	< 0.0001*	0.6687	< 0.0001*
Titanic	< 0.0001*	< 0.0001*	< 0.0001*	0.6160
Twonorm	0.0086*	0.1080	0.7809	0.0203*
Waveform	< 0.0001*	< 0.0001*	0.2868	0.0003*

our proposal outperformed RF in 12 out of 13 datasets, being the image dataset the only one in which RF performed better than our proposal.

Comparing MOMTS-S2 with LS-SVM-BR, we can note that the proposal obtained better results in 7 out of 13 datasets (banana, breast cancer, diabetes, image, splice, thyroid, and titanic), but it was outperformed in the rest of the datasets. In addition, it is worth noting that an improvement of more than 6% was reached in the splice dataset.

With respect to PSMS, a single-objective approach that considers different learning algorithms and hyper-parameters selection, we note that MOMTS-S2 performed better than PSMS in 12 out of 13 benchmark datasets. When comparing our proposal with SUMO, another evolutionary approach for model selection, we note that MOMTS-S2 got better generalization performance in 10 out of 13 datasets.

Regarding statistical assessment, the ANOVA test is used to compare the performance of the model selection approaches: LS-SVM-BR, PSMS, SUMO, and MOMTS-S2. Inasmuch as the goal is to compare the performance of MOMTS-S2 with the reference results, the Dunnett's test is used as the post-hoc test. These statistical tests were conducted independently for each dataset. The results of these are shown in Table 5.5,

Table 5.5: Reported results by ANOVA and Dunnett's tests, for comparing MOMTS-S2 against LS-SVM-BR, PSMS, and SUMO. The reported results are the p-value for ANOVA and the adjusted p-value (APV) for Dunnett's test. Cases whose p-value is below $\alpha = 0.05$ are marked with an asterisk (*).

	ANOVA	(APV) - MOMTS-S2 versus		
Dataset	(p)	LS-SVM-BR	PSMS	SUMO
Banana	$< 0.0001^*$	0.495	$< 0.0001^*$	$< 0.0001^*$
Breast Cancer	$< 0.0001^*$	0.097	$< 0.0001^*$	0.6710
Diabetes	$< 0.0001^*$	0.9930	$< 0.0001^*$	0.329
Flare Solar	$< 0.0001^*$	0.5380	0.9340	$< 0.0001^*$
German	$< 0.0001^*$	0.998	$< 0.0001^*$	0.986
Heart	$< 0.0001^*$	0.931	$< 0.0001^*$	0.1190
Image	0.0005^*	0.0021^*	0.0022^*	0.5463
Ringnorm	$< 0.0001^*$	0.1590	$< 0.0001^*$	0.2480
Splice	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$	$< 0.0001^*$
Thyroid	0.0357^*	0.1090	0.6026	0.0176^*
Titanic	$< 0.0001^*$	0.4570	$< 0.0001^*$	$< 0.0001^*$
Twonorm	$< 0.0001^*$	$< 0.0001^*$	0.0002^*	$< 0.0001^*$
Waveform	$< 0.0001^*$	0.8650	$< 0.0001^*$	0.8650

from which we can note that, for all cases, the analysis of variance revealed that there is a statistically significance difference at the level of $\alpha = 0.05$. Thus, the post-hoc test is applied.

According to the results shown in Table 5.5, statistical tests indicated that MOMTS-S2 significantly outperformed LS-SVM-BR in 2 datasets (image and splice), and it was significantly outperformed in one dataset (twonorm). Regarding SUMO, MOMTS-S2 performed significantly better in 5 out of 13 datasets (banana, flare solar, splice, thyroid, and titanic), and it was significantly outperformed in the twonorm dataset. On the other hand, MOMTS-S2 significantly outperformed PSMS in 10 of the benchmarks datasets (banana, breast cancer, diabetes, german, heart, image, ringnorm, splice, titanic, and waveform datasets), but it was significantly worse than PSMS in the twonorm dataset.

Overall, MOMTS-S2 is able to get lower error rates than the other model selection methods in 7 out of 13 datasets, while the Bayesian regularization approach does the same in 5 out of 13 datasets, and SUMO in 2 out of 13 datasets. There is not a clear advantage of LS-SVM-BR and MOMTS-S2 when multiple datasets are considered. In order to statistically assess the four model selection approaches over the suite of 13 benchmark datasets, the Friedman test is applied. As a post-hoc test, the Bonferroni-

Dunn test is used, to compare the performance of MOMTS-S2 with the references. According to these tests, MOMTS-S2 is statistically better than PSMS, but there is not a statistical significant difference between MOMTS-S2 and LS-SVM-BR and MOMTS-S2 and SUMO.

Another aspect to take into consideration is the computational cost of the methods. In this regard, we compare the execution time required by MOMTS against PSMS and SUMO. The average execution time of MOTMS was 54.29 minutes, whilst PSMS and SUMO required, respectively, 30.36 and 31.90 minutes on average.

Discussion

From the experimental results shown in Table 5.3, we can note how over-fitting can be present in model selection. Among the three strategies for constructing a final model, those based on ensembles proved being beneficial, reducing the over-fitting effect. Notwithstanding, we cannot say that ensemble approaches completely solve the problem. We can also note that in most cases, the use of the solutions in the whole non-dominated front in an ensemble achieved a better generalization performance than when a subset of these are considered for the ensemble. This is a surprising result, since it was expected that by taking into account diversity as a criterion for the ensemble construction, a better performance would be attained than when not doing so.

A comparison with RF showed the benefits of performing model selection against not doing so. This is specially remarkable in the ringnorm, splice, and twonorm datasets, in which an improvement above 4% is reached. Even though a simple RF outperformed our proposal in the image dataset, a pairwise comparison did not show a statistical significant difference between both. Therefore, it is worth performing the computational effort in order to construct a reliable classification model.

MOMTS-S2 significantly outperformed LS-SVM-BR on three benchmark datasets, but it was significantly worse in one dataset. The greatest improvement was obtained in the splice dataset, reducing the error rate in 6.07%. The greatest degradation was on the twonorm dataset, with a difference of 0.89%. In spite of this, the overall performance of both approaches was similar. Neither the reference nor the proposed approach were significantly better than each other. It is interesting to note that, although MOMTS-S2 deals with different model types and their corresponding hyper-parameters, it does not outperform LS-SVM-BR. Nevertheless, we can argue that in LS-SVM-BR there are only two parameters to be optimized, while in MOMTS, seven parameters are taken into consideration, which considerably increases the search space and makes it harder

to reach the "optimal" solutions with a lower number of iterations. Moreover, we gain generality without significantly over-fitting the models.

The experimental evaluations showed that MOMTS-S2 significantly outperformed PSMS. Although there was not a statistical significant difference between MOMTS-S2 and SUMO, when different datasets were considered, MOMTS-S2 significantly outperformed SUMO on several individual datasets. This gives evidence about the suitability of using a multi-objective approach in contrast to a single-objective approach, in spite of the computational cost of doing so.

As one could note, MOMTS is more time-consuming than the others. This is due to the fact that under the proposed approach two objectives have to be evaluated, while in PSMS and SUMO only a single objective is evaluated. In our case, estimating the model complexity through the VC-dimension implies to train and to test a model a number of times (10 times, according to the parameter that we used). Measuring the training error also implies to train and test such model. Hence, evaluating both objectives involves training and testing the model. This could represent a disadvantage with respect to the others, in terms of computational cost. Notwithstanding, we can argue that due to the nature of multi-objective optimization, several models are generated, which enables to explore a number of strategies for a final model construction without significantly increasing the computational cost. In addition to this, MOTMS has the advantage of getting highly competitive models, outperforming SUMO and PSMS in most of the datasets.

The experimental results showed that only minimizing the error rate estimated through k-fold cross validation could lead to choose a model with a small degree of over-fitting. However, the k-fold cross validation approach has the advantage of being free from the model assumptions, which makes it applicable to any learning algorithm and feasible to the full model selection formulation[5]. On the other hand, the use of the VC-dimension for controlling the model complexity, and avoiding over-fitting, as much as possible, also shows its potential for being applicable to different model types. Therefore, we believe that this approach can also be applicable to the full model selection formulation.

[5]The full model selection formulation consists of the task of finding the best combination of pre-processing, feature selection, and learning algorithms together with their parameters (Escalante et al., 2009; Sun et al., 2012).

5.4 Final Remarks

This chapter introduced MOMTS, a multi-objective approach for dealing with the problem of model type selection. MOMTS takes into account both the learning algorithm and the hyper-parameters during the search process. It uses the training error, or empirical error, and the model complexity, which is estimated through the VC dimension, as the objectives to be optimized. The adopted formulation showed the following advantages:

- the experimental way for measuring the VC dimension allows us to consider different learning algorithms in a general framework, and makes the method applicable to the full model selection problem;
- it had a competitive performance over different benchmark datasets, making it applicable to problems from diverse domains; and
- the multiple non-dominated solutions obtained through the multi-objective formulation makes it easy to extend it to ensembles of models.

The experimental results showed that constructing an ensemble of models performs better than choosing a single model. Furthermore, the ensemble approach showed to be effective for reducing the effect of over-fitting. The advantages of the multi-objective approach over a single-objective formulation such as PSMS were also supported by the experiments. The experimental results also show that highly competitive classification models were generated by our proposal, without significantly degrading the performance in most cases. Hence, we can conclude that our proposed approach can be an useful framework for model selection in real world problems.

In next chapters, we present the extension to the full model selection problem, i.e., considering feature selection and data pre-processing into the model selection process. We also study more effective ways for constructing an ensemble as well as study strategies based on surrogate-assisted optimization with the aim at reducing the computational cost due to the experimental estimation of the VC dimension.

Chapter 6

Evolutionary Nested Multi-Objective Full Model Selection with Pareto-based Ensemble

Would you tell me, please, which way I ought to walk from here? – Alice asked the Cheshire Cat
That depends a good deal on where you want to get to,– said the Cat.
I don't much care where– said Alice.
Then it doesn't matter which way you walk,– said the Cat.
. . . so long as I get somewhere,– Alice added as an explanation.
Oh, you're sure to do that,– said the Cat, –if you only walk long enough.

Lewis Carroll

Besides the large amount of learning algorithms available in the literature, a user can also account with a number of techniques for data pre-processing, such as data sampling, data transformation, feature selection, among others. Combining them into a single classification model is called a full model for the classification task. Hence, in the design of a full classification model, the user has to face mainly the issues of selecting the combination of pre-processing and learning algorithms and their customization by the definition of the hyper-parameters. This results in a relatively high degree of freedom to make these choices, which could suggest a challenging task into the community.

In this chapter, the design of a full classification model is seen as a hierarchical optimization problem. This means that there exists a dependency of the hyper-

parameters with the algorithm in which they are used. For instance, in an SVM with an RBF kernel, a γ value equal to 0.10 is meaningful for it, but not for another learning algorithm, such as a neural network. Inspired in this idea, the full model selection problem is approached as a nested optimization problem. Moreover, this chapter also describes several approaches for combining the non-dominated solutions into a single ensemble model. To this end, seven fusion schema of the Pareto optimal solutions are introduced. These strategies are the global Pareto ensemble, Pareto error reduce ensemble, Pareto complementary ensemble, margin distance, boosting, Pareto knee-based, and Pareto evolutionary selection.

This chapter first describes some related works in the full model selection problem. Second, the proposed method, called **EN-MOMS-PbE**: **E**volutionary **N**ested **M**ulti-**O**bjective Full **M**odel **S**election with **P**areto-**b**ased **E**nsemble, is introduced. Next, the experimental study and reported results are presented followed by some concluding remarks.

6.1 Related Works

The full model selection is defined as the problem of finding a combination of pre-processing methods and the learning algorithm together with the hyper-parameters for a given dataset. To the best of the author's knowledge, the works that address the full model selection problem are scarce.

Perhaps the first works on the full model selection problem were developed, in a parallel fashion, by Escalante et al. (2009) and Gorissen et al. (2009). Escalante et al. (2009) proposed an algorithm called Particle Swarm Model Selection (PSMS), in which the full model selection problem is approached as a single-objective optimization one. A simple particle swarm optimizer (PSO) is used as the search algorithm. In PSMS, both methods and hyper-parameters are encoded in a single real-valued representation. The balanced error rate is defined as the optimization criterion, which is estimated by means of the k-fold cross validation strategy. In a further study, Escalante et al. (2010) extended their former work by including three fusion schemes of the models in the PSO's population and the area under ROC curve is used as the optimization criterion. This approach is called Ensemble PSMS (EPSMS). Both approaches were evaluated on a suite of benchmark datasets and obtained promising results.

On the other hand, Gorissen et al. (2009) proposed an evolutionary approach for performing model selection. Their approach is based on an island model. In this proposal, the population is divided into sub-populations, where each sub-population

is composed by a model type, i.e., neural network, SVM, etc. Migrations among the sub-populations are allowed. One should note that different model types can be chosen for the crossover operation. Hence, authors defined how heterogeneous evolution is performed, which is approached by means of ensembles. The objective function here is more versatile, because it is defined as optimizing a criterion performance estimated through an evaluation strategy. For instance, one could try to minimize the classification error estimated via k-fold cross validation, or to minimize the mean square error estimated via hold-out.

Later, Sun et al. (2012) proposed the genetic algorithm and particle swarm optimization for full model selection (GPS, which stands for **G**A-**P**SO-FM**S**). In GPS, as the name suggests, authors combine a genetic algorithm with a particle swarm optimizer, two bio-inspired optimization techniques, for addressing the full model selection problem. Given a classification/regression problem, GPS finds a full model solution, which is represented as a directed acyclic graph. In their experimental study, they showed that GPS is able to outperform PSMS.

Thornton et al. (2013) proposed the Auto-Weka, in which the authors face the problem of a fully automated model selection problem. The authors combined three search and eight evaluation methods. The model can be composed by a feature selection and a learning algorithm. An special feature of this work is that it is able to work with a budget. Their experimental study is performed over small and large datasets, as well as in problems with a low or a high dimensionality. Their results were compared with a traditional grid search and showed to be quite efficient and effective.

Previous studies on the full model selection problem are, generally, formulated as a single-objective optimization ones. However, the advantages of using a multi-objective approach for hyper-parameters optimization has been pointed out by several authors (Chatelain et al., 2010; Gorissen et al., 2010; Jin and Sendhoff, 2008). To the best of the authors' knowledge, the full model selection problem has not been previously approached as a multi-objective one. We are aware that there exist proposals in which a multi-objective optimization is used to approach the model selection problem (e.g. Aydin et al. (2011); Chatelain et al. (2010); Suttorp and Igel (2006); Li et al. (2011); Pilat and Neruda (2013); Pilát and Neruda (2013)). They usually focused on the learning algorithm, but a combination with a pre-processing stage is not explored.

Inspired by the aforementioned, in this chapter we introduce **EN-MOMS-PbE**: **E**volutionary **N**ested **M**ulti-**O**bjective Full **M**odel **S**election with **P**areto-**b**ased **E**nsemble. EN-MOMS-PbE formulates the full model selection problem as a nested optimization

problem, where in an upper level the model structure[1] is optimized while the lower level deals with the hyper-parameters for a given model structure. Moreover, unlike previous studies that consider data transformation and/or feature selection as the pre-processing, EN-MOMS-PbE interprets in a broader sense the concept of full model by considering not only data transformation and feature selection, but also noise filtering and data sampling.

EN-MOMS-PbE further exploits the information in the Pareto front by means of ensembles. In this sense, it explores seven different strategies to fusion information, which are: Pareto global ensemble, Pareto error reduce, complementary incremental ensemble, margin distance ensemble, boosting, Pareto knee ensemble, evolutionary ensemble.

The main contributions of this chapter are the following:

- the formulation of the full model selection problem as a nested multi-objective optimization problem;
- the strategies that allow the information fusion from the Pareto front; and
- an experimental evaluation of the advantages of the proposed approach over a single-objective and other full model selection approaches.

We assess the performance of EN-MOMS-PbE over a set of 25 benchmark datasets. The following studies have been developed. First, a comparative study among different strategies for ensemble construction is performed. Second, the advantage of using an ensemble of the Pareto solutions against choosing a single solution is evaluated. Finally, results of EN-MOMS-PbE are compared with those reported by other state of the art model selection methods.

6.2 EN-MOMS-PbE: Evolutionary Nested-Multi-Objective Full Model Selection with Pareto-based Ensemble

EN-MOMS-PbE is described in a general fashion in Figure 6.1. As usually happens in evolutionary algorithms, EN-MOMS-PbE starts creating an initial population, in which each individual represents a full model structure, that is the algorithms and the way on how to combine them. The fitness value for each individual is computed. As we can see in Figure 6.1, in this stage is where the nested task is performed. For each full

[1] A model structure defines the pre-processing methods and learning algorithm and how they are combined into a full model.

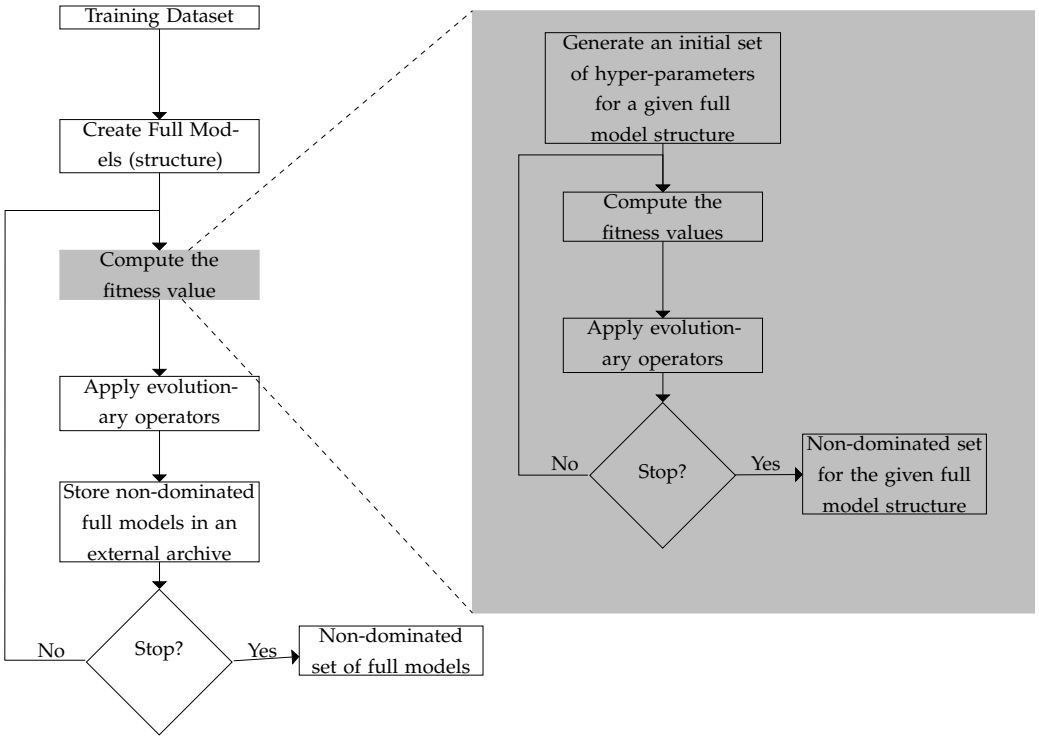


Figure 6.1: The proposed **EN-MOMS-PbE**: **E**volutionary **N**ested **M**ulti-**O**bjective Full **M**odel **S**election with **P**areto-**b**ased **E**nsemble

model structure, we optimize the set of hyper-parameters. Therefore, we have a sub-population of solutions for each full model structure. This sub-population is evolved to find the set of trade-off hyper-parameters for the structure at hand. By proceeding in this manner, EN-MOMS-PbE ensures that the lower level optimization only deals with the corresponding hyper-parameters to the given model structure. The result of the lower internal optimization is a set of non-dominated solutions, which are returned to the upper external optimizer for choosing the best non-dominated full models taking into consideration the results of other full model structures. Evolutionary operators are applied to create new full model structures and the process is repeated until a stopping criterion is met. The rest of this section explains in detail the proposal.

6.2.1 Nested-Multi-Objective Evolutionary Algorithm

The full model selection problem is approached as a nested-multi-objective optimization problem. In the upper level, the optimization problem is related to the full model selection structure; while in the lower level, with the hyper-parameters for a given structure. Therefore, the optimization algorithms should be able to deal with that.

Algorithm 8 Nested-MOEA

Require: N_u the size of the upper level population
N_l the size of the lower level population
A stopping criterion
Ensure: A set of non-dominated solutions
1: Initialize EP $\rightarrow \emptyset$
2: Generate a well-spread weight vector, $\omega = [\omega_1, \ldots, \omega_N]$
3: Generate an initial population $\mathbf{P}_u \left\{\mathbf{p}_u^1, \ldots, \mathbf{p}_u^{N_u}\right\}$ for the upper level
4: **for** each $\mathbf{p}_u^i \in \mathbf{P}_u$ **do**
5: Generate an initial lower level population $\mathbf{P}_l = \left\{\mathbf{p}_l^1, \ldots, \mathbf{p}_l^{Nl}\right\}$
6: Generate an approximation to the Pareto set, $\mathbf{PS}_l^{\mathbf{p}_u^i}$, using MOEA/D
7: **end for**
8: Store in EP the non-dominated solutions obtained from $\mathbf{PS}_l^{\mathbf{p}_u^i} : \forall i \in \{1, \ldots, Nu\}$
9: **while** stopping criterion is not satisfied **do**
10: **for** $i = 1$ **to** N **do**
11: Select parents either $\mathbf{P}_u$ or EP and then generate a new solution $\mathbf{y}$ by applying evolutionary operators
12: Initialize the lower level population from the nearest neighbors of $\mathbf{y}$ in EP
13: Generate an approximation to the Pareto set, $\mathbf{PS}_l^{\mathbf{y}}$, using MOEA/D
14: Update $\mathbf{p}_u^i$ if $\mathbf{y} : \mathbf{p}_l^{\mathbf{y}} \in \mathbf{PS}_l^{\mathbf{y}}$ is better in the aggregated function
15: Update of EP.
16: Refining the non-dominated solutions through a local search
17: **end for**
18: **end while**

With this aim, we present the Nested-Multi-Objective Evolutionary Algorithm, which is described in Algorithm 8.

Nested-MOEA initializes by creating a well spread vector of weights (step 2). After that, the population for the upper level is created (step 3) and for each individual in the upper level, a sub-population for the lower level is created. This sub-population is evolved using MOEA/D for finding an approximation to the Pareto optimal set in the lower level. During this stage, the variables that represents the upper level remain unchanged (steps 5 and 6). The non-dominated solutions by considering the two levels are stored in an external archive (step 8).

The next step is the evolutionary task, which implies the selection and creating of new individuals through evolutionary operators (step 11). A new individual in the

upper level is created taking into account both the current upper level population and the external archive. The parents to be used in the crossover are chosen as follows: first a random number with a uniform probability is generated and if it is lower than 0.5, a parent from the current upper level population is selected; otherwise, a subset of solutions are chosen from the external archive EP and the one with the highest distance is selected. The motivation to considering the external archive is to increase the probability of non-dominated solutions take part in the crossover. Since the solutions in the external archive are non-dominated, the criterion is based on distances to give more emphasis to those solutions in less crowded regions. After that, corresponding evolutionary operators are applied to create child $\mathbf{y}$. In the problem at hand, an upper level individual represents a full model structure.

For each new individual in the upper level, a sub-population for the lower level is created. Nevertheless, here this is slightly different, because the information from the previous explored solutions is taking into consideration (step 12). The creation of the sub-population of the new individual child considers previous explored solutions that are similar to $\mathbf{y}$, i.e., it is computed the similarity between $\mathbf{y}$ and each solution in EP and those that are the most similar, half of their sub-population is copied to the new sub-population, while the other half is randomly created. In this manner, some individuals are expected to be good solutions and they can help to improve convergence, while others can help to introduce diversity in the new sub-population. This sub-population is evolved using MOEA/D to find the approximation to the Pareto optimal set (step 13) and they are used for updating the upper level variable, which is done if the new solution is better than the previous one (step 14).

The final steps involve updating the external archive EP and the local search, this are the steps 15 and 16. As usually happens, a new individual is included into the external archive if the pair $\mathbf{y} : \mathbf{p}_l^{\mathbf{y}} \in \mathbf{PS}_l^{\mathbf{y}}$ is non-dominated with respect to those that are in the external archive. Moreover, if the new individual dominates some existing in the archive, those are removed. Regarding the local search, this could be performed for each new individual, this, however, would increase the computational cost. For this reason, the local search is performed on the solutions that are added to the external archive. The lower level variables are optimized and quadratic programming is used to this end. At this stage, the function to optimize is the following:

$$\begin{aligned} &\text{minimize: } f_k\left(\mathbf{p}_l^{\mathbf{y}}\right) \\ &\text{subject to: } f_i \leqslant o_i, \forall i \in \{1, \ldots, l\} \wedge i \neq k \end{aligned} \tag{6.1}$$

where o_i is the obtained fitness value in the i^{th} fitness function by MOEA/D. Thus,

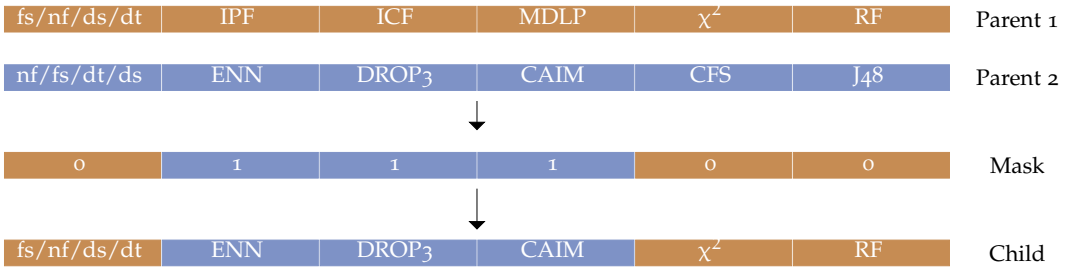


Figure 6.2: Croosover operation with the upper level model structure.

the local search is approached by an ϵ-constraint approach.

This process is repeated until a stopping criterion is met. Different stopping criteria can be adopted, such as reaching a fixed number of generations, the improvement in the non-dominated front be less than a threshold, or performing a maximum number of fitness evaluations, etc. The former two are here used.

6.2.2 Initialization

The first step to approach the full model selection problem as one of optimization is to define how to encode the solutions into a form of individuals. In the upper level, we want to optimize the full model structure, thus the representation is as follows:

$$\begin{aligned}\mathbf{p}_u^i = &[\mathtt{order}, \mathtt{noise-filtering}, \mathtt{data-sampling}, \mathtt{data-transformation}, \dots \\ &\mathtt{feature\ selection}, \mathtt{learning\ algorithm}]\end{aligned} \tag{6.2}$$

where $\mathtt{order}$ is a flag that indicates the order in which the pre-processing methods are applied; the variables $\mathtt{noise-filtering}$, $\mathtt{data-sampling}$, $\mathtt{data-transformation}$, $\mathtt{feature\ selection}$, and $\mathtt{learning\ algorithm}$ indicate the corresponding method used in each case. The list of considered methods in EN-MOMS-PbE is shown in Table 6.1.

The evolutionary operators for the upper level are shown in Figure 6.2 and Figure 6.3. Figure 6.2 graphically depicts the crossover operation, for the two chosen parents, a uniform crossover operator is applied and according to a mask vector the methods for the new individual are either chosen from the first parent or from the second parent. On the other hand, the mutation operation changes randomly one method for other one, as it is shown in Figure 6.3.

The lower level optimization deals with the hyper-parameters of the full model structure. Thus, in this nested optimization, individuals are real-valued and they strongly depend on the given structure.

Table 6.1: Considered Methods in EN-MOMS-PbE

Noise Filtering	Data-sampling	Data-transformation	Feature Selection	Learning Algorithm
ENN	Random	Unsupervised Dis.	Information gain	CART
IPF	SMOTE	CAIM	χ^2	J48
EF	ENN	Chi-Merge	CFS	JRip
None	DROP3	MDLP	LVF	K-NN
	FCNN	LLE	MrMr	K*
	ICF	PCA	Random	Logistic
	None	Nominal to Bin.	ReliefF	LVQ
		Normalize	None	Naive Bayes
		Standardize		Neural Net
		None		Random Forest
				SMO
				FURIA
				FK-NN
				RBF

Figure 6.3: Mutation operation with the upper level model structure.

For the lower level, we used differential evolution operator (Price et al., 2005), and polynomial mutation (Deb, 2001) as evolutionary operators. The differential evolution operator generates a new solution $\mathbf{y}^* = [y_1^*, \ldots, y_n^*]$ as follows:

$$y_j^* = \begin{cases} x_j = x_j^i + F \times \left(x_j^k - x_j^l\right) & \text{with probability } CR, \\ x_j^i & \text{with probability } 1 - CR \end{cases} \tag{6.3}$$

where CR and F are two control parameters and i, j, and k are the indexes of the parents.

Polynomial mutation generates the new solution, $\mathbf{y} = [y_1, \ldots, y_n]$ as follows:

$$y_j = \begin{cases} \bar{y_j} + \Delta_j \times (U_b - L_b) & \text{with probability } pm \\ \bar{y_j} & \text{with probability } 1 - pm, \end{cases} \tag{6.4}$$

where pm is the probability of mutation, U_b and L_b are the upper and lower bounds, respectively, and Δ_j is a polynomial distribution for random numbers generation in the following way:

$$\Delta_j = \begin{cases} (2 \times \text{rand})^{\frac{1}{\eta+1}} - 1 & \text{if rand } < 0.5 \\ 1 - [2 \times (1 - \text{rand})]^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases} \tag{6.5}$$

where rand is a uniform random number in $[0, 1]$, and η is the distribution index for the mutation operator.

6.2.3 Pareto-Based Ensemble for Multi-Objective Full Model Selection

The outcome of a MOEA is a set of non-dominated solutions that approximates the Pareto optimal set. All these solutions are equally acceptable for the problem at hand when there is no preference information available. However, in the problem that we face, the goal is to construct a reliable full model, which is used in the classification of unknown patterns. Thus, it is desirable to perform a post-processing over the trade-off solutions in order to get a final classification model. Notwithstanding, sometimes choosing a single solution could require some domain knowledge to set the appropriated preferences of each objective.

Due to the nature of multi-objective optimization, the solutions in the resulting non-dominated set would share some information about the training data. Thus, it could be a good idea to combine this information into a single one. This can be attained by means of an ensemble of these solutions. Moreover, since each solution in the non-dominated set corresponds to a full model trained with different hyper-parameters set and, possibly, different subsets of features or data transformations, this could lead to diverse models and an ensemble of classifiers can provide more information about the class label than a single classifier. This section extends the study performed in subsection 5.2.4 by including six new different strategies for combining solutions in the resulting non-dominated front, which are described in the following.

- ***Global Pareto ensemble*** (GPE): The basic idea here is to build an ensemble using all solutions in the resulting non-dominated set.
- ***Pareto error reduce*** (PER): The idea of this approach is not to use all solutions in the non-dominated set, but a subset of these. The ensemble is constructed in an incremental fashion. First, the solution with the lowest error on the dataset is included into the ensemble. The second solution included is the one that minimizes the error rate of the partial ensemble, and so on. Let $\mathcal{NC}$ be the set of classifiers in the resulting non-dominated set, the ensemble $\mathcal{E}$ is formed as follows:

$$\mathcal{E}_u = \underset{k}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(\mathcal{E}_{u-1} \cup \mathcal{NC}_k\left(\mathbf{x}_i\right), c_i\right) \tag{6.6}$$

  where the index k runs over all classifiers that are not already included into the ensemble and $\mathcal{E}_{u-1} \cup \mathcal{NC}_k\left(\mathbf{x}_i\right)$ is the predicted class when $\mathcal{NC}_k$ is included into the ensemble and c_i is the actual class label.

- ***Pareto Complementary Ensemble*** (PCE): This is also an incremental approach for the ensemble construction. The idea is to include, at each iteration, the classifier whose performance is the most complementary to that of the partial ensemble. As in error reduce, the first classifier included is the one that has the lowest error rate on the training samples. Subsequent classifiers are incorporated by adding the one that has the lowest error rate on the samples that were missclassified by the partial ensemble, i.e.,

$$\mathcal{E}_u = \underset{k}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(\mathcal{E}_{u-1} \cup \mathcal{NC}_k\left(\mathbf{x}_i\right), c_i\right) \left| \mathcal{L}\left(\mathcal{E}_{u-1}\left(\mathbf{x}_i\right), c_i\right) = 1 \right. \tag{6.7}$$

Similar to error reduce, the index k runs over all classifiers that are not already included into the ensemble.

- ***Margin Distance Ensemble*** (MDE): Margin distance minimization is a method for pruning bagging ensembles, which was introduced in (Martínez-Muñoz and Suárez, 2004); here, we approach it in the context of Pareto ensemble. In this approach, a signature vector $\mathbf{s}^{(t)}$ of a classifier k is defined as:

$$\mathbf{s}_i^{(k)} = 1 - 2 \times \mathcal{L}\left(\mathcal{NC}_k\left(\mathbf{x}_i\right), c_i\right), \forall i \in \{1, \ldots, N\} \tag{6.8}$$

where $\mathbf{s}_i^{(k)}$ is equal to 1 if the k^{th} classifier correctly predicts the i^{th} instance or -1 otherwise. The average signature vector is defined as:

$$\overline{\mathbf{s}_i} = \frac{1}{|\mathcal{NC}|} \sum_{k=1}^{|\mathcal{NC}|} \mathbf{s}_i^{(k)}, \forall i \in \{1, \ldots, N\} \tag{6.9}$$

An instance i is correctly classified by the ensemble if $\overline{\mathbf{s}_i} > 0$. The goal of this approach is to minimize the distance of the average signature vector to a positive reference point. Let $\mathbf{o}$ be the reference point, the classifier selected in the u iteration is determined by

$$\mathcal{E}_u = \underset{k}{\operatorname{argmin}} \left\| \mathbf{o} - \frac{1}{|\mathcal{NC}|} \left(\mathbf{s}^{(k)} + \sum_{t=1}^{u-1} \mathbf{s}^{(t)} \right) \right\| \tag{6.10}$$

where $\|d\|$ is the euclidean norm and $0 < o_i < 1, \forall i \in \{1, \ldots, N\}$.

- ***Boosting ordering***: This approach consists of selecting, at each iteration, the classifier that minimizes the weighted error rate. The weighting scheme used in the error computation follows the one given by Adaboost (Freund and Schapire, 1997). Here, instead of training a classifier with the weighted training set, the classifier with the lowest error rate on such dataset is selected from the pool of classifiers available in the resulting non-dominated set.

- ***Pareto Knee-Based*** (PKB): The basic idea in this approach is to combine the solutions on the knee region of the Pareto curve into an ensemble. In the absence of an explicit user preference, we can consider that the points in this region represent the preferences themselves. The issue of finding these solutions is faced in a straightforward fashion, which is inspired in the L-method (Salvador and Chan, 2004), in which they define a knee point as the one with the maximal

curvature. Here, instead of choosing a single point, the idea is to find the pair of points that best fit the set of non-dominated points with straight lines, and these define the knee region. For doing so, a bisection is performed over the set of non-dominated points and, for each section, the pair of points when they are connected with straight lines and that best approximate the entire set are chosen. Figure 6.4 shows an example where for the given trade-off curve, we have the straight lines that best approximate the curve and the points between the lines, enclosed with a dashed red line, constitute the knee region.

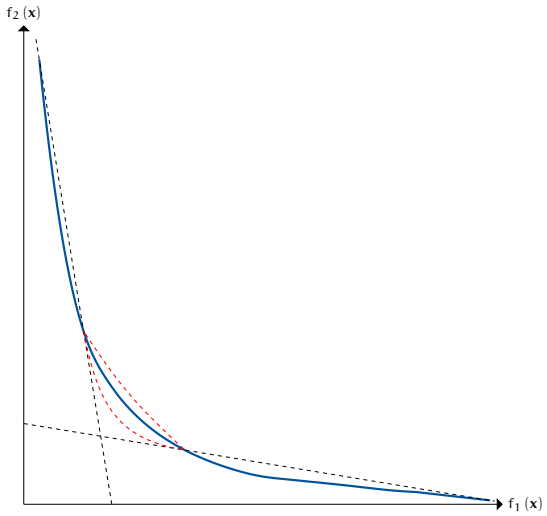


Figure 6.4: The solutions in the knee-region of a Pareto curve are selected to construct an ensemble. First, the set of points when connected with straight lines that best approximate the non-dominated set are chosen. The points in the enclosed red line represent the knee-region.

- ***Pareto evolutionary ensemble*** (PEE): In this strategy, differential evolution is performed to evolve classifier weights. The goal is to find a set of weights for each member in the ensemble that minimizes the error rate.

These schemes describe different ways of choosing a full classification model set from the non-dominated set. The next step is to combine the information given by each full model into a single one. We face this issue in a straightforward fashion by taking a majority vote given by the individual prediction of the models from an ensemble.

6.3 Experiments and Results

This section describes the experimental study performed to test our proposal. First, we compare among strategies for the ensemble construction. Next, we compare the best ensemble with respect to single-objective solution. After that, a comparison with full model selection state of the art methods is presented.

6.3.1 Experimental Settings

For the experimental evaluation, a set of 25 benchmark datasets is used from both the IDA (Rätsch et al., 2001) and the KEEL (Alcalá et al., 2011) repositories, which have been previously used in several supervised classification studies. Table 6.2 describes the suite of 25 datasets. For each dataset, it shows the total number of samples (including training and testing), the number of features, the number of classes and the number of partitions. Datasets from 1 to 13 are taken from the IDA repository, while from 14 to 25 are taken from the KEEL repository. These datasets are diverse both in number of samples, in number of attributes, and in number of classes. For each dataset, the model selection task is performed independently for each partition.

Regarding the configuration for EN-MOMS-PbE parameters, an external population size of 25, and an internal population size of 20 are used. Probabilities are fixed to 1.0 and 0.8 for external and internal crossover and $^1/_n$ for mutation both external and internal, where n is equal to the number of decision variables. The stopping criteria are defined as performing 200 and 350 fitness evaluations for internal and external optimizers, respectively, or the improvement of the non-dominated front being less than 0.1 in the hyper-volume measure.

Four different metrics are adopted to assess the performance of EN-MOMS-PbE. These are the accuracy, average accuracy, kappa statistic, and the AUC[2]. These metrics allow to evaluate different aspects, i.e., accuracy focuses on evaluating global performance of the model, while average accuracy evaluates the average performance per class, AUC does the same considering different thresholds, and kappa statistics measures the degree of agreement between the predicted and the real classes.

6.3.2 Comparing Among Pareto-based Ensemble Approaches

The obtained results when using the Pareto ensemble approaches are shown in Table 6.3. From this Table we can note that in average PEE gets the best scores over different measurements. The performances of GPE, Boosting, and PER are close to those

[2]A detailed description of theses measures can be found in subsection 4.3.1

Table 6.2: Description of the datasets. The number of instances, attributes, classes, and replications is shown for each dataset.

ID	Dataset	Samples	Atts.	Classes	Replications
1	Banana	5300	2	2	100
2	Breast Cancer	277	9	2	100
3	Diabetes	768	8	2	100
4	Flare Solar	1066	11	2	100
5	German	1000	20	2	100
6	Heart	270	13	2	100
7	Image	2310	20	2	20
8	Ringnorm	7400	20	2	100
9	Splice	3175	60	2	20
10	Thyroid	215	5	2	100
11	Titanic	2001	3	2	100
12	Twonorm	7400	20	2	100
13	Waveform	5000	21	2	100
14	Appendicitis	106	7	2	10
15	Balance	625	4	3	10
16	Ionosphere	351	33	2	10
17	Iris	150	4	3	10
18	Mammographic	830	5	2	10
19	Phoneme	5404	5	2	10
20	Pima	768	8	2	10
21	Sonar	208	60	2	10
22	Wdbc	569	30	2	10
23	Wine	178	13	3	10
24	Wisconsin	683	9	2	10
25	Yeast	1484	8	10	10

obtained by PEE. On the other hand, MDE is the one with the worst performance in all measurements.

Apart from the results, an statistical analysis over the results gathered by the different Pareto ensembles approaches is performed. The Wilcoxon signed rank test is used in order to conduct the pairwise comparisons among all of them. The considered level of significance in the statistical test is set to $\alpha = 0.05$. Table 6.4 summarizes the results obtained by the statistical tests. For each method in the rows, the column

Table 6.3: Obtained results over different scores when comparing the different Pareto ensemble approaches

Score	PER	PCE	MDE	Boosting	GPE	PKB	PEE
Accuracy	86.49 ± 11.36	81.78 ± 12.07	80.51 ± 13.08	85.51 ± 11.30	86.36 ± 11.03	85.17 ± 10.67	$\mathbf{86.81 \pm 11.22}$
Avg. Acc.	83.99 ± 13.75	78.61 ± 13.81	77.54 ± 14.56	82.46 ± 13.81	83.63 ± 13.29	82.30 ± 12.92	$\mathbf{84.40 \pm 13.50}$
Kappa	70.63 ± 24.76	60.29 ± 26.30	57.67 ± 28.05	68.03 ± 25.19	70.17 ± 24.07	67.64 ± 23.45	$\mathbf{71.29 \pm 24.30}$
AUC	86.42 ± 10.83	84.29 ± 10.56	79.38 ± 12.24	87.32 ± 10.34	88.40 ± 10.26	87.80 ± 10.27	$\mathbf{88.61 \pm 10.42}$

Table 6.4: Summary of the results from the Wilcoxon test

Dataset	Acc.		$\overline{\text{Acc.}}$		Kappa		AUC	
	$+$	$\pm$	$+$	$\pm$	$+$	$\pm$	$+$	$\pm$
PER	3	5	3	5	3	5	2	4
PCE	0	1	0	1	0	1	1	1
MDE	0	1	0	1	0	1	0	0
Boosting	2	4	2	4	2	4	2	4
GPE	4	6	4	6	4	6	4	6
PKB	2	3	2	3	2	3	2	5
PEE	5	6	5	6	5	6	5	6

represented by the symbol "+" indicates the number of variants of the Pareto ensemble approaches that were outperformed according to the Wilcoxon test. The column with the "$\pm$" symbol indicates the number of wins and ties obtained by the method in the row.

From the results and the statistical results presented in the Tables, we can highlight the following:

- Among the proposed Pareto ensemble strategies, the Pareto Evolutionary Ensemble (PEE) gets the highest performance in the four considered metrics. The second one seems to be GPE followed by PER and boosting.
- On the other hand, MDE and PCE are the ones with the worst performances. They were outperformed by the others. This is due to the way in which MDE and PCE are constructed. MDE gives equal importance to all instances, indicating that all of them would have the same margin. PCE adds a classification model only considering those instances that were misscorrectly classified and the effect over the ones that were correctly classified are not taken into consideration.
- Performance of different Pareto ensemble approaches were consistent over all different metrics.

Table 6.5: Average results over different scores when comparing with a single criterion approach

Score	Single	PEE
Average	85.26 ± 11.49	$\mathbf{86.81 \pm 11.22}$
Avg. Acc.	83.06 ± 13.58	$\mathbf{84.40 \pm 13.50}$
Kappa	68.33 ± 25.01	$\mathbf{71.29 \pm 24.30}$
AUC	85.73 ± 10.54	$\mathbf{88.61 \pm 10.42}$

- In spite of the lack of an statistical significant difference between GPE and PEE, the former considers all solutions in the resulting Pareto front, which can increase the processing and prediction time.

Therefore, based on the comparative study, the EN-MOMS-PbE with PEE strategy can be chosen as the best one and it is used hereafter in the following comparisons.

6.3.3 Comparing with a Single Aggregated Criterion

The goal of this section is to determine the suitability of the Pareto processing instead of choosing a single solution. To accomplish this, we choose a single solution from the resulting non-dominated front. Here, an aggregated function that captures in a single expression both criteria is defined as the criterion to determine what solution should be chosen. The aggregated formula is defined such that both criteria have the same weight and it is defined as follows:

$$f_{agg} = \max\left(\frac{1}{2}f_1, \frac{1}{2}f_2\right) \tag{6.11}$$

where f_1 and f_2 are the normalized objective values. Each solution in the generated Pareto front is evaluated using the aggregated function and the one that minimizes it is chosen as the single solution.

Table 6.5 shows the results obtained when a single solution is chosen (EN-MOMS-Single) and those reported by the proposal when using PEE as the Pareto processing. This Table shows the accuracy performance, the average accuracy, the kappa statistic, and the AUC score.

As it is suggested by Demšar (2006), the Wilcoxon signed rank test is applied to make the statistical evaluation of the classifiers' performance over different datasets. Table 6.6 shows the reported results with this test.

Based on these Tables, we can summarize this comparison as follows:

Table 6.6: Results obtained by the Wilcoxon signed rank test when comparing EN-MOMS-Single with EN-MOMS-PbE with PEE Pareto processing

Score	R^+	R^-	p-value
Accuracy	9	267	< 0.01
Avg. Acc.	48	277	0.002
Kappa	40	285	< 0.01
AUC	3	297	< 0.01

- Pareto processing outperforms the single solution approach. This behavior is consistent along different datasets and considered scores.
- Improvement above 1% for accuracy and average accuracy, and above 3% for kappa and AUC are reached.
- Wilcoxon test showed with $p < 0.01$ that PEE is able to statistically outperform the single solution in all evaluation criteria.
- Based on these observations, we can conclude that the Pareto processing is a competent method for performing classification tasks.

6.3.4 Comparing with Full Model Selection Methods

Once an evaluation among different Pareto processing approaches and a comparison of its advantage over a single criterion has been performed, the next step is to assess the performance of EN-MOMS-PbE with PEE Pareto processing against full model selection methods reported in the literature. EPSMS (Escalante et al., 2010) and Auto-Weka (Thornton et al., 2013) methods are selected to this aim. Both EPSMS and Auto-Weka allow constructing ensemble of classification models. This makes a fairer comparison with the EN-MOMS-PbE using the Pareto Evolutionary Ensemble approach.

The resulting accuracy, average accuracy, kappa statistic, and AUC score are reported in Table 6.7. For the statistical comparison, the Friedman test with the Holm's procedure as the post-hoc test are used. The results of these statistical tests are shown in Table 6.8.

Here, the reported results can be summarized as follows:

- EN-MOMS-PbE with PEE Pareto processing outperforms, in average, both EPSMS and Auto Weka.

Table 6.7: Average results over different scores when comparing with full model selection methods

Score	PEE	EPSMS	Auto-Weka
Accuracy	$\mathbf{86.81 \pm 11.22}$	83.76 ± 11.75	85.82 ± 13.38
Avg. Acc.	$\mathbf{84.40 \pm 13.50}$	82.57 ± 12.85	83.02 ± 16.24
Kappa	$\mathbf{71.29 \pm 24.30}$	66.58 ± 22.32	69.08 ± 26.35
AUC	$\mathbf{88.61 \pm 10.42}$	87.96 ± 10.34	85.54 ± 11.16

Table 6.8: Results of the statistical test. The average ranking reported by the Friedman test and the adjusted p-value (APV) obtained by the Holm's procedure are reported for each score

Method	Acc.		Avg. Acc.		Kappa		AUC	
	Rank	APV	Rank	APV	Rank	APV	Rank	APV
EPSMS	1.42	0.001	1.03	—	1.66	0.018	2.12	0.322
Auto-Weka	2.18	0.437	1.90	—	1.94	0.104	1.48	0.002
PPE	2.40	—	2.20	—	2.40	—	2.40	—

- EPSMS showed a better performance for the AUC score.
- The Friedman test and the Holm's procedure reveal that EN-MOMS-PbE-PEE statistically performs better than EPSMS in the accuracy score and kappa statistic, and performs better than Auto-Weka in the AUC score.
- The Friedman test is not able to find a statistical significant difference for the average accuracy score. In this score, the three methods are equivalent.
- EN-MOMS-PbE showed to be more robust over different evaluation criteria than the others.

6.3.5 Discussion

Based on the experimental study, we can note that the definition of the solutions to be chosen from the Pareto ensemble has an impact in the performance of the constructed model. The best Pareto ensemble strategies were PEE, GPE, and PER. On the other hand, MDE and PCE showed the lowest performance in the four considered measures. This is due to the assumptions behind these approaches. For instance, MDE computes a margin between instances and a reference point, which, however, gives the same importance to all samples in the dataset. This can result contradictory, since some

instances can be closer to the boundaries than others. PCE, on the other hand, tries to add to the sub-ensemble, the model that best complement the classification of the previous misclassified instances. PCE, however, does not take into account that by adding such model to the ensemble, can affect the instances that were previously correctly classified. These fundamental assumptions make these approaches unsuitable for the problem at hand.

The performance of PEE and GPE is almost similar and a statistical significant difference is not found, PEE performs a pruning step over the Pareto solutions, which results in an ensemble with a lower number of elements. This can be translated in reducing the processing time when a new instance is classified as well as a reduction in memory.

The experimental evaluation of PEE against a single solution showed the advantages of the Pareto processing. For most of the datasets, the performance of PEE is better than the one reported by the single solution approach. This is a well known fact in machine learning.

A comparison with two full model selection methods reported in the literature showed that EN-MOMS-PbE with PEE Pareto processing can significantly outperform them. EPSMS gets its best performance with the AUC score. This is due to the fact that EPSMS uses the AUC score as the optimization criterion. On the other hand, it should be noted that the proposal does not use the AUC score as the optimization criteria. Notwithstanding, the proposal is able to perform better than EPSMS in this score. Auto-weka, on the other hand, uses the error rate as the optimization criterion. This is closer to the optimization criterion used in EN-MOMS-PbE. It can be noted that when comparing EPSMS and auto-weka, the former outperforms in the criterion that was used in its optimization procedure. This is an interesting result since EN-MOMS-PbE outperforms the reference methods, even in those scores for which the others were trained. Therefore, EN-MOMS-PbE is able to provide full classification models without significantly overfitting to a given criterion.

6.4 Final Remarks

This chapter has introduced EN-MOMS-PbE, which formulates the full model selection problem as a hierarchical optimization one. EN-MOMS-PbE optimizes in an upper level the model structure and in a lower level the hyper-parameters corresponding to a given model structure. It further exploits the information contained in the resulting Pareto front through several ensemble approaches. The formulation of EN-MOMS-PbE

has shown the following advantages:

- the lower level optimization only deals with the hyper-parameters that are meaningful to the given model structure;
- different Pareto ensemble models can be constructed without significantly increasing the computational cost;
- it has a competitive performance over different problems;
- the generated models showed a good performance over different scores, even those that are not considered in the optimization task.

The experimental evaluation showed that with only a subset of the Pareto optimal solutions it is possible to construct a highly effective full classification model. Furthermore, the Pareto processing showed to perform better than choosing a single solution from the Pareto front. Moreover, the comparison of EN-MOMS-PbE against the full model selection methods reported in the literature and the statistical evaluation showed that EN-MOMS-PbE is a competitive method and a reference method to defeat.

One should note that due to the hierarchical formulation of the problem, this yields a relatively high number of fitness evaluations during the optimization task. This means that performing selection requires a higher computational effort due to the great amount of models that should be trained and tested. The reduction of the number of fitness function evaluations is a work to face in the next chapter.

Chapter 7

Surrogate-Assisted Multi-Objective Full Model Selection

We cannot solve our problems with the same thinking we used when we created them

Albert Einstein

The optimization of the hyper-parameters is a crucial step in order to achieve highly reliable classification models. This step, however, consumes most of the time, due to the evaluation of the performance of the hyper-parameters. In the last years, the use of surrogate models as cheap synthetic functions has gained interest as an alternative to reduce the cost of expensive optimization problems. Inspired in this, a surrogate optimization approach is here adopted to face the full model selection problem.

This chapter introduces **SEN-MOMS-PbE**: **S**urrogate **E**volutionary **N**ested **M**ulti-**O**bjective Full **M**odel **S**election, the surrogate-assisted version of EN-MOMS-PbE. First, some related works that address the model selection problem using surrogate optimization are described. Next, the adaptation of the surrogates to the problem at hand is presented, followed by the experimental study.

7.1 Related Works

Surrogate-assisted optimization aims at reducing the computational cost of expensive optimization problems by replacing the expensive function to be optimized by a cheap approximation. In the last years, a growing interest has emerged on using surrogate-assisted optimization to solve different problems, as those described in (Arias-Montano et al., 2012; Couckuyt et al., 2010; Shahrokhi and Jahangirian, 2010).

In the field of machine learning and pattern recognition, one of the problems is to determine an appropriated configuration of a learning algorithm to construct a

classification model. This task traditionally requires a large number of experiments, where different configurations are tested looking for the one that best behaves according to a given criterion. As a result of this large number of trials, the model selection task could become a computational expensive problem. Inspired by this, in the last years, several studies have exploited the ideas of surrogate-assisted optimization in the model selection problem.

Bergstra et al. (2011) used a Sequential Model-Based Optimization (SMBO) and Gaussian Process as the surrogate model to optimize the hyper-parameters of a deep belief network. When using the surrogate, the expected improvement is the function to optimize. Their proposal is tested on the MNIST dataset and showed that the surrogate based approach performs better than a manual tuning and a brute-force random search.

In other study, Bardenet et al. (2012, 2013) proposed the used of a collaborative approach to the hyper-parameter optimization problem, in which the idea is to learn from previous experiences. This past knowledge is used when the hyper-parameters for a new classification problem should be adjusted. SMBO and Gaussian Processes are used and AdaBoost as the classification method.

Pilát and Neruda (2013) proposed a memetic algorithm for which in the local search stage a surrogate model is used. They optimize the parameters for a multi-layer perceptron. Their approach does not only optimize the hyper-parameters for the MLP, but also performs a model selection for the surrogate to be used in the optimization. A meta-learning approach is used to perform the model selection for the surrogate model. The criteria to optimize are the error rate, the kappa statistic, and the root mean square error. In the decision making step; nonetheless, they only consider the error as the most important criterion.

Yogatama and Mann (2014) proposed a transfer learning method which is coupled with the optimization step to perform the selection of the hyper-parameters. The idea is to learn a function from all datasets that mimic the function error. They have also used SMBO and Gaussian processes. For a new dataset, a Gaussian process is fit and it is defined as a criterion to determine if a dataset is included.

In a more recent study, Eggensperger et al. (2015) proposed a benchmark of surrogate functions for an umbrella of datasets. In this benchmark, the hyper-parameters and the model performance are learned. The Sequential Model-based Algorithm Configuration and Gaussian processes are used as the search engine and surrogate, respectively.

These studies have generally focused on a single model type at the time. In the

problem of full model selection, perhaps the only exception is Auto-Weka Thornton et al. (2013), which uses Gaussian processes as the surrogate. Auto-Weka only considers feature selection as pre-processing, but in a broader interpretation data transformation, data sampling, and noise filtering can be encompassed during the data pre-processing step. This yields high degrees of freedom in the full model selection problem. The latter has been approached in chapter 6, where EN-MOMS-PbE is proposed. In this chapter, EN-MOMS-PbE is used as the base method to explore surrogate-assisted optimization in the full model selection problem. Thus, this results in an extended version called **SEN-MOMS-PbE**: **S**urrogate **E**volutionary **N**ested Full **M**odel **S**election with **P**areto-**b**ased **E**nsembles. The main contributions of this chapter are the following:

- the hybridization of EN-MOMS-PbE with a surrogate-optimization approach, which allows to significantly reduce the number of required fitness function evaluations;
- an experimental evaluation of the model performance and the reduction when the surrogates are used.

We assess the performance of SEN-MOMS-PbE using a set of 25 datasets and is compared against EN-MOMS-PbE. Next, the proposed approach is described in detail.

7.2 Surrogate-Assisted Evolutionary Nested Multi-Objective Full Model Selection

SEN-MOMS-PbE is described in Figure 7.1, which resembles Figure 6.1 showed in chapter 6. The main difference between both approaches is the hybridization of surrogate models. One should note that the most computationally expensive part of EN-MOMS-PbE is in the lower level optimization. This is because for each potential model structure, a search is performed to determine the hyper-parameters. The optimization aims at minimizing the conflicting criteria, which are determined in an experimental fashion. Hence, for each possible structure with a given hyper-parameter configuration, the model is trained and tested a number of times.

Here, SEN-MOMS-PbE takes the aforementioned into consideration and performs the lower level optimization using surrogates. Surrogates aim at approximating the error and the complexity functions from the hyper-parameters; i.e., given a model configuration, the surrogates provide an estimation of the expected response of the optimization criteria. This means that instead of training and testing each possible model configuration, only those with best trade-off according to the surrogates are

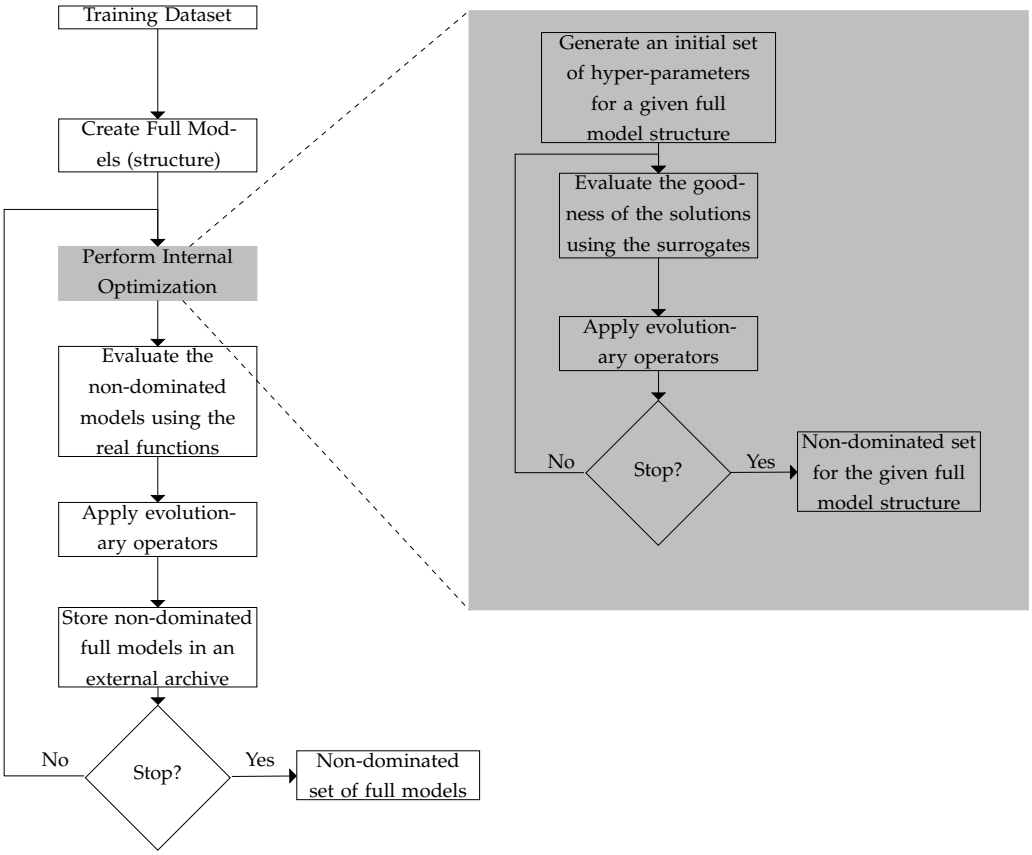


Figure 7.1: General architecture of SEN-MOMS-PbE

considered. These solutions can be used to update the surrogate. Here, the definition of this approach is as follows:

- **Surrogate model**: The surrogate model, here, is used as an inexpensive function to approximate the behavior of the optimization criteria. However, we need to define the regression functions to be used. For this sake, radial basis functions (RBFs) are used. We choose this type of surrogate modeling because it allows modeling the two criteria in a single training phase. RBFs have hyper-parameters to define, such as the kernel type and the kernel's parameters. In order to avoid facing the hyper-parameter optimization, we perform an ensemble of RBFs, for which three different kernel types are considered: linear, cubic, and splines. In this way, we hope to get a balance between exploration and exploitation. The individuals predictions given for each surrogate are combined in order to make a single prediction; this is graphically depicted in Figure 7.2.

- **Updating criterion**: After the internal optimization is performed with the surro-

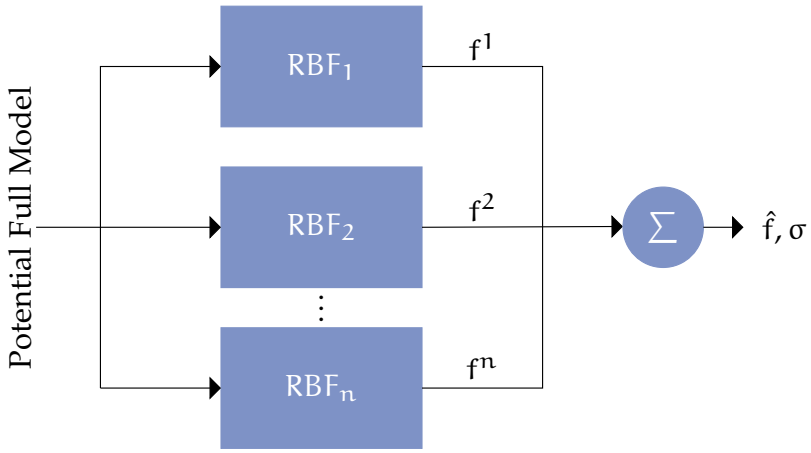


Figure 7.2: Combining the individual predictions given for each surrogate model into a single approximation. f^1, f^2, and f^n represent the individual predictions; $\hat{f}$ and σ represent the mean value of the predictions and the corresponding standard deviation.

gate models, a set of non-dominated solutions is obtained. This set can be used to update the landscape of the surrogates. An option is to use all of them in the updating stage. Here, however, we approach this in a different way. Since an ensemble of surrogates is used, it is possible to compute the mean and the standard deviation from the approximation values. From these values, we can construct a Region of the approximation which is used to determine which surrogates are going to be updated. Figure 7.3 shows two examples of the uncertainty region. The procedure to update the surrogate is as follows:

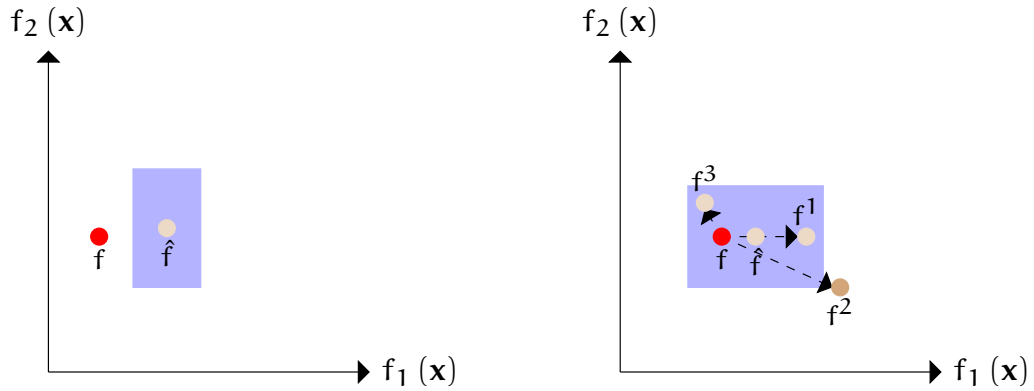


Figure 7.3: Uncertainty region when using surrogates. $\hat{f}$ represents the mean value of the approximation and f the obtained one when the solution is evaluated with the expensive fitness functions.

1. Determine the area of the uncertainty region, which is defined by the mean and standard deviation per each axes, of each of the obtained Pareto solutions from the internal optimizer and choose the one with the highest area;
2. from the selected one, determine whether the point, when it is evaluated with the expensive function, is inside or outside from the region
   - if the point is outside of the uncertainty region, this could mean that the approximations are not accurate enough and the solution is used for updating all surrogates;
   - if the point is inside of the uncertainty region, compute the similarity from those points obtained with the surrogates and the one with the expensive functions is computed. For doing so, the Euclidean distance is computed from the desired point to each approximation and the surrogate that provides the least similarity approximation is updated with the solution. In this case, not all surrogates are updated, but only those whose approximation would be the most different to the real one.

   In Figure 7.3 these two cases are depicted. The left figure represents the first case when the real value is outside from the region; in the right figure, the second case when is inside the region. In this second case, the Euclidean distance from the real point to each approximation is computed.

These two main mechanisms are integrated into EN-MOMS-PbE in order to enable the inclusion of the surrogate. Next, we present the experimental evaluation of this.

7.3 Experiments and Results

This section presents the experimental evaluation performed. First, we present the experimental settings and after that, the experiments performed that allow assessing the performance of the proposal

7.3.1 Experimental Settings

For the experimental comparison, we used the set of 25 benchmark datasets described in Table 6.2, which are in the IDA (Rätsch et al., 2001) and in the KEEL (Alcalá et al., 2011) repositories. The same partitions and number of replications are performed for each case, by proceeding in this manner we can make a fair comparison between

EN-MOMS-PbE and SEN-MOMS-PbE. The same parameter configuration of EN-MOMS-PbE is used for the surrogate-assisted version.

Regarding the performance assessment, two types of comparisons are performed. The first one aims at comparing the quality of the approximation of the Pareto front, while the second one aims at comparing the quality of the full classification models generated by both approaches. These comparisons are presented next. For the first goal, we used two measures, which are the hyper-volume and the number of fitness evaluations. For the second goal, we used a set of four measures, which are classification rate, average classification, kappa statistic, and AUC.

7.3.2 Experimental Results

The goal of this section is twofold. First, we compare the approximation in terms of the Pareto front from the one obtained by EN-MOMS-PbE and the one generated by SEN-MOMS-PbE. After that, we perform a comparison in terms of the quality of the generated full classification models.

Comparing the Approximations of the Pareto Fronts

The obtained results in terms of the quality of the approximation and the fitness evaluations performed per each approach are presented in Table 7.1. One should note that both objectives are normalized and the point $(1, 1)$ is used as the reference point in the hyper-volume computation. The use of non-parametrical statistical tests is recommended for comparing evolutionary algorithms (Derrac et al., 2011; García et al., 2009). Here, the Wilcoxon signed rank test is used for comparing the performance in the hyper-volume score.

Based on the results reported in Table 7.1 and the Wilcoxon test, we summarize the following:

- The average performance of both approaches, in terms of the hyper-volume score, is quite similar. According to the Wilcoxon test, there is not a statistically significant difference at the $\alpha = 0.05$ level. The reported p-value for the Wilcoxon test is 0.184.

- There is a great difference between the number of fitness evaluations performed per each method. When the surrogates are used, a reduction around an 80% of the fitness evaluations is achieved. Therefore, the hybridization with the surrogates allows reducing the number of the expensive evaluations during the optimization

Table 7.1: Results in hyper-volume (HV) and number of fitness evaluations performed for each approach. The reported results are the average and standard deviation over the replications and the best one is shown in **boldface**

Dataset	EN-MOMS-PbE		SEN-MOMS-PbE	
	HV	No. Evals.	HV	No. Evals.
Banana	**0.88 ± 0.06**	14361.20 ± 2793.91	0.84 ± 0.08	2516.50 ± 139.84
Breast cancer	**0.38 ± 0.04**	15180.00 ± 3590.35	0.37 ± 0.05	2966.30 ± 433.09
Diabetis	0.72 ± 0.16	19549.60 ± 2800.69	**0.74 ± 0.14**	3532.70 ± 491.40
Flare solar	**0.67 ± 0.16**	18666.20 ± 3155.84	**0.67 ± 0.16**	2671.50 ± 228.43
German	**0.41 ± 0.03**	20896.00 ± 3152.66	**0.41 ± 0.03**	3286.70 ± 390.13
Heart	**0.89 ± 0.08**	20085.80 ± 1731.52	0.87 ± 0.10	3012.60 ± 388.41
Image	**0.98 ± 0.02**	15772.40 ± 2381.73	**0.98 ± 0.01**	3223.10 ± 599.06
Ringnorm	**0.99 ± 0.01**	11105.50 ± 2467.16	**0.99 ± 0.01**	3741.10 ± 575.08
Splice	**0.96 ± 0.03**	17112.20 ± 4109.32	0.94 ± 0.04	3161.60 ± 400.14
Thyroid	**0.96 ± 0.03**	12482.00 ± 4140.11	**0.96 ± 0.03**	2707.00 ± 638.84
Titanic	**0.64 ± 0.13**	11201.40 ± 2444.64	0.57 ± 0.17	2684.30 ± 514.60
Twonorm	**0.97 ± 0.01**	12568.50 ± 4412.21	**0.97 ± 0.01**	3068.70 ± 529.15
Waveform	**0.94 ± 0.07**	17316.60 ± 2903.21	**0.94 ± 0.07**	2858.10 ± 220.21
Appendicitis	**0.64 ± 0.04**	11286.10 ± 4084.95	0.63 ± 0.05	2969.40 ± 631.36
Balance	0.87 ± 0.08	12494.30 ± 3196.84	**0.93 ± 0.06**	3080.80 ± 679.86
Ionosphere	0.95 ± 0.05	14936.60 ± 3170.41	**0.97 ± 0.04**	3182.20 ± 719.75
Iris	**0.96 ± 0.01**	10370.50 ± 2153.58	**0.96 ± 0.01**	2988.10 ± 555.86
Mammographic	0.80 ± 0.11	17820.60 ± 3153.16	**0.85 ± 0.07**	2873.00 ± 567.95
Phoneme	0.83 ± 0.08	18468.10 ± 2942.81	**0.85 ± 0.08**	2896.80 ± 251.78
Pima	**0.48 ± 0.02**	12519.50 ± 2680.59	**0.48 ± 0.02**	3059.40 ± 517.74
Sonar	0.81 ± 0.12	20752.20 ± 3166.33	**0.93 ± 0.02**	3063.80 ± 404.24
Wdbc	**0.98 ± 0.02**	14736.40 ± 3414.73	**0.98 ± 0.03**	3110.30 ± 589.22
Wine	**0.99 ± 0.01**	8826.30 ± 2554.06	0.98 ± 0.01	2876.80 ± 355.79
Wisconsin	**0.96 ± 0.00**	15159.20 ± 2649.93	**0.96 ± 0.00**	3663.30 ± 855.68
Yeast	0.29 ± 0.14	14101.50 ± 3313.74	**0.32 ± 0.11**	3315.00 ± 315.90
Ave.	**0.80 ± 0.21**	15110.75 ± 3387.77	**0.80 ± 0.21**	3060.36 ± 289.49

step without significantly degrading the quality of the approximation to the generated Pareto front.

- EN-MOMS-PbE required, in average, a computational time of 4.29 ± 16.53 hours, while SEN-MOMS-PbE required 0.72 ± 2.60 hours. This represents a significant saving in computational time.

Once the performance of both approaches in terms of the Pareto front approximation is almost the same, we present the comparison in terms of the classification performance of the full models selected for each method.

7.3.3 Comparing the Performance of the Full Classification Models

In this section, we compare in terms of the performance reached by the classification models. It is worth noting that the Pareto ensemble approach used in SEN-MOMS-PbE is PEE due to the fact that this is the one that gives the best performance with the non-surrogate-assisted version. Table 7.2 shows the results for the four considered scores in classification both for EN-MOMS-PbE and SEN-MOMS-PbE. These scores are the standard accuracy performance, the average accuracy performance, the kappa statistic, and the AUC score.

Additionally to the results, statistical tests for each score are performed. The Wilcoxon signed rank test is used to compare the performance between the two approaches on each score. Table 7.3 summarizes the results reported by the Wilcoxon test.

From previous Tables, we can highlight the following:

- The performance of both approaches in the standard accuracy is almost the same.
- With respect to average accuracy and kappa, EN-MOMS-PbE has a slightly better performance. However, the Wilcoxon test revealed that this difference is not statistically significant at the $\alpha = 0.05$ level.
- When considering the AUC score, SEN-MOMS-PbE, the surrogate-assisted version, improves EN-MOMS-PbE in around a 1.5%. The Wilcoxon test showed that this is an statistically significant difference at the $\alpha = 0.05$ level.

In summary, when hybridizing EN-MOMS-PbE with a surrogate model, we can note that we can significantly reduce the number of fitness evaluations required to get a good approximation of the Pareto front without significantly degrading it. This is

Table 7.2: Reported results for accuracy, average accuracy, kappa statistic, and AUC for EN-MOMS-PbE and SEN-MOMS-PbE. The reported results are the average and standard deviation over the replications and the best one is shown in **boldface**

Dataset	EN-MOMS-PbE				SEN-MOMS-PbE			
	Acc	Avg. Acc.	Kappa	AUC	Acc	Avg. Acc.	Kappa	AUC
Banana	**88.34 ± 0.69**	87.74 ± 0.83	76.19 ± 1.48	93.88 ± 1.08	88.30 ± 0.44	**88.07 ± 0.41**	**76.29 ± 0.88**	**95.26 ± 0.66**
Breast cancer	70.26 ± 3.04	61.03 ± 4.98	23.48 ± 10.30	68.09 ± 5.37	**71.69 ± 4.02**	**61.31 ± 5.67**	**24.71 ± 11.34**	**70.30 ± 5.47**
Diabetis	**76.47 ± 1.64**	69.75 ± 2.10	42.99 ± 3.94	**82.39 ± 1.56**	75.23 ± 1.86	**70.97 ± 2.52**	**43.19 ± 4.44**	82.25 ± 1.42
Flare solar	**66.72 ± 2.36**	**66.36 ± 2.96**	**32.57 ± 5.54**	**74.62 ± 1.91**	66.20 ± 1.42	66.07 ± 1.83	31.87 ± 3.36	70.08 ± 2.14
German	75.17 ± 2.75	**69.53 ± 2.51**	39.71 ± 4.32	77.88 ± 2.86	**75.80 ± 3.35**	69.37 ± 3.47	**40.13 ± 6.88**	**79.02 ± 3.27**
Heart	**82.40 ± 4.15**	**81.86 ± 4.05**	**64.17 ± 8.19**	**88.76 ± 2.78**	79.70 ± 3.80	79.04 ± 3.75	58.58 ± 7.59	88.23 ± 3.26
Image	97.92 ± 0.53	97.81 ± 0.63	95.75 ± 1.10	99.34 ± 0.54	**98.02 ± 0.42**	**97.95 ± 0.45**	**95.96 ± 0.86**	**99.82 ± 0.05**
Ringnorm	**98.29 ± 0.22**	**98.29 ± 0.23**	**96.59 ± 0.45**	99.27 ± 0.50	98.28 ± 0.46	98.27 ± 0.47	96.55 ± 0.92	**99.85 ± 0.02**
Splice	95.95 ± 0.59	95.96 ± 0.63	91.89 ± 1.19	98.47 ± 0.66	**96.15 ± 0.60**	**96.22 ± 0.58**	**92.30 ± 1.20**	**99.12 ± 0.20**
Thyroid	94.53 ± 3.01	92.03 ± 5.00	86.07 ± 8.38	97.18 ± 0.98	**94.67 ± 2.46**	**92.10 ± 3.77**	**86.78 ± 6.34**	**98.96 ± 1.70**
Titanic	**76.99 ± 1.49**	67.50 ± 3.28	39.70 ± 6.07	**71.85 ± 1.88**	76.54 ± 2.33	**67.99 ± 2.26**	**40.17 ± 4.67**	64.26 ± 1.94
Twonorm	**97.40 ± 0.18**	**97.40 ± 0.18**	**94.81 ± 0.36**	99.58 ± 0.16	92.59 ± 14.17	92.58 ± 14.20	85.17 ± 28.39	**99.74 ± 0.04**
Waveform	**90.40 ± 0.55**	**90.22 ± 0.65**	**78.75 ± 0.97**	96.43 ± 0.73	89.87 ± 1.38	89.01 ± 1.17	77.32 ± 2.74	**96.82 ± 0.58**
Appendicitis	**85.00 ± 7.42**	**76.11 ± 14.67**	**50.09 ± 26.52**	71.58 ± 9.89	84.27 ± 10.64	72.64 ± 15.71	47.78 ± 31.99	**80.00 ± 12.55**
Balance	**97.92 ± 1.25**	**96.13 ± 4.51**	**96.33 ± 2.22**	91.06 ± 7.38	94.41 ± 4.56	89.18 ± 10.19	90.24 ± 7.91	**97.19 ± 2.70**
Ionosphere	**96.29 ± 1.83**	**95.39 ± 2.39**	**91.84 ± 4.05**	95.37 ± 1.68	93.18 ± 4.58	91.78 ± 5.46	84.88 ± 10.11	**98.47 ± 1.45**
Iris	**95.33 ± 4.27**	**95.33 ± 4.27**	**93.00 ± 6.40**	94.87 ± 0.27	**95.33 ± 4.27**	**95.33 ± 4.27**	**93.00 ± 6.40**	**99.60 ± 0.61**
Mammographic	**83.46 ± 4.18**	**83.48 ± 3.92**	**66.86 ± 8.16**	89.51 ± 3.74	83.35 ± 4.23	83.34 ± 4.12	66.60 ± 8.41	**90.29 ± 3.79**
Phoneme	91.34 ± 1.72	**89.29 ± 1.25**	**79.85 ± 2.90**	96.60 ± 0.77	**91.73 ± 1.28**	88.91 ± 1.61	79.60 ± 3.13	**96.83 ± 0.80**
Pima	**72.80 ± 3.29**	**67.67 ± 4.57**	**36.91 ± 8.59**	78.65 ± 4.96	71.88 ± 3.44	65.38 ± 4.56	33.03 ± 9.07	**79.67 ± 5.04**
Sonar	85.05 ± 10.38	84.99 ± 10.68	69.84 ± 21.17	88.87 ± 7.48	**86.38 ± 11.11**	**86.06 ± 11.60**	**72.29 ± 22.94**	**93.85 ± 7.52**
Wdbc	97.18 ± 1.98	96.58 ± 2.64	**96.58 ± 2.64**	96.42 ± 2.37	**97.89 ± 2.05**	**97.44 ± 2.51**	95.42 ± 4.48	**99.27 ± 1.15**
Wine	**97.78 ± 3.69**	**97.88 ± 3.80**	**96.62 ± 5.63**	95.58 ± 0.53	96.60 ± 4.56	96.65 ± 4.80	94.81 ± 6.99	**99.66 ± 0.58**
Wisconsin	**97.28 ± 2.43**	**97.64 ± 2.41**	**94.10 ± 5.26**	97.96 ± 0.43	96.84 ± 1.78	97.31 ± 1.79	93.22 ± 3.70	**99.26 ± 0.28**
Yeast	59.84 ± 2.47	**54.10 ± 4.57**	47.57 ± 3.14	71.00 ± 12.16	**60.11 ± 2.55**	53.88 ± 4.31	**48.02 ± 3.23**	**72.54 ± 13.53**
Ave.	**86.81 ± 11.22**	**84.40 ± 13.50**	**71.29 ± 24.30**	88.61 ± 10.42	86.20 ± 10.91	83.47 ± 13.22	69.92 ± 23.69	**90.01 ± 11.33**

Table 7.3: Reported results for the Wilcoxon test when comparing EN-MOMS-PbE and SEN-MOMS-PbE

Score	R^+	R^-	p-value
Accuracy	220	105	0.119
Avg. Acc.	226	99	0.085
Kappa	211	89	0.079
AUC	51	274	0.003

also reflected in the performance of the selected full classification models when they are evaluated under different scores.

One should note that the inclusion of the surrogate model adds a computational cost to the optimization step. This is because this model must be trained to get an approximation of the landscape of the optimization criteria. Here, RBFs are used as surrogate models. The learning stage of an RBF involves the matrix inversion, which in the worst case requires $\mathcal{O}\left(n^3\right)$. For this reason, one should be careful on the data used to train the model. Based on this, we take into consideration the approximation of the surrogate to that given by the expensive fitness functions to determine which model should be updated. Notwithstanding the computational cost of training the surrogates, one should also note that the surrogates are used as an inexpensive evaluation of a number of solutions. This implies that while the algorithm spends time in training a surrogate, it also saves time in the evaluation[1] of a number of full classification models. Thus, a reduction in the number of evaluations can be translated to a reduction in the time required to perform this task.

7.4 Final Remarks

In this chapter we presented the hybridization of EN-MOMS-PbE with a surrogate-assisted optimization approach that aims at reducing the number of required fitness evaluations to get an approximation of the Pareto front. This hybrid approach is called SEN-MOMS-PbE, which stands for Surrogate Evolutionary Nested Multi-Objective full Model Selection with Pareto-based Ensembles. Our proposal showed the following:

- The adaptation of the surrogates as an inexpensive function evaluation showed to significantly reduce the number of fitness evaluations to get almost the same quality of the approximation to the Pareto front.

[1] One should recall that the evaluation of the fitness functions requires to train and to test the full model a number of times.

- The performance of both approaches, i.e., the non-surrogate and the surrogate ones, is quite similar under different scores widely used in pattern recognition.

The experimental evaluation shows that the reduction in the number of fitness function evaluations attained is around 80% and the statistical assessment revealed that the difference in the classification performance for both is not significant for 3 out of 4 of the scores, it only shows an statistically significant difference in the AUC score. Hence, we can conclude that SEN-MOMS-PbE is able to construct highly effective classification models without requiring a high computational effort when doing so.

Part IV

General Conclusion

Chapter 8

Conclusions

We can only see a short distance ahead, but we can see plenty there that needs to be done.

Alan Turing

This thesis has dealt with the problem of full model selection for classification tasks. The main goal of it is to advance the state of the art in this topic. For doing so, three main approaches have been proposed, which are listed in the following:

- In chapter 5, MOMTS: Multi-Objective Model Type Selection is presented, which makes use of the VC Dimension theory to estimate in a general fashion the model complexity for different model types. This enables the formulation of the model type selection problem as a multi-objective one. This has represented the first step in order to achieve our general goal and enables the exploration of the full model selection in a more straightforward fashion. We found that the experimental estimation of the VC dimension is an alternative tool for computing the model complexity in a general fashion. This allow us considering different model types in a single representation. Moreover, it also allows penalizing models and, in this manner, controlling the overfitting of the generated models.

- In chapter 6, EN-MOMS-PbE: Evolutionary Nested Multi-Objective full Model Selection with Pareto-based Ensembles is introduced. EN-MOMS-PbE formulates the full model selection problem as a nested multi-objective optimization problem. The motivation of this formulation relies on the fact that the hyper-parameters are only meaningful for the algorithm they belong to. Moreover, EN-MOMS-PbE has extended in a wider way the concept of pre-processing methods in the full model selection method by considering not only data transformation and feature selection, but also data sampling and noise filtering. This has yield a relatively high number of degrees of freedom in the full model selection task.

EN-MOMS-PbE also explores different ways to construct a final classification model based on ensembles of solutions from the Pareto front. We empirically found that an ensemble based on evolutionary optimization (PEE: Pareto Evolutionary Ensemble) performed better than the other proposals. The Global Pareto Ensemble (GPE) was the second best in terms of performance. Nonetheless, GPE showed the disadvantage of requiring a larger number of full models than PEE. This is because GPE uses all full models in the non-dominated front, whereas PEE a subset of them.

We also noted that Margin Distance Ensemble (MDE) and Pareto Complementary Ensemble (PCE) were the ones with the worst performance among the Pareto-based Ensembles. MDE gives equal weight to all instances, which violates the idea of the margin in where some instances are closer to the boundary decision region than others. On the other hand, PCE tries to find the models that best complements the errors incurred by the subensemble, but without considering the joint effect in the previous instances correctly classified.

In the experimental evaluation, the performance of EN-MOMS-PbE was assessed on four different measures. It showed good performance under them, without overfitting the constructed full model to a given criterion. For instance, its performance in the AUC score showed to be highly effective when comparing with EPSMS, even when the later takes this as the optimization criterion and EN-MOMS-PbE does not.

- In chapter 7, it is proposed an extension to EN-MOMS-PbE by exploring the idea of using surrogate-assisted optimization and is called SEN-MOMS-PbE, Surrogate EN-MOMS-PbE. In the experimental evaluation, SEN-MOMS-PbE is able to get similar performance to EN-MOMS-PbE but with a lower number of fitness evaluations. A reduction rate around 80% in the number of fitness functions evaluation is reached when the surrogates are hybridize with the proposal.

These three proposals describe the contributions of this thesis. We have tested them using several benchmark datasets widely used in several pattern recognition and model selection studies and comparing with model selection methods reported in the state of the art. The results have been contrasted with statistical tests. All these give evidence of the success of the proposed methods and that the goals of this research have been reached.

However, this is not all, we are aware that there is a large path to explore. As part of this future work, we consider the following:

- Take advantage of the previously selected models to the landscape learning during the optimization. When performing a model selection, during the search a great amount of information is generated, which can be used to gain knowledge about the fitness landscape and this task is performing several times. Using the knowledge of the previous models can be used to construct a more general surrogate function to a wide range of datasets.
- Exploring techniques such as transfer learning for dealing with semi-supervised problems. One disadvantage is that the proposed approach makes use of sampling techniques to experimentally estimate the optimization criteria. This could be a shortcoming when dealing with problems with a scarce amount of data, such as semi-supervised learning problems, where only a small fraction of the data is labelled.
- Including ideas from meta-learning for the surrogate selection/adaptation during the search. Recently, meta-learning have been explored to perform model selection in surrogate-assisted optimization in an inexpensive way, as well as allowing adapting the surrogate during the optimization.
- Testing the proposal on real world problems that require a classification model. During this research, we used standard classification datasets available in common repositories; however, assessing its performance on real world problems is a valuable task to perform.

References

Aha, D. W., Kibler, D., Albert, M. K., 1991. Instance-based learning algorithms. Machine Learning 6 (1), 37–66.

Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F., 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.

Amthauer, H. A., 2008. Applying machine learning methods to suggest network involvement and functionality of genes in saccharomyces cerevisiae. Ph.D. thesis, University of Kansas.

Angiulli, F., 2007. Fast nearest neighbor condensation for large data sets classification. Knowledge and Data Engineering, IEEE Transactions on 19 (11), 1450–1464.

Antonelli, M., Ducange, P., Marcelloni, F., 2014. A fast and efficient multi-objective evolutionary learning scheme for fuzzy rule-based classifiers. Information Sciences 283, 36 – 54.

Arias-Montano, A., Coello Coello, C. A., Mezura Montes, E., June 2012. Multi-objective airfoil shape optimization using a multiple-surrogate approach. In: Evolutionary Computation (CEC), 2012 IEEE Congress on. pp. 1–8.

Ayat, N., Cheriet, M., Suen, C., 2005. Automatic model selection for the optimization of SVM kernels. Pattern Recognition 38 (10), 1733 – 1745.

Aydin, I., Karakose, M., Akin, E., 2011. A multi-objective artificial immune algorithm for parameter optimization in support vector machine. Applied Soft Computing 11 (1), 120 – 129.

Bao, Y., Hu, Z., Xiong, T., 2013. A PSO and pattern search based memetic algorithm for SVMs parameters optimization. Neurocomputing 117 (0), 98 – 106.

Bardenet, R., Brendel, M., Kégl, B., Sebag, M., 2012. Scot: surrogate-based collaborative tuning for hyperparameter learning that remembers the past. In: NIPS workshop on Bayesian Optimization and Decision Making.

Bardenet, R., Brendel, M., Kégl, B., Sebag, M., 2013. Collaborative hyperparameter tuning. In: Dasgupta, S., Mcallester, D. (Eds.), Proceedings of the 30th International Conference on Machine Learning (ICML-13). JMLR Workshop and Conference Proceedings, pp. 199–207.

Bengio, Y., 2000. Continuous optimization of hyper-parameters. In: Proceedings of the International Joint Conference on Neural Networks. Vol. 1. pp. 305–310.

Bergstra, J. S., Bardenet, R., Bengio, Y., Kégl, B., 2011. Algorithms for hyper-parameter optimization. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (Eds.), Advances in Neural Information Processing Systems 24. Curran Associates, Inc., pp. 2546–2554.

Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J., 1984. Classification and Regression Trees. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A.

Brodley, C. E., Friedl, M. A., 1999. Identifying mislabeled training data. Journal of Artificial Intelligence Research, 131–167.

Cawley, G. C., Talbot, N. L., 2010. On over-fitting in model selection and subsequent selection bias in performance evaluation. Journal of Machine Learning Research 11, 2079–2107.

Cawley, G. C., Talbot, N. L. C., 2007. Preventing over-fitting during model selection via bayesian regularisation of the hyper-parameters. Journal of Machine Learning Research 8, 841–861.

Chang, C.-C., Lin, C.-J., 2011. Libsvm: A library for support vector machines. ACM Trans. Intell. Syst. Technol. 2 (3), 27:1–27:27.

Chatelain, C., Adam, S., Lecourtier, Y., Heutte, L., Paquet, T., 2010. A multi-model selection framework for unknown and/or evolutive misclassification cost problems. Pattern Recognition 43 (3), 815 – 823.

Cherkassky, V., Mülier, F. M., 2007. Learning from data: concepts, theory, and methods. Wiley.

Cleary, J. G., Trigg, L. E., 1995. K*: An instance-based learner using an entropic distance measure. In: In Proceedings of the 12th International Conference on Machine Learning. Morgan Kaufmann, pp. 108–114.

Coates, A., Ng, A. Y., Lee, H., 2011. An analysis of single-layer networks in unsupervised feature learning. In: International conference on artificial intelligence and statistics. pp. 215–223.

Coello Coello, C. A., Lamont, G. B., Veldhuizen, D. A. V., 2007. Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd Edition. Genetic and Evolutionary Computation. Springer US.

Couckuyt, I., Declercq, F., Dhaene, T., Rogier, H., Knockaert, L., 2010. Surrogate-based infill optimization applied to electromagnetic problems. International Journal of RF and Microwave Computer-Aided Engineering 20 (5), 492–501.

Cover, T., Hart, P., 1967. Nearest neighbor pattern classification. IEEE Transactions on Information Theory 13 (1), 21–27.

De Jong, K. A., 2006. Evolutionary Computation: A Unified Approach. MIT Press. URL https://books.google.com.mx/books?id=OIRQAAAAMAAJ

Deb, K., 2001. Multi-Objective Optimization Using Evolutionary Algorithms. Wiley.

Deb, K., 2012. Optimization for engineering design: Algorithms and examples. PHI Learning Pvt. Ltd.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6 (2), 182–197.

Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30.

Derrac, J., García, S., Molina, D., Herrera, F., 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), 3 – 18.

Dietterich, T., 2000. Ensemble methods in machine learning. In: Multiple Classifier Systems. Vol. 1857 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1–15.

Duda, R. O., Hart, P. E., Stork, D. G., 2001. Pattern classification, 2nd Edition. John Wiley & Sons.

Edgeworth, F. Y., 1881. Mathematical psychics: An essay on the application of mathematics to the moral sciences. C. Keagann Paul.

Eggensperger, K., Hutter, F., Hoos, H., Leyton-Brown, K., 2015. Efficient benchmarking of hyperparameter optimizers via surrogates. In: Twenty-Ninth AAAI Conference on Artificial Intelligence. pp. 1114–1120.

Eiben, A. E., Smith, J. E., 2003. Introduction to Evolutionary Computing. SpringerVerlag.

Escalante, H., Montes, M., Sucar, E., July 2010. Ensemble particle swarm model selection. In: The 2010 International Joint Conference on Neural Networks (IJCNN). pp. 1–8.

Escalante, H. J., Montes, M., Sucar, L. E., 2009. Particle swarm model selection. Journal of Machine Learning Research 10, 405–440.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., Lin, C.-J., Jun. 2008. Liblinear: A library for large linear classification. Journal of Machine Learning Research 9, 1871–1874.

Fayyad, U. M., Irani, K. B., 1993. Multi-interval discretization of continuous-valued attributes for classification learning. In: IJCAI. pp. 1022–1029.

Fogel, D. B., 1992. Evolving artificial intelligence. Ph.D. thesis, University of California at San Diego, La Jolla, CA, USA, uMI Order No. GAX93-03240.

Fogel, L., Owens, A., Walsh, M., 1966. Artificial intelligence through simulated evolution. Wiley & Sons.

Fonseca, C. M., Fleming, P. J., 1993. Genetic algorithms for multiobjective optimization: Formulation discussion and generalization. In: Proceedings of the 5th ICGA. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 416–423.

Frenay, B., Verleysen, M., May 2014. Classification in the presence of label noise: A survey. IEEE Transactions on Neural Networks and Learning Systems 25 (5), 845–869.

Freund, Y., Schapire, R. E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 55 (1), 119–139.

Friedrichs, F., Igel, C., 2005. Evolutionary tuning of multiple SVM parameters. Neurocomputing 64 (0), 107 – 117.

García, S., Derrac, J., Cano, J., Herrera, F., March 2012. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (3), 417–435.

García, S., Fernández, A., Luengo, J., Herrera, F., 2009. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Computing 13 (10), 959–977.

García, S., Luengo, J., Herrera, F., 2014. Data Preprocessing in Data Mining. Springer Publishing Company, Incorporated.

García, S., Luengo, J., Sáez, J., López, V., Herrera, F., April 2013. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. IEEE Transactions on Knowledge and Data Engineering 25 (4), 734–750.

Goldberg, D. E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning, 1st Edition. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Gorissen, D., Couckuyt, I., Laermans, E., Dhaene, T., 2010. Multiobjective global surrogate modeling, dealing with the 5-percent problem. Engineering with Computers 26 (1), 81–98.

Gorissen, D., Dhaene, T., Turck, F. D., 2009. Evolutionary model type selection for global surrogate modeling. Journal of Machine Learning Research 10, 2039–2078.

Guyon, I., 2009. A practical guide to model selection. In: Proceedings of the machine learning summer school. Springer Series in Statistics. Springer.

Guyon, I., Saffari, A., Dror, G., Cawley, G., 2008. Analysis of the IJCNN 2007 agnostic learning vs. prior knowledge challenge. Neural Networks 21 (2–3), 544 – 550, advances in Neural Networks Research: IJCNN 07 2007 International Joint Conference on Neural Networks IJCNN 07.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H., 2009. The weka data mining software: an update. SIGKDD Explor. Newsl. 11 (1), 10–18.

Hall, M. A., 1998. Correlation-based feature subset selection for machine learning. Ph.D. thesis, University of Waikato, Hamilton, New Zealand.

Hand, D., Till, R., 2001. A simple generalisation of the area under the roc curve for multiple class classification problems. Machine Learning 45 (2), 171–186.

Handl, J., Knowles, J., 2007. An evolutionary approach to multiobjective clustering. IEEE Transactions on Evolutionary Computation 11 (1), 56–76.

Hastie, T., Tibshirani, R., Friedman, J., 2009. The elements of statistical learning: data mining, inference and prediction, 2nd Edition. Springer.

Holland, J. H., 1975. Adaptation in Natural and Artificial Systems. The University of Michigan Press.

Horn, J., Nafpliotis, N., Goldberg, D., 1994. A niched pareto genetic algorithm for multiobjective optimization. In: IEEE WCCI. P. pp. 82 –87 vol.1.

Hwang, C., Masud, A., 1979. Multiple objective decision making, methods and applications: a state-of-the-art survey. Lecture notes in economics and mathematical systems. Springer-Verlag.

Jin, Y., Sendhoff, B., 2008. Pareto-based multiobjective machine learning: An overview and case studies. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 38 (3), 397–415.

Kerber, R., 1992. Chimerge: Discretization of numeric attributes. In: Proceedings of the Tenth National Conference on Artificial Intelligence. AAAI'92. AAAI Press, pp. 123–128.

Khoshgoftaar, T., Rebours, P., 2007. Improving software quality prediction by noise filtering techniques. Journal of Computer Science and Technology 22 (3), 387–396.

Kira, K., Rendell, L. A., 1992. The feature selection problem: Traditional methods and a new algorithm. In: Proceedings of the Tenth National Conference on Artificial Intelligence. AAAI'92. AAAI Press, pp. 129–134.

Knowles, J., Corne, D., 2000. Approximating the nondominated front using the pareto archived evolution strategy. Evolutionary Computation 8 (2), 149–172.

Koza, J. R., 1992. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA.

Kuncheva, L. I., Whitaker, C. J., 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning 51 (2), 181–207.

Kurgan, L., Cios, K., Feb 2004. Caim discretization algorithm. Knowledge and Data Engineering, IEEE Transactions on 16 (2), 145–153.

Li, W., Liu, L., Gong, W., 2011. Multi-objective uniform design as a SVM model selection tool for face recognition. Expert Systems with Applications 38 (6), 6689 – 6695.

Lin, S.-W., Ying, K.-C., Chen, S.-C., Lee, Z.-J., 2008. Particle swarm optimization for parameter determination and feature selection of support vector machines. Expert Systems with Applications 35 (4), 1817 – 1824.

Liu, H., Setiono, R., 1995. Chi2: Feature selection and discretization of numeric attributes. In: Proceedings of the Seventh International Conference on Tools with Artificial Intelligence. TAI '95. IEEE Computer Society, Washington, DC, USA, pp. 388–391.

Lorena, A. C., de Carvalho, A. C., 2008. Evolutionary tuning of SVM parameter values in multiclass problems. Neurocomputing 71 (16–18), 3326 – 3334.

Martín, D., Rosete, A., Alcalá-Fdez, J., Herrera, F., 2014a. A new multiobjective evolutionary algorithm for mining a reduced set of interesting positive and negative quantitative association rules. IEEE Transactions on Evolutionary Computation 18 (1), 54–69.

Martín, D., Rosete, A., Alcalá-Fdez, J., Herrera, F., 2014b. QAR-CIP-NSGA-II: A new multi-objective evolutionary algorithm to mine quantitative association rules. Information Sciences 258, 1 – 28.

Martínez-Muñoz, G., Suárez, A., 2004. Aggregation ordering in bagging. In: Proc. of the IASTED International Conference on Artificial Intelligence and Applications. pp. 258–263.

McCulloch, W., Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics 5 (4), 115–133.

Mckay, M. D., Beckman, R. J., Conover, W. J., 2000. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 42 (1), 55–61.

Miettinen, K., 1999. Nonlinear Multiobjective Optimization. Vol. 12 of International Series in Operations Research and Management Science. Kluwer Academic Publishers, Dordrecht.

Minaei-Bidgoli, B., Barmaki, R., Nasiri, M., 2013. Mining numerical association rules via multi-objective genetic algorithms. Information Sciences 233, 15 – 24.

Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S., 2015. A survey of multiobjective evolutionary clustering. ACM Computing Surveys 47 (4), 61:1–61:46.

Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S., Coello Coello, C. A., 2014a. A survey of multiobjective evolutionary algorithms for data mining: Part I. IEEE Transactions on Evolutionary Computation 18 (1), 4–19.

Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S., Coello Coello, C. A., 2014b. Survey of multiobjective evolutionary algorithms for data mining: Part II. IEEE Transactions on Evolutionary Computation 18 (1), 20–35.

Pareto, V., 1964. *Cours d'economie politique*. Librairie Droz.

Pilát, M., Neruda, R., 2013. Multi-objectivization and surrogate modelling for neural network hyper-parameters tuning. In: Huang, D.-S., Gupta, P., Wang, L., Gromiha, M. (Eds.), Emerging Intelligent Computing Technology and Applications. Vol. 375 of Communications in Computer and Information Science. Springer Berlin Heidelberg, pp. 61–66.
URL http://dx.doi.org/10.1007/978-3-642-39678-6_11

Pilat, M., Neruda, R., 2013. Multiobjectivization for classifier parameter tuning. In: Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion. GECCO '13 Companion. ACM, New York, NY, USA, pp. 97–98.

Pinto, N., Doukhan, D., DiCarlo, J. J., Cox, D. D., 2009. A high-throughput screening approach to discovering good forms of biologically inspired visual representation. PLoS Comput Biol 5 (11), e1000579.

Price, K., Storn, R. M., Lampinen, J. A., 2006. Differential evolution: A practical approach to global optimization. Springer.

Price, K. V., Storn, R. M., Lampinen, J. A., 2005. Differential Evolution: A Practical Approach to Global Optimization. Natural Computing Series. Springer Berlin Heidelberg.

Quinlan, J. R., 1986. Induction of decision trees. Mach. Learn. 1 (1), 81–106.

Quinlan, J. R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Rao, S., 1984. Optimization: Theory and Applications. Wiley, New York.

Rätsch, G., Onoda, T., Müller, K.-R., 2001. Soft margins for adaboost. Machine Learning 42 (3), 287–320.

Ravindran, A., Reklaitis, G. V., Ragsdell, K. M., 2006. Engineering optimization: Methods and applications. John Wiley & Sons.

Rechenberg, I., 1965. Cybernetic solution path of an experimental problem. Tech. rep., Ministry of Aviation, Royal Aircraft Establishment.

Rechenberg, I., 1973. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Ieice Transactions.

Rosales-Pérez, A., Gonzalez, J. A., Coello Coello, C. A., Escalante, H. J., Reyes-Garcia, C. A., 2014. Multi-objective model type selection. Neurocomputing 146, 83 – 94.

Rosales-Pérez, A., Gonzalez, J. A., Coello Coello, C. A., Escalante, H. J., Reyes-Garcia, C. A., 2015. Surrogate-assisted multi-objective model selection for support vector machines. Neurocomputing 150, Part A, 163 – 172.

Rosenblatt, F., Nov. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review 65 (6), 386–408.

Rosenblatt, F., 1962. Principles of neurodynamics: perceptrons and the theory of brain mechanisms. Report (Cornell Aeronautical Laboratory). Spartan Books.

Salvador, S., Chan, P., Nov 2004. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In: 16th IEEE International Conference on Tools with Artificial Intelligence, 2004. pp. 576–584.

Sánchez, J., Pla, F., Ferri, F., 1997. Prototype selection for the nearest neighbour rule through proximity graphs. Pattern Recognition Letters 18 (6), 507 – 513.

Sánchez-Monedero, J., Gutiérrez, P. A., Fernández-Navarro, F., Hervás-Martínez, C., 2011. Weighting efficient accuracy and minimum sensitivity for evolving multi-class classifiers. Neural processing letters 34 (2), 101–116.

Schaffer, J. D., 1985. Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the 1st International Conference on Genetic Algorithms. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, pp. 93–100.

Schütze, O., Esquivel, X., Lara, A., Coello Coello, C. A., Aug 2012. Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization. IEEE Transactions on Evolutionary Computation 16 (4), 504–522.

Schwefel, H.-P., 1977. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Vol. 26. Birkhäuser, Basel/Stuttgart.

Shahrokhi, A., Jahangirian, A., 2010. A surrogate assisted evolutionary optimization method with application to the transonic airfoil design. Engineering Optimization 42 (6), 497–515.

Srinivas, N., Deb, K., 1994. Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation 2 (3), 221–248.

Storn, R., Price, K., 1997. Differential Evolution: A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization 11 (4), 341–359.

Sun, Q., Pfahringer, B., Mayo, M., 2012. Full model selection in the space of data mining operators. In: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion. GECCO Companion '12. ACM, New York, NY, USA, pp. 1503–1504.

Suttorp, T., Igel, C., 2006. Multi-objective optimization of support vector machines. In: Jin, Y. (Ed.), Multi-Objective Machine Learning. Vol. 16 of Studies in Computational Intelligence. Springer Berlin Heidelberg, pp. 199–220.

Thornton, C., Hutter, F., Hoos, H. H., Leyton-Brown, K., 2013. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In: Proc. of KDD-2013. pp. 847–855.

Vapnik, V., Levin, E., Le Cun, Y., 1994. Measuring the VC-dimension of a learning machine. Neural Computation 6 (5), 851–876.

Vapnik, V. N., 1995. The nature of statistical learning theory. Springer-Verlag New York, Inc., New York, NY, USA.

Wang, C.-M., Huang, Y.-F., 2009. Evolutionary-based feature selection approaches with new criteria for data mining: A case study of credit approval data. Expert Systems with Applications 36 (3, Part 2), 5900 – 5908.

Wang, Z., Li, M., Li, J., 2015. A multi-objective evolutionary algorithm for feature selection based on mutual information with a new redundancy measure. Information Sciences 307, 73 – 88.

Werbos, P., 1974. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. thesis, Harvard University, Cambridge, MA.

Wilson, D. L., 1972. Asymptotic properties of nearest neighbor rules using edited data. IEEE Transactions on Systems, Man and Cybernetics SMC-2 (3), 408–421.

Wilson, D. R., Martinez, T. R., 2000. Reduction techniques for instance-basedlearning algorithms. Machine Learning 38 (3), 257–286.

Wolpert, D. H., 1996. The lack of a priori distinctions between learning algorithms. Neural Computation 8 (7), 1341–1390.

Wong, A. K., Chiu, D., Nov 1987. Synthesizing statistical knowledge from incomplete mixed-mode data. Pattern Analysis and Machine Intelligence, IEEE Transactions on PAMI-9 (6), 796–805.

Yogatama, D., Mann, G., 2014. Efficient transfer learning method for automatic hyperparameter tuning. In: International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, pp. 1077–1085.

Yu, X., Gen, M., 2010. Introduction to Evolutionary Algorithms. Decision Engineering. Springer London.

Zhang, Q., Li, H., 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation 11 (6), 712–731.

Zhu, X., Wu, X., 2004. Class noise vs. attribute noise: A quantitative study of their impacts. Artificial Intelligence Review 22 (3), 177–210.

Zitzler, E., Deb, K., Thiele, L., Jun. 2000. Comparison of multiobjective evolutionary algorithms: Empirical results. Evol. Comput. 8 (2), 173–195.

Zitzler, E., Laumanns, M., Thiele, L., 2001. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Giannakoglou, K. C., Tsahalis, D. T., Périaux, J., Papailiou, K. D., Fogarty, T. (Eds.), Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems. International Center for Numerical Methods in Engineering, pp. 95–100.

Zitzler, E., Thiele, L., 1999. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation 3 (4), 257 –271.

Appendices

Appendix A

Datasets Description

This appendix presents a description of the classification datasets used in the experimental evaluation through a set of data complexity measures. First, this set of measures is described in detail. After that, they are used to describe the classification problems.

A.1 Data Complexity Measures

Ho and Basu (2002) presents a set of 12 data complexity measures is presented, which can be categorized into three main groups:

- **measures of overlap in the features values from different classes**: these measures focus on the capacity of individual features to separate example from different classes;

- **measures of class separability**: these measures estimate to what extent the classes are separable by examining the length and the linearity of the class boundary; and

- **measures of geometry, topology, and density of manifolds**: these measures provide an indirect characterization of the class separability. They assume that the problem is composed of several manifolds spanned by each class. The shape, position, and interconnectedness of these manifolds give some hints on how well the classes are separated and on the density or population of each manifold.

A.1.1 Measures of Overlap in the Features Values from Different Classes

This group encompass the following measures:

Maximum Fisher's discriminant ratio (F1)

This measure computes the maximum discriminative power of each attribute. The discriminative power of each attribute is computed as follows:

$$f_j = \frac{\left(\mu_1^j - \mu_2^j\right)^2}{\left(\sigma_1^j\right)^2 + \left(\sigma_2^j\right)^2} \tag{A.1}$$

where μ_k^j and $\left(\sigma_k^j\right)^2$ are the mean and variance of the k^{th} class in the j^{th} attribute. Nominal attributes are usually converted to a integer values.

For problems with more than 2-classes, the discriminative power of each attribute is computed as:

$$f_j = \frac{\sum_{k=1}^{C-1} \sum_{l=k+1}^{C} p_k p_l \left(\mu_k^j - \mu_l^j\right)^2}{\sum_{k=1}^{C} pk \left(\sigma_k^j\right)^2} \tag{A.2}$$

where C is the maximum number of classes, p_k is portions of instances from the k^{th} class.

F1 ranges in the interval $[0, \mu_k]$. High values of the Fisher's discriminant ratio indicate that at least one of the attributes enables the learner to separate the instances of different classes with partitions that are parallel to an axis of the feature space. Low values of this measure do not imply that the classes are not linearly separable, but that they cannot be discriminated by hyperplanes parallel to one of the axis of the feature space (Macià Antolínez, 2011).

The overlap of the per-class bounding boxes (F2)

F2 measures the amount of overlap of the bounding boxes of two classes. It is the product of per-feature overlap ratios, each of which is the width of the overlap interval normalized by the width of the entire interval encompassing the two classes (Ho and Basu, 2002). F2 is computed as:

$$F2 = \prod_{j=1}^{d} \frac{Min\_Max_j - Max\_Min_j}{Max\_Max_j - Min\_Min_j} \tag{A.3}$$

where d is the number of features, Min_Max_j is the minimum of the maximum values of the j^{th} feature, Max_Min_j is the maximum of the minimum values of the j^{th} feature, Max_Max_j and Min_Min_j are the maximum of the maximums and the minimum of the minimums values of the j^{th} feature.

F2 ranges in the interval $[0, 1]$. Low values of this measure mean that the attributes can discriminate the instances of different classes.

The maximum (individual) feature efficiency (F3)

This measure computes the discriminative power of individual features . It only considers separating hyperplanes perpendicular to the feature axes.

F3 is computed using the following heuristic. For each attribute, the overlapping region is considered and the ratio of the number of instances that are not in this overlapping region to the total number of instances is returned. Then, the maximum discriminative ratio is taken as measure F3.

F3 ranges in the interval $[0, 1]$. High values of this measure indicate that there is an attribute which is able to discriminate between instances of different classes (Macià Antolínez, 2011).

A.1.2 Measures of Class Separability

The measures in this group are described below.

The Minimized Sum of Error Distance by a Linear Classifier (L1)

This measure determines the linear separability. L1 returns the sum of the differences between the prediction of a linear classifier and the current class value. L1 ranges from 0 to 1. A zero value in L1 indicates that the problem is linearly separable.

The Training error of a Linear Classifier (L2)

This measure provides information to what extent the training data is linearly separable. L2 constructs a linear classifier and returns the percentage of samples incorrectly classified.

L2 ranges from 0 to 1. A zero value indicates that the training data is linearly separable.

The Fraction of Points on the Class Boundary (N1)

This method constructs a Minimum Spanning Tree (MST) over the entire dataset by connecting all samples using the Euclidean distance. It counts the number of samples that are connected with samples of different classes. The fraction of such samples over all in the dataset is used as a measure. If a sample is connected to more than one node of different class, it is counted only once.

N1 can take values from 0 to 1. High values of N1 indicates that the majority of the samples are located close to the class boundary.

The Ratio of Average Intra/Inter Class Nearest Neighbor Distance (N2)

This measure compares the intra-class spread and the inter-class spread. N2 computes the distance of each instance in the dataset with its nearest neighbor from the same class (intra distance) and its nearest neighbor from different class (inter distance) and returns the ratio of the average intra distances to the average of the inter distances.

N2 ranges in the interval $[0, \infty]$. Low values suggest that instances from the same class lie closely in the feature space.

The Leave-One-Out Error Rate of the Nearest Neighbor Classifier (N3)

This measure returns the leave-one-out error rate of the one-nearest neighbor classifier. N3 ranges in the interval $[0, 1]$. High values of this measure indicate that most of the samples are in the class boundary.

A.1.3 Measures of Geometry, Topology, and Density of Manifolds

In this group, we can find four measures that are described in the following.

The Non-Linearity of a Linear Classifier (L3)

Given a training dataset, L3 creates a test set by means of a linear interpolation with random coefficients between a pair of points randomly selected from the same class. This measure returns the error rate on the artificial test set when a linear classifier is trained using the original dataset.

L3 ranges from 0 to 1. High values of L3 can indicate that there is a high interleaving between classes.

The Non-Linearity of the Nearest Neighbor Classifier (N4)

This measure creates an artificial test set, similar as L3 and returns the test error when the one nearest neighbor is used with the original dataset as the training set.

N4 ranges in the interval $[0, 1]$, where a high value expresses a high interleaving between classes.

The Fraction of Maximum Covering Spheres (T1)

This measure uses the concept of *adherence subsets*, which, roughly speaking, is a sphere centered on an instance and grows as much as possible before touching an instance from another class. This measures only considers the biggest spheres and the redundant ones are removed. T1 returns the number of spheres normalized by the total number of points.

T1 ranges in the interval $[0, 1]$, where low values indicates that instances are grouped in compact clusters.

The Average Number of Points per Dimension (T2)

This measure returns the ratio of the number of instances to the number of attributes. T2 ranges from 0 to m, where m is the number of instances in the dataset.

A.2 Datasets Description

For the experimental evaluation, a total of 25 classification datasets have been used, which are available in IDA[1] (Rätsch et al., 2001) and KEEL[2] (Alcalá et al., 2011) repositories. Table A.1 shows the description of the datasets, including the intrinsic dimensionality (I.Dim.) and the data complexity measures.

[1] http://www.raetschlab.org/Members/raetsch/benchmark
[2] http://sci2s.ugr.es/keel/category.php?cat=clas

Table A.1: Description of the datasets

Dataset	I.Dim.	F1	F2	F3	L1	L2	L3	N1	N2	N3	N4	T1	T2
Appendicitis	3	0.838	0.045	0.286	0.422	0.200	0.500	0.295	0.642	0.200	0.252	0.971	15.000
Balance	4	0.208	3.000	0.000	0.134	0.077	0.712	0.311	0.677	0.213	0.331	0.915	156.000
Banana	2	0.010	0.625	0.005	0.897	0.448	0.500	0.192	0.152	0.127	0.346	0.781	2649.500
Breast Cancer	8	0.302	0.188	0.022	0.576	0.290	0.500	0.438	0.757	0.315	0.306	0.989	30.667
Diabetis	8	0.575	0.252	0.007	0.690	0.351	0.500	0.450	0.846	0.293	0.278	0.999	95.875
Flare Solar	7	0.292	0.000	0.125	0.727	0.325	0.323	0.485	0.392	0.461	0.450	1.000	118.333
German	18	0.353	0.662	0.014	0.701	0.282	0.482	0.434	0.862	0.299	0.246	1.000	49.950
Heart	12	0.754	0.196	0.015	0.530	0.152	0.126	0.335	0.782	0.234	0.113	0.996	20.692
Image	10	0.486	0.000	0.158	0.627	0.264	0.259	0.065	0.220	0.022	0.307	0.982	128.278
Ionosphere	24	0.614	0.000	0.191	0.456	0.117	0.151	0.234	0.636	0.134	0.180	0.949	10.606
Iris	2	16.369	0.054	0.570	0.024	0.000	0.007	0.094	0.199	0.040	0.007	0.886	37.250
Mammographic	1	0.857	0.088	0.014	0.582	0.206	0.178	0.379	0.490	0.261	0.229	0.955	192.000
Phoneme	5	0.270	0.271	0.122	0.667	0.257	0.336	0.190	0.258	0.088	0.270	0.987	1080.600
Pima	4	0.200	0.882	0.027	0.722	0.348	0.500	0.506	0.800	0.346	0.342	1.000	95.875
Ringnorm	19	0.054	0.000	0.060	0.719	0.224	0.190	0.441	0.853	0.245	0.281	1.000	369.950
Sonar	17	0.463	0.000	0.048	0.648	0.193	0.164	0.382	0.689	0.174	0.152	0.986	3.450
Splice	56	0.608	1.000	0.000	0.862	0.143	0.105	0.362	0.827	0.244	0.020	1.000	52.900
Thyroid	5	0.263	0.001	0.187	0.582	0.299	0.500	0.103	0.322	0.033	0.152	0.902	42.800
Titanic	3	0.489	1.000	0.000	0.448	0.224	0.303	0.249	0.028	0.245	0.292	0.989	733.333
Twonorm	19	0.436	0.004	0.010	0.831	0.023	0.006	0.084	0.782	0.054	0.018	1.000	369.950
Waveform	18	1.230	0.003	0.204	1.283	0.121	0.042	0.216	0.719	0.149	0.074	1.000	238.048
Wdbc	1	3.401	0.000	0.516	0.538	0.049	0.026	0.136	0.161	0.085	0.114	0.794	18.933
Wine	1	2.639	0.000	0.814	0.022	0.000	0.039	0.395	0.183	0.237	0.184	0.887	13.615
Wisconsin	7	3.475	0.248	0.119	0.461	0.036	0.014	0.064	0.334	0.043	0.041	0.807	77.556
Yeast	7	0.845	0.197	1.000	0.426	0.431	0.881	0.655	0.949	0.475	0.405	1.000	185.375

References

Alcalá, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F., 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.

Ho, T. K., Basu, M., Mar 2002. Complexity measures of supervised classification problems. IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (3), 289–300.

Macià Antolínez, N., 2011. Data complexity in supervised learning: A far-reaching implication. Ph.D. thesis, Universitat Ramon Llull.

Rätsch, G., Onoda, T., Müller, K.-R., 2001. Soft margins for adaboost. Machine Learning 42 (3), 287–320.

Appendix B

Experimental Results

This section reports the full list of results from the experiments performed in chapter 6. This appendix is divided in three sections, which report the following results:

- Experimental Results when Comparing Among Pareto-based Ensemble Approaches,
- Experimental Results when Comparing with a Single Criterion Approach, and
- Experimental Results when Comparing with Full Model Selection Methods.

Each section reports the results obtained by each of the considered scores:

- Accuracy,
- Average accuracy,
- Kappa statistic, and
- Area under Receiver Operating Curve (AUC).

The reported results are the averages over all replications of each dataset.

B.1 Experimental Results when Comparing Among Pareto-based Ensemble Approaches

The obtained results when using the Pareto ensemble approaches are shown from Table B.1 to Table B.4. Table B.1 shows the accuracy performance, Table B.2 the average accuracy, Table B.3 the results of the kappa statistic, and Table B.4 the resulting AUC score.

Table B.1: Results in classification accuracy for each Pareto ensemble method. The reported results are the average and the standard deviation over the replications and the best one is shown in **boldface**

Dataset	PER	PCE	MDE	Boosting	GPE	PKB	PEE
Banana	**88.58 ± 0.66**	71.26 ± 12.90	61.63 ± 9.58	85.77 ± 4.03	87.66 ± 1.16	82.24 ± 10.23	88.34 ± 0.73
Breast Cancer	70.13 ± 5.54	70.26 ± 3.38	67.40 ± 5.91	69.87 ± 4.57	**70.65 ± 4.00**	70.52 ± 3.92	70.26 ± 3.21
Diabetes	74.33 ± 2.18	71.83 ± 3.53	70.73 ± 4.52	74.90 ± 3.06	75.67 ± 1.88	75.97 ± 1.62	**76.47 ± 1.73**
Flare Solar	64.45 ± 4.02	61.38 ± 4.64	60.93 ± 4.96	65.45 ± 4.44	66.58 ± 2.21	**66.75 ± 2.21**	66.73 ± 2.49
German	**75.50 ± 2.51**	74.93 ± 2.14	73.50 ± 2.00	74.40 ± 2.96	74.50 ± 2.46	74.50 ± 2.23	75.17 ± 2.90
Heart	81.80 ± 4.07	78.00 ± 10.92	77.70 ± 10.58	81.90 ± 4.89	82.20 ± 4.42	81.00 ± 4.64	**82.40 ± 4.38**
Image	96.88 ± 2.26	96.12 ± 2.93	64.38 ± 10.50	96.04 ± 2.52	97.72 ± 0.79	90.94 ± 16.89	**97.92 ± 0.56**
Ringnorm	98.08 ± 0.35	97.95 ± 0.41	97.95 ± 0.47	98.11 ± 0.24	98.17 ± 0.23	98.16 ± 0.21	**98.29 ± 0.24**
Splice	**96.21 ± 0.76**	77.35 ± 8.41	75.08 ± 6.23	94.37 ± 1.60	93.94 ± 1.51	92.51 ± 1.70	95.95 ± 0.63
Thyroid	94.40 ± 3.60	**94.67 ± 3.38**	92.80 ± 4.96	92.93 ± 5.15	93.87 ± 3.57	93.47 ± 4.00	94.53 ± 3.17
Titanic	77.35 ± 0.55	72.10 ± 7.04	74.96 ± 4.15	76.78 ± 1.21	**78.02 ± 0.67**	76.86 ± 1.62	76.99 ± 1.57
Twonorm	96.97 ± 0.59	94.60 ± 7.18	96.29 ± 2.55	95.35 ± 4.52	97.20 ± 0.30	90.56 ± 13.06	**97.40 ± 0.19**
Waveform	90.19 ± 0.77	83.67 ± 6.90	78.06 ± 16.36	89.85 ± 0.83	**90.60 ± 0.38**	90.05 ± 0.82	90.40 ± 0.55
Appendicitis	85.00 ± 7.42	84.00 ± 7.35	80.36 ± 11.66	**85.91 ± 8.56**	85.00 ± 7.42	84.09 ± 7.20	85.00 ± 7.42
Balance	**97.92 ± 1.25**	93.22 ± 7.12	**97.92 ± 1.25**	94.86 ± 4.81	95.33 ± 5.14	95.18 ± 5.31	**97.92 ± 1.25**
Ionosphere	95.44 ± 2.29	86.32 ± 18.11	92.30 ± 5.73	**96.29 ± 2.87**	96.02 ± 3.15	94.88 ± 3.77	**96.29 ± 1.83**
Iris	95.33 ± 4.27	95.33 ± 4.27	**96.00 ± 4.42**	95.33 ± 4.27	95.33 ± 4.27	95.33 ± 4.27	95.33 ± 4.27
Mammographic	83.36 ± 4.24	71.51 ± 13.82	66.71 ± 14.97	80.96 ± 5.86	82.73 ± 3.90	82.52 ± 4.68	**83.46 ± 4.18**
Phoneme	91.27 ± 1.50	78.42 ± 4.20	82.79 ± 2.34	77.42 ± 3.07	83.88 ± 2.82	85.12 ± 5.56	**91.34 ± 1.72**
Pima	72.67 ± 3.47	70.47 ± 4.69	**72.91 ± 4.69**	72.54 ± 4.67	72.66 ± 4.57	71.24 ± 4.90	72.80 ± 3.29
Sonar	85.95 ± 9.75	73.12 ± 12.30	81.19 ± 9.15	**86.48 ± 10.59**	88.40 ± 9.49	86.02 ± 9.12	85.05 ± 10.38
Wdbc	96.83 ± 2.46	95.42 ± 3.90	96.12 ± 2.94	**97.71 ± 1.78**	97.18 ± 1.80	96.31 ± 2.42	97.18 ± 1.98
Wine	97.22 ± 3.73	97.78 ± 3.69	97.78 ± 3.69	**98.33 ± 3.56**	**98.33 ± 3.56**	**98.33 ± 3.56**	97.78 ± 3.69
Wisconsin	96.85 ± 2.46	97.00 ± 2.34	97.28 ± 2.67	97.14 ± 2.39	97.28 ± 2.43	**97.42 ± 2.54**	97.28 ± 2.43
Yeast	59.57 ± 2.65	57.82 ± 3.76	**59.98 ± 2.22**	59.10 ± 2.94	**59.98 ± 2.22**	59.24 ± 3.01	59.84 ± 2.47
Ave.	86.49 ± 11.36	81.78 ± 12.07	80.51 ± 13.08	85.51 ± 11.30	86.36 ± 11.03	85.17 ± 10.67	**86.81 ± 11.22**

Table B.2: Results for average accuracy for each Pareto ensemble method. The reported results are the average and the standard deviation over the replications and the best one is shown in **boldface**

Dataset	PER	PCE	MDE	Boosting	GPE	PKB	PEE
Banana	**88.16 ± 0.61**	68.51 ± 14.28	58.50 ± 8.69	84.64 ± 4.47	86.91 ± 1.34	80.69 ± 11.16	87.74 ± 0.83
Breast Cancer	60.40 ± 6.73	60.64 ± 5.62	**61.23 ± 6.73**	60.22 ± 6.13	60.83 ± 5.64	60.31 ± 7.55	61.03 ± 4.98
Diabetis	67.40 ± 2.92	65.09 ± 6.10	62.11 ± 7.56	66.83 ± 4.18	68.07 ± 2.51	69.21 ± 1.95	**69.75 ± 2.10**
Flare Solar	64.36 ± 5.20	61.70 ± 5.01	61.88 ± 6.36	64.24 ± 5.40	66.18 ± 2.62	**66.64 ± 2.55**	66.36 ± 2.96
German	67.44 ± 3.15	67.82 ± 2.59	68.95 ± 3.39	69.29 ± 3.05	**69.70 ± 2.64**	69.23 ± 3.06	69.53 ± 2.51
Heart	81.18 ± 3.72	77.38 ± 10.12	77.21 ± 9.84	81.42 ± 4.63	81.65 ± 4.00	80.05 ± 4.31	**81.86 ± 4.05**
Image	97.06 ± 1.88	95.87 ± 3.15	62.60 ± 10.83	95.67 ± 2.86	97.59 ± 0.81	91.63 ± 14.18	**97.81 ± 0.63**
Ringnorm	98.07 ± 0.34	97.94 ± 0.39	97.94 ± 0.45	98.11 ± 0.23	98.16 ± 0.22	98.15 ± 0.20	**98.29 ± 0.23**
Splice	**96.25 ± 0.73**	76.63 ± 8.13	74.02 ± 6.13	94.26 ± 1.64	93.82 ± 1.55	92.29 ± 1.72	95.96 ± 0.63
Thyroid	91.93 ± 5.32	**92.62 ± 4.95**	90.20 ± 6.16	89.42 ± 7.27	91.12 ± 4.39	90.44 ± 5.47	92.03 ± 5.00
Titanic	67.81 ± 1.69	60.77 ± 7.91	64.03 ± 7.19	67.99 ± 2.94	**68.48 ± 1.66**	67.71 ± 2.99	67.50 ± 3.28
Twonorm	96.97 ± 0.56	94.60 ± 6.80	96.28 ± 2.43	95.36 ± 4.26	97.20 ± 0.28	90.59 ± 12.35	**97.40 ± 0.18**
Waveform	89.11 ± 1.42	78.44 ± 12.17	74.51 ± 14.31	89.15 ± 1.50	90.10 ± 0.84	88.75 ± 1.56	**90.22 ± 0.65**
Appendicitis	80.00 ± 12.17	77.43 ± 11.55	72.29 ± 14.17	**80.56 ± 12.92**	78.96 ± 13.30	78.40 ± 12.75	76.11 ± 14.67
Balance	**96.13 ± 4.51**	85.00 ± 15.78	**96.13 ± 4.51**	88.24 ± 14.26	89.84 ± 13.29	89.17 ± 14.34	**96.13 ± 4.51**
Ionosphere	94.24 ± 3.17	85.57 ± 16.28	89.86 ± 6.94	**95.62 ± 3.21**	95.56 ± 3.25	94.88 ± 3.49	95.39 ± 2.39
Iris	95.33 ± 4.27	95.33 ± 4.27	**.00 ± 4.42**	95.33 ± 4.27	95.33 ± 4.27	95.33 ± 4.27	95.33 ± 4.27
Mammographic	83.22 ± 4.27	71.05 ± 14.41	65.82 ± 15.77	80.99 ± 5.45	82.69 ± 3.81	82.26 ± 4.74	**83.48 ± 3.92**
Phoneme	88.25 ± 1.66	71.05 ± 8.89	74.47 ± 4.43	63.98 ± 5.58	76.11 ± 5.03	76.28 ± 9.96	**89.29 ± 1.25**
Pima	66.31 ± 4.73	65.37 ± 4.69	**68.51 ± 5.88**	66.97 ± 5.97	66.88 ± 6.27	64.53 ± 6.22	67.67 ± 4.57
Sonar	85.97 ± 10.12	73.51 ± 11.82	80.48 ± 9.76	86.26 ± 11.01	**88.33 ± 9.62**	85.83 ± 9.43	84.99 ± 10.68
Wdbc	96.50 ± 2.65	94.89 ± 4.40	95.37 ± 3.64	**97.20 ± 2.22**	96.61 ± 2.20	95.81 ± 2.65	96.58 ± 2.64
Wine	97.22 ± 3.96	97.88 ± 3.80	97.81 ± 3.87	**98.36 ± 3.74**	**98.36 ± 3.74**	**98.36 ± 3.74**	97.88 ± 3.80
Wisconsin	97.03 ± 2.52	97.05 ± 2.43	97.64 ± 2.60	97.43 ± 2.38	97.64 ± 2.41	**97.65 ± 2.55**	97.64 ± 2.41
Yeast	53.37 ± 5.36	53.09 ± 7.01	**54.72 ± 4.82**	53.85 ± 4.71	**54.72 ± 4.82**	53.34 ± 5.06	54.10 ± 4.57
Ave.	83.99 ± 13.75	78.61 ± 13.81	77.54 ± 14.56	82.46 ± 13.81	83.63 ± 13.29	82.30 ± 12.92	**84.40 ± 13.50**

Table B.3: Results for kappa statistic for each Pareto ensemble method. The reported results are the average and the standard deviation over the replications and the best one is shown in **boldface**

Dataset	PER	PCE	MDE	Boosting	GPE	PKB	PEE
Banana	**76.77 ± 1.26**	37.77 ± 28.52	17.98 ± 17.79	70.55 ± 8.44	74.72 ± 2.37	62.47 ± 21.97	76.19 ± 1.48
Breast Cancer	22.31 ± 14.02	22.73 ± 11.38	22.60 ± 13.28	22.00 ± 12.65	23.44 ± 11.63	21.67 ± 15.51	**23.48 ± 10.30**
Diabetis	37.77 ± 5.50	31.80 ± 11.76	25.89 ± 15.39	37.39 ± 8.44	39.98 ± 4.84	41.81 ± 3.85	**42.99 ± 3.94**
Flare Solar	28.18 ± 10.06	22.71 ± 9.85	22.51 ± 12.29	28.62 ± 10.58	32.24 ± 4.88	**32.98 ± 4.78**	32.57 ± 5.54
German	37.24 ± 5.53	37.21 ± 4.12	37.21 ± 4.55	38.62 ± 4.89	39.20 ± 3.57	38.55 ± 4.37	**39.71 ± 4.32**
Heart	62.90 ± 7.61	55.29 ± 20.46	54.77 ± 19.84	63.20 ± 9.20	63.78 ± 8.17	61.02 ± 8.79	**64.17 ± 8.19**
Image	93.70 ± 4.29	92.02 ± 5.83	25.16 ± 21.63	91.84 ± 5.06	95.33 ± 1.53	83.07 ± 28.47	**95.75 ± 1.10**
Ringnorm	96.16 ± 0.67	95.90 ± 0.77	95.90 ± 0.90	96.22 ± 0.46	96.34 ± 0.44	96.31 ± 0.40	**96.59 ± 0.45**
Splice	**92.41 ± 1.45**	53.95 ± 16.19	48.99 ± 12.27	88.69 ± 3.08	87.83 ± 2.90	84.92 ± 3.28	91.89 ± 1.19
Thyroid	85.76 ± 9.29	**86.55 ± 8.77**	82.41 ± 11.11	82.08 ± 12.47	84.91 ± 7.95	83.72 ± 9.48	86.07 ± 8.38
Titanic	40.66 ± 2.41	24.40 ± 17.94	31.85 ± 15.84	40.17 ± 4.54	**42.30 ± 2.41**	39.90 ± 5.67	39.70 ± 6.07
Twonorm	93.95 ± 1.11	89.20 ± 13.62	92.57 ± 4.84	90.71 ± 8.56	94.40 ± 0.56	81.16 ± 24.72	**94.81 ± 0.36**
Waveform	77.88 ± 1.86	58.50 ± 23.14	50.80 ± 27.97	77.31 ± 1.68	**79.03 ± 0.80**	77.47 ± 1.92	78.75 ± 0.97
Appendicitis	55.69 ± 20.61	52.05 ± 20.25	42.50 ± 28.66	**58.25 ± 24.07**	54.24 ± 22.26	52.40 ± 21.26	50.09 ± 26.52
Balance	**96.33 ± 2.22**	87.59 ± 13.37	**96.33 ± 2.22**	90.69 ± 9.01	91.58 ± 9.49	91.27 ± 9.83	**96.33 ± 2.22**
Ionosphere	89.88 ± 5.12	72.17 ± 32.14	82.52 ± 13.00	**91.92 ± 6.15**	91.42 ± 6.71	89.11 ± 7.89	91.84 ± 4.05
Iris	**94.00 ± 6.63**	93.00 ± 6.40	93.00 ± 6.40	**94.00 ± 6.63**	93.00 ± 6.40	93.00 ± 6.40	93.00 ± 6.40
Mammographic	66.50 ± 8.53	41.99 ± 28.73	31.54 ± 31.38	61.91 ± 11.25	65.33 ± 7.74	64.73 ± 9.43	**66.86 ± 8.16**
Phoneme	78.43 ± 3.55	43.21 ± 17.27	53.84 ± 7.79	33.39 ± 11.55	57.02 ± 8.91	58.11 ± 17.95	**79.85 ± 2.90**
Pima	34.96 ± 9.25	32.11 ± 9.39	**38.09 ± 11.19**	35.70 ± 11.80	35.60 ± 12.37	31.19 ± 12.87	36.91 ± 8.59
Sonar	71.67 ± 19.93	46.74 ± 23.71	61.49 ± 19.28	72.57 ± 21.66	**76.66 ± 19.18**	71.72 ± 18.71	69.84 ± 21.17
Wdbc	96.50 ± 2.65	94.89 ± 4.40	95.37 ± 3.64	**97.20 ± 2.22**	96.61 ± 2.20	95.81 ± 2.65	96.58 ± 2.64
Wine	95.77 ± 5.70	96.62 ± 5.63	96.60 ± 5.64	**97.45 ± 5.45**	**97.45 ± 5.45**	**97.45 ± 5.45**	96.62 ± 5.63
Wisconsin	93.15 ± 5.32	93.43 ± 5.09	94.12 ± 5.76	93.78 ± 5.18	94.10 ± 5.26	**94.39 ± 5.52**	94.10 ± 5.26
Yeast	47.16 ± 3.50	45.38 ± 4.68	**47.75 ± 2.82**	46.56 ± 3.87	**47.75 ± 2.82**	46.65 ± 3.90	47.57 ± 3.14
Ave.	70.63 ± 24.76	60.29 ± 26.30	57.67 ± 28.05	68.03 ± 25.19	70.17 ± 24.07	67.64 ± 23.45	**71.29 ± 24.30**

Table B.4: Results in AUC for each Pareto ensemble method. The reported results are the average and the standard deviation over the replications and the best one is shown in **boldface**

Dataset	PER	PCE	MDE	Boosting	GPE	PKB	PEE
Banana	93.68 ± 2.45	86.85 ± 3.87	57.41 ± 9.10	89.93 ± 9.08	**93.98 ± 0.87**	92.84 ± 2.76	93.88 ± 1.08
Cancer	68.08 ± 5.20	67.50 ± 5.05	67.23 ± 6.61	67.66 ± 5.45	**68.30 ± 5.19**	68.27 ± 5.55	68.09 ± 5.37
Diabetis	81.16 ± 1.70	75.69 ± 5.39	70.02 ± 12.48	81.80 ± 2.06	**82.57 ± 1.45**	80.94 ± 2.55	82.39 ± 1.56
Solar	72.08 ± 4.64	69.62 ± 5.42	64.13 ± 7.46	71.64 ± 6.23	74.55 ± 1.95	**74.67 ± 1.62**	74.62 ± 1.91
German	77.69 ± 3.31	77.41 ± 2.86	76.34 ± 2.48	77.88 ± 2.66	**77.68 ± 2.75**	77.73 ± 2.66	**77.88 ± 2.86**
Heart	87.84 ± 2.66	84.30 ± 9.06	86.94 ± 3.58	86.93 ± 3.89	88.73 ± 2.90	88.03 ± 2.99	**88.76 ± 2.78**
Image	**99.70 ± 0.15**	97.91 ± 1.96	71.89 ± 10.26	98.88 ± 1.14	99.27 ± 0.62	99.52 ± 0.31	99.34 ± 0.54
Ringnorm	98.55 ± 0.89	98.60 ± 0.85	98.42 ± 1.00	99.10 ± 0.51	**99.28 ± 0.48**	99.01 ± 0.46	99.27 ± 0.50
Splice	**99.03 ± 0.34**	93.73 ± 9.82	74.03 ± 6.13	98.58 ± 0.34	98.37 ± 0.81	98.17 ± 1.00	98.47 ± 0.66
Thyroid	96.81 ± 1.30	97.05 ± 1.36	95.79 ± 2.08	**97.36 ± 0.87**	97.20 ± 1.07	97.15 ± 1.11	97.18 ± 0.98
Titanic	70.62 ± 1.76	70.73 ± 2.20	69.34 ± 3.46	71.33 ± 1.54	71.66 ± 1.51	71.80 ± 2.10	**71.85 ± 1.88**
Twonorm	98.80 ± 1.05	98.38 ± 1.76	97.59 ± 2.74	99.35 ± 0.31	99.57 ± 0.17	98.53 ± 1.73	**99.58 ± 0.16**
Waveform	95.35 ± 2.04	93.25 ± 2.76	78.44 ± 16.45	96.26 ± 0.68	**96.54 ± 0.75**	96.44 ± 0.82	96.43 ± 0.73
Appendicitis	**73.39 ± 7.44**	72.57 ± 7.59	67.60 ± 10.26	72.20 ± 8.04	72.05 ± 8.91	70.38 ± 7.40	71.58 ± 9.89
Balance	82.79 ± 5.99	84.51 ± 6.98	82.83 ± 5.96	89.57 ± 7.81	**91.06 ± 7.47**	89.17 ± 7.34	**91.06 ± 7.38**
Ionosphere	74.12 ± 5.49	81.12 ± 18.40	70.90 ± 10.33	85.60 ± 10.21	**95.62 ± 1.56**	87.97 ± 9.54	95.37 ± 1.68
Iris	94.67 ± 0.56	94.83 ± 0.34	91.60 ± 8.89	**94.93 ± 0.20**	94.80 ± 0.31	94.80 ± 0.31	94.87 ± 0.27
Mammographic	**90.17 ± 4.18**	78.47 ± 11.73	66.42 ± 23.51	88.86 ± 3.32	89.32 ± 3.76	89.64 ± 3.59	89.51 ± 3.74
Phoneme	96.45 ± 0.98	79.99 ± 11.05	89.85 ± 1.61	88.48 ± 2.64	90.50 ± 1.96	93.73 ± 1.59	**96.60 ± 0.77**
Pima	78.59 ± 4.15	76.41 ± 7.05	76.05 ± 5.63	79.16 ± 5.95	**78.90 ± 4.90**	77.87 ± 6.75	78.65 ± 4.96
Sonar	77.79 ± 11.59	76.18 ± 10.32	73.48 ± 8.91	87.75 ± 8.61	**89.09 ± 6.90**	87.14 ± 7.26	88.87 ± 7.48
Wdbc	89.82 ± 15.38	90.82 ± 14.06	95.91 ± 2.51	96.40 ± 2.53	**96.44 ± 2.37**	96.32 ± 2.68	96.42 ± 2.37
Wine	**95.58 ± 0.53**	**95.58 ± 0.53**	93.39 ± 6.51	**95.58 ± 0.53**	**95.58 ± 0.53**	**95.58 ± 0.53**	**95.58 ± 0.53**
Wisconsin	97.92 ± 0.43	97.78 ± 0.85	97.89 ± 0.44	97.90 ± 0.47	97.95 ± 0.42	**98.00 ± 0.40**	97.96 ± 0.43
Yeast	69.75 ± 12.29	68.00 ± 14.03	71.10 ± 11.98	69.89 ± 12.18	71.10 ± 11.98	**71.22 ± 13.44**	71.00 ± 12.16
Ave.	86.42 ± 10.83	84.29 ± 10.56	79.38 ± 12.24	87.32 ± 10.34	88.40 ± 10.26	87.80 ± 10.27	**88.61 ± 10.42**

B.2 Experimental Results when Comparing with a Single Criterion Approach

Table B.5 to Table B.8 show the results obtained when a single solution is chosen (EN-MOMS-Single) and those reported by the proposal when using PEE as the Pareto processing. These tables show the accuracy performance, the average accuracy, the kappa statistic, and the AUC score, respectively.

Table B.5: Results in classification accuracy for the EN-MOMS-single solution and EN-MOMS-Pbe with PEE Pareto processing. The reported results are the average and standard deviation over the replications and the best one is shown in **boldface**

Dataset	Single	PEE
Banana	88.25 ± 1.37	$\mathbf{88.34 \pm 0.69}$
Cancer	70.00 ± 3.69	$\mathbf{70.26 \pm 3.04}$
Diabetis	72.40 ± 4.69	$\mathbf{76.47 \pm 1.64}$
Solar	64.95 ± 3.09	$\mathbf{66.72 \pm 2.36}$
German	73.80 ± 2.04	$\mathbf{75.17 \pm 2.75}$
Heart	79.00 ± 4.27	$\mathbf{82.40 \pm 4.15}$
Image	$\mathbf{97.98 \pm 0.40}$	97.92 ± 0.53
Ringnorm	98.12 ± 0.27	$\mathbf{98.29 \pm 0.22}$
Splice	91.73 ± 7.33	$\mathbf{95.95 \pm 0.59}$
Thyroid	92.27 ± 5.68	$\mathbf{94.53 \pm 3.01}$
Titanic	75.73 ± 1.82	$\mathbf{76.99 \pm 1.49}$
Twonorm	96.77 ± 0.81	$\mathbf{97.40 \pm 0.18}$
Waveform	89.11 ± 1.15	$\mathbf{90.40 \pm 0.55}$
Appendicitis	81.18 ± 11.46	$\mathbf{85.00 \pm 7.42}$
Balance	97.76 ± 1.07	$\mathbf{97.92 \pm 1.25}$
Ionosphere	93.45 ± 4.43	$\mathbf{96.29 \pm 1.83}$
Iris	$\mathbf{96.00 \pm 4.42}$	95.33 ± 4.27
Mammographic	82.73 ± 4.77	$\mathbf{83.46 \pm 4.18}$
Phoneme	87.45 ± 1.90	$\mathbf{91.34 \pm 1.72}$
Pima	70.71 ± 4.27	72.80 ± 3.29
Sonar	82.67 ± 8.86	$\mathbf{85.05 \pm 10.38}$
Wdbc	95.07 ± 3.45	$\mathbf{97.18 \pm 1.98}$
Wine	$\mathbf{97.78 \pm 3.69}$	$\mathbf{97.78 \pm 3.69}$
Wisconsin	$\mathbf{97.28 \pm 2.67}$	$\mathbf{97.28 \pm 2.43}$
Yeast	59.30 ± 1.77	$\mathbf{59.84 \pm 2.47}$
Ave.	85.26 ± 11.49	$\mathbf{86.81 \pm 11.22}$

Table B.6: Results in classification average accuracy for the EN-MOMS-single solution and EN-MOMS-Pbe with PEE Pareto processing. The reported results are the average and standard deviation over the replications and the best one is shown in **boldface**

Dataset	Single	PEE
Banana	**87.99 ± 1.40**	87.74 ± 0.83
Cancer	60.33 ± 6.90	**61.03 ± 4.98**
Diabetis	66.04 ± 8.53	**69.75 ± 2.10**
Solar	63.38 ± 5.28	**66.36 ± 2.96**
German	68.91 ± 2.86	**69.53 ± 2.51**
Heart	78.36 ± 4.33	**81.86 ± 4.05**
Image	**97.92 ± 0.46**	97.81 ± 0.63
Ringnorm	98.11 ± 0.26	**98.29 ± 0.23**
Splice	91.71 ± 7.46	**95.96 ± 0.63**
Thyroid	89.09 ± 7.42	**92.03 ± 5.00**
Titanic	**69.50 ± 1.71**	67.50 ± 3.28
Twonorm	96.77 ± 0.82	**97.40 ± 0.18**
Waveform	88.45 ± 1.78	**90.22 ± 0.65**
Appendicitis	**77.64 ± 11.28**	76.11 ± 14.67
Balance	96.01 ± 4.13	**96.13 ± 4.51**
Ionosphere	93.95 ± 3.79	**95.39 ± 2.39**
Iris	**96.00 ± 4.42**	95.33 ± 4.27
Mammographic	82.54 ± 4.78	**83.48 ± 3.92**
Phoneme	82.32 ± 2.81	**89.29 ± 1.25**
Pima	65.56 ± 5.47	**67.67 ± 4.57**
Sonar	82.60 ± 9.08	**84.99 ± 10.68**
Wdbc	94.43 ± 4.13	**96.58 ± 2.64**
Wine	97.69 ± 3.98	**97.88 ± 3.80**
Wisconsin	97.56 ± 2.70	**97.64 ± 2.41**
Yeast	53.64 ± 5.03	**54.10 ± 4.57**
Ave.	83.06 ± 13.58	**84.40 ± 13.50**

Table B.7: Results in kappa statistic for the EN-MOMS-single solution and EN-MOMS-Pbe with PEE Pareto processing. The reported results are the average and standard deviation over the replications and the best one is shown in **boldface**

Dataset	Single	PEE
Banana	76.18 ± 2.80	$\mathbf{76.19 \pm 1.48}$
Cancer	21.46 ± 13.98	$\mathbf{23.48 \pm 10.30}$
Diabetis	32.95 ± 17.25	$\mathbf{42.99 \pm 3.94}$
Solar	26.90 ± 10.50	$\mathbf{32.57 \pm 5.54}$
German	37.53 ± 4.07	$\mathbf{39.71 \pm 4.32}$
Heart	57.16 ± 8.72	$\mathbf{64.17 \pm 8.19}$
Image	$\mathbf{95.88 \pm 0.82}$	95.75 ± 1.10
Ringnorm	96.23 ± 0.53	$\mathbf{96.59 \pm 0.45}$
Splice	83.39 ± 14.78	$\mathbf{91.89 \pm 1.19}$
Thyroid	80.69 ± 13.91	$\mathbf{86.07 \pm 8.38}$
Titanic	$\mathbf{41.27 \pm 2.64}$	39.70 ± 6.07
Twonorm	93.53 ± 1.62	$\mathbf{94.81 \pm 0.36}$
Waveform	75.72 ± 2.62	$\mathbf{78.75 \pm 0.97}$
Appendicitis	$\mathbf{50.34 \pm 21.77}$	50.09 ± 26.52
Balance	96.08 ± 1.85	$\mathbf{96.33 \pm 2.22}$
Ionosphere	86.24 ± 8.82	$\mathbf{91.84 \pm 4.05}$
Iris	$\mathbf{94.00 \pm 6.63}$	93.00 ± 6.40
Mammographic	65.20 ± 9.59	$\mathbf{66.86 \pm 8.16}$
Phoneme	68.07 ± 5.09	$\mathbf{79.85 \pm 2.90}$
Pima	32.35 ± 10.95	$\mathbf{36.91 \pm 8.59}$
Sonar	65.10 ± 18.12	$\mathbf{69.84 \pm 21.17}$
Wdbc	94.43 ± 4.13	$\mathbf{96.58 \pm 2.64}$
Wine	96.60 ± 5.65	$\mathbf{96.62 \pm 5.63}$
Wisconsin	$\mathbf{94.10 \pm 5.78}$	$\mathbf{94.10 \pm 5.26}$
Yeast	46.86 ± 2.13	$\mathbf{47.57 \pm 3.14}$
Ave.	68.33 ± 25.01	$\mathbf{71.29 \pm 24.30}$

Table B.8: Results in AUC for each of the EN-MOMS-single solution and EN-MOMS-Pbe with PEE Pareto processing. The reported results are the average and standard deviation over the replications and the best one is shown in **boldface**

Dataset	Single	PEE
Banana	90.28 ± 2.61	$\mathbf{93.88 \pm 1.08}$
Cancer	67.13 ± 5.40	$\mathbf{68.09 \pm 5.37}$
Diabetis	78.47 ± 6.21	$\mathbf{82.39 \pm 1.56}$
Solar	70.18 ± 4.16	$\mathbf{74.62 \pm 1.91}$
German	76.77 ± 2.65	$\mathbf{77.88 \pm 2.86}$
Heart	86.50 ± 4.62	$\mathbf{88.76 \pm 2.78}$
Image	$\mathbf{99.48 \pm 0.79}$	99.34 ± 0.54
Ringnorm	98.39 ± 0.76	$\mathbf{99.27 \pm 0.50}$
Splice	95.43 ± 6.38	$\mathbf{98.47 \pm 0.66}$
Thyroid	95.25 ± 4.15	$\mathbf{97.18 \pm 0.98}$
Titanic	70.24 ± 1.15	$\mathbf{71.85 \pm 1.88}$
Twonorm	98.57 ± 1.05	$\mathbf{99.58 \pm 0.16}$
Waveform	93.12 ± 3.59	$\mathbf{96.43 \pm 0.73}$
Appendicitis	$\mathbf{71.58 \pm 6.76}$	$\mathbf{71.58 \pm 9.89}$
Balance	85.37 ± 8.10	$\mathbf{91.06 \pm 7.38}$
Ionosphere	80.34 ± 9.63	$\mathbf{95.37 \pm 1.68}$
Iris	94.67 ± 0.67	$\mathbf{94.87 \pm 0.27}$
Mammographic	87.44 ± 6.06	$\mathbf{89.51 \pm 3.74}$
Phoneme	92.11 ± 2.03	$\mathbf{96.60 \pm 0.77}$
Pima	78.20 ± 5.15	$\mathbf{78.65 \pm 4.96}$
Sonar	76.48 ± 7.51	$\mathbf{88.87 \pm 7.48}$
Wdbc	93.49 ± 7.30	$\mathbf{96.42 \pm 2.37}$
Wine	$\mathbf{95.58 \pm 0.53}$	$\mathbf{95.58 \pm 0.53}$
Wisconsin	97.95 ± 0.41	$\mathbf{97.96 \pm 0.43}$
Yeast	70.23 ± 12.64	$\mathbf{71.00 \pm 12.16}$
Ave.	85.73 ± 10.54	$\mathbf{88.61 \pm 10.42}$

B.3 Experimental Results when Comparing with Full Model Selection Methods

The resulting accuracy, average accuracy, kappa statistic, and AUC score are reported, respectively, from Table B.9 to Table B.12. It compares the results with two methods reported in the literature: EPSMS and Auto-Weka.

Table B.9: Results in classification accuracy for EN-MOMS-PbE with PPE Pareto processing, EPSMS, and Auto Weka. The reported results are the average and standard deviation over the replications and the best one is shown in **boldface**

Dataset	PEE	EPSMS	Auto-Weka
Banana	88.34 ± 0.69	$\mathbf{88.70 \pm 1.05}$	88.62 ± 0.73
Cancer	70.26 ± 3.04	66.75 ± 5.23	$\mathbf{70.52 \pm 2.79}$
Diabetis	$\mathbf{76.47 \pm 1.64}$	73.33 ± 2.11	75.90 ± 1.61
Solar	66.72 ± 2.36	65.15 ± 1.78	$\mathbf{66.88 \pm 1.84}$
German	75.17 ± 2.75	72.83 ± 2.82	$\mathbf{76.20 \pm 1.44}$
Heart	$\mathbf{82.40 \pm 4.15}$	80.80 ± 3.40	79.70 ± 3.41
Image	97.92 ± 0.53	97.03 ± 0.97	$\mathbf{98.41 \pm 0.32}$
Ringnorm	98.29 ± 0.22	89.79 ± 3.90	$\mathbf{98.41 \pm 0.12}$
Splice	95.95 ± 0.59	93.23 ± 1.00	$\mathbf{96.10 \pm 0.36}$
Thyroid	94.53 ± 3.01	$\mathbf{95.07 \pm 2.31}$	93.60 ± 2.44
Titanic	$\mathbf{76.99 \pm 1.49}$	72.12 ± 8.23	76.58 ± 2.87
Twonorm	$\mathbf{97.41 \pm 0.18}$	97.22 ± 0.26	$\mathbf{97.41 \pm 0.22}$
Waveform	$\mathbf{90.40 \pm 0.55}$	89.02 ± 0.66	90.08 ± 0.83
Appendicitis	$\mathbf{85.00 \pm 7.42}$	78.91 ± 11.56	83.27 ± 8.82
Balance	97.92 ± 1.25	94.26 ± 5.08	$\mathbf{99.68 \pm 0.95}$
Ionosphere	$\mathbf{96.29 \pm 1.83}$	88.02 ± 4.93	94.04 ± 3.14
Iris	95.33 ± 4.27	$\mathbf{96.00 \pm 4.42}$	$\mathbf{96.00 \pm 4.42}$
Mammographic	$\mathbf{83.46 \pm 4.18}$	$\mathbf{83.46 \pm 4.93}$	82.73 ± 4.89
Phoneme	91.34 ± 1.72	83.03 ± 2.53	$\mathbf{91.76 \pm 1.25}$
Pima	$\mathbf{72.80 \pm 3.29}$	69.52 ± 3.88	72.54 ± 5.02
Sonar	$\mathbf{85.05 \pm 10.38}$	76.69 ± 16.14	84.10 ± 7.61
Wdbc	$\mathbf{97.18 \pm 1.98}$	95.95 ± 3.34	97.01 ± 1.58
Wine	$\mathbf{97.78 \pm 3.69}$	97.75 ± 3.72	97.19 ± 3.75
Wisconsin	$\mathbf{97.28 \pm 2.43}$	93.56 ± 2.73	97.14 ± 2.56
Yeast	$\mathbf{59.84 \pm 2.47}$	55.76 ± 3.16	41.67 ± 18.03
Ave.	$\mathbf{86.81 \pm 11.22}$	83.76 ± 11.75	85.82 ± 13.38

Table B.10: Results in average accuracy for EN-MOMS-PbE with PPE Pareto processing, EPSMS, and Auto Weka. The reported results are the average and standard deviation over the replications and the best one is shown in **boldface**

Dataset	PEE	EPSMS	Auto-Weka
Banana	87.74 ± 0.83	$\mathbf{88.50 \pm 0.94}$	88.42 ± 0.66
Cancer	61.03 ± 4.98	$\mathbf{64.45 \pm 6.20}$	58.49 ± 5.39
Diabetis	69.75 ± 2.10	$\mathbf{72.90 \pm 1.20}$	70.36 ± 2.05
Solar	66.36 ± 2.96	$\mathbf{66.98 \pm 1.53}$	67.13 ± 1.82
German	69.53 ± 2.51	$\mathbf{71.70 \pm 2.68}$	67.44 ± 2.56
Heart	$\mathbf{81.86 \pm 4.05}$	80.67 ± 3.15	79.37 ± 3.29
Image	97.81 ± 0.63	97.06 ± 0.92	$\mathbf{98.36 \pm 0.32}$
Ringnorm	98.29 ± 0.23	89.74 ± 3.93	$\mathbf{98.41 \pm 0.12}$
Splice	95.96 ± 0.63	93.35 ± 1.04	$\mathbf{96.13 \pm 0.35}$
Thyroid	92.03 ± 5.00	$\mathbf{94.07 \pm 3.08}$	91.70 ± 4.19
Titanic	67.50 ± 3.28	$\mathbf{68.05 \pm 3.81}$	66.56 ± 5.72
Twonorm	$\mathbf{97.41 \pm 0.18}$	97.22 ± 0.26	$\mathbf{97.41 \pm 0.22}$
Waveform	$\mathbf{90.22 \pm 0.65}$	89.78 ± 0.39	89.89 ± 0.62
Appendicitis	76.11 ± 14.67	$\mathbf{78.06 \pm 13.01}$	71.25 ± 16.20
Balance	96.13 ± 4.51	90.72 ± 10.50	$\mathbf{99.77 \pm 0.69}$
Ionosphere	$\mathbf{95.39 \pm 2.39}$	83.93 ± 6.95	93.58 ± 3.11
Iris	95.33 ± 4.27	$\mathbf{96.00 \pm 4.42}$	$\mathbf{96.00 \pm 4.42}$
Mammographic	$\mathbf{83.48 \pm 3.92}$	83.29 ± 5.06	82.38 ± 5.03
Phoneme	$\mathbf{89.29 \pm 1.25}$	82.18 ± 2.41	88.71 ± 1.91
Pima	$\mathbf{67.67 \pm 4.57}$	67.45 ± 4.79	65.28 ± 6.67
Sonar	$\mathbf{84.99 \pm 10.68}$	77.12 ± 15.88	83.51 ± 7.97
Wdbc	96.58 ± 2.64	95.22 ± 4.33	$\mathbf{96.64 \pm 2.00}$
Wine	97.88 ± 3.80	$\mathbf{97.96 \pm 3.48}$	97.22 ± 3.96
Wisconsin	$\mathbf{97.64 \pm 2.41}$	91.49 ± 3.88	97.43 ± 2.48
Yeast	$\mathbf{54.10 \pm 4.57}$	46.34 ± 6.94	33.96 ± 21.23
Ave.	$\mathbf{84.40 \pm 13.50}$	82.57 ± 12.85	83.02 ± 16.24

Table B.11: Results in kappa statistic for EN-MOMS-PbE with PEE Pareto processing, EPSMS, and Auto Weka. The reported results are the average and standard deviation over the replications and the best one is shown in **boldface**

Dataset	PEE	EPSMS	Auto-Weka
Banana	76.19 ± 1.48	$\mathbf{77.11 \pm 2.06}$	76.96 ± 1.45
Cancer	23.48 ± 10.30	$\mathbf{27.02 \pm 12.27}$	18.82 ± 10.82
Diabetis	42.99 ± 3.94	$\mathbf{43.65 \pm 3.04}$	43.18 ± 4.37
Solar	$\mathbf{32.57 \pm 5.54}$	32.45 ± 2.92	33.13 ± 1.84
German	39.71 ± 4.32	$\mathbf{40.10 \pm 4.84}$	37.92 ± 4.99
Heart	$\mathbf{64.17 \pm 8.19}$	61.26 ± 6.61	58.86 ± 6.70
Image	95.75 ± 1.10	93.96 ± 1.98	$\mathbf{96.75 \pm 0.64}$
Ringnorm	96.59 ± 0.45	79.56 ± 7.83	$\mathbf{96.82 \pm 0.23}$
Splice	91.89 ± 1.19	86.47 ± 2.01	$\mathbf{92.19 \pm 0.72}$
Thyroid	86.07 ± 8.38	$\mathbf{88.23 \pm 5.49}$	84.55 ± 5.85
Titanic	$\mathbf{39.70 \pm 6.07}$	37.26 ± 9.97	37.50 ± 12.65
Twonorm	$\mathbf{94.81 \pm 0.36}$	94.43 ± 0.53	$\mathbf{94.81 \pm 0.43}$
Waveform	$\mathbf{78.75 \pm 0.97}$	76.21 ± 1.12	78.05 ± 1.53
Appendicitis	$\mathbf{50.09 \pm 26.52}$	47.58 ± 24.85	42.25 ± 32.32
Balance	96.33 ± 2.22	89.99 ± 9.01	$\mathbf{99.45 \pm 1.64}$
Ionosphere	$\mathbf{91.84 \pm 4.05}$	71.80 ± 12.73	87.15 ± 6.41
Iris	93.00 ± 6.40	$\mathbf{94.00 \pm 6.63}$	$\mathbf{94.00 \pm 6.63}$
Mammographic	$\mathbf{66.86 \pm 8.16}$	66.67 ± 9.97	65.07 ± 9.94
Phoneme	$\mathbf{79.85 \pm 2.90}$	61.23 ± 4.29	78.77 ± 4.10
Pima	$\mathbf{36.91 \pm 8.59}$	34.08 ± 8.66	33.20 ± 13.84
Sonar	$\mathbf{69.84 \pm 21.17}$	53.86 ± 31.54	67.60 ± 15.80
Wdbc	$\mathbf{96.58 \pm 2.64}$	91.14 ± 7.48	93.55 ± 3.44
Wine	$\mathbf{96.62 \pm 5.63}$	87.39 ± 29.64	95.70 ± 5.75
Wisconsin	$\mathbf{94.10 \pm 5.26}$	85.25 ± 6.30	93.80 ± 5.51
Yeast	$\mathbf{47.57 \pm 3.14}$	43.69 ± 3.88	26.90 ± 20.57
Ave.	$\mathbf{71.29 \pm 24.30}$	66.58 ± 22.32	69.08 ± 26.35

Table B.12: Results in AUC for each of EN-MOMS-PbE with PEE Pareto processing, EPSMS, and Auto Weka. The reported results are the average and standard deviation over the replications and the best one is shown in **boldface**

Dataset	PEE	EPSMS	Auto-Weka
Banana	93.88 ± 1.08	$\mathbf{95.78 \pm 0.52}$	93.31 ± 2.88
Cancer	68.09 ± 5.37	$\mathbf{68.65 \pm 6.15}$	64.62 ± 7.79
Diabetis	$\mathbf{82.39 \pm 1.56}$	81.63 ± 1.12	80.70 ± 4.44
Solar	$\mathbf{74.62 \pm 1.91}$	72.07 ± 2.15	69.34 ± 1.88
German	77.88 ± 2.86	78.09 ± 3.01	$\mathbf{79.01 \pm 2.40}$
Heart	$\mathbf{88.76 \pm 2.78}$	88.64 ± 2.85	85.73 ± 3.56
Image	99.34 ± 0.54	99.23 ± 0.44	$\mathbf{99.77 \pm 0.06}$
Ringnorm	$\mathbf{99.27 \pm 0.50}$	95.41 ± 2.90	99.15 ± 0.59
Splice	98.47 ± 0.66	97.26 ± 0.79	$\mathbf{99.09 \pm 0.13}$
Thyroid	97.18 ± 0.98	$\mathbf{97.47 \pm 0.77}$	88.27 ± 11.26
Titanic	$\mathbf{71.85 \pm 1.88}$	69.00 ± 5.46	68.11 ± 6.30
Twonorm	99.58 ± 0.16	$\mathbf{99.68 \pm 0.06}$	99.05 ± 0.97
Waveform	$\mathbf{96.43 \pm 0.73}$	96.02 ± 0.37	95.64 ± 2.52
Appendicitis	71.58 ± 9.89	$\mathbf{74.11 \pm 10.80}$	69.24 ± 10.30
Balance	91.06 ± 7.38	$\mathbf{93.16 \pm 5.73}$	80.51 ± 10.09
Ionosphere	$\mathbf{95.37 \pm 1.68}$	93.34 ± 2.96	93.52 ± 7.61
Iris	$\mathbf{94.87 \pm 0.27}$	94.57 ± 0.50	72.83 ± 13.09
Mammographic	$\mathbf{89.51 \pm 3.74}$	89.33 ± 4.22	88.16 ± 4.75
Phoneme	$\mathbf{96.60 \pm 0.77}$	91.10 ± 1.82	95.80 ± 2.25
Pima	$\mathbf{78.65 \pm 4.96}$	74.25 ± 4.77	77.97 ± 5.57
Sonar	$\mathbf{88.87 \pm 7.48}$	84.19 ± 11.95	83.28 ± 9.70
Wdbc	96.42 ± 2.37	$\mathbf{97.59 \pm 0.76}$	97.49 ± 1.07
Wine	95.58 ± 0.53	$\mathbf{95.66 \pm 0.21}$	87.05 ± 9.04
Wisconsin	$\mathbf{97.96 \pm 0.43}$	97.87 ± 0.68	97.33 ± 1.10
Yeast	71.00 ± 12.16	$\mathbf{74.78 \pm 12.91}$	73.41 ± 14.33
Ave.	$\mathbf{88.61 \pm 10.42}$	87.96 ± 10.34	85.54 ± 11.16