UNIVERSIDADE DA CORUÑA

FACULTY OF INFORMATICS

Department of Computer Science

PH.D. THESIS

Novel feature selection methods for high dimensional data

Author: Verónica Bolón Canedo

Advisors: Amparo Alonso Betanzos

Noelia Sánchez Maroño

A Coruña, March 2014

March 12, 2014
UNIVERSITY OF A CORUÑA

FACULTY OF INFORMATICS
Campus de Elviña s/n
15071 - A Coruña (Spain)

Stay hungry. Stay foolish.

Acknowledgements

The path to becoming a doctor has been long. It started back in the kindergarten, when I was curious about everything and the necessity of investigating arose. From primary to high school, I was taught how to read, how to write, how to add and subtract. It was in University when I learned how to fall seven times and stand up eight. After graduating, I started my PhD, without knowing exactly what it meant. Now that I have spent all my life learning, and this stage is about to finish, I know that the term *philosophy* in PhD has another meaning: *love of wisdom*. It was this love of wisdom which encouraged me to start this journey and to stay intellectually hungry, and I hope that it will never leave me.

As I have said, the path to becoming a doctor is long and hard, but also full of people to acknowledge. First of all, I would like to thank my parents. They raised me, taught me, supported me, encouraged me, loved me and provided me with everything I needed. For all these reasons, to them I dedicate this thesis.

Most of all, I would like to thank my thesis advisors, Amparo and Noelia. All these years they have trained me to be a student, a researcher, a doctor. There has always been an answer to my questions, a solution to the troubles I have encountered. Without their patient guidance I would not have been made it to this point.

I would like to express my gratitude to my lab mates. Not only did they help me with everything I needed, but they also became good friends. The coffee break was the best moment of the working day. Not only for the coffee –nor for the break–, but for the nice company. Thanks for that.

In this journey, I had also the opportunity to travel. I would like to acknowledge Dr. Principe for welcoming me to sunny Florida. A special thanks goes to all the friends I made in the CNEL Lab. They were my little family during those warm months, filling me with experiences I will never forget. *See you later, alligators.*

I also wish to thank José Manuel Benítez and Paco Herrera for receiving me with so much warmth during my short visit to Granada. They have taught me so much and are still doing so.

Fortunately, the path is full of distractions, and I also want to acknowledge them. I am truly thankful to all my friends and family for always helping me to occupy my free time and bringing nothing but good moments to my life. A special mention goes to my sister. *There is no better friend than a sister. And there is no better sister than you.*

Last but not least, I want to thank my husband Gabriel for supporting me in this journey. I cannot think of a better person to share my life with.

Verónica Bolón Canedo

Resumo

A selección de características defínese como o proceso de detectar as características relevantes e descartar as irrelevantes, co obxectivo de obter un subconxunto de características máis pequeno que describa axeitadamente o problema dado cunha degradación mínima ou incluso cunha mellora do rendemento. Coa chegada dos conxuntos de alta dimensión –tanto en mostras como en características–, volveuse indispensable a identificación axeitada das características relevantes en escenarios do mundo real. Neste contexto, os diferentes métodos dispoñibles atópanse cun novo reto en canto a aplicabilidade e escalabilidade. Ademais, é necesario desenvolver novos métodos que teñan en conta estas particularidades da alta dimensión. Esta tese está adicada á investigación en selección de características e á súa aplicación a datos reais de alta dimensión.

A primeira parte deste traballo trata da análise dos métodos de selección de características existentes, para comprobar a súa idoneidade fronte a diferentes retos e para poder proporcionar novos resultados aos investigadores de selección de características. Para isto, aplicáronse as técnicas máis populares a problemas reais, co obxectivo de obter non só melloras no rendemento senón tamén para permitir a súa aplicación en tempo real. Ademais da eficiencia, a escalabilidade tamén é un aspecto crítico nas aplicacións de grande escala. A eficacia dos métodos de selección de características pode verse significativamente degradada, se non totalmente inaplicable, cando o tamaño dos datos medra continuamente. Por este motivo, a escalabilidade dos métodos de selección de características tamén debe ser analizada.

Despois de levar a cabo unha análise en profundidade dos métodos de selección de características existentes, a segunda parte desta tese céntrase no desenvolvemento de novas técnicas. Debido a que a maooría dos métodos de selección existentes precisan que os datos sexan discretos, a primeira aproximación proposta consiste na combinación dun discretizador, un filtro e un clasificador, obtendo resultados prometedores en escenarios diferentes. Nun intento de introducir diversidade, a segunda proposta trata de empregar un conxunto de filtros en lugar dun só, co obxectivo de liberar ao usuario de ter que decidir que técnica é a máis axeitada para un problema dado. A terceira técnica proposta nesta tese non só considera a relevancia das características senón tamén o

seu custo asociado –económico ou canto a tempo de execución–, polo que se presenta unha metodoloxa xeral para selección de características baseada en custo. Por último, propóñense varias estratexias para distribuir e paralelizar a selección de características, xa que transformar un problema de grande escala en varios problemas de pequena escala pode levar a melloras no tempo de procesado e, nalgunhas ocasións, na precisión de clasificación.

Resumen

La selección de características se define como el proceso de detectar las características relevantes y descartar las irrelevantes, con el objetivo de obtener un subconjunto de características más pequeño que describa adecuadamente el problema dado con una degradación mínima o incluso con una mejora del rendimiento. Con la llegada de los conjuntos de alta dimensión –tanto en muestras como en características–, se ha vuelto indispensable la identificación adecuada de las características relevantes en escenarios del mundo real. En este contexto, los diferentes métodos disponibles se encuentran con un nuevo reto en cuanto a aplicabilidad y escalabilidad. Además, es necesario desarrollar nuevos métodos que tengan en cuenta estas particularidades de la alta dimensión. Esta tesis está dedicada a la investigación en selección de características y a su aplicación a datos reales de alta dimensión.

La primera parte de este trabajo trata del análisis de los métodos de selección de características existentes, para comprobar su idoneidad frente a diferentes retos y para poder proporcionar nuevos resultados a los investigadores de selección de características. Para esto, se han aplicado las técnicas más populares a problemas reales, con el objetivo de obtener no sólo mejoras en rendimiento sino también para permitir su aplicación en tiempo real. Además de la eficiencia, la escalabilidad también es un aspecto crítico en aplicaciones de gran escala. La eficacia de los métodos de selección de características puede verse significativamente degradada, si no totalmente inaplicable, cuando el tamaño de los datos se incrementa continuamente. Por este motivo, la escalabilidad de los métodos de selección de características también debe ser analizada.

Tras llevar a cabo un análisis en profundidad de los métodos de selección de características existentes, la segunda parte de esta tesis se centra en el desarrollo de nuevas técnicas. Debido a que la mayoría de métodos de selección existentes necesitan que los datos sean discretos, la primera aproximación propuesta consiste en la combinación de un discretizador, un filtro y un clasificador, obteniendo resultados prometedores en escenarios diferentes. En un intento de introducir diversidad, la segunda propuesta trata de usar un conjunto de filtros en lugar de uno sólo, con el objetivo de liberar al usuario de tener que decidir qué técnica es la más adecuada para un problema dado. La tercera

técnica propuesta en esta tesis no sólo considera la relevancia de las características sino también su coste asociado –económico o en cuanto a tiempo de ejecución–, por lo que se presenta una metodologa general para selección de características basada en coste. Por último, se proponen varias estrategias para distribuir y paralelizar la selección de características, ya que transformar un problema de gran escala en varios problemas de pequeña escala puede llevar a mejoras en el tiempo de procesado y, en algunas ocasiones, en precisión de clasificación.

Abstract

Feature selection can be defined as the process of detecting the relevant features and discarding the irrelevant ones, with the goal of obtaining a small subset of features that describes properly a given problem with a minimum degradation or even improvement in performance. With the advent of high dimensionality –both in samples and features–, the adequate identification of the relevant features of the data has become indispensable in real world scenarios. In this context, the different methods available encounter a new challenge regarding application and scalability. Also, new methods that take into account the peculiarities of high dimension need to be developed. This thesis is devoted to feature selection research and its application to real high dimensional data.

The first part of this work deals with the analysis of existing feature selection methods, to check their adequacy toward different challenges and to be able to provide new findings for feature selection researchers. To this end, the most popular techniques are applied to real-life problems, in an attempt to obtain not only improvements in performance but also to allow a real-time application of the techniques. Apart from efficiency, scalability is also a critical issue in large-scale applications. The effectiveness of feature selection methods may be significantly downgraded, if not totally inapplicable, when the data size increases steadily. For this reason, scalability in feature selection is analyzed as well.

After carrying out an in-depth analysis of existing feature selection methods, the second part of this thesis is focused on proposing novel techniques aimed at solving some of the problems detected in the field. Since most of the existing feature selection methods need data to be discrete, the first proposed approach consists of a combination of a discretizer, a filter method and a classifier, obtaining promising results in different scenarios. In an attempt to introduce diversity, the second proposal lies on employing an ensemble of filters instead of a single one, with the aim of releasing the user from the decision of which technique is the most appropriate for a given problem. The third technique proposed in this thesis considers not only the relevance of the features but also their related cost –economic or in terms of processing time–, so that a framework for cost-based feature selection is described. Finally, several approaches for distributed and

parallel feature selection are also proposed, since transforming the large-scale problem into several small-scale problems can lead to improvements in processing time and, sometimes, in classification accuracy.

Contents

List of figures

List of tables

List of algorithms

CHAPTER 1

Introduction

The advent of high dimensional data has brought unprecedented challenges to machine learning researchers, making the learning task more complex and computationally demanding. The term high dimensionality is applied to a database that presents one of the following characteristics: (a) the number of samples is very high; (b) the number of features is very high; or (c) both the number of samples and features are very high. There exists in the literature some controversy about the term *high-dimensionality*, since some authors claim that it only refers to the feature space whereas others use it indistinctly for both features and samples. In this thesis the latter alternative will be adopted and a dataset will be considered of very high dimensionality when having more than 10000 data (where data means *features x samples*) according to Z. A. Zhao and Liu (2011).

When dealing with high-dimensional data, learning algorithms can degenerate their performance due to overfitting, learned models decrease their interpretability as they are more complex, and finally speed and efficiency of the algorithms decline in accordance with size. Machine learning can take advantage of feature selection methods to be able to reduce the dimensionality of a given problem. *Feature selection* is the process of detecting the relevant features and discarding the irrelevant and redundant ones, with the goal of obtaining a small subset of features that describes properly the given problem with a minimum degradation or even improvement in performance (Guyon, 2006). Feature selection, as it is an important activity in data preprocessing, has been an active research area in the last decade, finding success in many different real world applications, especially those related with classification problems.

This thesis is devoted to feature selection research and its application to high dimensional data. The work presented herein flows from general principles to proposing novel methods. First, a critical analysis of existing feature selection methods is performed, to check their adequacy toward different challenges and to be able to provide some recommendations to the users. Bearing this analysis in mind, the most adequate

techniques are applied to several real-life problems, obtaining a notable improvement in performance. Apart from efficiency, another critical issue in large-scale applications is scalability. The effectiveness of feature selection methods may be significantly downgraded, if not totally inapplicable, when the data size increases steadily. For this reason, scalability in feature selection is analyzed.

Then, new techniques for large-scale feature selection are proposed. In the first place, as most of the existing feature selection techniques need data to be discrete, a new approach is proposed that consists of a combination of a discretizer, a filter method and a very simple classical classifier, obtaining promising results. Another proposal is to employ a ensemble of filters instead of a single one, releasing the user from the decision of which technique is the most appropriate for a given problem. An interesting topic is also to consider the cost related with the different features (economic, or associated with memory or time requirements), therefore a framework for cost-based feature selection is proposed, demonstrating its adequacy in a real-life scenario. Finally, it is well-known that a manner of handling large-scale data is to transform the large-scale problem into several small-scale problems, by distributing the data. With this aim, several approaches for distributed and parallel feature selection are proposed.

Before diving into the specific aspects of each topic, in this introduction a summary of the main goals of the present thesis in each part is given.

1.1 Analysis of feature selection

Feature selection methods usually come in three flavors: *filter*, *wrapper*, and *embedded* methods (Guyon, 2006). The *filter* model relies on the general characteristics of training data and carries out the feature selection process as a pre-processing step with independence of the induction algorithm. On the contrary, *wrappers* involve optimizing a predictor as a part of the selection process. Halfway these two models one can find *embedded* methods, which perform feature selection in the process of training and are usually specific to given learning machines.

There exists a vast body of feature selection methods in the literature, including filters based on distinct metrics (e.g. entropy, probability distributions or information theory) and embedded and wrappers methods using different induction algorithms. The proliferation of feature selection algorithms, however, has not brought about a general

methodology that allows for intelligent selection from existing algorithms. In order to make a correct choice, a user not only needs to know the domain well, but also is expected to understand technical details of available algorithms. On top of this, most algorithms were developed when dataset sizes were much smaller, but nowadays distinct compromises are required for the case of small-scale and large-scale (big data) learning problems. Small-scale learning problems are subject to the usual approximation-estimation trade-off. In the case of large-scale learning problems, the trade-off is more complex because it involves not only the accuracy of the selection but also other aspects, such as stability (i.e. the sensitivity of the results to training set variations) or scalability.

The first part of this thesis is devoted to analyzing the state of the art feature selection methods and demonstrating their adequacy on real applications. The main goals of this block of the thesis are the following:

- Critical review of the most popular feature selection methods in the literature by checking their performance in an artificial controlled experimental scenario. In this manner, the ability of the algorithms to select the relevant features and to discard the irrelevant ones without permitting noise or redundancy to obstruct this process is evaluated.
- Analysis of the behavior of feature selection in a very challenging field: DNA microarray classification. DNA microarray data is a hard challenge for machine learning researchers due to the high number of features (around 10 000) but small sample size (typically one hundred or less). For this purpose, it is necessary to review the most up-to-date algorithms developed ad-hoc for this type of data, as well as studying their particularities.
- Application of classic feature selection to real problems in order to check their adequacy. Specifically, testing the effectiveness of feature selection in two problems from the medical domain: tear film lipid layer classification and K-complex identification in sleep apnea.
- Analysis of the issue of scalability in feature selection. With the advent of high dimensionality, machine learning researchers are not focused only in accuracy, but also in the scalability of the solution. Therefore, this issue must be addressed. First, the influence of feature selection to scaling up machine learning algorithms is tested. Then, a study in detail of the scalability of feature selection methods is also necessary.

1.2 Novel feature selection methods

The second part of the thesis is devoted to developing novel feature selection capable of being applied to high dimensional datasets. Although the benefits of the feature selection process have been extensively proved, most researchers agree that there is not a so-called "best method" and their efforts are focused on finding a good method for a specific problem setting. For this reason, new feature selection methods are constantly emerging using different strategies. In fact, the current tendency in feature selection is not toward developing new algorithmic measures, but toward favoring the combination or modification of existing algorithms. Therefore, the objective of this part of the thesis is focused in exploring different strategies to deal with the new problematics which have emerged derived from the big data explosion.

Our first proposal is related with preprocessing techniques, so a discretization stage was introduced prior to feature selection trying to improve the performance of the induction methods. Another interesting and popular line of research in classification is ensemble learning, based on the assumption that a set of experts is better than a single expert, so inspired on this idea an ensemble of filters and classifiers is proposed. It is also interesting to consider cases in which features have their own risk or cost, since this factor must be taken into account as well as the accuracy. For this reason, feature selection methods which tackle cost are also proposed. Finally, a recent topic of interest has arisen which consists of distributing the feature selection process, trying to improve accuracy whilst reducing the training time. Some proposals are presented in this thesis aiming at covering this issue.

To sum up, the main goals of the second part of this thesis are the following:

- Development of a new framework which consists of combining discretization and filter methods. This framework is successfully applied to intrusion detection and microarray data classification.
- Development of a novel method for dealing with high-dimensional data: an ensemble of filters and classifiers. The idea of this ensemble is to apply several filters based on different metrics and then joining the results obtained after training a classifier with the selected subset of features. In this manner, the user is released from the task of choosing an adequate filter for each dataset.

- Proposal for a new framework for cost-based feature selection. In this manner, the scope of feature selection is broaden by taking into consideration not only the relevance of the features but also their associated costs. The proposed framework consists of adding a new term to the evaluation function of a filter method so that the cost is taken into account.

- Distributed and parallel feature selection. There are two common types of data distribution: (a) horizontal distribution wherein data are distributed in subsets of instances; and (b) vertical distribution wherein data are distributed in subsets of attributes. Both approaches are tested, employing for this sake filter and wrapper methods. Since in some cases the partitioning of the datasets can introduce some redundancy among features, new partitioning schemes are being investigated, for example by dividing the features according to some goodness measure.

1.3 Overview of this thesis

This chapter has introduced the main topics to be presented in this work. Figure 1.1 depicts the organization of the thesis. Part I is covered by chapters 2 - 6. Chapter 2 presents the foundations of feature selection, as well as a description of the feature selection methods which will be employed in this thesis. Then, Chapter 3 reviews the most popular methods in the literature and checks their performance in an artificial controlled scenario, proposing some guidelines about their appropriateness in different domains. Chapter 4 analyzes the up-to-date contributions of feature selection research applied to the field of DNA microarray classification, whereas Chapter 5 is devoted to proving the benefits of feature selection in other real applications such as classification of the tear film lipid layer and K-complex classification. Chapter 6 closes Part I by studying the scalability of existing feature selection methods.

Part II is covered by chapters 7 - 10. Chapter 7 presents a method which consists of a combination of discretizers, filters and classifiers. The proposed method is applied over an intrusion detection benchmark dataset, as well as other challenging scenarios such as DNA microarray data. Chapter 8 introduces an ensemble of filters to be applied to different scenarios. The idea builds on the assumption that an ensemble of filters is better than a single method, since it is possible to take advantage of their individual strengths and overcome their weak points at the same time. Chapter 9 proposes a new framework for cost-based feature selection. The objective is to solve

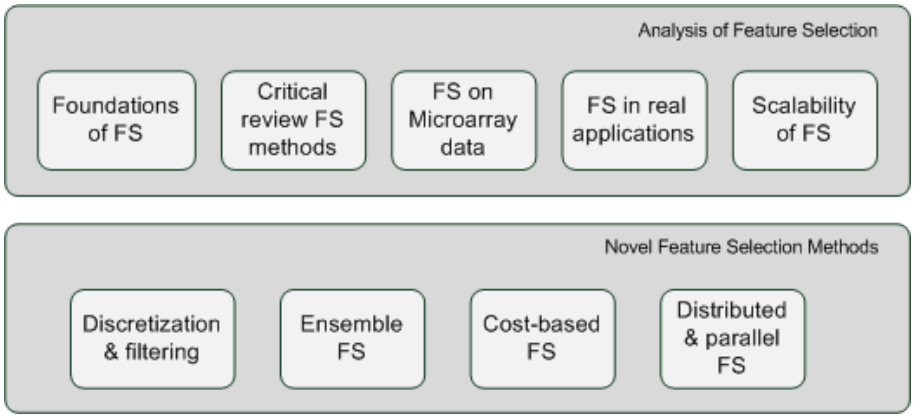


Figure 1.1: Organization of the thesis

problems in which it is interesting not only to minimize the classification error, but also to reduce costs that may be associated to input features. Chapter 10 presents some approaches for distributed and parallel feature selection, splitting the data both vertically and horizontally. Finally, Chapter 11 summarizes the main conclusions and contributions of this thesis. Notice that Appendix I presents the materials and methods used throughout this thesis and Appendix II reports the author's key publications and mentions.

PART I

Analysis of feature selection

CHAPTER 2

Foundations of feature selection

In the last years, several datasets with high dimensionality have become publicly available on the Internet. This fact has brought an interesting challenge to the research community, since for the machine learning methods it is difficult to deal with a high number of input features. To confront the problem of the high number of input features, dimensionality reduction techniques can be applied to reduce the dimensionality of the original data and improve learning performance. These dimensionality reduction techniques usually come in two flavors: *feature selection* and *feature extraction*.

Feature selection and feature extraction each have their own merits (Z. A. Zhao & Liu, 2011). On the one hand, feature extraction techniques achieve dimensionality reduction by combining the original features. In this manner, they are able to generate a set of new features, which is usually more compact and of stronger discriminating power. It is preferable in applications such as image analysis, signal processing, and information retrieval, where model accuracy is more important than model interpretability. On the other hand, feature selection achieves dimensionality reduction by removing the irrelevant and redundant features. It is widely used in data mining applications, such as text mining, genetics analysis, and sensor data processing. Due to the fact that feature selection maintains the original features, it is especially useful for applications where the original features are important for model interpreting and knowledge extraction.

This chapter will present the foundations of feature selection, as well as a description of the feature selection methods which will be employed in this thesis.

2.1 Feature selection

Feature selection can be defined as the process of detecting the relevant features and discarding the irrelevant and redundant ones with the goal of obtaining a subset of

features that describes properly the given problem with a minimum degradation of performance. It has several advantages (Guyon, 2006), such as:

- Improving the performance of the machine learning algorithms.
- Data understanding, gaining knowledge about the process and perhaps helping to visualize it.
- General data reduction, limiting storage requirements and perhaps helping in reducing costs.
- Feature set reduction, saving resources in the next round of data collection or during utilization.
- Simplicity, possibility of using simpler models and gaining speed.

2.1.1 Feature relevance

Intuitively, it can be determined that a feature is relevant if it contains some information about the target. More formally, Kohavi & John classified features into three disjoint categories, namely, strongly relevant, weakly relevant, and irrelevant features (Kohavi & John, 1997). In their approach, the relevance of a feature X is defined in terms of an ideal Bayes classifier. A feature X is considered to be *strongly relevant* when the removal of X results in a deterioration of the prediction accuracy of the ideal Bayes classifier. A feature X is said to be *weakly relevant* if it is not strongly relevant and there exists a subset of features S, such that the performance of the ideal Bayes classifier on S is worse than the performance on $S \cup \{X\}$. A feature is defined as *irrelevant* if it is neither strongly nor weakly relevant.

2.1.2 Feature redundancy

A feature is usually considered as redundant in terms of feature correlation (Yu & Liu, 2004a). It is widely accepted that two features are redundant to each other if their values are completely correlated, but it might not be so easy to determine feature redundancy when a feature is correlated with a set of features. According to Yu and Liu (2004a), a feature is redundant and hence should be removed if it is weakly relevant and

has a Markov blanket (Koller & Sahami, 1995) within the current set of features. Since irrelevant features should be removed anyway, they are excluded from this definition of redundant features.

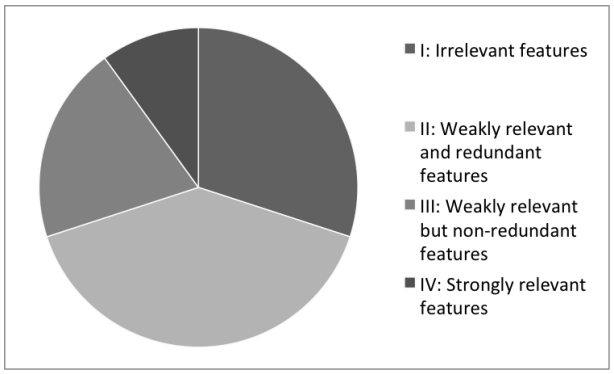


Figure 2.1: Overview of feature relevance and redundancy

Figure 2.1 visualizes an overview of the relationship between feature relevance and redundancy. The entire feature set can be conceptually divided into four basic disjoint parts: irrelevant features (I), weakly relevant and redundant features (II), weakly relevant but non-redundant features (III) and strongly relevant features (IV) (Yu & Liu, 2004a). Notice that the optimal subset would cointain all the features in parts III and IV.

2.2 Feature selection methods

Feature selection methods can be divided according to two approaches: *individual evaluation* and *subset evaluation* (Yu & Liu, 2004a). Individual evaluation is also known as feature ranking and assesses individual features by assigning them weights according to their degrees of relevance. On the other hand, subset evaluation produces candidate feature subsets based on a certain search strategy. Each candidate subset is evaluated by a certain evaluation measure and compared with the previous best one with respect to this measure. While the individual evaluation is incapable of removing redundant features because redundant features are likely to have similar rankings, the subset evaluation approach can handle feature redundancy with feature relevance. However, methods in this framework can suffer from an inevitable problem caused by searching through all feature subsets required in the subset generation step, and thus, both approaches are worth it to be studied.

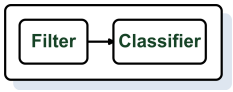
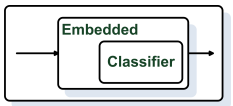
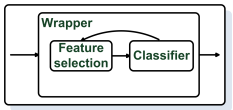
Aside from this classification, three major approaches can be distinguished based upon the relationship between a feature selection algorithm and the inductive learning method used to infer a model (Guyon, 2006):

- *Filters*, which rely on the general characteristics of training data and carry out the feature selection process as a pre-processing step with independence of the induction algorithm. This model is advantageous for its low computational cost and good generalization ability.
- *Wrappers*, which involve a learning algorithm as a black box and consists of using its prediction performance to assess the relative usefulness of subsets of variables. In other words, the feature selection algorithm uses the learning method as a subroutine with the computational burden that comes from calling the learning algorithm to evaluate each subset of features. However, this iteration with the classifier tends to give better performance results than filters.
- *Embedded methods*, which perform feature selection in the process of training and are usually specific to given learning machines. Therefore, the search for an optimal subset of features is built into the classifier construction and can be seen as a search in the combined space of feature subsets and hypotheses. This approach is able to capture dependencies at a lower computational cost than wrappers.

Table 2.1 provides a summary of the characteristics of the three feature selection methods, indicating the most prominent advantages and disadvantages.

Considering that there exist several algorithms for each one of the previously commented approaches, there is a vast body of feature selection methods. Most researchers agree that "the best method" simply does not exist and their efforts are focused on finding an good method for a specific problem setting. In that sense, different methods have been developed to deal with large scale datasets where the importance of feature selection is beyond doubt, since it is essential to minimize training time and allocated memory while maintaining accuracy. Nevertheless, it is important to bear in mind that most feature selection methods use the performance of the learned model as part of the selection process. In fact, from the three categories shown above (filters, wrappers and embedded), only filters are algorithm-independent. This property makes filters computationally simple and fast, being able to handle extremely large-scale datasets. However, most filters are univariate, i.e. they consider each feature independently of

Table 2.1: Feature selection techniques.

Method	Advantages	Disadvantages
Filter 	Independence of the classifier Lower computational cost than wrappers Fast Good generalization ability	No interaction with the classifier
Embedded 	Interaction with the classifier Lower computational cost than wrappers Captures feature dependencies	Classifier-dependent selection
Wrapper 	Interaction with the classifier Captures feature dependencies	Computationally expensive Risk of overfitting Classifier-dependent selection

other features, a drawback that can be overcome by multivariate techniques which usually demand more computational resources.

2.2.1 Filter methods

Filter methods are based on performance evaluation metric calculated directly from the data, without direct feedback from predictors that will finally be used on data with reduced number of features (Guyon, 2006). As mentioned above, these algorithms are usually computationally less expensive than wrappers or embedded methods. In this subsection, the most popular filters are described, which will be used throughout this thesis.

2.2.1.1 Chi-squared

This is an univariate filter based on the χ^2 statistic (Liu & Setiono, 1995) and which evaluates each feature independently with respect to the classes. The higher the value of chi-squared, the more relevant is the feature with respect to the class.

2.2.1.2 Information Gain

The Information Gain filter (Quinlan, 1986) is one of the most common univariate methods of evaluation attributes. This filter evaluates the features according to their information gain and considers a single feature at a time. It provides an orderly classification of all the features, and then a threshold is required to select a certain number of them according to the order obtained.

2.2.1.3 Correlation-based Feature Selection, CFS

This is a simple multivariate filter algorithm that ranks feature subsets according to a correlation based heuristic evaluation function (M. A. Hall, 1999).The bias of the evaluation function is toward subsets that contain features that are highly correlated with the class and uncorrelated with each other. Irrelevant features should be ignored because they will have low correlation with the class. Redundant features should be screened out as they will be highly correlated with one or more of the remaining features. The acceptance of a feature will depend on the extent to which it predicts classes in areas of the instance space not already predicted by other features.

2.2.1.4 Consistency-based Filter

The filter based on consistency (Dash & Liu, 2003) evaluates the worth of a subset of features by the level of consistency in the class values when the training instances are projected onto the subset of attributes. From the space of features, the algorithm generates a random subset S in each iteration. If S contains fewer features than the current best subset, the inconsistency index of the data described by S is compared with the index of inconsistency in the best subset. If S is as consistent or more than

the best subset, S becomes the best subset. The criterion of inconsistency, which is the key to success of this algorithm, specify how large can be the reduction of dimension in the data. If the rate of consistency of the data described by selected characteristics is smaller than a set threshold, it means that the reduction in size is acceptable. Notice that this method is multivariate.

2.2.1.5 Fast Correlation-Based Filter, FCBF

The fast correlated-based filter method (Yu & Liu, 2003) is a multivariate algorithm that measures feature-class and feature-feature correlation. FCBF starts by selecting a set of features that is highly correlated with the class based on symmetrical uncertainty (SU), which is defined as the ratio between the information gain and the entropy of two features. Then, it applies three heuristics that remove the redundant features and keep the features that are more relevant to the class. FCBF was designed for high-dimensionality data and has been shown to be effective in removing both irrelevant and redundant features. However, it fails to take into consideration the interaction between features.

2.2.1.6 INTERACT

The INTERACT algorithm (Z. Zhao & Liu, 2007) uses the same goodness measure as FCBF filter, i.e. SU, but it also includes the consistency contribution, which is an indicator about how significantly the elimination of a feature will affect consistency. The algorithm consists of two major parts. In the first part, the features are ranked in descending order based on their SU values. In the second part, features are evaluated one by one starting from the end of the ranked feature list. If the consistency contribution of a feature is less than an established threshold, the feature is removed, otherwise it is selected. The authors stated that this method can handle feature interaction, and efficiently selects relevant features.

2.2.1.7 ReliefF

The filter ReliefF (Kononenko, 1994) is an extension of the original Relief algorithm (Kira & Rendell, 1992). The original Relief works by randomly sampling an instance

from the data and then locating its nearest neighbor from the same and opposite class. The values of the attributes of the nearest neighbors are compared to the sampled instance and used to update relevance scores for each attribute. The rationale is that a useful attribute should differentiate between instances from different classes and have the same value for instances from the same class. ReliefF adds the ability of dealing with multiclass problems and is also more robust and capable of dealing with incomplete and noisy data. This method may be applied in all situations, has low bias, includes interaction among features and may capture local dependencies which other methods miss.

2.2.1.8 minimum Redundancy Maximum Relevance, mRMR

The mRMR method (H. Peng, Long, & Ding, 2005) selects features that have the highest relevance with the target class and are also minimally redundant, i.e., selects features that are maximally dissimilar to each other. Both optimization criteria (Maximum-Relevance and Minimum-Redundancy) are based on mutual information.

2.2.1.9 $\mathcal{M}_d$

The $\mathcal{M}_d$ filter (Seth & Principe, 2010) is an extension of mRMR which uses a measure of monotone dependence (instead of mutual information) to assess relevance and irrelevance. One of its contributions is the inclusion of a free parameter (λ) that controls the relative emphasis given on relevance and redundancy. In this thesis, two values of lambda will be tested: 0 and 1. When λ is equal to zero, the effect of the redundancy disappears and the measure is based only on maximizing the relevance. On the other hand, when λ is equal to one, it is more important to minimize the redundancy among variables. These two values of λ were chosen in this thesis because we are interested in checking the performance of the method when the effect of the redundancy disappears. Also, Seth and Principe (2010) stated that $\lambda = 1$ performs better than other λ values.

Table 2.2 reports the main characteristics of the filters employed in this thesis. With regard to the computational cost, it can be noticed that some of the proposed filter techniques are univariate. This means that each feature is considered separately, thereby ignoring feature dependencies, which may lead to worse classification performance when compared to other types of feature selection techniques. However, they

have the advantage, in theory, of being scalable. Multivariate filter techniques were introduced, aiming to incorporate feature dependencies to some degree, but at the cost of reducing their scalability.

Table 2.2: Summary of filters

	Uni/Multivariate	Ranker/Subset
Chi-Squared	Univariate	Ranker
Information Gain	Univariate	Ranker
ReliefF	Multivariate	Ranker
mRMR	Multivariate	Ranker
$\mathcal{M}_d$	Multivariate	Ranker
CFS	Multivariate	Subset
FCBF	Multivariate	Subset
INTERACT	Multivariate	Subset
Consistency	Multivariate	Subset

2.2.2 Embedded methods

In contrast to filter and wrapper approaches, embedded methods do not separate the learning from the feature selection part. Embedded methods include algorithms, which optimize a regularized risk function with respect to two sets of parameters: the parameters of the learning machine and the parameters indicating which features are selected. The embedded methods used in this thesis are subsequently described.

2.2.2.1 Recursive Feature Elimination for Support Vector Machines, SVM-RFE

This embedded method was introduced by Guyon in (Guyon, Weston, Barnhill, & Vapnik, 2002). It performs feature selection by iteratively training a SVM classifier with the current set of features and removing the least important feature indicated by the weights in the SVM solution.

2.2.2.2 Feature Selection - Perceptron, FS-P

FS-P (Mejía-Lavalle, Sucar, & Arroyo, 2006) is an embedded method based on a perceptron. A perceptron is a type of artificial neural network that can be seen as the simplest kind of feedforward neural network: a linear classifier. The basic idea of this method consists on training a perceptron in the context of supervised learning. The interconnection weights are used as indicators of which features could be the most relevant and provide a ranking.

2.2.3 Wrapper methods

The idea of the wrapper approach is to select a feature subset using a learning algorithm as part of the evaluation function. Instead of using subset sufficiency, entropy or another explicitly defined evaluation function, a kind of "black box" function is used to guide the search. The evaluation function for each candidate feature subset returns an estimate of the quality of the model that is induced by the learning algorithm. This can be rather time consuming, since, for each candidate feature subset evaluated during the search, the target learning algorithm is usually applied several times (e.g. in the case of 10-fold cross validation being used to estimate model quality). In this thesis it will be used the wrapper following described, which can be combined with any learning algorithm.

Weka (M. Hall et al., 2009) provides the **WrapperSubsetEval** method, which evaluates attribute sets by using a learning scheme. Cross validation is used to estimate the accuracy of the learning scheme for a set of attributes. The algorithm starts with the empty set of attributes and searches forward, adding attributes until performance does not improve further.

2.2.4 Other approaches

There exist numerous papers and books proving the benefits of the feature selection process (Guyon, 2006; Z. A. Zhao & Liu, 2011; Kohavi & John, 1997). However, most researchers agree that there is not a so-called "best method" and their efforts are focused on finding a good method for a specific problem setting. Therefore, new feature

selection methods are constantly appearing using different strategies: a) combining several feature selection methods, which could be done by using algorithms from the same approach, such as two filters (Y. Zhang, Ding, & Li, 2008), or coordinating algorithms from two different approaches, usually filters and wrappers (Y. Peng, Wu, & Jiang, 2010; El Akadi, Amine, El Ouardighi, & Aboutajdine, 2011); b) combining feature selection approaches with other techniques, such as feature extraction (Vainer, Kraus, Kaminka, & Slovin, 2011) or tree ensembles (Tuv, Borisov, Runger, & Torkkola, 2009); c) reinterpreting existing algorithms (Sun & Li, 2006), sometimes to adapt them to specific problems (Sun, Todorovic, & Goodison, 2008a); d) creating new methods to deal with still unresolved situations (Chidlovskii & Lecerf, 2008; Loscalzo, Yu, & Ding, 2009) and e) using an ensemble of feature selection techniques to ensure a better behavior (Saeys, Abeel, & Van de Peer, 2008; Bolón-Canedo, Sánchez-Maroño, & Alonso-Betanzos, 2012).

2.3 Summary

To confront the problem of the high dimensionality of data, feature selection algorithms have become indispensable components of the learning process. Hence, a correct selection of the features can lead to an improvement of the inductive learner, either in terms of learning speed, generalization capacity or simplicity of the induced model.

Feature selection methods may roughly be divided into three types: filters, wrappers and embedded methods. This chapter has summarized the advantages and disadvantages of each approach, as well as described the algorithms that will be used throughout this thesis.

CHAPTER 3

A critical review of feature selection methods

In the past few years, feature selection algorithms have become indispensable components of the learning process to get rid of irrelevant and redundant features. As mentioned in the previous chapter, there exist two major approaches in feature selection: *individual evaluation* and *subset evaluation*. Individual evaluation is also known as feature ranking (Guyon & Elisseeff, 2003) and assesses individual features by assigning them weights according to their degrees of relevance. On the other hand, subset evaluation produces candidate feature subsets based on a certain search strategy. Besides this classification, feature selection methods can also be divided into three models: *filters*, *wrappers* and *embedded* methods (Guyon, 2006). With such a vast body of feature selection methods, the need arises to find out some criteria that enable users to adequately decide which algorithm to use (or not) in certain situations.

There are several situations that can hinder the process of feature selection, such as the presence of irrelevant and redundant features, noise in the data or interaction between attributes. In the presence of hundreds or thousands of features, such as DNA microarray analysis, researchers notice (Y. Yang & Pedersen, 1997; Yu & Liu, 2004a) that commonly a large number of features is not informative because they are either irrelevant or redundant with respect to the class concept. Moreover, when the number of features is high but the number of samples is small, machine learning gets particularly difficult, since the search space will be sparsely populated and the model will not be able to distinguish correctly the relevant data and the noise (Provost, 2000).

This chapter reviews the most popular feature selection methods in the literature and checks their performance in an artificial controlled experimental scenario. In this manner, it is contrasted the ability of the algorithms to select the relevant features and to discard the irrelevant ones without permitting noise or redundancy to obstruct this process. A scoring measure will be introduced to compute the degree of matching between the output given by the algorithm and the known optimal solution, as well as the classification accuracy.

3.1 Existing reviews of feature selection methods

Feature selection, since it is an important activity in data preprocessing, has been widely studied in the past years by the machine learning researchers. This technique has found success in many different real world applications like DNA microarray analysis (Yu & Liu, 2004b), intrusion detection (Bolón-Canedo, Sánchez-Maroño, & Alonso-Betanzos, 2011; W. Lee, Stolfo, & Mok, 2000), text categorization (Forman, 2003; Gomez, Boiy, & Moens, 2012) or information retrieval (Egozi, Gabrilovich, & Markovitch, 2008), including image retrieval (Dy, Brodley, Kak, Broderick, & Aisen, 2003) or music information retrieval (Saari, Eerola, & Lartillot, 2011).

Bearing in mind the large amount of feature selection methods available, it is easy to note that carrying out a comparative study is an arduous task. Another problem is to test the effectiveness of these feature selection methods when real data sets are employed, usually without knowing the relevant features. In these cases the performance of the feature selection methods clearly rely on the performance of the learning method used afterwards and it can vary notably from one method to another. Moreover, performance can be measured using many different metrics such as computer resources (memory and time), accuracy, ratio of features selected, etc. Besides, datasets may include a great number of challenges: multiple class output, noisy data, huge number of irrelevant features, redundant or repeated features, ratio number of samples/number of features very close to zero and so on. It can be noticed that a comparative study tackling all these considerations could be unapproachable and therefore, most of the interesting comparative studies are focused on the problem to be solved. So, for example, Forman (2003) presented an empirical comparison of twelve feature selection methods evaluated on a benchmark of 229 text classification problem instances; another comparative study (Sun, Babbs, & Delp, 2006) is used for the detection of breast cancers in mammograms. Other works are devoted to a specific approach (Liu, Liu, & Zhang, 2008), in which an experimental study of eight typical filter mutual information based feature selection algorithms on thirty-three datasets is presented; or an evaluation of the capability of the survival ReliefF algorithm (sReliefF) and of a tuned sReliefF approach to properly select the causative pair of attributes (Beretta & Santaniello, 2011). Similarly, there are works examining different feature selection methods to obtain good performance results using a specific classifier (naive Bayes in (M.-L. Zhang, Peña, & Robles, 2009), C4.5 in (Perner & Apte, 2000) or the theoretical review for support vector machines (SVMs) in (Victo Sudha & Raj, 2011)). Related to dataset challenges, there are several works trying to face the problem of high dimensionality, in both di-

mensions (samples and features) or in one of them, i.e. a high number of features versus low number of samples; most of these studies also tackle with the multiple class problems (Chidlovskii & Lecerf, 2008; Li, Zhang, & Ogihara, 2004; Hua, Tembe, & Dougherty, 2009; Bontempi & Meyer, 2010; Aliferis, Statnikov, Tsamardinos, Mani, & Koutsoukos, 2010). Also, the majority of current real datasets (microarray, text retrieval, etc.) also present noisy data, however no specific feature selection comparative studies dealing with this complex problem were found in the literature, although some interesting works have been proposed, see for example (Y. Zhang et al., 2008; Byeon & Rasheed, 2008; S.-H. Yang & Hu, 2008). Focusing on non-linear methods is worth mentioning the study of Guyon and col. (Guyon, Bitter, Ahmed, Brown, & Heller, 2005). Finally, from a theoretical perspective, Liu and Yu (2005) presented a survey of feature selection methods, providing some guidelines in selecting feature selection algorithms, paving the way to build an integrated system for intelligent feature selection.

More experimental work on feature selection algorithms for comparative purposes can be found (Molina, Belanche, & Nebot, 2002; Doak, 1992; Jain & Zongker, 1997; Kudo & Sklansky, 1997; Liu & Setiono, 1998), some of which were performed over artificially generated data, like the widely-used *Parity*, *Led* or *Monks* problems (Thrun et al., 1991). Several authors choose to use artificial data since the desired output is known, therefore a feature selection algorithm can be evaluated with independence of the classifier used. Although the final goal of a feature selection method is to test its effectiveness over a real dataset, the first step should be on synthetic data. The reason for this is two-fold (Belanche & González, 2011):

1. Controlled experiments can be developed by systematically varying chosen experimental conditions, like adding more irrelevant features or noise in the input. This fact facilitates to draw more useful conclusions and to test the strengths and weaknesses of the existing algorithms.

2. The main advantage of artificial scenarios is the knowledge of the set of optimal features that must be selected, thus the degree of closeness to any of these solutions can be assessed in a confident way.

3.2 Experimental settings

As was stated in Section 3.1, the first step to test the effectiveness of a feature selection method should be on synthetic data, since the knowledge of the optimal features and the chance to modify the experimental conditions allows to draw more useful conclusions. The datasets chosen for this study try to cover different problems: increasing number of irrelevant features, redundancy, noise in the output, alteration of the inputs, non-linearity of the data, etc. These factors complicate the task of the feature selection methods, which are very affected by them as it will be shown afterwards. Besides, some of the datasets have a significantly higher number of features than samples, which implies an added difficulty for the correct selection of the relevant features. The characteristics of the eleven synthetic datasets employed (Corral, Corral-100, XOR-100, Parity3+3, Led-25, Led-100, Monk3, SD1, SD2, SD3 and Madelon) can be consulted in Appendix I, Table I.1.

Twelve different feature selection methods are tested and compared in order to draw useful conclusions. The chosen methods are CFS, consistency-based, INTERACT, Information Gain, ReliefF, mRMR, M_d, SVM-RFE, FS-P and the wrapper combined with SVM and C4.5; a description of all of them can be found in Chapter 2. In the case of M_d, two different values of λ will be tested: 0 and 1. As for SVM-RFE, two kernels will be considered: a linear kernel, which is the kernel used by default, and a Gaussian kernel in an attempt to solve more complex problems (Rakotomamonjy, 2003), which will be referred in this chapter as SVM-RFE-G.

As was mentioned in Chapter 2, there exist two major approaches in feature selection: *individual evaluation*, that provides an ordered ranking of the features; and *subset evaluation*, that produces a candidate feature subset. When a ranking of the features is returned, it is necessary to establish a threshold in order to discard those less relevant for the algorithm. Unfortunately, where to establish the threshold is not an easy-to-solve question. Belanche and González (2011) opted for discarding those weights associated to the ranking which were further than two variances from the mean. On the other hand, Mejía-Lavalle et al. (2006) use a threshold defined by the largest gap between two consecutive ranked attributes, and other authors (Sánchez-Maroño, Alonso-Betanzos, & Tombilla-Sanromán, 2007) just studied the whole ranking paying more attention to the first ranked features. However, in this experimental study it is impossible to use a threshold related to the weights associated to the ranking, since some of the ranker methods (SVM-RFE, mRMR and M_d) eliminate chunks of features at a time and do

not provide weights. To solve this problem and for the sake of fairness, in these experiments we heuristically set the following rules to decide the number of features that ranker methods should return, according to the number of total features (N):

- if N $<$ 10, select 75% of features
- if 10 $<$ N $<$ 75, select 40% of features
- if 75 $<$ N $<$ 100, select 10% of features
- if N $>$ 100, select 3% of features

At this point it has to be clarified that the datasets SD (see Appendix I), due to their particularities, will be analyzed in a different manner which will be explained later. According to the rules showed above, the number of features that will be returned by ranker methods is 5 for the datasets Corral, Parity3+3 and Monk3, 10 for the datasets Corral-100, XOR-100 and both versions of Led, and 15 for Madelon.

A scoring measure was defined in order to fairly compare the effectiveness showed by the different feature selection methods. The measure presented is a index of success *suc.*, see (3.1), which attempts to reward the selection of relevant features and to penalize the inclusion of irrelevant ones, in two situations:

- The solution is *incomplete*: there are relevant features lacking.
- The solution is *incorrect*: there are some irrelevant features.

$$suc. = \left[\frac{R_s}{R_t} - \alpha\frac{I_s}{I_t}\right] \times 100, \tag{3.1}$$

where R_s is the number of relevant features selected, R_t is the total number of relevant features, I_s is the number of irrelevant features selected and I_t is the total number of irrelevant features. The term α was introduced to ponder that choosing an irrelevant feature is better than missing a relevant one (i.e. we prefer an incorrect solution rather than an incomplete one). The parameter α is defined as $\alpha = \min\{\frac{1}{2}, \frac{R_t}{I_t}\}$. Note that the higher the success, the better the method, and 100 is the maximum.

In the case of ranker methods and in order to be fair, if all the optimal features are selected before any irrelevant feature, the index of success will be 100, due to the fact that the number of features that ranker methods are forced to select is always larger than the number of relevant features.

As was explained above, the evaluation of the feature selection methods is done by counting the number of correct/wrong features. However, it is also interesting and a common practice in the literature (Mamitsuka, 2006) to see the classification accuracy obtained in a 10-fold cross-validation, in order to check if the true model (that is, the one with an index of success of 100) is also unique (that is, if is the only one that can achive the best percentage of classification success). For this purpose, four well-known classifiers, based on different models, were chosen: C4.5, naive Bayes (NB), k-NN and SVM (see Appendix I). Experimental evidence has shown that decision trees, such as C4.5, exhibit a degradation in the performance when faced with many irrelevant features. Similarly, instance-based learners, such as k-NN, are also very susceptible to irrelevant features. It has been shown that the number of training instances needed to produce a predetermined level of performance for instance-based learning increases exponentially with the number of irrelevant features present (Langley & Iba, 1993). On the other hand, algorithms such as naive Bayes are robust with respect to irrelevant features, degrading their performance very slowly when more irrelevant features are added. However, the performance of such algorithms deteriorate quickly by adding redundant features, even if they are relevant to the concept. Finally, SVM can indeed suffer in high dimensional spaces where many features are irrelevant (Weston et al., 2000).

3.3 Experimental results

In all tables of this section, the best index of success and the best accuracy for each classifier are highlighted in bold face. Columns "Rel." (relevant), "Irr." (irrelevant) and "suc." (success, see equation (3.1)) refer to the evaluation via counting the number of correct features selected, whilst the remaining columns show the classification accuracy obtained by four different classifiers. It has to be noted that for the calculation of the index of success, the redundant attributes selected have the same penalization as the irrelevant features.

3.3.1 Dealing with correlation and redundancy: CorrAL

Two versions of this well-known datasets were used: CorrAL (the classic dataset) and CorrAL-100, formed by adding 93 irrelevant binary features (see Appendix I, Section I.2.1.1). The desired behavior of a feature selection method is to select the 4 relevant features and to discard the irrelevant ones, as well as detecting the correlated feature and not selecting it.

Table 3.1: Results for CorrAL. "C" indicates if the correlated feature is selected (✓) or not (✗).

Method	Rel.	C	Irr.	suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
CFS	–	✓	0	-25	75.00	75.00	59.38	75.00
Consistency	–	✓	0	-25	75.00	75.00	59.38	75.00
INTERACT	–	✓	0	-25	75.00	75.00	59.38	75.00
InfoGain	–	✓	0	-25	75.00	75.00	59.38	75.00
ReliefF	1-4	✓	0	75	62.50	**81.25**	96.88	**87.50**
mRMR	1-4	✓	0	75	62.50	**81.25**	96.88	**87.50**
$M_d(\lambda = 0)$	1-4	✓	0	75	62.50	**81.25**	96.88	**87.50**
$M_d(\lambda = 1)$	1-4	✓	0	75	62.50	**81.25**	96.88	**87.50**
SVM-RFE	1-4	✓	0	75	62.50	**81.25**	96.88	**87.50**
SVM-RFE-G	1-4	✗	1	75	**81.25**	78.13	81.25	71.86
FS-P	1-4	✗	0	**100**	**81.25**	78.13	**100.00**	81.25
Wrapper SVM	–	✓	0	-25	75.00	75.00	59.38	75.00
Wrapper C4.5	–	✓	0	-25	75.00	75.00	59.38	75.00

Tables 3.1 and 3.2 show the results obtained over the datasets Corral and Corral-100, respectively. Over Corral, FS-P was able to select the desired set of features, which led to 100% classification accuracy obtained by k-NN classifier. Regarding Corral-100, it is curious that the best classification accuracy was obtained by SVM-RFE, which has an index of success of 25, but this fact can be explained because in this dataset there are some irrelevant features that are informative to the classifiers. This fact will be further analyzed in Section 3.5.

Table 3.2: Results for CorrAL-100. "C" indicates if the correlated feature is selected (✓) or not (✗).

Method	Rel.	C	Irr.	suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
CFS	–	✓	0	-2	75.00	75.00	59.38	75.00
Consistency	–	✓	0	-2	75.00	75.00	59.38	75.00
INTERACT	–	✓	0	-2	75.00	75.00	59.38	75.00
InfoGain	–	✓	0	-2	75.00	75.00	59.38	75.00
ReliefF	1-3	✓	6	75	53.13	84.38	87.50	81.25
mRMR	1-4	✓	5	**99**	53.13	81.25	**90.63**	90.63
$M_d(\lambda = 0)$	1-4	✓	5	**99**	65.63	81.25	87.50	81.25
$M_d(\lambda = 1)$	1-4	✓	5	**99**	59.38	84.38	81.25	87.50
SVM-RFE	4	✓	8	25	62.50	**87.50**	68.75	**96.88**
SVM-RFE-G	–	✓	9	-44	68.75	68.75	62.50	75.00
FS-P	1,3,4	✓	6	75	53.13	**87.50**	84.38	87.50
Wrapper SVM	–	✓	0	-2	75.00	75.00	59.38	75.00
Wrapper C4.5	–	✓	2	-13	**84.38**	75.00	75.00	75.00

3.3.2 Dealing with non-linearity: XOR and Parity

In this subsection, the ability of feature selection methods to deal with relevance, irrelevance and redundancy will be checked over two non-linear scenarios, XOR and Parity (see Appendix I, Sections I.2.1.2 and I.2.1.3). XOR-100 contains 2 relevant features and 97 irrelevant features whilst Parity3+3 has 3 relevant, 3 redundant and 6 irrelevant features. In the case of XOR-100, there exists the added handicap of a small ratio between number of samples and number of features. For the sake of completeness, SVM and naive Bayes will be applied over these two datasets. However, bear in mind that those methods cannot solve non-linear problems (SVM uses a linear kernel) and thus good results are not expected, so they will not be the focus of the analysis.

Tables 3.3 and 3.4 show that the methods CFS, Consistency, INTERACT and InfoGain do not appear because they were not able to solve these non-linear problems, returning an empty subset of features. On the other hand, the filter ReliefF and the embedded method SVM-RFE-G detected the relevant features both in XOR-100 and in Parity3+3, achieving the best indices of success and leading to high classification accuracies.

Table 3.3: Results for XOR-100.

Method	Rel.	Irr.	suc.	Accuracy (%)			
				C4.5	NB	k-NN	SVM
ReliefF	1,2	0	**100**	**100.00**	64.00	**100.00**	70.00
mRMR	1	9	50	52.00	74.00	64.00	72.00
$M_d(\lambda = 0)$	1	9	50	54.00	74.00	58.00	70.00
$M_d(\lambda = 1)$	1	9	50	58.00	70.00	62.00	62.00
SVM-RFE	–	10	-21	48.00	68.00	56.00	**78.00**
SVM-RFE-G	1,2	0	**100**	**100.00**	64.00	**100.00**	70.00
FS-P	1	9	50	62.00	**76.00**	62.00	74.00
Wrapper SVM	–	1	-2	66.00	66.00	60.00	66.00
Wrapper C4.5	1,2	2	99	**100.00**	70.00	96.00	50.00

Table 3.4: Results for Parity3+3

Method	Rel.	Red.	Irr.	suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
ReliefF	1,2,3	2	0	93	**90.63**	29.69	**100.00**	37.50
mRMR	2,3	0	3	56	60.94	59.38	59.38	59.38
$M_d(\lambda = 0)$	–	0	5	-19	53.13	57.81	50.00	59.38
$M_d(\lambda = 1)$	–	0	5	-19	54.69	54.69	54.69	57.81
SVM-RFE	3	0	4	19	54.69	59.38	46.88	57.81
SVM-RFE-G	1,2,3	0	0	**100**	**90.63**	31.25	**100.00**	25.00
FS-P	–	0	5	-19	51.56	57.81	56.25	57.81
Wrapper SVM	–	0	1	-4	64.06	**64.06**	57.81	**64.06**
Wrapper C4.5	–	0	1	-4	64.06	**64.06**	57.81	**64.06**

3.3.3 Dealing with noise in the inputs: Led

The Led dataset (see Appendix I, Section I.2.1.4) consists of correctly identifying seven leds that represent numbers between 0 and 9. Some irrelevant features were added forming the Led-25 dataset (17 irrelevant features) and the Led-100 dataset (92 irrelevant attributes). In order to make these datasets more complex, different levels of noise in the inputs (2%, 6%, 10%, 15% and 20%) were added. It has to be noted that, as the attributes take binary values, adding noise means assigning to the relevant features an incorrect value.

In Tables 3.5 and 3.6 one can see detailed results of these experiments. It is interesting to note that subset filters (CFS, Consistency and INTERACT) and the ranker filter Information Gain (which has a behavior similar to subset filters) do not select any of the irrelevant features in any case, at the expense of discarding some of the relevant ones, especially with high levels of noise. With regard to the classification accuracy, it decreases as the level of noise increases, as expected.

Table 3.5: Results for Led-25 dataset with different levels of noise (N) in inputs.

N(%)	Method	Relevant	Irr. No.	suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
0	CFS	1-5,7	0	86	92.00	**100.00**	**100.00**	96.00
	Consistency	1-5	0	71	**92.00**	**100.00**	**100.00**	96.00
	INTERACT	1-5,7	0	86	**92.00**	**100.00**	**100.00**	96.00
	InfoGain	1-7	0	**100**	**92.00**	**100.00**	**100.00**	96.00
	ReliefF	1-7	3	93	**92.00**	96.00	84.00	96.00
	mRMR	1-7	3	93	**92.00**	98.00	90.00	98.00
	$M_d(\lambda = 0)$	1-5,7	4	76	90.00	92.00	82.00	98.00
	$M_d(\lambda = 1)$	1-5,7	4	76	90.00	92.00	82.00	96.00
	SVM-RFE	1-7	3	93	**92.00**	98.00	80.00	96.00
	SVM-RFE n-l	1-7	0	**100**	**92.00**	**100.00**	**100.00**	96.00
	FS-P	1-7	0	**100**	**92.00**	**100.00**	**100.00**	96.00
	Wrapper SVM	1-5	2	67	**92.00**	90.00	82.00	**100.00**
	Wrapper C4.5	1-5	0	71	**92.00**	90.00	82.00	96.00

Continued on next page

Table 3.5 – continued from previous page

N(%)	Method	Relevant	Irr. No.	suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
2	CFS	1-5	0	71	**90.00**	**98.00**	**96.00**	94.00
	Consistency	1-5	0	71	**90.00**	**98.00**	**96.00**	94.00
	INTERACT	1-5	0	71	**90.00**	**98.00**	**96.00**	94.00
	InfoGain	1-7	0	**100**	**90.00**	96.00	94.00	94.00
	ReliefF	1-7	3	93	86.00	88.00	82.00	88.00
	mRMR	1-7	3	93	84.00	88.00	82.00	88.00
	$M_d(\lambda = 0)$	1-5,7	4	76	84.00	84.00	76.00	94.00
	$M_d(\lambda = 1)$	1-5,7	4	76	84.00	84.00	78.00	88.00
	SVM-RFE	1-7	3	93	86.00	88.00	86.00	94.00
	SVM-RFE n-l	1-7	3	93	88.00	92.00	80.00	86.00
	FS-P	1-7	0	**100**	**90.00**	96.00	94.00	94.00
	Wrapper SVM	1-5	2	67	**90.00**	88.00	80.00	**96.00**
	Wrapper C4.5	1-5	0	71	**90.00**	**98.00**	**96.00**	94.00
6	CFS	1,2,4,5,7	0	71	70.00	**76.00**	**66.00**	68.00
	Consistency	1,2,4,5,7	0	71	70.00	**76.00**	**66.00**	68.00
	INTERACT	1,2,4,5,7	0	71	70.00	**76.00**	**66.00**	68.00
	InfoGain	1,2,4,5,7	0	71	70.00	**76.00**	**66.00**	68.00
	ReliefF	1-7	3	**93**	64.00	62.00	**66.00**	64.00
	mRMR	1-7	3	**93**	64.00	62.00	**66.00**	64.00
	$M_d(\lambda = 0)$	1-5,7	4	76	62.00	62.00	60.00	68.00
	$M_d(\lambda = 1)$	1-5,7	4	76	62.00	64.00	60.00	66.00
	SVM-RFE	1-4,7	5	59	62.00	62.00	58.00	68.00
	SVM-RFE-G	1-5	5	59	60.00	62.00	58.00	64.00
	FS-P	1-6	4	76	64.00	56.00	56.00	62.00
	Wrapper SVM	1,2,4,5	3	50	68.00	66.00	58.00	**70.00**
	Wrapper C4.5	1,2,4-6	1	69	**72.00**	68.00	58.00	66.00
10	CFS	1,2,4,7	0	57	60.00	50.00	**58.00**	46.00
	Consistency	1,2,4,7	0	57	60.00	50.00	**58.00**	46.00
	INTERACT	1,2,4,7	0	57	60.00	50.00	**58.00**	46.00
	InfoGain	1,2,4,7	0	57	60.00	50.00	**58.00**	46.00
	ReliefF	1-5,7	4	76	56.00	50.00	52.00	54.00
	mRMR	1-5,7	4	76	56.00	50.00	52.00	54.00
	$M_d(\lambda = 0)$	1-5,7	4	76	58.00	52.00	52.00	60.00
	$M_d(\lambda = 1)$	1-5,7	4	76	56.00	50.00	52.00	54.00
	SVM-RFE	2,4,7	7	26	44.00	40.00	42.00	50.00
	SVM-RFE n-l	1-5	5	59	58.00	48.00	46.00	56.00
	FS-P	1-7	3	**93**	60.00	48.00	**58.00**	52.00
	Wrapper SVM	1,2,4-6	2	67	66.00	54.00	52.00	**68.00**
	Wrapper C4.5	1-4	0	57	**68.00**	**64.00**	**58.00**	62.00

Continued on next page

Table 3.5 – continued from previous page

N(%)	Method	Relevant	Irr. No.	suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
15	CFS	1,7	0	29	28.00	28.00	32.00	36.00
	Consistency	1,7	0	29	28.00	28.00	32.00	36.00
	INTERACT	1,7	0	29	28.00	28.00	32.00	36.00
	InfoGain	1,7	0	29	28.00	28.00	32.00	36.00
	ReliefF	1-7	3	**93**	48.00	38.00	46.00	52.00
	mRMR	1-7	3	**93**	48.00	38.00	46.00	52.00
	$M_d(\lambda = 0)$	1-5,7	4	76	48.00	42.00	44.00	52.00
	$M_d(\lambda = 1)$	1-5,7	4	76	44.00	36.00	46.00	48.00
	SVM-RFE	2,7	8	9	26.00	34.00	28.00	36.00
	SVM-RFE n-l	1,3,4,7	6	43	34.00	30.00	26.00	38.00
	FS-P	1-7	3	**93**	48.00	40.00	**48.00**	54.00
	Wrapper SVM	1,2	3	21	52.00	48.00	40.00	**56.00**
	Wrapper C4.5	1,2,5,7	0	57	**58.00**	44.00	40.00	54.00
20	CFS	1	0	14	28.00	20.00	28.00	28.00
	Consistency	1	0	14	28.00	20.00	28.00	28.00
	INTERACT	1	0	14	28.00	20.00	28.00	28.00
	InfoGain	1	0	14	28.00	20.00	28.00	28.00
	ReliefF	1-7	3	**93**	24.00	30.00	**40.00**	42.00
	mRMR	1-7	3	**93**	24.00	30.00	**40.00**	42.00
	$M_d(\lambda = 0)$	1-5,7	4	76	36.00	36.00	34.00	44.00
	$M_d(\lambda = 1)$	1-5,7	4	76	36.00	32.00	38.00	44.00
	SVM-RFE	5,6	8	9	16.00	24.00	18.00	20.00
	SVM-RFE n-l	1,2,4,5	6	43	38.00	36.00	20.00	34.00
	FS-P	1-7	3	**93**	34.00	34.00	36.00	40.00
	Wrapper SVM	1,2,5,7	3	50	44.00	38.00	38.00	**56.00**
	Wrapper C4.5	1,2,5	3	36	**48.00**	**40.00**	34.00	42.00

Table 3.6: Results for Led-100 dataset with different levels of noise (N) in inputs.

N(%)	Method	Relevant	Irr. No.	suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
0	CFS	1-5,7	0	86	**92.00**	**100.00**	**100.00**	96.00
	Consistency	1-5	0	71	**92.00**	**100.00**	**100.00**	96.00
	INTERACT	1-5,7	0	86	**92.00**	**100.00**	**100.00**	96.00
	InfoGain	1-7	0	**100**	**92.00**	**100.00**	**100.00**	96.00
	ReliefF	1-7	3	99	**92.00**	94.00	96.00	**100.00**
	mRMR	1-5,7	4	85	**92.00**	94.00	88.00	96.00
	$M_d(\lambda = 0)$	1-5,7	4	85	86.00	92.00	76.00	92.00
	$M_d(\lambda = 1)$	1-5,7	4	85	86.00	92.00	90.00	96.00
	SVM-RFE	3-7	5	71	46.00	54.00	48.00	48.00
	SVM-RFE-G	1-6	4	85	**92.00**	92.00	80.00	94.00
	FS-P	1-7	3	99	**92.00**	92.00	86.00	96.00
	Wrapper SVM	1-5	2	71	**92.00**	90.00	82.00	**100.00**
	Wrapper C4.5	1-5	0	71	**92.00**	**100.00**	**100.00**	96.00
2	CFS	1-5	0	71	**90.00**	**98.00**	**96.00**	94.00
	Consistency	1-5	0	71	**90.00**	**98.00**	**96.00**	94.00
	INTERACT	1-5	0	71	**90.00**	**98.00**	**96.00**	94.00
	InfoGain	1-7	0	**100**	**90.00**	96.00	94.00	94.00
	ReliefF	1-7	3	99	**90.00**	90.00	84.00	92.00
	mRMR	1-5,7	4	85	88.00	86.00	80.00	86.00
	$M_d(\lambda = 0)$	1-5,7	4	85	84.00	86.00	76.00	84.00
	$M_d(\lambda = 1)$	1-5,7	4	85	84.00	86.00	76.00	84.00
	SVM-RFE	3-7	5	71	68.00	70.00	54.00	70.00
	SVM-RFE-G	1-6	4	85	**90.00**	90.00	74.00	88.00
	FS-P	1-7	3	99	**90.00**	86.00	82.00	90.00
	Wrapper SVM	1-5	2	71	**90.00**	88.00	80.00	**96.00**
	Wrapper C4.5	1-5	0	71	**90.00**	**98.00**	**96.00**	94.00
6	CFS	1,2,4,5,7	0	71	72.00	**78.00**	**72.00**	70.00
	Consistency	1,2,4,5,7	0	71	72.00	**78.00**	**72.00**	70.00
	INTERACT	1,2,4,5,7	0	71	72.00	**78.00**	**72.00**	70.00
	InfoGain	1,2,4,5,7	0	71	72.00	**78.00**	**72.00**	70.00
	ReliefF	1-5,7	4	85	60.00	66.00	68.00	72.00
	mRMR	1-5,7	4	85	60.00	66.00	68.00	72.00
	$M_d(\lambda = 0)$	1,2,4,5,7	5	71	58.00	56.00	56.00	62.00
	$M_d(\lambda = 1)$	1-5,7	4	85	58.00	64.00	66.00	64.00
	SVM-RFE	2,3,5	7	42	52.00	50.00	34.00	52.00
	SVM-RFE-G	1-6	4	85	70.00	72.00	50.00	72.00
	FS-P	1-6	4	85	72.00	56.00	62.00	70.00
	Wrapper SVM	1-7	15	**99**	56.00	54.00	58.00	**84.00**
	Wrapper C4.5	1,2,4,5	2	57	**76.00**	72.00	66.00	72.00

Continued on next page

Table 3.6 – continued from previous page

N(%)	Method	Relevant	Irr. No.	suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
10	CFS	1,2,4,7	0	57	60.00	50.00	58.00	46.00
	Consistency	1,2,4,7	0	57	60.00	50.00	58.00	46.00
	INTERACT	1,2,4,7	0	57	60.00	50.00	58.00	46.00
	InfoGain	1,2,4,7	0	57	60.00	50.00	58.00	46.00
	ReliefF	1,2,4,5,7	5	71	74.00	54.00	66.00	64.00
	mRMR	1,2,4,5,7	5	71	66.00	**60.00**	66.00	66.00
	$M_d(\lambda = 0)$	1,2,4,5,7	5	71	68.00	58.00	**72.00**	60.00
	$M_d(\lambda = 1)$	1,2,4,5,7	5	71	74.00	56.00	62.00	66.00
	SVM-RFE	2,3,5,7	6	57	44.00	36.00	38.00	42.00
	SVM-RFE-G	1,3,5	7	42	26.00	34.00	30.00	40.00
	FS-P	1-6	4	**85**	60.00	46.00	48.00	58.00
	Wrapper SVM	1,2,4	9	42	72.00	56.00	56.00	**78.00**
	Wrapper C4.5	1,2,4	3	43	**76.00**	58.00	56.00	66.00
15	CFS	1,7	0	29	28.00	28.00	32.00	36.00
	Consistency	1,7	0	29	28.00	28.00	32.00	36.00
	INTERACT	1,7	0	29	28.00	28.00	32.00	36.00
	InfoGain	1,7	0	29	28.00	28.00	32.00	36.00
	ReliefF	1,2,4,5,7	5	**71**	54.00	**50.00**	**54.00**	**64.00**
	mRMR	1,2,4,5,7	5	**71**	54.00	**50.00**	**54.00**	**64.00**
	$M_d(\lambda = 0)$	1,2,4,5,7	5	**71**	54.00	**50.00**	**54.00**	**64.00**
	$M_d(\lambda = 1)$	1,2,4,5,7	5	**71**	**58.00**	**50.00**	52.00	56.00
	SVM-RFE	3,5,7	7	42	30.00	20.00	16.00	26.00
	SVM-RFE-G	1,5	8	28	16.00	24.00	12.00	16.00
	FS-P	1,3,5,6,7	5	**71**	30.00	28.00	22.00	26.00
	Wrapper SVM	1,2,6	5	42	50.00	**50.00**	42.00	**64.00**
	Wrapper C4.5	1,2,5,7	2	57	**58.00**	**50.00**	46.00	52.00
20	CFS	1	0	14	28.00	20.00	28.00	28.00
	Consistency	1	0	14	28.00	20.00	28.00	28.00
	INTERACT	1	0	14	28.00	20.00	28.00	28.00
	InfoGain	1	0	14	28.00	20.00	28.00	28.00
	ReliefF	1,2,5,7	6	57	30.00	**38.00**	**44.00**	44.00
	mRMR	1,2,5,7	6	57	34.00	**38.00**	42.00	**48.00**
	$M_d(\lambda = 0)$	1,2,5,7	6	57	32.00	**38.00**	38.00	32.00
	$M_d(\lambda = 1)$	1,2,5,7	6	57	38.00	32.00	28.00	34.00
	SVM-RFE	–	10	-1	8.00	26.00	20.00	20.00
	SVM-RFE-G	1,2,3,5	6	57	32.00	32.00	14.00	26.00
	FS-P	1-3,5,6	5	**71**	18.00	24.00	24.00	20.00
	Wrapper SVM	1	3	14	36.00	**38.00**	28.00	44.00
	Wrapper C4.5	1,5	4	28	**44.00**	32.00	28.00	32.00

3.3.4 Dealing with noise in the target: Monk3

In this subsection, the Monk3 problem, which includes a 5% of misclassifications, i.e. noise in the target, will be tested. The relevant features are x_2, x_4 and x_5. However, as it was stated by Kohavi and John (1997), for a feature selection algorithm it is easy to find the variables x_2 and x_5, which together yield the second conjunction in the expression seen in Section I.2.1.5 (Appendix I). According to the experimental results presented by Kohavi and John (1997), selecting those features can lead to a better performance than selecting the three relevant ones. This additional information can help to explain the fact that in Table 3.7 several algorithms selected only two of the relevant features.

Table 3.7: Results for Monk3. Relevant features: 2,4,5.

Method	Relevant	Irr. No.	suc.	Accuracy (%)			
				C4.5	NB	k-NN	SVM
CFS	2,5	0	67	**93.44**	88.52	89.34	79.51
Consistency	2,5	0	67	**93.44**	88.52	89.34	79.51
INTERACT	2,5	0	67	**93.44**	88.52	89.34	79.51
InfoGain	2,5	0	67	**93.44**	88.52	89.34	79.51
ReliefF	2,5,4	0	**100**	**93.44**	88.52	**90.98**	80.33
mRMR	2,5	3	17	92.62	88.52	80.33	78.69
$M_d(\lambda = 0)$	2,4,5	2	67	**93.44**	88.52	84.43	81.97
$M_d(\lambda = 1)$	2,4,5	2	67	**93.44**	88.52	84.43	81.97
SVM-RFE	2,4,5	2	67	**93.44**	88.52	84.43	**84.43**
SVM-RFE-G	2,4,5	2	67	**93.44**	88.52	84.43	**84.43**
FS-P	2,4,5	2	67	**93.44**	88.52	84.43	**84.43**
Wrapper SVM	2,4,5	1	83	**93.44**	**89.34**	82.79	79.51
Wrapper C4.5	2,5	0	67	**93.44**	88.52	89.34	79.51

Studying the index of success in Table 3.7, one can see that only ReliefF achieved a value of 100. The worst behavior was showed by mRMR, since it selected the three irrelevant features. As was justified above, many methods selected only two of the relevant features and it can be considered a good comportment. For k-NN classifier, the best accuracy corresponds to ReliefF, which also obtained the best result in terms of index of success. Notice that C4.5 classifier achieves a very high classification accuracy (93.44%), especially bearing in mind that 5% of the samples are missclassifications and cannot be correctly identified.

3.3.4.1 Dealing with a small ratio samples/features: SD1, SD2 and SD3

These synthetic datasets (see Section I.2.1.6 in Appendix I) have a small ratio between number of samples and features, which makes difficult the task of feature selection. This is the problematic present in microarray data, a hard challenge for machine learning researchers. Besides these particularities of the data, there is a high number of irrelevant features for the task of gene classification and also the presence of redundant variables is a critical issue.

Table 3.8: Features selected by each algorithm on synthetic dataset SD1

	(#)	OPT(2)	Red	Irr	suc	Accuracy (%)			
						C4.5	NB	k-NN	SVM
CFS	28	2	1	25	**100**	57.33	82.67	69.33	77.33
Cons	8	2	0	6	**100**	54.67	76.00	60.00	66.67
INT	23	2	0	21	**100**	60.00	81.33	66.67	80.00
IG	42	2	15	25	**100**	58.67	72.00	70.67	78.67
ReliefF[1]	2	1	1	0	50	40.00	45.33	44.00	46.67
ReliefF[2]	20	2	13	5	**100**	60.00	61.33	70.67	73.33
mRMR[1]	2	1	0	1	50	41.33	49.33	34.67	50.67
mRMR[2]	20	1	0	19	50	54.67	82.67	68.00	78.67
SVM-RFE[1]	2	2	0	0	**100**	56.00	60.00	52.00	57.33
SVM-RFE[2]	20	2	3	15	**100**	46.67	**88.00**	**76.00**	92.00
FS-P[1]	2	0	0	2	0	37.33	49.33	41.33	50.67
FS-P[2]	20	1	2	17	50	53.33	76.00	65.33	73.33
$\mathcal{M}_d(\lambda = 0)^1$	2	1	1	0	50	41.33	48.00	32.00	44.00
$\mathcal{M}_d(\lambda = 0)^2$	20	2	17	1	**100**	56.00	62.67	46.67	66.67
$\mathcal{M}_d(\lambda = 1)^1$	2	1	0	1	50	48.00	61.33	58.67	57.33
$\mathcal{M}_d(\lambda = 1)^2$	20	2	17	1	**100**	54.67	62.67	46.67	66.67
W-SVM	19	1	0	18	50	44.00	74.67	58.67	**94.67**
W-C4.5	10	0	0	10	0	**77.33**	38.67	40.00	38.67

[1] Selecting the optimal number of features.

[2] Selecting 20 features.

For these datasets, besides of using the index of success and classification accuracy, we will use the measures employed in (Zhu, Ong, & Zurada, 2010), which are more

specific for this problem. Hence, the performance of SD1, SD2 and SD3 will be also evaluated in terms of:

- **(#)**: number of selected features.
- **OPT(x)**: number of selected features within the optimal subset, where x indicates the optimal number of features.
- **Red**: number of redundant features.
- **Irr**: number of irrelevant features.

Table 3.9: Features selected by each algorithm on synthetic dataset SD2

	(#)	OPT(4)	Red	Irr	suc	Accuracy (%)			
						C4.5	NB	k-NN	SVM
CFS	21	4	0	17	**100**	64.00	**84.00**	72.00	81.33
Cons	9	4	0	5	**100**	54.67	70.67	60.00	72.00
INT	20	3	0	17	75	70.67	80.00	**74.67**	81.33
IG	40	4	19	17	**100**	61.33	69.33	61.33	76.00
ReliefF[1]	4	0	0	4	0	48.00	64.00	50.67	52.00
ReliefF[2]	20	1	9	10	25	54.67	60.00	61.33	70.67
mRMR[1]	4	1	0	3	25	54.67	64.00	60.00	57.33
mRMR[2]	20	1	0	19	25	60.00	70.67	44.00	68.00
SVM-RFE[1]	4	3	1	0	75	46.67	62.67	54.67	65.33
SVM-RFE[2]	20	4	4	12	**100**	57.33	82.67	69.33	**84.00**
FS-P[1]	4	0	0	20	0	42.67	54.67	40.00	57.33
FS-P[2]	20	0	0	20	0	52.00	68.00	42.67	61.33
$\mathcal{M}_d(\lambda = 0)^1$	4	2	2	0	50	56.00	56.00	26.67	50.67
$\mathcal{M}_d(\lambda = 0)^2$	20	4	16	0	**100**	54.67	64.00	49.33	68.00
$\mathcal{M}_d(\lambda = 1)^1$	4	1	0	3	25	46.67	69.33	56.00	69.33
$\mathcal{M}_d(\lambda = 1)^2$	20	1	9	10	25	52.00	62.67	60.00	74.67
W-SVM	13	1	0	12	25	44.00	60.00	45.33	77.33
W-C4.5	6	1	0	5	25	**72.00**	46.67	34.67	42.67

[1] Selecting the optimal number of features.

[2] Selecting 20 features.

For the ranker methods ReliefF, mRMR, $\mathcal{M}_d$, SVM-RFE and FS-P, two different cardinalities were tested: the optimal number of features and 20, since the subset

methods have a similar cardinality. It has to be noted that in this problem and for the calculation of the index of success, redundant features are treated the same as irrelevant features in equation (3.1). Notice that the index of success is 100 even with 25 irrelevant features selected, due to the high number of irrelevant features (4000).

Table 3.10: Features selected by each algorithm on synthetic dataset SD3

	(#)	OPT(6)	Red	Irr	suc	Accuracy (%)			
						C4.5	NB	k-NN	SVM
CFS	23	4	2	17	**67**	64.00	80.00	**73.33**	70.67
Cons	9	3	0	6	50	58.67	76.00	62.67	76.00
INT	19	4	1	14	**67**	61.33	82.67	70.67	66.67
IG	49	4	31	14	**67**	62.67	65.33	65.33	73.33
ReliefF[1]	6	1	5	0	17	50.67	57.33	45.33	53.33
ReliefF[2]	20	1	9	10	17	56.00	69.33	61.33	68.00
mRMR[1]	6	1	0	5	17	62.67	62.67	66.67	65.33
mRMR[2]	20	1	0	19	17	50.67	77.33	52.00	66.67
SVM-RFE[1]	6	3	0	3	50	56.00	70.67	61.33	65.33
SVM-RFE[2]	20	4	2	14	**67**	49.33	**85.33**	70.67	**82.67**
FS-P[1]	6	0	0	6	0	36.00	54.67	34.67	46.67
FS-P[2]	20	1	0	19	17	38.67	61.33	45.33	56.00
$\mathcal{M}_d(\lambda = 0)^1$	6	1	5	0	17	52.00	58.67	40.00	54.67
$\mathcal{M}_d(\lambda = 0)^2$	20	3	15	2	50	45.33	57.33	50.67	54.67
$\mathcal{M}_d(\lambda = 1)^1$	6	1	5	0	17	52.00	58.67	42.67	53.33
$\mathcal{M}_d(\lambda = 1)^2$	20	1	9	10	17	54.67	66.67	60.00	70.67
W-SVM	10	1	0	9	17	48.00	61.33	61.33	81.33
W-C4.5	5	1	0	4	17	**68.00**	50.67	37.33	48.00

[1] Selecting the optimal number of features.

[2] Selecting 20 features.

Tables 3.8, 3.9 and 3.10 show the results for the datasets SD1, SD2 and SD3, respectively. Regarding the selected features, the subset filters and InfoGain (which exhibits a similar behavior) showed excellent results, in all SD1, SD2 and SD3. Also SVM-RFE obtained good results, although the version with a Gaussian kernel could not been applied on these datasets due to memory complexity. With respect to the classifiers, SVM achieves the highest accuracies.

3.3.5 Dealing with a complex dataset: Madelon

Madelon (see Section I.2.1.7 in Appendix I) is a very complex artificial dataset which is distorted by adding noise, flipping labels, shifting and rescaling. It is also a non-linear problem, so it conforms a challenge for feature selection researchers. The desired behavior for a feature selection method is to select the relevant features (1-5) and discard the redundant and irrelevant ones.

Table 3.11 shows the relevant features selected by the feature selection methods, as well as the number of redundant and irrelevant features elected by them and the classification accuracy. Notice that for the calculation of index of success, the redundant attributes selected stand for irrelevant features. Again the results for SVM and naive Bayes will not be analyzed, since they are linear classifiers. The best result in terms of index of success was obtained by the wrapper with C4.5, selecting all the 5 relevant features, which also led to the best classification accuracy for C4.5.

Table 3.11: Results for Madelon. Relevant features: 1-5.

Method	Relevant	Red. No.	Irr. No.	suc.	Accuracy (%)			
					C4.5	NB	k-NN	SVM
CFS	3	7	0	20	80.92	69.58	86.83	66.08
Consistency	3,4	10	0	40	83.54	69.67	**90.83**	66.83
INTERACT	3,4	10	0	40	83.54	69.67	**90.83**	66.83
InfoGain	3,4	10	0	40	83.54	69.67	**90.83**	66.83
ReliefF	1,3,4,5	11	0	80	84.21	69.83	89.88	66.46
mRMR	–	1	14	0	64.92	62.25	53.13	57.08
$M_d(\lambda = 0)$	3,4	10	2	40	83.23	**70.21**	85.29	66.42
$M_d(\lambda = 1)$	3,4	10	2	40	83.23	**70.21**	85.29	66.42
SVM-RFE	1,3,4,5	4	7	80	86.42	66.88	81.25	67.42
FS-P	3,4	3	10	40	70.50	66.17	62.54	66.96
Wrapper SVM	3	0	16	20	66.63	66.04	54.08	**67.54**
Wrapper C4.5	1-5	5	15	**99**	**87.04**	70.00	75.42	66.33

3.4 Cases of study

After presenting the experimental results, and before discussing and analyzing them in detail, we will describe several cases of study in order to decide among similar methods. These cases of study will be based on the index of success, to make this analysis classifier-independent.

3.4.1 Case of study I: different kernels for SVM-RFE

Two different kernels were applied on the embedded method SVM-RFE. The Gaussian kernel allows to solve non-linear problems, but at the expense of being more computationally demanding. In fact, SVM-RFE-G could not be applied on the datasets SD and Madelon, due to the space complexity. In Figure 3.2 one can see a comparison of these two versions of the method. Note that as for both Led-25 and Led-100 datasets there are results for 6 different levels of noise in the inputs, we have opted for computing the average of the index of success.

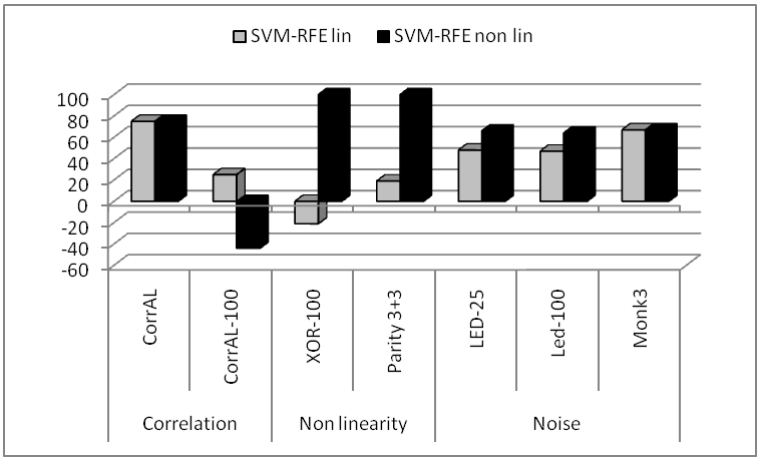


Figure 3.2: SVM-RFE: Linear vs Gaussian kernel. The vertical axis represents the index of success.

As expected, the linear kernel is not able to deal with non-linear problems (XOR-100 and Parity3+3). On the other hand, the Gaussian kernel achieves a poor result over Corral-100 dataset (where the number of irrelevant features increases considerably). In the remaining datasets, the Gaussian kernel maintains or increases the performance of the linear kernel. In these cases, it is necessary to bear in mind that the Gaussian kernel

raises the computational time requirements of the algorithms and it cannot be applied over high-dimensional datasets (such as Madelon and the SD family). For example, over XOR-100 dataset, the time used by the Gaussian kernel quadruplicates that of the linear one. We suggest to use the Gaussian kernel when there is some knowledge about the non-linearity of the problem, and to use the linear kernel in the remaining cases, specially when dealing with large amounts of data.

3.4.2 Case of study II: mRMR *vs* $\mathcal{M}_d$

The filter method $\mathcal{M}_d$ is an extension of mRMR which instead of mutual information, uses a measure of dependence to assess relevance and irrelevance. Besides, it included a free parameter (λ) that controls the relative emphasis given on relevance and irrelevance. In light of the above, the authors think that it is interesting to compare the behaviors showed by these two methods over the artificial datasets studied in this work. Two values of lambda were tested, 0 and 1, and it is also important to see the difference between them. When λ is equal to zero, the effect of the redundancy disappears and the measure is based only on maximizing the relevance. On the other hand, when λ is equal to one, it is more important to minimize the redundancy. For the sake of fairness, note that for the SD family of datasets, we considered the results achieved selecting 20 features.

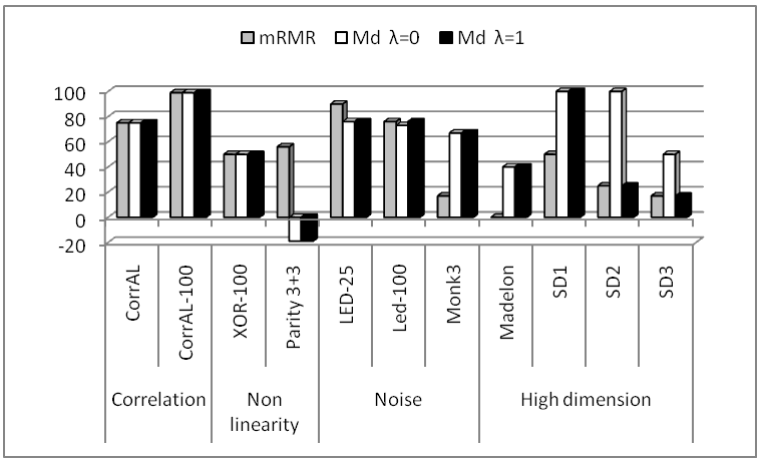


Figure 3.3: mRMR *vs* $\mathcal{M}_d$. The vertical axis represents the index of success.

With regard to the different values of λ, one can se in Figure 3.3 that the index of succes is the same for most of the datasets tested (8 out of 11). However, there is a important improvement in SD2 and SD3 when the value of λ is zero. Therefore, using

this value of λ is recommended, although the appropriate value of λ is not an easy-to-solve question that requires to be studied further and seems to be very dependent of the nature of data.

Comparing $\mathcal{M}_d$ and mRMR, the latter performs better in two datasets (Parity3+3 and Led25) whereas $\mathcal{M}_d$ is better in 5 datasets (Monk3, Madelon, and the SD family). In the remaining datasets, the index of success achieved by both methods is the same. In light of these results, the use of $\mathcal{M}_d$ is recommended, except in datasets with high non-linearity.

3.4.3 Case of study III: subset filters

Subset evaluation produces candidate feature subsets based on a certain search strategy. Each candidate subset is evaluated by a certain evaluation measure and compared with the previous best one with respect to this measure. This approach can handle feature redundancy together with feature relevance, besides of releasing the user from the task of choosing how many features to retain.

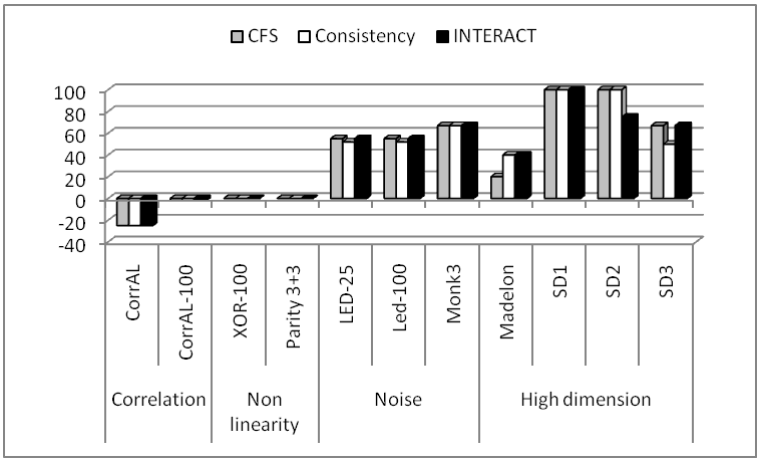


Figure 3.4: Subset filters. The vertical axis represents the index of success.

In Figure 3.4 one can see a comparison among the three subset filters studied in this work (CFS, INTERACT and Consistency-based) with regard to the index of success. All the three methods show in general a very similar behavior, although some differences have been found. Consistency-based is slightly worse on datasets which present noise (Led25, Led100). This can be explained because for this filter, a pattern is considered inconsistent if there exists at least two instances such that they match all but their class

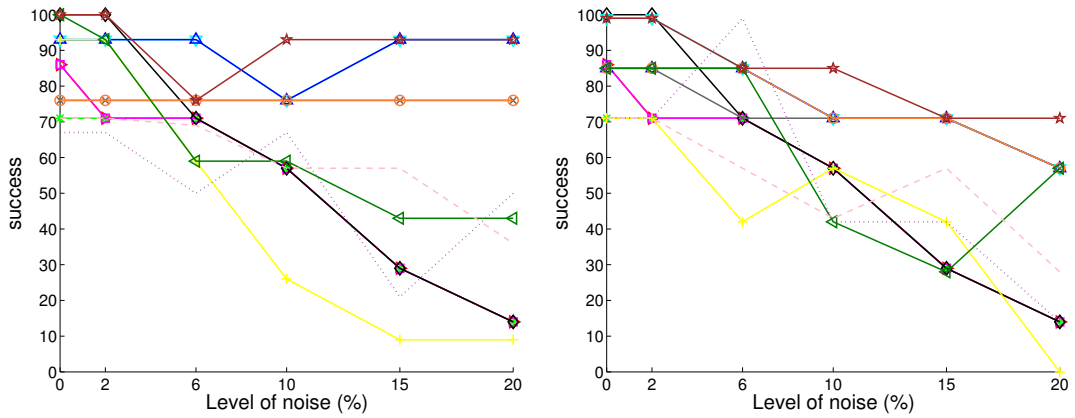
labels, and therefore the given subset of features is inconsistent and the features are discarded. This case can happen when the data has been distorted with noise. On the other hand, CFS decays on Madelon. This method aims to maximize the correlation of the selected features with the class and to minimize the intercorrelation among the selected features. However, this algorithm cannot identify really strong interactions as the ones which may appear in parity problems (remind that Madelon is a generalization of a parity problem). In light of the above, it is recommended to use INTERACT.

3.4.4 Case of study IV: different levels of noise in the input

Figure 3.5 shows an overview of the behavior of feature selection methods with regard to different levels of noise, according to the index of success described in (3.1). As we would have expected, in general the index of success decreases when the level of noise increases, and worse performances were obtained over Led-100 due to the higher number of irrelevant features. It may seem strange that in some cases the index of success improves with higher levels of noise (for example, in Led-100 from 10% to 15% of noise), but this fact can be explained by the random generation of the noise. Notice that the influence of each relevant feature is not the same in this problem, so adding noise to one or another may cause different results. In fact, the first five features (see Figure I.1 in Appendix I) are enough to distinguish among the ten digits, therefore if these attributes are distorted, the result may be altered.

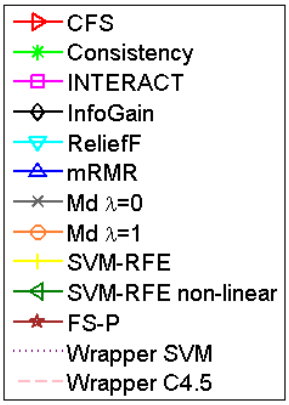
Several conclusions can be extracted from the graphs in Figure 3.5. Regarding the wrapper model, both versions tested degrade their results with the presence of noise, both in Led-25 and Led-100. With respect to embedded methods, two opposite behaviors have been observed. On the one hand, FS-P achieved very promising results on both datasets, without showing degradation as the level of noise increases. In fact, the index of success oscillates between 76 and 100 on Led-25 –subfigure 3.5a– and between 71 and 99 on Led-100 –subfigure 3.5b–. On the other hand, SVM-RFE (specially the version with the linear kernel) deteriorates considerably its behavior with high levels of noise. Note that SVM-RFE with linear kernel obtained 9 as index of success on Led-25 and -1 on Led-100, which is the worst result for all the feature selection methods tested.

Concerning the filter model, mRMR and ReliefF are the methods that achieve the best indices of success, being ReliefF slightly better in two cases (Led-100 with 0% and



(a) Led-25

(b) Led-100



(c) Legend

Figure 3.5: Results for Led-25 and Led-100

2% of noise). These two filters obtain very good results without being very affected by noise. On the contrary, the subset filters (CFS, Consistency and INTERACT) and Information Gain are affected by high levels of noise although they are robust to the addition of irrelevant features. Finally, with respect to $\mathcal{M}_d$, it attains constant results, particularly on Led-25, and no significant differences have been found between the two values of λ tested.

It is curious the opposite behaviors of Information Gain and mRMR, bearing in mind that both come from the Information Theory field. However, this fact can be explained because Information Gain is a univariate measure that considers the entropy between a given feature and the class level. On the other hand, mRMR takes into account the mutual information among features. The latter is a multivariate measure and therefore a better behavior is expected when noise is present in data, because although some

features may be affected by noise in a sample, not all of them are supposed to suffer it. This is why Information Gain obtains excellent results with low levels of noise but as this increases, its performance decays until reaching an index of success with value 14.

To sum up, the filters mRMR and ReliefF and the embedded method FS-P are the methods most tolerant to noise in the inputs and the subsets filters (CFS, Consistency and INTERACT) and Information Gain are the most affected by noise.

3.5 Analysis and Discussion

In this section an analysis and discussion of the results presented in Section 3.3 will be carried out, trying to check which method is the best and to explain some behaviors showed in the experimental results. We will start by analyzing the index of success and then we will discuss the relation between index of success and classification accuracy focusing on the specific problems studied in this work.

3.5.1 Analysis of index of success

Table 3.12 shows the average of success for each feature selection method over each scenario and also an overall average for each method (last column). For Led25 and Led100 only one result is presented, respectively, corresponding to the average of the results for the distinct levels of noise tested. Analogously, for the SD family of datasets, only the average result of the 3 datasets is shown.

We are interested in an analysis of the index of success (regardless of the classification accuracy) in order to check the behavior of the feature selection methods in a classifier-independent manner. In light of the results shown in Table 3.12, the best method according to the index of success is the filter ReliefF, followed by the filters mRMR and both versions of $\mathcal{M}_d$. However, the subset filters and Information Gain showed poor results. Regarding the embedded model, FS-P is slightly better than SVM-RFE, and both of them are in the middle of the ranking. Finally, wrapper methods turned out to be the worst option in this study, since they achieved the poorest averages of success.

Table 3.12: Average of success for every feature selection method tested. "N/A" stands for "Not Applicable".

	Correlation		Non-linear		Noise			High dimension		
Method	Corr.	Corr100	XOR100	Par3+3	Led25	Led100	Monk3	SD	Mad.	Av.
CFS	-25	-2	0	0	55	55	67	89	20	29
Consistency	-25	-2	0	0	52	52	67	83	40	30
INTERACT	-25	-2	0	0	55	55	67	81	40	30
InfoGain	-25	-2	0	0	62	62	67	89	40	33
ReliefF	75	75	100	93	90	80	100	47	80	82
mRMR	75	99	50	56	90	76	17	31	0	55
M_d[1]	75	99	50	-19	76	73	67	83	40	60
M_d[2]	75	99	50	-19	76	76	67	47	40	57
SVM-RFE	75	25	-21	19	48	47	67	89	80	48
SVM-RFE-G	75	-44	100	100	66	64	67	N/A	N/A	N/A
FS-P	100	75	-19	-19	93	85	67	22	40	49
Wrap. SVM	-25	-2	-4	-4	54	57	83	31	20	23
Wrap. C4.5	-25	-13	-4	-4	60	55	67	22	99	29

[1] $\lambda = 0$

[2] $\lambda = 1$

In light of the results presented in Table 3.12, some guidelines are suggested:

- In complete ignorance of the particulars of data, the authors suggest to use the filter ReliefF. It detects relevance in a satisfactory manner, even in complex datasets such as XOR-100, and it is tolerant to noise (both in the inputs and in the output). Moreover, due to the fact that it is a filter, it has the implicit advantage of its low computational cost.
- When dealing with high non-linearity in data (such as XOR-100 and Parity3+3), SVM-RFE with a Gaussian kernel is an excellent choice, since it is able to solve these complex problems. However, at the expense of being computationally more expensive than the remaining approaches seen in these experiments.
- In the presence of altered inputs, the best option is to use the embedded method FS-P, since it has proved to be very robust to noise. A less expensive alternative is the use of the filters ReliefF or mRMR, which also shown good behaviors over this scenario. With low levels of noise (up to 6%), the use of the filter Information Gain is also recommended.
- When the goal is to select the smallest number of irrelevant features (even at the expense of selecting fewer relevant features), we suggest to employ one of the subset filters (CFS, Consistency-based or INTERACT). This kind of methods

have the advantage of releasing the user from the task of deciding how many features to choose.

- When dealing with datasets with a small ratio between number of samples and features and a high number of irrelevant attributes, which is part of the problematics of microarray data, the subset filters and Information Gain presented a promising behavior. SVM-RFE performs also adequately, but because of being an embedded method is computationally expensive, especially in high-dimensional datasets like these.

- In general, the authors suggest the use of filters (specifically ReliefF), since they carry out the feature selection process with independence of the induction algorithm and are faster than embedded and wrapper methods. However, in case of using another approach, we suggest to use the embedded method FS-P.

As was stated in Chapter 2, filters are the methods with the lower computational cost whilst wrappers are computationally the most expensive. To illustrate this fact, in Figure 3.6 one can see the execution time of the different feature selection methods over two datasets: XOR-100 and Led-100 (with no noise). In order to be fair, mRMR was not included in this study because it was executed in a different machine, however one can expect a computational time similar to the one required by $\mathcal{M}_d$.

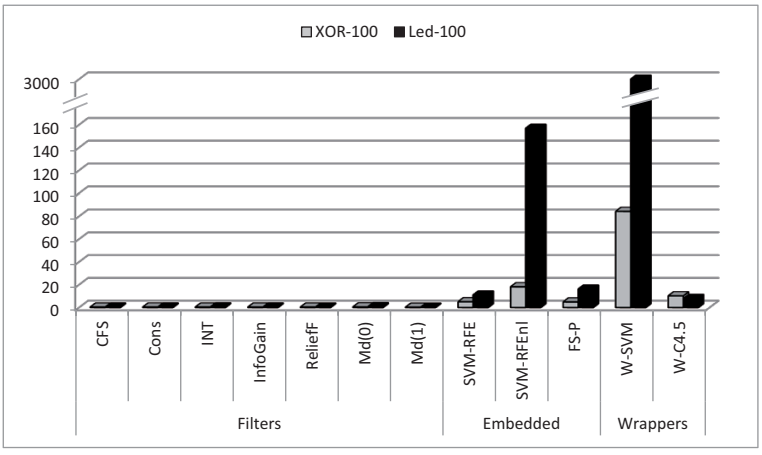


Figure 3.6: Time in seconds for the datasets XOR-100 and Led-100.

As expected, the filter model achieves the lowest execution times, always below 1 second. The embedded methods require more computational time, specially SVM-RFE with a Gaussian kernel. The time required by this method over Led-100 is especially high because this is a multiclass dataset, a fact that also increases the computational

time. Wrapper SVM (which uses a linear kernel), contrary to Wrapper C4.5 is a very demanding method, particularly with Led-100, which needed almost 1 hour.

3.5.2 Analysis of classification accuracy

Although the previous analysis is interesting because it is not classifier-dependent, one may want to see the classification accuracy in order to check if the desired subset of features is unique and if it is the best option. For the sake of brevity, only the two extreme values of noise for Led-25 and Led-100 datasets were included in this study. Table 3.13 shows, for each classifier and dataset, the best accuracy obtained, as well as the corresponding index of success and feature selection method employed. In this manner, it is easy to see at a glance if the best accuracy matches with the best index of success. In fact, this happens for all datasets except Led-100 with 20% of noise, where the inputs are clearly disturbed. This may be explained because the irrelevant features (randomly generated) are adding some information useful to the classifier, whilst the disturbed relevant features are not so informative.

The k-NN classifier, based on nearest neighbors, seems to be the best match for the proposed index of success, since it obtains the best result for classification when the index obtains also its best result, specifically in 5 out of 13 datasets tested. Instance-based learners are very susceptible to irrelevant features, therefore when a feature selection method only selects the relevant features, its index of success is high and also the classification accuracy obtained by this classifier. It has to be also noted that k-NN is a non-linear classifier therefore it is capable to correctly classify problems such as XOR-100 or Parity3+3, achieving 100% of classification accuracy when other methods like SVM obtained poor results.

SVM obtained the highest classification accuracy in 7 out of 13 datasets showed in Table 3.13, however it only coincides with the highest index of success in SD2 dataset. This predictor takes advantage of the embedded method SVM-RFE and Wrapper SVM, both methods using this classifier performance to select the features. In fact, the highest accuracies were obtained after applying one of those methods for all datasets except for Led-25 with 20% of noise.

Although the behavior of the classifiers is interesting, one may want to focus on the problems studied in this work. For dealing with correlation and redundancy, two datasets were evaluated in this chapter: CorrAL and CorrAL-100 (see Tables 3.1 and

3.2). Focusing on Corral, the subset filters, InfoGain and the wrappers selected only the correlated feature, which leads to an accuracy of 75% for all the classifiers except k-NN, which apparently is not able to take advantage of the relation between this feature and the class. When the four relevant features plus the correlated one are selected (rows 6-10 in Tables 3.1 and 3.2), one can see that the correlated feature (since it is correlated for 75% of data) is hindering the process of classification, preventing the predictors to correctly classify all samples. FS-P was the only method that selected the four relevant features and discarded the irrelevant and correlated ones; k-NN was able to achieve a 100% of classification accuracy, whilst the other methods were not. This fact is explained because of the complexity of the problem that may cause that a given classifier may not solve the problem satisfactorily, even with the proper features. Regarding Corral-100, the highest accuracy (96.88%) was obtained by SVM having only one of the relevant features, the correlated one, and 8 irrelevant ones. This fact can seem surprising but it can be explained because the irrelevant features (randomly generated) are informative in this problem. Classifying only with the relevant features and the correlated one, SVM achieves 65.62% of classification accuracy, therefore it is clear that the irrelevant features are adding some useful information to the learner. In fact, by randomly generating 94 binary features and having only 32 samples, the probability that some of these irrelevant features could be correlated with the class is very high. This situation happens again with Wrapper C4.5 and C4.5 classifier, whilst the remaining methods exhibit a similar behavior to Corral.

Non-linearity is a difficult problem to deal with. In fact, two of the classifiers employed in this work (SVM with linear kernel and naive Bayes) and several feature selection methods do not turn very good results. This problematic is present in XOR-100 and Parity3+3 datasets, in Tables 3.3 and 3.4. As we have said, naive Bayes and SVM cannot deal with non-linearity therefore they will not be the focus in this Section. On the other hand, k-NN (and C4.5 only over XOR-100) achieve 100% of classification accuracy when the desired features are selected. Over XOR-100, C4.5 obtains also 100% of prediction accuracy after applying its own wrapper, even when it selected two extra irrelevant features. It has to be noted that this classifier performs an embedded selection of the features, therefore it may be using a subset of features smaller than the one given by the feature selection method. Finally, it needs to be remarked the improvement in SVM-RFE when using a Gaussian kernel for this kind of datasets. SVM-RFE over XOR-100 did not select any of the relevant features, which led to a classification accuracy below the baseline accuracy. However, the computational time when using a Gaussian kernel is almost four times the one needed using a linear one (see Table 3.6) so when choosing one or the other it is a fact to take into account.

Table 3.13: Summary of results grouped by classifier. 'Rf' stands for 'ReliefF', 'SR' for 'SVM-RFE', 'SRG' for 'SVM-RFE-G' and 'IG' for 'Information Gain', respectively.

	C4.5			NB			k-NN			SVM		
Dataset	Best Acc.	suc.	Method	Best Acc.	suc.	Method	Best Acc.	suc.	Method	Best Acc.	suc.	Method
CorrAL	81.25	**100** 75	FS-P SRG	81.25	75	Rf,mRMR,M_d,SR	**100.00**	**100**	FS-P	87.50	75	Rf,mRMR,M_d,SR
CorrAL-100	84.38	-13	W-C4.5	87.50	75 25	FS-P SR	90.63	**99**	mRMR	**96.88**	25	SR
XOR-100	**100.00**	**100** 99	Rf,SRG W-C4.5	76.00	50	FS-P	**100.00**	**100**	Rf,SRG	78.00	-21	SR
Parity3+3	90.63	**100** 93	SRG Rf	64.06	-4	W-SVM,W-C4.5	**100.00**	**100** 93	SRG Rf	64.06	-4	W-SVM,W-C4.5
Led-25 (0%)	92.00	**100** 93 86 71 67	IG,SRG,FS-P Rf,mRMR,SR CFS,INT Cons,W-C4.5 W-SVM	**100.00**	**100** 86 71	IG,SRG,FS-P CFS,INT Cons	**100.00**	**100** 86 71	IG,SRG,FS-P CFS,INT Cons	**100.00**	67	W-SVM
Led-25 (20%)	48.00	36	W-C4.5	40.00	36	W-C4.5	40.00	**93**	Rf,mRMR	**56.00**	50	W-SVM
Led-100 (0%)	92.00	**100** 99 86 85 71	IG Rf,FS-P CFS,INT mRMR,SRG Cons,W-SVM,W-C4.5	**100.00**	**100** 86 71	IG CFS,INT Cons,W-C4.5	**100.00**	**100** 86 71	IG CFS,INT Cons,W-C4.5	**100.00**	99 71	Rf W-SVM
Led-100 (20%)	44.00	28	W-C4.5	38.00	**57*** 14	Rf,mRMR,$M_d(0)$ W-SVM	44.00	**57***	Rf	**48.00**	**57***	mRMR
Monk3	**93.44**	**100** 83 67	Rf W-SVM Rest except mRMR	89.34	83	W-SVM	90.98	**100**	Rf	84.43	67	SR SRG FS-P
SD1	77.33	0	W-C4.5	88.00	**100**	SRG	76.00	**100**	SRG	**94.67**	50	W-SVM
SD2	72.00	25	W-C4.5	**84.00**	**100**	CFS	74.67	75	INT	**84.00**	**100**	SRG
SD3	68.00	17	W-C4.5	**85.33**	**67**	SRG	73.33	**67**	CFS	82.67	**67**	SRG
Madelon	87.04	**99**	W-C4.5	70.21	40	M_d	**90.83**	40	Cons,INT,IG	67.54	20	W-SVM

* This is not the highest index of success achieved.

Different levels of noise in the input features were tested over Led-25 (Table 3.5) and Led-100 (Table 3.6) datasets. As expected, the classification accuracy decreases when the level of noise increases. It has to be noted that, for both datasets, selecting 5 out of the 7 relevant features is enough to achieve 100% classification accuracy. Because segments 1-5 (see Figure I.1, Appendix I) are enough to distinguish the 10 digits (actually, 5 binary features allow to represent 32 different states). In fact, when the level of noise is 6%, the first four methods miss the third feature (which allows to distinguish between digit 5 and 6) and the performance decays in 24%, that cannot be ascribed to the level of noise. This is a case of classification showing that the true model is not unique. On the other hand, it is curious that in some cases such as ReliefF over Led-25 with 20% of noise, where it achieves an index of success of 93 (selecting the 7 relevant features), the maximum classification accuracy obtained with these features was 40% (SVM) which is not the result expected. This fact can be explained because of the high level of noise, which corrupts the relevant features and makes the classification task very complex. In thoses cases with high levels of noise, wrappers appear to be a good alternative, since they are classifier-dependent and try to search for the best features to the given classifier. To sum up, the filters mRMR and ReliefF and the embedded method FS-P are the methods most tolerant to noise in the inputs and the subsets filters (CFS, Consistency and INTERACT) and Information Gain are the most affected by noise, although wrappers are also a good choice if one is interested in maximizing the classification accuracy.

Monk3 dataset (see Table 3.7) is studied to deal with noise in the target. As was explained in Section 3.3.4, there are evidences that features x_2 and x_5 are enough for certain classifier, which in fact happens in the experiments presented in this work. This is an example of the optimal feature subset being different than the subset of relevant features. On the other hand, again one can see the implicit capacity of C4.5 to select features, since it achieves the same result in cases where different subsets of features were selected, although for mRMR some of the irrelevant features caused the incorrect classification of one extra feature. This is not the case for k-NN classifier, which achieves the highest accuracy only when the "known" set of relevant features is selected. Naive Bayes and SVM seem to be more affected by misclassifications, since they obtain the worst results and do not take advantage of the best indices of success.

SD1, SD2 and SD3 (Tables 3.8, 3.9 and 3.10) introduce the problematic of microarray data: a small ratio between number of samples and features and a high number of redundant and irrelevant features. In general, the classification results are poor, because this kind of problems are very difficult to solve since the classifiers tend to

overfit. Moreover, the accuracy decreases when the complexity of the dataset increases (SD3). The embedded method SVM-RFE achieves very good results, specially with the SVM classifier. CFS and INTERACT filters also work satisfactorily together with naive Bayes and k-NN classifiers. The small ratio between number of samples and features prevents the use of wrappers, which have the risk of overfitting due to the small sample size. In fact, one can see in Tables 3.8, 3.9 and 3.10 that the wrappers obtain high accuracies in conjunction with their corresponding classifiers, but the performance decreases when using other classifiers. Regarding the classifiers, SVM achieves good results, specially over SD1. SVMs have many mathematical properties that make them attractive for gene expression analysis, including their flexibility in choosing a similarity function, sparseness of solution when dealing with large data sets, the ability to handle large feature spaces, and the ability to identify outliers (M. P. Brown et al., 2000). Naive Bayes obtained also high accuracies, specially over SD2 and SD3. This learner is robust with respect to irrelevant features, although it deteriorates quickly by adding redundant features. In fact, it obtains the best accuracies when a small number of redundant features are present.

Madelon (Table 3.11) is a complex dataset which includes noise, flipping labels and non-linearity. Due to the latter, naive Bayes and SVM cannot obtain satisfactory results so they will not be analyzed. C4.5 obtained its highest accuracy after applying its own wrapper, as expected. It is more surprising the behavior of k-NN, which obtained the highest prediction accuracy after applying methods that achieve poor indices of success. However, this fact can be explained because these methods selected a high number of redundant features. These redundant features were built by multiplying the useful features by a random matrix, therefore they are also informative.

Table 3.14 shows the behavior of the different feature selection methods over the different problems studied, where the larger the number of dots, the better the behavior. To decide which methods were the most suitable under a given situation, it was computed a trade-off between index of success and classification accuracy. In light of these results, ReliefF turned out to be the best option independently of the particulars of the data, with the added benefit that it is a filter, which is the model with the lowest computational cost. However, SVM-RFE-G showed outstanding results, although its computational time is in some cases prohibitive (in fact, it could not be applied over some datasets). Wrappers have proven to be an interesting choice in some domains, nevertheless they must be applied together with their own classifiers and it has to be reminded that this is the model with the highest computational cost. In addition to this, Table 3.14 provides some guidelines for specific problems.

Table 3.14: General guidelines for specific problems

Method	Correlation and redundancy	Non Linearity	Noise Inputs	Noise Target	No. feat >> No. samples
CFS	•	•	•	•••	••••
Consistency	•	•	•	•••	••
INTERACT	•	•	•	•••	•••
InfoGain	•	•	•	•••	•••
ReliefF	••••	•••••	•••••	•••••	••
mRMR	••••	•••	•••••	••	•
$M_d(\lambda = 0)$	••••	••	•••	•••	•••
$M_d(\lambda = 1)$	••••	••	•••	•••	•••
SVM-RFE	••••	•	•	••••	•••••
SVM-RFE-G	••••	•••••	•••	••••	–
FS-P	•••••	••	••••	••••	•
Wrapper SVM	•	•	•••	••••	••
Wrapper C4.5	••	•••	•••	•••	•••

3.6 Summary

Feature selection has been an active and fruitful field of research in machine learning. Its importance is beyond doubt and it has proven effective in increasing predictive accuracy and reducing complexity of machine learning models. However, choosing the appropriate feature selection method for a given scenario is not an easy-to-solve question. In this chapter, a review of eleven feature selection methods applied over eleven synthetic datasets was presented aimed at studying their performance with respect to several situations that can hinder the process of feature selection. The suite of synthetic datasets chosen covers phenomena such as presence of irrelevant and redundant features, noise in the data or interaction between attributes. A scenario with a small ratio between number of samples and features where most of the features are irrelevant was also tested. It reflects the problematic of datasets such as microarray data, a well-known and hard challenge in the machine learning field where feature selection becomes indispensable.

Within the feature selection field, three major approaches were evaluated: filters, wrappers and embedded methods. To test the effectiveness of the studied methods, an evaluation measure was introduced trying to reward the selection of the relevant

features and to penalize the inclusion of the irrelevant ones. Besides, four classifiers were selected to measure the effectiveness of the selected features and to check if the true model was also unique.

Besides a detailed analysis of the findings of this chapter, some cases of study were also presented in order to decide among methods that showed similar behaviors and helping to find the adequacy of them in different situations. Finally, some guidelines about the appropriateness of the different feature selection methods in diverse scenarios have been proposed.

So far, this thesis has described the fundamentals and state-of-the-art of feature selection. The next chapters are dedicated to deeply study feature selection in applications such as DNA microarray classification or the medical domain.

CHAPTER 4

Feature selection in DNA microarray classification

As mentioned in previous chapters, feature selection has been widely studied in the past years by the machine learning researchers. This technique has found success in many different real world applications, among them it is worth highlighting DNA microarray analysis, since it is an important representative of high-dimensional data.

During the last two decades, the advent of DNA microarray datasets stimulated a new line of research both in bioinformatics and in machine learning. This type of data is used to collect information from tissue and cell samples regarding gene expression differences that could be useful for diagnosis disease or for distinguishing a specific tumor type. Although there are usually very few samples (often less than 100 patients) for training and testing, the number of features in the raw data ranges from 6000 to 60000, since it measures the gene expression en masse. A typical classification task is to separate healthy patients from cancer patients based on their gene expression "profile" (binary approach). There are also datasets where the goal is to distinguish among different types of tumors (multiclass approach), making the task even more complicated.

Therefore, microarray data pose a great challenge for machine learning researchers. Having so many fields relative to so few samples, create a high likelihood of finding "false positives" that are due to chance (both in finding relevant genes and in building predictive models) (Piatetsky-Shapiro & Tamayo, 2003). It becomes necessary to find robust methods to validate the models and assess their likelihood. Furthermore, additional experimental complications like noise and variability render the analysis of microarray data an exciting domain (Saeys, Inza, & Larrañaga, 2007).

Several studies have shown that most genes measured in a DNA microarray experiment are not relevant for an accurate classification among different classes of the problem (Golub et al., 1999). To avoid the "curse of dimensionality" (Jain & Zongker,

1997), feature (gene) selection plays a crucial role in DNA microarray analysis. It soon became indispensable among the researchers, not only to remove redundant and irrelevant features, but also to help biologists to identify the underlying mechanism that relates gene expression to diseases. This research area has received significant attention in recent years (most of the work has been published in the last decade), and new algorithms emerge as alternatives to the existing ones. However, when a new method is proposed, there is a lack of standard state-of-the-art results to perform a fair comparative study. Besides, there is a broad suite of microarray datasets to be used in the experiments, some of them even are named the same, but the number of samples or characteristics are different in different studies, which makes this task more complicated.

The main goal of this chapter is to provide a framework for ongoing studies, paying attention to the datasets used in the experimental analysis, the intrinsic data characteristics and the experimental analysis of classical feature selection algorithms available in data mining software tools used for microarray data. In this manner, it is possible to be aware of the particularities of this type of data as well as its problematics, such as the imbalance of the data, their complexity, the presence of overlapping and outliers, or the so-called dataset shift. These problematics render the analysis of microarray data an exciting domain.

An experimental framework has been designed in such a way that well-founded conclusions can be extracted. A set of nine binary microarray datasets is used, which suffer from problems such as class imbalance, overlapping or dataset shift. Some of these datasets come originally divided into training and test datasets, so a holdout validation is performed on them. For the remaining datasets, they will be evaluated with a k-fold cross-validation, since it is a common choice in the literature. However, it has been shown that cross-validation can potentially introduce dataset shift, so another strategy has been included to create the partitioning, called *Distribution optimally balanced stratified cross-validation* (DOB-SCV) (Moreno-Torres, Sáez, & Herrera, 2012). C4.5, Support Vector Machine (SVM) and naive Bayes were selected as classifiers, and we use classification accuracy, sensitivity and specificity on the test partitions as the evaluation criteria.

4.1 Background: the problem and first attempts

All cells have a nucleus, and inside nucleus there is DNA, which encodes the "program" for future organisms. DNA has coding and non-coding segments. The coding segments, also known as genes, specify the structure of proteins, which do the essential work in every organism. Genes make proteins in two steps: DNA is transcribed into mRNA and then mRNA is translated into proteins. Advances in molecular genetics technologies, such as DNA microarrays, allow us to obtain a global view of the cell, where it is possible to measure the simultaneous expression of tens of thousands of genes (Piatetsky-Shapiro & Tamayo, 2003). Figure 4.1 displays the general process of acquiring the gene expression data from a DNA microarray. These gene expression profiles can be used as inputs to large-scale data analysis, for example, to increase our understanding of normal and disease states.

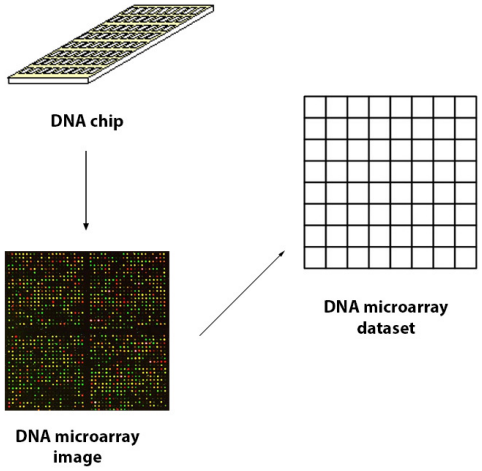


Figure 4.1: General process of acquiring the gene expression data from DNA microarray

Microarray datasets began to be dealt with at the end of the nineties. Soon feature (gene) selection was considered a *de facto* standard in this field. Further work was carried out in the beginning of the 2000s (Saeys et al., 2007). The univariate paradigm, which is fast and scalable but ignores feature dependencies, has dominated the field during the 2000s (Dudoit, Fridlyand, & Speed, 2002; Li et al., 2004; J. Lee, Lee, Park, & Song, 2005). However, there were also attempts to tackle microarray data with multivariate methods, which are able to model feature dependencies, but at the cost of being slower and less scalable than univariate techniques. Besides of applying

multivariate filter methods (Ding & Peng, 2005; Yeung & Bumgarner, 2003; I. Wang Y.and Tetko et al., 2005; Gevaert, Smet, Timmerman, Moreau, & Moor, 2006), the microarray problem was also treated with more complex techniques such as wrappers and embedded methods (Blanco, Larrañaga, Inza, & Sierra, 2004; Jirapech-Umpai & Aitken, 2005; Inza, Larrañaga, Blanco, & Cerrolaza, 2004; Ruiz, Riquelme, & Aguilar-Ruiz, 2006).

4.2 Intrinsic characteristics of microarray data

As mentioned at the beginning of this chapter, microarray data classification posed a great challenge for computational techniques, because of their large dimensionality (up to several tens of thousands of genes) while small sample sizes. Furthermore, there exist additional experimental complications that render the analysis of microarray data an exciting domain.

4.2.1 Small sample size

The first problem that one may find when dealing with microarray data is related with the small samples size (usually less than 100). A key point in this situation is that error estimation is greatly impacted by small samples (E. Dougherty, 2001). Without the appropriate estimation of the error, there exists an unsound application of classification methods, which has generated a large number of publications and an equally large amount of unsubstantiated scientific hypotheses (Braga-Neto, 2007). For example, Michiels, Koscielny, and Hill (2005) reported that reanalysis of data from the seven largest published microarray-based studies that have attempted to predict prognosis of cancer patients reveals that five out of those seven did not classify patients better than by chance. To overcome this problem, it becomes necessary to select a correct validation method for estimating the classification error.

4.2.2 Class imbalance

A common problem in microarray data is the so-called *class imbalance problem*. This occurs when a dataset is dominated by a major class or classes which have significantly

more instances than the other rare/minority classes in the data (He & Garcia, 2009; Sun, Wong, & Kamel, 2009; López, Fernández, García, Palade, & Herrera, 2013). Typically, people have more interests in learning rare classes. For example, in the domain at hand, the cancer class tends to be rarer than the non-cancer class because usually there are more healthy patients. However, it is important for practitioners to predict and prevent the apparition of cancer. In these cases, standard classifier learning algorithms have a bias toward the classes with greater number of instances, since rules that correctly predict those instances are positively weighted in favor of the accuracy metric, whereas specific rules that predict examples from the minority class are usually ignored (treated as noise), because more general rules are preferred. Therefore, minority class instances are more often misclassified than those from the other classes (Galar, Fernández, Barrenechea, Bustince, & Herrera, 2012). Although class imbalance does not hinder the learning task by itself, there are some difficulties related to this problem that turn up, such as a small sample size, as it is the case with microarray data. This problematic is of special importance when the imbalance is more marked in the test set than in the training set, as will be further discussed in subsection 4.2.4 dealing with the dataset shift problem. Multiclass datasets also suffer this problem. For example, the widely-used Lymphoma dataset (Alizadeh et al., 2000) has 9 classes but the majority class takes 48% of the samples. Traditional preprocessing techniques to overcome this issue are undersampling methods, which create a subset of the original dataset by eliminating instances; oversampling methods, which create a superset of the original dataset by replicating some instances or creating new instances from existing ones; and finally, hybrid methods that combine both sampling methods. One of the most employed resampling techniques is the so-called SMOTE (Chawla, Bowyer, Hall, & Kegelmeyer, 2002), where the minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. This technique was applied by Blagus and Lusa (2012) on microarray data, although the authors stated that it does not attenuate the bias towards the classification in the majority class for most classifiers. In recent years, ensemble of classifiers have arisen as a possible solution to the class imbalance problem attracting great interest among researchers (Galar et al., 2012; Galar, Fernández, Barrenechea, & Herrera, 2013), in several cases combined with preprocessing techniques such as SMOTE. Ensemble-based algorithms have proven to improve the results that are obtained by the usage of data preprocessing techniques and training a single classifier. For all these reasons, it is worth considering this problematic when dealing with unbalanced microarray datasets.

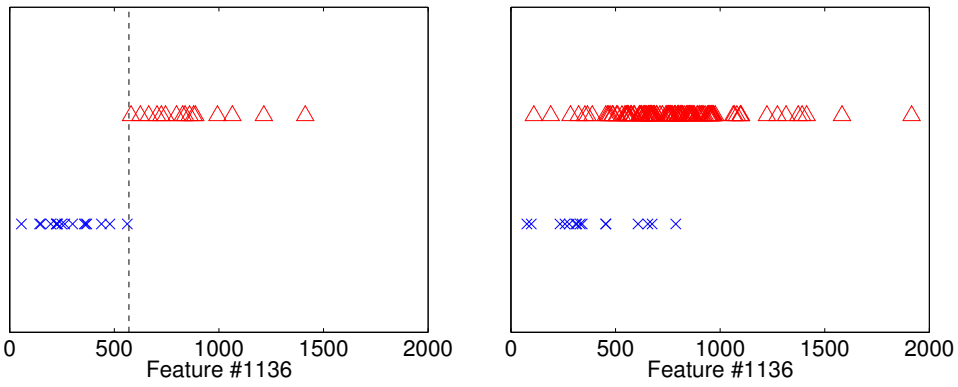
4.2.3 Data complexity

Data complexity measures are a recent proposal to represent characteristics of the data which are considered difficult in classification tasks, such as the overlapping among classes, their separability or the linearity of the decision boundaries (Saez, Luengo, & Herrera, 2013; T. K. Ho & Basu, 2002). Particularly, these measures have been referred to gene expression analysis by Lorena, Costa, Spolaôr, and de Souto (2012) and Okun and Priisalu (2009), demonstrating that low complexity corresponds to small classification error. In particular, the measures of *class overlapping*, such as F1 (*maximum Fisher's discriminant ratio*) (Saez et al., 2013), focus on the effectiveness of a single feature dimension in separating the classes. They examine the range and spread of values in the dataset within each class and check for overlapping among different classes.

4.2.4 Dataset shift

Another common problem when datasets come originally divided in training and test sets, is the so-called *dataset shift*. It occurs when the testing (unseen) data experience a phenomenon that leads to a change in the distribution of a single feature, a combination of features, or the class boundaries (Moreno-Torres, Raeder, Alaiz-Rodríguez, Chawla, & Herrera, 2012). As a result, the common assumption that the training and testing data follow the same distributions is often violated in real-world applications and scenarios, which may hinder the process of feature selection and classification. For example, Lung (Gordon et al., 2002) and Prostate (Singh et al., 2002) datasets have separated training and test sets. In the case of Lung, there is a single feature (#1136) which can correctly classify all the samples in the training set, as shown in Figure 4.2a, where different colors and shapes stand for different classes and the dashed line shows a clear linear separation between them. However, the same feature is not that informative in the test set and the class is not linearly separable, as displayed in Figure 4.2b. Besides, note that there is a enormous disparity in the class distribution: 50%-50% in the training set whilst 90%-10% in the test set.

The Prostate dataset poses a big challenge for machine learning methods since the test dataset was extracted from a different experiment and has a nearly 10-fold difference in overall microarray intensity from the training data. In fact, the test distribution (26%-74%) differs significantly from the train distribution (49%-51%) and with an inappropriate feature selection, some classifiers just assign all the samples to one of the classes (Bolón-Canedo, Sánchez-Maroño, & Alonso-Betanzos, 2010, 2013a).



(a) Lung Train (b) Lung Test

Figure 4.2: Feature #1136 in Lung dataset

Dataset shift can also appear when performing a cross-validation technique, which divides the whole training set into several subsets of data. In this case, there are some partitioning methods which may solve the problem (Moreno-Torres, Sáez, & Herrera, 2012).

4.2.5 Outliers

An important aspect that has been neglected in the literature is to detect outliers (Barnett & Lewis, 1994) in microarray samples. In some microarray datasets, there are samples incorrectly labeled or identified as likely contaminated which should be designated outliers, since they can exert a negative effect on the selection of informative genes for sample classification. Kadota, Tominaga, Akiyama, and Takahashi (2003) developed a method which found some outlying samples in the well-known Colon dataset. Therefore, analysis of samples designated as outliers should be considered as a pre-processing step in classification of microarray datasets because they can have a negative effect in the gene subset selection and, as a consequence, in the final prediction (Gonzalez-Navarro, 2011).

4.3 Algorithms for feature selection on microarray data: a review

Feature selection methods are constantly emerging and, for this reason, there is a wide suite of methods that deal with microarray gene data. The aim of this section is to present those methods developed in the last few years. As traditionally in the feature selection domain, the most employed gene selection methods fall into the filter approach (see subsection 4.3.1). Most of the novel filter methods proposed are based on information theory, although issues such as robustness or division in multiple binary problems are emerging topics. Discretization as a step prior to feature selection has also received some amount of attention. On the other hand, and due to the heavy computational consumption of resources and the high risk of overfitting, the wrapper approach has been mostly avoided in the literature (see subsection 4.3.2). Although the embedded model had not received enough attention during the infancy of microarray data classification, there appeared several proposals in the last years, as reported in subsection 4.3.3. It is also worth noticing that the review of the up-to-date literature showed a tendency to mix algorithms, either in the form of hybrid methods or ensemble methods. Also, it is well-known that genes interact with each other through gene regulative networks, so clustering methods have been also proposed. These novel approaches will be described in subsection 4.3.4.

4.3.1 Filters

Filter methods evaluate the goodness of gene subsets by observing only intrinsic data characteristics (i.e. statistical measures), where typically a single gene or a subset of genes is evaluated against the class label. Classical filter methods are usually applied to microarray data, such as Correlation Feature Selection (CFS), Fast Correlation-Based Filter (FCBF), ReliefF, or the consistency-based filter (see Chapter 2). Also, the well-known and widely-used minimum Redundancy Maximum Relevance (mRMR) has proven to be an appropriate tool for gene selection. During the last lustrum, besides the application of known methods, an important number of filter approaches have been proposed and applied to microarray datasets, and this subsection will review the most interesting ones. An important number of filters are based on information theory, as can be seen in subsection 4.3.1.1. On the other hand, several approaches include a preprocessing step to discretize data, since some filter require the data to be discrete,

as reported in subsection 4.3.1.2. Subsection 4.3.1.3 presents the filters which can deal with multiple binary problems and subsection 4.3.1.4 describes methods which are related with another issues such as robustness. A summary of all the filters reviewed in this subsection can be found in Table 4.1.

4.3.1.1 Information Theory

Firstly, the methods based on information theory will be presented, which despite being born years ago, are still the focus of much attention. A novel filter framework is presented by J. Wang, Wu, Kong, Li, and Zhang (2013) to select optimal feature subset based on a maximum weight and minimum redundancy (MWMR) criterion. The weight of each feature indicates its importance for some ad-hoc tasks (e.g. clustering or classification) and the redundancy represents the correlation among features. With this method it is possible to select the feature subset in which the features are most beneficial to the subsequent tasks while the redundancy among them is minimal. Experimental results on five datasets (two of them based on DNA microarray) demonstrated the advantage and efficiency of MWMR.

In a previous work (Bolón-Canedo, Seth, Sánchez-Maroño, Alonso-Betanzos, & Principe, 2011) a statistical dependence measure is presented for gene selection in the context of DNA microarray classification. The proposed method is also based on a maximum relevance minimum redundancy approach, but it uses a simple measure of monotone dependence ($\mathcal{M}_d$) to quantify both relevance and redundancy. $\mathcal{M}_d$ was compared against the well-known minimum redundancy maximum relevance (mRMR) method, and was shown to obtain better or equal performance over binary datasets.

Also related with information theory, Meyer, Schretter, and Bontempi (2008) introduced MASSIVE, a new information-theoretic filter approach for mining microarray data. This filter relies on a criterion which consists of maximizing a term appearing both in the lower bound and in an upper bound of the mutual information of a subset. The experimental results showed that the proposed method is competitive with five state-of-the-art approaches.

An entropic filtering algorithm (EFA) (González Navarro & Belanche Muñoz, 2009) was proposed as a fast feature selection method based on finding feature subsets that jointly maximize the normalized multivariate conditional entropy with respect to the

classification ability of tumors. The solutions achieved are of comparable quality to previous results, obtained in a maximum of half an hour computing time and using a very low number of genes.

L. Song, Smola, Gretton, Bedo, and Borgwardt (2012) introduced a framework for feature selection based on dependence maximization between the selected features and the labels of an estimation problem, using the Hilbert-Schmidt Independence Criterion. Their proposed method, BAHSIC, is a filter method that demonstrated good performance on microarray data, compared with more specialized methods.

4.3.1.2 Discretization

After presenting the measures related with information theory, the topic of discretization (García, Luengo, Sáez, López, & Herrera, 2013) related to feature selection will be discussed. Although the use of a feature selection method when dealing with microarray data is a common practice, discretization has not received the same amount of attention. Ferreira and Figueiredo (2012) proposed not only new techniques for feature selection, but they also added a previous discretization step. They performed scalar feature discretization with the well-known Linde-Buzo-Gray algorithm, using a stopping criterion based on bit allocation. Then, the feature selection step applies a simple unsupervised criterion with indicators on the original numeric features and on the discretized features. They also devised two efficient relevance/redundancy feature selection algorithms (RFS and RRFS) in order to remove redundant features.

The necessity of a prior discretization of the data is introduced for two main reasons: the first one is to help the filtering process and the second one is related to the high number of genes with very unbalanced values present in microarray datasets (Bolón-Canedo et al., 2010). Results on ten datasets demonstrated that the combination method, discretizer+filter, outperformed the results achieved by previous approaches, in some cases with improvements in the classification accuracy and descents in the number of genes needed. This methodology will be explained in detail in Chapter 7.

4.3.1.3 Multiple binary problems

The same scheme of discretizer+filter was employed again (Sánchez-Maroño, Alonso-Betanzos, García-González, & Bolón-Canedo, 2010), but in this case to be applied only to multiclass datasets. While studies on feature selection using the multiclass approach (a method that can deal directly with multiple classes) are relatively frequent in the literature (Bolón-Canedo, Seth, et al., 2011; Meyer et al., 2008; L. Song et al., 2012; Bolón-Canedo et al., 2010; Ferreira & Figueiredo, 2012; Nie, Huang, Cai, & Ding, 2010; Lan & Vucetic, 2011), very few studies employ the multiple binary sub-problems approach. Two well-known methods were employed for generating binary problems from a multiple class dataset: one versus one and one versus rest. The methodology was applied on 21 datasets, including a microarray dataset (Leukemia). On this dataset, the best results were obtained when applying feature selection. Specifically, the one versus rest approach obtained promising accuracy results along with a drastic reduction in the number of features needed.

Student and Fujarewicz (2012) also proposed a method based on Partial Least Squares (PLS) and decomposition to a set of two-class sub-problems; again using one versus one and one versus rest. They state that it is more effective to solve a multiclass feature selection by splitting it into a set of two-class problems and merging the results in one gene list. In this way, they obtained a very good accuracy rate and stability, as well as providing easy interpretation of the results by biologists.

4.3.1.4 Other approaches

Robustness is a trending issue on feature selection. Nie et al. (2010) proposed a new robust feature selection method (RFS) with emphasizing joint $\ell_{2,1}$-norm minimization on both loss function and regularization. This method is robust to outliers and also efficient in calculation.

Finally, a very interesting and novel filter approach was proposed by Lan and Vucetic (2011) based in so-called multi-task learning. When the number of labelled microarrays is particularly small (e.g. less than 10), the amount of available information diminishes to the level that even the most carefully designed classification approaches are bound to outperform. An alternative approach is to utilize information from the external microarray datasets, so accuracy on the target classification task can be significantly

increased if data from the auxiliary tasks are consulted during learning. The multi-task filter (M_FS) was evaluated on microarray data showing that this method is successful when applied in conjunction with both single-task and multi-tasks classifiers.

Table 4.1: Filter methods used on microarray data. Type of evaluation (ranker/subset) and type of data (binary/multiclass).

Method	Original Ref.	Type (r/s)	Data (b/m)
BAHSIC	(L. Song et al., 2012)	r	m
Discretizer+filter	(Bolón-Canedo et al., 2010; Sánchez-Maroño et al., 2010)	s	m
EFA	(González Navarro & Belanche Muñoz, 2009)	s	b
$\mathcal{M}_d$	(Bolón-Canedo, Seth, et al., 2011)	r	m
M_FS	(Lan & Vucetic, 2011)	r	m
MASSIVE	(Meyer et al., 2008)	r	m
MWMR	(J. Wang et al., 2013)	s	b
PLS	(Student & Fujarewicz, 2012)	r	m
RFS	(Ferreira & Figueiredo, 2012)	r	m
RFS	(Nie et al., 2010)	r	m
RRFS	(Ferreira & Figueiredo, 2012)	r	m

4.3.2 Wrappers

As mentioned before, the wrapper approach has not received the same amount of attention than the filter methods, due to its high computational cost. As the number of features grows, the space of feature subsets grows exponentially, something that becomes a critical aspect when having tens of thousands of features. Besides, they have the risk of overfitting due to the small sample size of microarray data. As a result, the wrapper approach has been mostly avoided in the literature.

Some works using the wrapper approach can be found in the earliest years of the investigation of microarray data. Notice that in a typical wrapper, a search is conducted in the space of genes, evaluating the goodness of each found gene subset by the estimation of the accuracy percentage of the specific classifier to be used, training the classifier only with the found genes. For example, Inza et al. (2002) evaluated classical wrapper search algorithms (sequential forward and backward selection, floating selection and best-first search) on three microarray datasets. Another example has been provided by Ruiz et al. (2006), in which an incremental wrapper called BIRS is presented for gene selection. Although the use of wrappers on microarray data has not evolved in the same line than the other feature selection methods, some examples were found in the last years.

Sharma, Imoto, and Miyano (2012) proposed an algorithm called successive feature selection (SFS). It is well-known that most of the conventional feature selection algorithms (e.g. individual ranking and forward selection schemes) have the drawback that a weakly ranked gene that could perform well in terms of classification with an appropriate subset of genes will be left out of the selection. Trying to overcome this shortcoming, the proposed SFS consists of first partitioning the features into smaller blocks. Once the top features from each of the blocks are obtained according to their classification performance, they are compared among themselves to obtain the best feature subset. This algorithm provides high classification accuracy on several DNA microarray datasets.

Wanderley, Gardeux, Natowicz, and Braga (2013) presented an evolutionary wrapper method (GA-KDE-Bayes). It uses a non-parametric density estimation method and a Bayesian classifier. The authors state that non-parametric methods are a good alternative for scarce and sparse data, such as bioinformatics problem, since they do not make any assumptions about its structure and all the information come from data itself. Results on six microarray datasets showed better performance than other presented in the literature.

Table 4.2 visualizes the wrapper methods described, along with the original reference, the type of evaluation (ranker or subset) and the type of data that they can deal with (binary or multiclass).

Table 4.2: Wrapper methods used on microarray data. Type of evaluation (ranker/subset) and type of data (binary/multiclass).

Method	Original Ref.	Type (r/s)	Data (b/m)
GA-KDE-Bayes	(Wanderley et al., 2013)	s	b
SPS	(Sharma et al., 2012)	s	m

4.3.3 Embedded

Despite their lower time consumption, a main disadvantage of the filter approach is the fact that it does not interact with the classifier, usually leading to worse performance results than wrappers. However, it has been seen that the wrapper model comes with a

expensive computational cost, especially aggravated by the high dimensionality of microarray data. An intermediate solution for researchers is the use of embedded methods, which use the core of the classifier to establish a criteria to rank features. Probably the most famous embedded method is Support Vector Machine based on Recursive Feature Elimination (SVM-RFE), proposed by Guyon et al. (2002) to specifically deal with gene selection for cancer classification. This method soon joined the group of algorithms which represent the state-of-the-art for gene selection, and therefore multiple extensions and modifications of it have been proposed. Next, we will describe several embedded approaches designed to deal with microarray data that we found reviewing the up-to-date literature (for a summary of them, consult Table 4.3).

Maldonado, Weber, and Basak (2011) introduced a new embedded method. It simultaneously selects relevant features during classifier construction by penalizing each feature's use in the dual formulation of support vector machines (SVM). The approach is called kernel-penalized SVM (KP-SVM) and it optimizes the shape of an anisotropic RBF Kernel eliminating features that have low relevance for the classifier. The experiments on two benchmark microarray datasets and two real-world datasets showed that KP-SVM outperformed the alternative approaches and determined consistently fewer relevant features.

G. Wang, Song, Xu, and Zhou (2013) proposed a FOIL (First Order Inductive Learner) rule based feature subset selection algorithm, called FRFS. This method firstly generates the FOIL classification rules using a modified propositional implementation of the FOIL algorithm. Then, it combines the features that appeared in the antecedents of all rules together, and achieves a candidate feature subset that excludes redundant features and reserves the interactive ones. Lastly, it measures the relevance of the features in the candidate feature subset by their proposed new metric CoverRatio and identifies and removes the irrelevant features.

Shah, Marchand, and Corbeil (2012) focused not only on obtaining a small number of genes but also on having verifiable future performance guarantees. They investigated the premise of learning conjunctions of decision stumps and proposed three formulations based on different learning principles, which embed the feature selection as a part of the learning process itself. One of their proposals, Probably Approximately Correct (PAC) Bayes, yields competitive classification performance while at the same time utilizing significantly fewer attributes.

Canul-Reich, Hall, Goldgof, Korecki, and Eschrich (2012) introduced the iterative perturbation method (IFP), which is an embedded gene selector applied to four microarray datasets. This algorithm uses a backward elimination approach and a criterion to determine which features are the least important relying on the classification performance impact that each feature has when perturbed by noise. If adding noise leads to a big change in the classification performance, then the feature is considered relevant. The IFP approach resulted in comparable or superior average class accuracy when compared to well-studied SVM-RFE on three out of the four datasets.

To overcome the problem of the imbalance of some microarray datasets, a new embedded method based on random forest algorithm is presented by Anaissi, Kennedy, and Goyal (2011). Its strategy is composed of different methods and algorithms. First, an algorithm to find the best training error cost for each class is run, in order to deal with the imbalance of the data. Then, random forest is run to select the relevant features. Finally, a strategy to avoid overfitting is also applied. The method was designed ad-hoc to deal with a complex gene expression dataset for Leukemia malignancy, showing a very acceptable outcome.

Table 4.3: Embedded methods used on microarray data. Type of evaluation (ranker/subset) and type of data (binary/multiclass).

Method	Original Ref.	Type (r/s)	Data (b/m)
FRFS	(G. Wang et al., 2013)	s	m
IFP	(Canul-Reich et al., 2012)	s	b
KP-SVM	(Maldonado et al., 2011)	s	m
PAC-Bayes	(Shah et al., 2012)	r	b
Random Forest	(Anaissi et al., 2011)	s	m

4.3.4 Other algorithms

Nowadays, the trend is to use not only classical feature selection methods (filters, wrappers and embedded) but also to focus on new combinations such as hybrid or ensemble methods.

Hybrid methods usually combine two or more feature selection algorithms of different conceptual origin in a sequential manner. Mundra and Rajapakse (2010) combined two of the most famous feature selection methods for microarray data: SVM-RFE

and mRMR. They propose an approach that incorporates a mutual-information-based mRMR filter in SVM-RFE to minimize the redundancy among selected genes. Their approach improved the accuracy of classification and yielded smaller gene sets compared with mRMR and SVM-RFE, as well as other popular methods.

Shreem, Abdullah, Nazri, and Alzaqebah (2012) also used mRMR in their hybrid method. In this case, the proposed approach combines ReliefF, mRMR and GA (Genetic Algorithm) coded as R-m-GA. In the first stage, the candidate gene set is identified by applying ReliefF. Then, the redundancy is minimized with the help of mRMR, which facilitates the selection of effectual gene subset from the candidate set. In the third stage, GA with classifier (used as a fitness function by the GA) is applied to select the most discriminating genes. The proposed method is capable of finding the smallest gene subset that offers the highest classification accuracy.

Chuang, Yang, Wu, and Yang (2011) proposed a hybrid method called CFS-TGA, which combines correlation-based feature selection (CFS) and the Taguchi-genetic algorithm, where the K-nearest neighbor served as a classifier. The proposed method obtained the highest classification accuracy in ten out the eleven gene expression datasets in which it was tested on.

C. Lee and Leu (2011) developed another hybrid method. It first uses a genetic algorithm with dynamic parameter setting (GADP) to generate a number of subsets of genes and to rank the genes according to their occurrence frequencies in the gene subsets. Then, χ^2 is used to select a proper number of the top-ranked genes for data analysis. Finally, a SVM is employed to verify the efficiency of the selected-genes. The experimental results on six microarray datasets showed that the GADP method is better than the existing methods in terms of the number of selected genes and the prediction accuracy.

Leung and Hung (2010) proposed a multiple-filter-multiple-wrapper (MFMW) method. The rationale behind this proposal is that filters are fast but their predictions are inaccurate whilst wrappers maximize classification accuracy at the expense of a formidable computation burden. MFMW is based on previous hybrid approaches that maximize the classification accuracy for a chosen classifier with respect to a filtered set of genes. The drawback of the previous hybrid methods which combine filters and wrappers is that the classification accuracy is dependent on the choice of specific filter and wrapper. MFMW overcomes this problem by making use of multiple filters and multiple wrappers to improve the accuracy and robustness of the classification.

Ensemble feature selection builds on the assumption that combining the output of multiple experts is better than the output of any single expert. Typically, ensemble learning has been applied to classification, but it has recently been applied to microarray gene selection. An ensemble of filters (EF) is proposed (Bolón-Canedo et al., 2012), which will be further explained in chapter 8. The rationale of this approach is behind the variability of results of each available filter over different microarray datasets. That is, a filter can obtain excellent classification results in a given dataset while performing poorly in another dataset, even in the same domain. This ensemble obtains a classification prediction for every different filter conforming the ensemble, and then combines these predictions by simple voting. Experiments on 10 microarray datasets showed that the ensemble obtained the lowest average of classification error for the four classifiers tested. Recently, the same authors introduced new ensembles to improve performance (E1-cp, E1-nk, E1-ns, E2)(Bolón-Canedo et al., 2013a).

From the perspective of pattern analysis, researchers must focus not only in classification accuracy but also in producing a stable or robust solution. Trying to improve the robustness of feature selection algorithms, F. Yang and Mao (2011) proposed an ensemble method called multicriterion fusion-based recursive feature elimination (MCF-RFE). Experimental studies on microarray datasets demonstrated that the MCF-RFE method outperformed the commonly used benchmark feature selection algorithm SVM-RFE both in classification performance and stability of feature selection results.

Abeel, Helleputte, Van de Peer, Dupont, and Saeys (2010) are also concerned with the analysis of the robustness of biomarker selection techniques. For this sake, they proposed a general experimental setup for stability analysis that can be easily included in any biomarker identification pipeline. In addition, they also presented a set of ensemble feature selection methods improving biomarker stability and classification performance on four microarray datasets. They used recursive feature elimination (RFE) as a baseline method and a bootstrapping method to generate diversity in the selection. Then, two different schemes were proposed to aggregate the different rankings of features. Their findings were that when decreasing the number of selected features, the stability of RFE tends to degrade while ensemble methods offer significantly better stability.

Ye, Wu, Zhexue Huang, Ng, and Li (2013) proposed a stratified sampling method to select the feature subspaces for random forest (SRF). The key idea is to stratify features into two groups. One group will contain strong informative features and the other weak informative features. Then, for feature subset selection, features are randomly chosen

from each group proportionally. The advantage of stratified sampling is that it can ensure that each subspace contains enough informative features for classification in high dimensional data.

Clustering methods for microarray data have been also recently proposed. Most of the gene selection techniques are based on the assumption of the independence between genes (actually a typical approach is to rank them individually). However, it is well known that genes interact with each other through gene regulative networks. To overcome this problem, Lovato et al. (2012) presented a novel feature selection scheme, based on the Counting Grid (GC) model, which can measure and consider the relation and the influence between genes.

Table 4.4: Other feature selection methods used on microarray data. Type of evaluation (ranker/subset) and type of data (binary/multiclass).

Method	Original Ref.	Type (r/s)	Data (b/m)
CFS-TGA	(Chuang et al., 2011)	s	m
E1-cp	(Bolón-Canedo et al., 2013a)	s	b
E1-nk	(Bolón-Canedo et al., 2013a)	s	b
E1-ns	(Bolón-Canedo et al., 2013a)	s	b
E2	(Bolón-Canedo et al., 2013a)	s	b
Ensemble RFE	(Abeel et al., 2010)	s	b
EF	(Bolón-Canedo et al., 2012)	s	m
FAST	(Q. Song, Ni, & Wang, 2013)	s	m
GADP	(C. Lee & Leu, 2011)	s	m
GC	(Lovato et al., 2012)	r	b
MCF-RFE	(F. Yang & Mao, 2011)	s	b
MFMW	(Leung & Hung, 2010)	s	b
R-m-GA	(Shreem et al., 2012)	s	b
SRF	(Ye et al., 2013)	s	m
SVM-RFE with MRMR	(Mundra & Rajapakse, 2010)	r	b

Q. Song et al. (2013) presented a fast clustering-based feature selection algorithm (FAST) which works in two steps. In the first step, features are divided into clusters by using graph-theoretic clustering methods. In the second step, the most representative feature that is strongly related to target classes is selected from each cluster to form a subset of features. Since features in different clusters are relatively independent, the clustering-based strategy of FAST has a high probability of producing a subset

of useful and independent features. The exhaustive evaluation carried out on 35 real-world datasets (14 of them in the microarray domain) demonstrated that FAST not only produced smaller subsets of features, but also improved the performances for four types of classifiers.

Table 4.4 depicts a summary of the methods presented in this section. The original reference is displayed, as well as the type of evaluation (ranker or subset) and the type of the data they can deal with (binary or multiclass).

4.4 A framework for feature selection evaluation in microarray datasets

The goal of performing feature selection on microarray data can be two-fold: class prediction or biomarkers identification. If the goal is class prediction, there is a demand for machine learning techniques such as supervised classification. However, if the objective is to find informative genes, the classification performance is ignored and the selected genes have to be individually evaluated. The experiments that will be presented in this section are focused on class prediction, which is an important reason to use feature selection methods in microarray analysis. The typical microarray pipeline is formed by a feature selection step, followed by a classification stage and providing an error estimation, as seen in Figure 4.3.

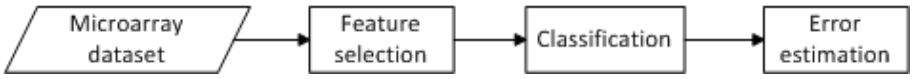


Figure 4.3: DNA microarray classification pipeline.

The rest of this section is devoted to the importance of the validation techniques usually applied on microarray data and to analyze the characteristics of the datasets whilst providing classification accuracy results obtained with classical feature selection methods.

4.4.1 Validation techniques

To evaluate the goodness of the selected set of genes, it is necessary to have an independent test set with data which have not been seen by the feature selection method. In some cases, the data come originally distributed into training and test sets, so the training set is usually employed to perform the feature selection process and the test set is used to evaluate the appropriateness of the selection. However, not all the datasets found in the literature are originally partitioned. For overcoming this issue, there exist several validation techniques, where the most used ones in the microarray domain are k-fold cross validation, leave-one-out cross validation, bootstrap and holdout validation (see Appendix I, Section I.3).

Probably cross-validation would be the most famous technique. However, it has been shown (Moreno-Torres, Sáez, & Herrera, 2012) that it can potentially introduce dataset shift, a harmful factor that is often not taken into account and can result in inaccurate performance estimation. To solve this problem, *Distribution optimally balanced stratified cross-validation* (DOB-SCV) (Moreno-Torres, Sáez, & Herrera, 2012) is based on the idea that by assigning close-by examples to different folds, each fold will end up with enough representatives of every region, thus avoiding dataset shift. To achieve this goal, DOB-SCV starts on a random unassigned example, and then finds its $k-1$ nearest unassigned neighbors of the same class. Once it has found them, it assigns each of those examples to a different fold. The process is repeated until all examples are assigned.

The selection of a validation technique on the microarray domain is not an easy-to-solve question. This is due to the fact that microarray data is characterized by a extremely high number of features and comparatively small number of samples. This situation is commonly referred to as a *small-sample* scenario, which means that application of traditional pattern recognition methods must be carried out with judgment to avoid pitfalls (Braga-Neto, 2007). A key point for microarray classification is that error estimation is greatly impacted by small samples (E. Dougherty, 2001), so the choice of a validation technique must be further discussed. In fact, there are several works in the literature dealing with this issue. Ambroise and McLachlan (2002) recommended to use 10-fold cross validation rather than leave-one-out and, concerning the bootstrap, they suggest using the so called 0.632+ bootstrap error estimate (which weighs the bootstrapped resubstitution error) designed to handle overfitted prediction rules. Braga-Neto and Dougherty (2004) performed an extensive simulation study by

comparing cross-validation, resubstitution and bootstrap estimation. They stated that while cross-validation error estimation is much less biased than resubstitution, it displays excessive variance, which makes individual estimates unreliable for small samples. Bootstrap methods provide improved performance relative to variance, but at a high computational cost and often with increased bias.

In this situation, it does not exist the so-called best validation technique for microarray data. In fact, reviewing the recent literature one can find examples of the four methods described above. k-fold cross-validation is a common choice (Meyer et al., 2008; L. Song et al., 2012; Nie et al., 2010; Shah et al., 2012; Canul-Reich et al., 2012; Shreem et al., 2012; Ye et al., 2013), as well as holdout validation (González Navarro & Belanche Muñoz, 2009; Ferreira & Figueiredo, 2012; Sharma et al., 2012; Maldonado et al., 2011; Anaissi et al., 2011; C. Lee & Leu, 2011; Lovato et al., 2012). Bootstrap sampling was less used (Student & Fujarewicz, 2012; Mundra & Rajapakse, 2010; F. Yang & Mao, 2011), probably due to its high computational cost, and there are also some representatives of leave-one-out cross-validation (Chuang et al., 2011; Leung & Hung, 2010).

4.4.2 On the datasets characteristics

In this study, nine widely-used binary microarray datasets have been considered, which are available for download in (Feature Selection at ASU, n.d.; Statnikov, Aliferis, & Tsamardinos, n.d.; Kent Ridge, n.d.). The reason to choose binary datasets is because they are much more common in the literature than the multiclass ones. As a matter of fact, a typical microarray dataset consists of distinguishing between having a given cancer or not, therefore the great majority of the datasets are binary. Tables 4.5 and 4.6 show the imbalance ratio (IR) and the F1 (*maximum Fisher's discriminant ratio*) of the datasets used, whereas more details about them (such as number of features or samples) can be found in Appendix I, Section I.2.4. The imbalance ratio (Orriols-Puig & Bernadó-Mansilla, 2009) is defined as the number of negative class examples that are divided by the number of positive class examples, where a high level indicates that the dataset is highly imbalanced. F1 (see Section 4.2.3) checks for overlapping among the classes where the higher the F1, the more separable the data is. Notice that for the datasets in Table 4.5, which come originally divided in training and test sets, these measures are shown for both partitions. For these two datasets, it is shown that both of them present more imbalance in the test set than in the training set,

especially in the case of Prostate dataset. As for the F1 measure, the higher amount of overlapping occurs on Breast training set, with a value of 0.68. Regarding the information depicted in Table 4.6, the most unbalanced dataset is GLI and the one more affected by overlapping is SMK, with a F1 value of 0.41.

Table 4.5: Imbalance ratio and F1 of the binary datasets used in the holdout experimental study

	Train		Test	
Dataset	IR	F1	IR	F1
Breast	1.29	0.68	1.71	4.98
Prostate	1.04	2.05	2.78	11.35

Table 4.6: Imbalance ratio and F1 of the binary datasets used in the k-fold cross-validation experimental study

Dataset	IR	F1
Brain	2.00	0.89
CNS	1.86	0.45
Colon	1.82	1.08
DLBCL	1.04	2.91
GLI	2.27	2.35
Ovarian	1.78	6.94
SMK	1.08	0.41

4.4.3 Feature selection methods

Seven classical feature selection methods were chosen to be applied on this study: CFS, FCBF, INTERACT, Information Gain, ReliefF, mRMR and SVM-RFE (see Chapter 2). All of them are available in the well-known Weka tool (M. Hall et al., 2009), except for mRMR filter, whose implementation is available for Matlab ®. These methods are used extensively in the literature, and their performance can serve as a reference for the interested reader.

4.4.4 Evaluation measures

In order to evaluate the behavior of the feature selection methods after applying a classifier, three well-known measures are used: accuracy, sensitivity and specificity (see Appendix I, Section I.6). In general, sensitivity indicates how well the test predicts the actual positives (e.g. the percentage of cancer patients who are correctly identified as having the condition) while specificity measures how well the test identifies the negatives (e.g. the percentage of healthy patients who are correctly identified as not having cancer). A perfect predictor would be described as 100% sensitive (e.g. predicting all patients with cancer as such) and 100% specific (e.g. not predicting any patient from the healthy group as having cancer). Accuracy is expected to measure how well the test predicts both categories.

4.5 A practical evaluation: Analysis of results

The goal of this section is to perform an experimental study using the most representative binary datasets, described in Section 4.4.2 and some classical feature selection methods. To evaluate the adequacy of these methods over microarray data, three well-known classifiers were chosen: C4.5, naive Bayes and SVM (see Appendix I). As reported in Section 4.4.1, there is no consensus in the literature about which validation technique to use when dealing with microarray data. In light of those facts, two studies will be performed. In the first one, a holdout validation will be applied on those datasets which come originally divided in training and test datasets. As revealed in Section 4.2.4, the training and test data of those datasets were extracted under different conditions, which means an added challenge for the machine learning methods. If the two sets are joined in an unique dataset (e.g. for later applying a k-fold cross-validation), the new situation would be easier for the learner, and this particular characteristic of microarray data would be overlooked. The second study will consist on applying a 5-fold cross-validation over those datasets which provide a unique training test, where 5 folds have been chosen because with the standard value of 10, for some datasets the test set would remain with only a couple of samples. However, as mentioned in Section 4.4.1, in some cases cross-validation can potentially introduce dataset shift, so we will include DOB-SCV in the experimental study trying to overcome this problem.

The first three feature selection methods (CFS, FCBF and INTERACT) return a subset of features, whilst the remaining four (IG, ReliefF, mRMR and SVM-RFE) provide an ordered ranker of the features. For the ranker methods, the performance when retaining the top 10 and 50 features is shown. For those methods which return a subset of features, the number of features selected for each dataset is revealed in Table 4.7. Notice that for the datasets involved in the cross-validation study (Brain, CNS, Colon, DLBCL, Gli85, Ovarian and Smk) this number is the mean average of the number of features selected in each fold. Since two types of partitions are tried, both values are shown in the table (regular cross-validation / DOB-SCV).

Table 4.7: Number of features selected by subset methods on binary datasets

Method	Brain	Breast	CNS	Colon	DLBCL	Gli85	Ovarian	Prostate	Smk
CFS	36/49	130	44/44	24/25	61/65	141/156	35/33	89	107/103
FCBF	1/1	99	33/35	14/15	35/37	116/118	27/26	77	50/55
INT	36/49	102	33/34	14/16	45/51	117/123	32/31	73	51/51

4.5.1 Holdout validation study

This section reports the experimental results achieved over the binary datasets that are originally divided into training and test sets (see Table 4.5). Tables 4.8, 4.9 and 4.10 show the results achieved by C4.5, naive Bayes and SVM, respectively. These tables depict the classification accuracy (Ac), sensitivity (Se) and specificity (Sp) on the test datasets. For the sake of comparison, the first row shows those values without applying feature selection techniques. Notice that C4.5 algorithm does a feature selection because not all the attributes are considered when constructing the tree. The best results for dataset and classifier are marked in bold face.

Analyzing these tables, it is easy to note that the results obtained with SVM outperformed notably those achieved by C4.5 or naive Bayes. In fact, it has been in the literature a clear tendency to use SVM as the standard *de facto* method to estimate performance measures and Gonzalez-Navarro (2011) stated that the superiority of SVMs over other several classifiers seems out of doubt. As mentioned in Section 4.2.4, the Prostate dataset suffers from dataset shift, since the test dataset was extracted from a different experiment, and apparently C4.5 and naive Bayes classifiers cannot solve this problem and opted for assigning all the examples to the majority class.

Table 4.8: Experimental results for C4.5 classifier on binary datasets after performing holdout validation

		Breast			Prostate		
		Ac	Se	Sp	Ac	Se	Sp
no FS		0.74	**1.00**	0.58	0.26	**1.00**	0.00
CFS		0.68	0.71	0.66	0.26	**1.00**	0.00
FCBF		0.58	0.28	0.75	0.26	**1.00**	0.00
INT		**0.79**	0.71	0.83	0.26	**1.00**	0.00
IG	#10	0.47	0.28	0.58	0.26	**1.00**	0.00
	#50	0.53	0.42	0.58	0.29	**1.00**	0.04
ReliefF	#10	0.58	0.28	0.75	0.26	**1.00**	0.00
	#50	0.42	0.71	0.25	0.29	**1.00**	0.04
SVM-RFE	#10	0.58	**1.00**	0.33	0.32	**1.00**	0.08
	#50	0.58	**1.00**	0.33	0.26	**1.00**	0.00
mRMR	#10	0.58	0.71	0.50	**0.41**	0.88	**0.24**
	#50	0.74	0.42	**0.91**	0.35	**1.00**	0.12

Table 4.9: Experimental results for naive Bayes classifier on binary datasets after performing holdout validation

		Breast			Prostate		
		Ac	Se	Sp	Ac	Se	Sp
no FS		0.37	**1.00**	0.00	**0.26**	**1.00**	0.00
CFS		0.37	**1.00**	0.00	**0.26**	**1.00**	0.00
FCBF		0.37	**1.00**	0.00	**0.26**	**1.00**	0.00
INT		0.37	**1.00**	0.00	**0.26**	**1.00**	0.00
IG	#10	0.32	0.85	0.00	**0.26**	0.88	0.04
	#50	0.37	**1.00**	0.00	0.24	0.88	0.00
ReliefF	#10	0.74	0.71	0.75	0.21	0.55	**0.08**
	#50	**0.89**	0.85	**0.91**	0.21	0.77	0.00
SVM-RFE	#10	0.68	0.85	0.58	0.18	0.55	0.04
	#50	0.63	**1.00**	0.41	**0.26**	**1.00**	0.00
mRMR	#10	0.37	**1.00**	0.00	**0.26**	**1.00**	0.00
	#50	0.37	**1.00**	0.00	**0.26**	**1.00**	0.00

Table 4.10: Experimental results for SVM classifier on binary datasets after performing holdout validation

		Breast			Prostate		
		Ac	Se	Sp	Ac	Se	Sp
no FS		0.58	0.85	0.41	0.53	**1.00**	0.36
CFS		0.68	0.85	0.58	**0.97**	**1.00**	**0.96**
FCBF		0.58	0.28	0.75	**0.97**	**1.00**	**0.96**
INT		0.74	0.71	0.75	0.71	**1.00**	0.60
IG	#10	0.58	0.71	0.50	**0.97**	**1.00**	**0.96**
	#50	0.79	0.57	**0.91**	**0.97**	**1.00**	**0.96**
ReliefF	#10	**0.84**	**1.00**	0.75	0.94	0.88	**0.96**
	#50	**0.84**	0.85	0.83	**0.97**	**1.00**	**0.96**
SVM-RFE	#10	0.68	**1.00**	0.50	0.79	**1.00**	0.72
	#50	0.68	**1.00**	0.50	0.74	**1.00**	0.64
mRMR	#10	0.63	0.71	0.58	0.44	**1.00**	0.24
	#50	0.68	0.71	0.66	0.91	0.77	**0.96**

4.5.2 Cross-validation study

This section reveals the classification results obtained when applying the well-known cross-validation technique. A 5-fold cross-validation was performed over the binary datasets presented in Table 4.6, which only have training set available. Since in some cases cross-validation can potentially introduce the dataset shift problem, another strategy has been used. Distribution optimally balanced stratified cross-validation (DOB-SCV) tries to avoid dataset shift by assigning close-by examples to different folds. Subsection 4.5.2.1 analyzes the behavior of the feature selection methods studied on the datasets, whilst subsection 4.5.2.2 compares the performance of regular cross-validation against DOB-SCV. Finally, subsection 4.5.2.3 will analyze the influence of the datasets characteristics.

Tables 4.11 - 4.16 show the results achieved by C4.5, naive Bayes and SVM, for the two types of cross validation. The results shown in the tables are the average test results for the 5 folds, depicting the classification accuracy (Ac), sensitivity (Se) and specificity (Sp). Again, the first row reports those values without applying feature selection and the best results are marked in bold face.

Table 4.11: Experimental results for C4.5 classifier on binary datasets after performing regular 5-fold cross-validation.

			Brain	CNS	Colon	DLBCL	Gli85	Ovarian	Smk	Avg
No FS		Ac	**1.00**	0.58	0.74	0.70	0.75	0.97	0.65	0.77
		Se	**1.00**	0.64	0.60	0.69	0.81	0.95	0.66	0.77
		Sp	**1.00**	0.48	0.82	0.70	0.63	0.98	0.62	0.75
CFS		Ac	**1.00**	0.62	0.79	0.75	0.79	0.98	0.64	**0.79**
		Se	**1.00**	0.64	0.68	0.78	0.81	0.95	0.56	0.78
		Sp	**1.00**	**0.58**	0.85	0.71	0.75	**0.99**	**0.71**	**0.80**
FCBF		Ac	0.86	0.48	0.79	0.73	0.82	**0.99**	0.61	0.75
		Se	0.80	0.49	0.64	0.74	0.86	**0.99**	0.65	0.74
		Sp	0.86	0.50	0.87	0.70	0.75	**0.99**	0.56	0.75
INT		Ac	**1.00**	0.55	0.79	0.70	0.78	0.98	0.59	0.77
		Se	**1.00**	0.54	0.72	0.74	0.81	0.98	0.51	0.76
		Sp	**1.00**	**0.58**	0.82	0.66	0.71	0.98	0.66	0.77
IG	#10	Ac	0.71	0.62	0.72	0.75	**0.85**	0.96	0.60	0.74
		Se	0.70	0.69	0.78	0.79	0.88	0.93	0.71	0.78
		Sp	0.70	0.48	0.70	0.71	**0.79**	0.97	0.48	0.69
	#50	Ac	0.81	0.63	**0.84**	0.73	0.81	0.96	0.65	0.78
		Se	0.70	0.67	**0.83**	0.69	0.86	0.96	0.62	0.76
		Sp	0.87	**0.58**	0.85	0.74	0.71	0.97	0.67	0.77
ReliefF	#10	Ac	0.72	0.47	0.72	**0.85**	**0.85**	0.97	0.65	0.75
		Se	0.20	0.59	0.50	0.83	0.88	0.94	**0.80**	0.68
		Sp	**1.00**	0.25	0.85	**0.87**	0.77	**0.99**	0.47	0.74
	#50	Ac	0.62	0.53	0.82	0.73	0.82	**0.99**	0.61	0.73
		Se	0.20	0.60	0.68	0.74	0.88	**0.99**	0.61	0.67
		Sp	0.86	0.44	**0.90**	0.70	0.70	**0.99**	0.62	0.74
SVM-RFE	#10	Ac	0.57	**0.65**	0.71	0.81	0.81	0.98	0.60	0.73
		Se	0.00	**0.74**	0.60	0.82	0.85	0.98	0.65	0.66
		Sp	0.87	0.48	0.77	0.79	0.75	0.98	0.55	0.74
	#50	Ac	0.70	0.57	0.80	0.82	0.79	0.98	0.65	0.76
		Se	**1.00**	0.61	0.77	**0.84**	0.83	**0.99**	0.62	**0.81**
		Sp	0.56	0.49	0.82	0.79	0.70	0.98	0.66	0.72
mRMR	#10	Ac	0.86	0.55	0.82	0.75	0.79	0.98	**0.68**	0.77
		Se	0.90	0.72	0.68	0.79	0.86	0.96	0.71	0.80
		Sp	0.87	0.23	**0.90**	0.70	0.61	**0.99**	0.64	0.70
	#50	Ac	0.86	0.58	0.82	0.73	0.80	0.97	0.62	0.77
		Se	0.90	0.70	0.77	0.69	**0.91**	0.96	0.66	0.80
		Sp	0.87	0.39	0.85	0.74	0.54	0.98	0.57	0.71

Table 4.12: Experimental results for C4.5 classifier on binary datasets after performing DOB-SCV with 5 folds.

			Brain	CNS	Colon	DLBCL	Gli85	Ovarian	Smk	Avg
No FS		Ac	**0.92**	0.52	0.72	0.72	0.77	0.96	0.56	0.74
		Se	0.80	0.58	0.55	0.70	0.79	0.93	0.63	0.71
		Sp	**1.00**	0.39	0.80	0.74	0.74	0.97	0.48	0.73
CFS		Ac	**0.92**	0.52	0.79	0.80	0.75	0.96	0.59	0.76
		Se	0.80	0.53	0.65	**0.82**	0.77	0.93	0.59	0.73
		Sp	**1.00**	0.48	0.87	0.78	0.70	**0.98**	0.59	0.77
FCBF		Ac	0.72	0.52	0.81	0.72	0.76	**0.98**	0.59	0.73
		Se	0.70	0.56	0.69	0.70	0.79	**0.98**	0.53	0.71
		Sp	0.76	0.44	0.87	0.74	0.70	**0.98**	0.65	0.74
INT		Ac	**0.92**	0.53	0.81	0.74	0.71	0.97	0.67	0.76
		Se	0.80	0.51	0.69	0.65	0.69	0.94	0.58	0.69
		Sp	**1.00**	**0.56**	0.87	0.84	0.74	**0.98**	**0.75**	**0.82**
IG	#10	Ac	0.72	0.60	0.79	0.78	0.82	0.96	0.62	0.76
		Se	0.80	0.67	0.59	0.74	0.81	0.94	0.61	0.74
		Sp	0.73	0.49	**0.90**	0.83	**0.85**	0.96	0.63	0.77
	#50	Ac	0.77	0.54	**0.84**	0.80	0.81	**0.98**	0.60	0.76
		Se	0.80	0.59	**0.74**	**0.82**	0.79	**0.98**	0.58	0.76
		Sp	0.80	0.44	**0.90**	0.79	**0.85**	0.97	0.62	0.77
ReliefF	#10	Ac	0.48	0.55	0.77	**0.84**	0.81	0.96	0.66	0.73
		Se	0.10	**0.76**	0.69	**0.82**	**0.91**	0.93	0.67	0.70
		Sp	0.63	0.15	0.82	0.87	0.57	0.98	0.64	0.67
	#50	Ac	0.53	0.56	0.76	0.80	**0.85**	**0.98**	0.68	0.74
		Se	0.60	0.69	0.63	**0.82**	0.86	**0.98**	**0.75**	0.76
		Sp	0.50	0.34	0.82	0.79	0.81	**0.98**	0.60	0.69
SVM-RFE	#10	Ac	0.59	0.56	0.76	0.82	0.78	**0.98**	0.60	0.73
		Se	0.40	0.66	0.64	0.73	0.79	0.96	0.66	0.69
		Sp	0.70	0.37	0.82	**0.92**	0.74	**0.98**	0.53	0.72
	#50	Ac	0.70	**0.62**	0.78	0.78	0.76	0.97	0.65	0.75
		Se	0.70	0.65	0.56	0.69	0.76	0.95	0.69	0.71
		Sp	0.73	**0.56**	**0.90**	0.88	0.73	**0.98**	0.61	0.77
mRMR	#10	Ac	0.80	0.60	0.79	0.78	0.80	**0.98**	0.68	**0.78**
		Se	0.80	0.71	0.65	0.78	0.79	**0.98**	**0.75**	**0.78**
		Sp	0.83	0.38	0.87	0.80	0.81	**0.98**	0.61	0.75
	#50	Ac	0.84	0.60	0.82	0.76	0.78	**0.98**	**0.71**	**0.78**
		Se	**0.90**	0.66	0.69	0.78	0.76	**0.98**	0.71	**0.78**
		Sp	0.83	0.47	**0.90**	0.74	0.82	**0.98**	0.70	0.78

Table 4.13: Experimental results for naive Bayes classifier on binary datasets after performing regular 5-fold cross-validation.

			Brain	CNS	Colon	DLBCL	Gli85	Ovarian	Smk	Avg
No FS		Ac	0.67	0.60	0.55	0.92	0.84	0.93	0.63	0.73
		Se	0.00	0.64	0.69	**0.96**	0.88	0.99	0.60	0.68
		Sp	**1.00**	0.52	0.47	0.88	0.73	0.89	0.66	0.74
CFS		Ac	0.81	0.67	**0.85**	0.90	0.82	**1.00**	0.65	**0.81**
		Se	0.50	0.75	0.76	**0.96**	**0.90**	0.99	0.67	0.79
		Sp	**1.00**	0.54	**0.90**	0.84	0.67	**1.00**	0.62	0.79
FCBF		Ac	0.61	**0.70**	0.80	0.90	0.85	0.99	0.69	0.79
		Se	**1.00**	0.77	0.76	**0.96**	**0.90**	**1.00**	0.72	**0.87**
		Sp	0.40	0.58	0.82	0.84	0.74	0.99	0.65	0.72
INT		Ac	0.81	**0.70**	0.77	0.90	0.82	**1.00**	0.64	**0.81**
		Se	0.50	0.77	0.76	**0.96**	0.88	**1.00**	0.72	0.80
		Sp	**1.00**	0.58	0.77	0.83	0.71	0.99	0.55	0.78
IG	#10	Ac	**0.86**	0.63	0.79	**0.94**	0.85	0.96	0.61	**0.81**
		Se	0.70	0.67	0.72	**0.96**	0.88	0.95	0.59	0.78
		Sp	0.93	0.58	0.82	**0.92**	0.77	0.96	0.64	0.80
	#50	Ac	0.81	0.63	0.77	0.92	0.85	0.98	0.66	0.80
		Se	0.50	0.75	0.76	**0.96**	0.86	0.96	0.67	0.78
		Sp	**1.00**	0.42	0.77	0.88	0.81	0.98	0.65	0.79
ReliefF	#10	Ac	0.20	0.63	0.82	**0.94**	0.86	0.96	0.67	0.73
		Se	0.20	0.72	0.72	**0.96**	0.88	0.95	0.71	0.73
		Sp	0.20	0.48	0.87	**0.92**	0.81	0.96	0.63	0.70
	#50	Ac	0.19	0.67	0.84	0.92	**0.89**	0.98	0.67	0.74
		Se	0.50	0.72	0.77	**0.96**	0.86	0.95	0.72	0.78
		Sp	0.07	0.58	0.87	0.88	**0.97**	0.99	0.61	0.71
SVM-RFE	#10	Ac	0.62	0.68	0.73	0.92	0.82	0.99	**0.71**	0.78
		Se	0.30	0.77	0.61	0.91	0.83	**1.00**	**0.77**	0.74
		Sp	0.76	0.54	0.80	**0.92**	0.81	0.98	0.64	0.78
	#50	Ac	0.67	**0.70**	0.76	0.92	0.88	0.99	0.70	0.80
		Se	0.20	**0.82**	0.69	0.91	0.86	**1.00**	0.73	0.75
		Sp	0.90	0.49	0.80	**0.92**	0.93	0.98	0.65	**0.81**
mRMR	#10	Ac	0.73	0.63	0.80	0.92	0.85	0.99	0.67	0.80
		Se	0.60	0.79	0.78	**0.96**	0.88	0.96	0.68	0.81
		Sp	0.86	0.33	0.82	0.88	0.77	**1.00**	0.65	0.76
	#50	Ac	0.63	0.62	0.80	**0.94**	0.80	0.99	0.67	0.78
		Se	0.20	0.75	**0.86**	**0.96**	0.81	0.98	0.67	0.75
		Sp	0.86	0.38	0.77	**0.92**	0.77	0.99	**0.67**	0.77

Table 4.14: Experimental results for naive Bayes classifier on binary datasets after performing DOB-SCV with 5 folds.

			Brain	CNS	Colon	DLBCL	Gli85	Ovarian	Smk	Avg
No FS		Ac	0.67	0.63	0.58	**0.98**	0.87	0.94	0.63	0.76
		Se	0.00	0.69	0.72	**1.00**	0.91	0.98	0.58	0.70
		Sp	**1.00**	0.54	0.50	0.96	0.77	0.91	0.67	0.77
CFS		Ac	**0.92**	0.67	0.82	0.93	0.85	0.98	0.68	**0.84**
		Se	**0.80**	0.76	0.77	**1.00**	**0.93**	0.98	0.74	0.85
		Sp	**1.00**	0.50	0.85	0.87	0.66	0.98	0.62	0.78
FCBF		Ac	0.71	0.58	0.79	0.96	**0.88**	0.99	0.66	0.80
		Se	**0.80**	0.69	0.77	**1.00**	**0.93**	0.99	0.70	0.84
		Sp	0.70	0.40	0.80	0.92	0.77	**0.99**	0.62	0.74
INT		Ac	**0.92**	0.62	**0.84**	0.93	0.86	0.99	0.70	**0.84**
		Se	**0.80**	0.69	**0.86**	**1.00**	0.91	**1.00**	0.74	**0.86**
		Sp	**1.00**	0.50	0.82	0.87	0.73	**0.99**	0.65	0.79
IG	#10	Ac	**0.92**	0.60	0.77	0.91	0.87	0.96	0.68	0.82
		Se	**0.80**	0.59	0.72	0.96	0.89	0.95	0.66	0.80
		Sp	**1.00**	**0.64**	0.80	0.87	0.81	0.96	**0.71**	**0.83**
	#50	Ac	**0.92**	0.65	0.79	0.96	0.87	0.98	0.70	**0.84**
		Se	**0.80**	0.72	0.82	0.96	0.90	0.96	0.68	0.83
		Sp	**1.00**	0.54	0.77	0.96	0.81	**0.99**	**0.71**	**0.83**
ReliefF	#10	Ac	0.26	0.65	**0.84**	0.93	0.84	0.96	0.67	0.74
		Se	0.30	0.69	0.72	0.96	0.88	0.95	0.71	0.74
		Sp	0.20	0.57	**0.90**	0.91	0.74	0.96	0.62	0.70
	#50	Ac	0.21	0.67	**0.84**	0.96	0.86	0.98	0.68	0.74
		Se	0.30	0.71	0.72	0.96	0.86	0.95	**0.77**	0.75
		Sp	0.13	0.58	**0.90**	0.96	**0.85**	**0.99**	0.59	0.71
SVM-RFE	#10	Ac	0.58	**0.72**	0.76	0.94	0.83	**1.00**	0.70	0.79
		Se	0.20	0.76	0.64	0.91	0.86	**1.00**	0.75	0.73
		Sp	0.73	0.63	0.82	0.96	0.74	**0.99**	0.63	0.79
	#50	Ac	0.71	0.69	0.76	0.92	0.87	0.99	0.68	0.80
		Se	0.10	**0.77**	0.69	**1.00**	0.88	**1.00**	0.76	0.74
		Sp	**1.00**	0.54	0.80	0.84	**0.85**	**0.99**	0.59	0.80
mRMR	#10	Ac	0.75	0.68	0.82	**0.98**	0.87	0.99	**0.71**	0.83
		Se	**0.80**	0.74	0.77	**1.00**	**0.93**	0.98	0.74	0.85
		Sp	0.73	0.58	0.85	0.96	0.73	**0.99**	0.66	0.79
	#50	Ac	0.77	0.67	0.79	**0.98**	0.85	0.98	0.67	0.82
		Se	0.40	0.71	0.82	0.96	0.87	0.96	0.70	0.78
		Sp	**1.00**	0.59	0.77	**1.00**	0.77	**0.99**	0.64	0.82

Table 4.15: Experimental results for SVM classifier on binary datasets after performing regular 5-fold cross-validation.

			Brain	CNS	Colon	DLBCL	Gli85	Ovarian	Smk	Avg
No FS		Ac	**0.68**	0.67	0.77	**0.96**	**0.92**	**1.00**	**0.72**	**0.82**
		Se	0.20	0.82	0.60	0.96	**0.98**	**1.00**	0.79	0.77
		Sp	0.93	0.38	0.87	**0.96**	0.78	**1.00**	0.63	**0.79**
CFS		Ac	0.61	0.62	0.81	0.88	0.88	**1.00**	0.64	0.78
		Se	0.60	0.70	0.69	0.86	0.93	**1.00**	0.66	0.78
		Sp	0.66	0.49	0.87	0.88	0.77	**1.00**	0.61	0.76
FCBF		Ac	0.67	0.65	0.84	0.81	0.87	**1.00**	0.71	0.79
		Se	0.00	0.80	0.73	0.82	0.93	**1.00**	0.76	0.72
		Sp	**1.00**	0.39	0.90	0.79	0.73	**1.00**	0.64	0.78
INT		Ac	0.61	0.62	0.81	0.88	0.88	**1.00**	0.66	0.78
		Se	0.60	0.75	0.64	0.91	0.91	**1.00**	0.69	0.79
		Sp	0.66	0.39	0.90	0.83	0.81	**1.00**	0.63	0.75
IG	#10	Ac	0.48	0.63	0.81	0.94	0.91	0.98	0.64	0.77
		Se	0.00	0.82	0.59	0.96	**0.98**	0.96	0.74	0.72
		Sp	0.70	0.30	**0.92**	0.92	0.74	0.99	0.53	0.73
	#50	Ac	0.66	0.67	**0.85**	0.94	0.86	**1.00**	**0.72**	0.81
		Se	**0.80**	0.77	**0.81**	0.96	0.90	**1.00**	0.73	**0.85**
		Sp	0.66	0.48	0.87	0.92	0.77	0.99	0.70	0.77
ReliefF	#10	Ac	0.50	0.68	0.81	0.94	0.87	0.98	0.69	0.78
		Se	0.00	0.87	0.60	0.96	0.96	0.94	**0.82**	0.74
		Sp	0.73	0.34	**0.92**	0.92	0.66	0.99	0.54	0.73
	#50	Ac	0.35	**0.73**	**0.85**	0.92	0.89	**1.00**	0.69	0.78
		Se	0.00	0.82	0.72	**1.00**	0.93	**1.00**	0.74	0.74
		Sp	0.53	**0.58**	0.92	0.84	0.82	**1.00**	0.64	0.76
SVM-RFE	#10	Ac	0.62	**0.73**	0.73	0.87	0.86	**1.00**	0.70	0.79
		Se	0.20	0.84	0.56	0.87	0.88	**1.00**	0.78	0.73
		Sp	0.86	0.53	0.82	0.88	0.81	**1.00**	0.61	**0.79**
	#50	Ac	0.48	0.72	0.71	0.88	0.89	**1.00**	**0.72**	0.77
		Se	0.20	0.82	0.57	0.91	0.91	**1.00**	0.74	0.74
		Sp	0.63	0.53	0.80	0.84	**0.85**	**1.00**	**0.68**	0.76
mRMR	#10	Ac	0.53	0.65	0.77	0.92	0.89	**1.00**	0.68	0.78
		Se	0.60	**0.95**	0.56	0.96	0.95	0.99	0.74	0.82
		Sp	0.56	0.10	0.90	0.87	0.77	**1.00**	0.62	0.69
	#50	Ac	0.49	0.70	0.84	**0.96**	0.89	**1.00**	0.68	0.79
		Se	0.20	0.77	0.77	**1.00**	0.95	**1.00**	0.74	0.78
		Sp	0.63	0.57	0.87	0.92	0.77	**1.00**	0.62	0.77

Table 4.16: Experimental results for SVM classifier on binary datasets after performing DOB-SCV with 5 folds.

			Brain	CNS	Colon	DLBCL	Gli85	Ovarian	Smk	Avg
No FS		Ac	0.67	0.65	0.84	0.93	**0.91**	**1.00**	**0.72**	0.82
		Se	0.30	0.77	**0.82**	0.95	0.97	**1.00**	0.77	0.80
		Sp	0.86	0.44	0.85	0.92	0.77	**1.00**	0.66	**0.79**
CFS		Ac	**0.80**	0.66	0.82	0.94	0.89	**1.00**	0.68	**0.83**
		Se	0.70	0.74	0.78	**1.00**	0.97	0.99	0.70	0.84
		Sp	0.87	0.54	0.85	0.88	0.73	**1.00**	0.66	**0.79**
FCBF		Ac	0.67	0.58	0.79	0.92	0.90	0.99	0.66	0.79
		Se	0.00	0.71	0.68	**1.00**	0.93	0.98	0.68	0.71
		Sp	**1.00**	0.35	0.85	0.84	**0.81**	**1.00**	0.63	0.78
INT		Ac	**0.80**	0.60	0.77	0.91	0.87	**1.00**	**0.72**	0.81
		Se	0.70	0.66	0.63	0.95	0.91	0.99	0.74	0.80
		Sp	0.87	0.49	0.85	0.88	0.77	**1.00**	0.68	**0.79**
IG	#10	Ac	0.62	0.65	0.79	0.91	**0.91**	0.96	0.68	0.79
		Se	0.00	0.81	0.58	0.96	**0.98**	0.94	0.67	0.71
		Sp	0.93	0.34	0.90	0.87	0.73	0.97	0.70	0.78
	#50	Ac	0.66	0.67	**0.85**	**0.98**	0.87	**1.00**	0.65	0.81
		Se	**0.80**	0.74	**0.82**	**1.00**	0.93	**1.00**	0.70	**0.86**
		Sp	0.63	0.54	0.87	**0.96**	0.73	**1.00**	0.60	0.76
ReliefF	#10	Ac	0.45	0.63	0.82	0.95	0.84	0.98	0.65	0.76
		Se	0.10	0.84	0.58	**1.00**	0.93	0.93	0.75	0.73
		Sp	0.60	0.25	**0.95**	0.91	0.62	**1.00**	0.54	0.70
	#50	Ac	0.64	0.63	0.84	0.94	0.88	0.99	**0.72**	0.81
		Se	0.30	0.74	0.81	**1.00**	0.95	0.98	**0.83**	0.80
		Sp	0.80	0.43	0.85	0.88	0.73	**1.00**	0.58	0.75
SVM-RFE	#10	Ac	0.62	0.67	0.76	0.94	0.85	**1.00**	0.67	0.79
		Se	0.10	0.71	0.60	**1.00**	0.95	**1.00**	0.74	0.73
		Sp	0.86	0.58	0.85	0.88	0.61	**1.00**	0.60	0.77
	#50	Ac	0.67	0.65	0.81	0.91	0.86	**1.00**	0.71	0.80
		Se	0.30	0.72	0.78	0.95	0.91	**1.00**	0.70	0.77
		Sp	0.86	0.54	0.82	0.88	0.73	**1.00**	**0.71**	**0.79**
mRMR	#10	Ac	0.45	0.65	0.82	**0.98**	0.87	**1.00**	0.71	0.78
		Se	0.20	**0.87**	0.72	**1.00**	0.95	**1.00**	0.74	0.78
		Sp	0.60	0.24	0.87	**0.96**	0.69	**1.00**	0.66	0.72
	#50	Ac	0.58	**0.70**	0.84	0.93	0.87	**1.00**	0.66	0.80
		Se	0.10	0.74	0.78	0.95	0.95	**1.00**	0.73	0.75
		Sp	0.80	**0.63**	0.87	0.92	0.70	**1.00**	0.57	0.78

4.5.2.1 Analysis of algorithms

This subsection aims to analyze the behavior of the feature selection methods as well as the influence of the classifier on the studied datasets. Some interesting conclusions can be extracted by looking at the results reported by Tables 4.11 - 4.16.

1. The best performances are obtained by SVM and naive Bayes classifiers. As mentioned above, some studies (Gonzalez-Navarro, 2011) stated the superiority of SVMs over other classifiers. On the other hand, the performance of C4.5 may be affected by its embedded feature selection, in some cases leading to a extremely reduced set of features which can degrade the classification accuracy.

2. Focusing on the feature selection methods, in average for all datasets, the subsets filters show an outstanding behavior, specially CFS and INTERACT. It is surprising that SVM-RFE did not achieve the best results when combined with SVM classifier, but the poor performance of the ranker methods can be explained by the restriction of having to establish a threshold for the number of features to retain. In the case of the subsets filters, the number of features which conform the final subset of features is the optimal one for a given dataset and method. However, in the case of rankers, since this number has to be set *a priori*, it may result too small or too large, being this the main disadvantage of using this type of methods.

4.5.2.2 Cross-validation Vs. DOB-SCV

The purpose of this subsection is to check the adequacy of performing DOB-SCV instead of regular 5-fold cross-validation. In average for all datasets, DOB-SCV obtains better results than regular cross-validation for SVM and naive Bayes classifiers, which are the classifiers which showed the best overall performance. It is interesting to focus on the case of Brain dataset, which presents a high imbalance and an important amount of overlapping, as can be seen in Table 4.6. For this dataset, DOB-SCV outperforms regular cross-validation for SVM and naive Bayes classifiers, although the highest accuracy was achieved by C4.5 combined with the regular cross-validation. In the case of CNS, another complex dataset, there are no important differences between the two validation methods. On the other hand, there is the DLBCL dataset, which is in theory a simpler dataset than CNS and Brain (see Table 4.6). In fact, the accuracy obtained by

the classifiers is in most of the cases around 90%. Nevertheless, it is interesting to note that for this dataset, DOB-SCV outperforms, in average, the regular cross-validation. This datasets will be studied in detail in Section 4.5.2.3.

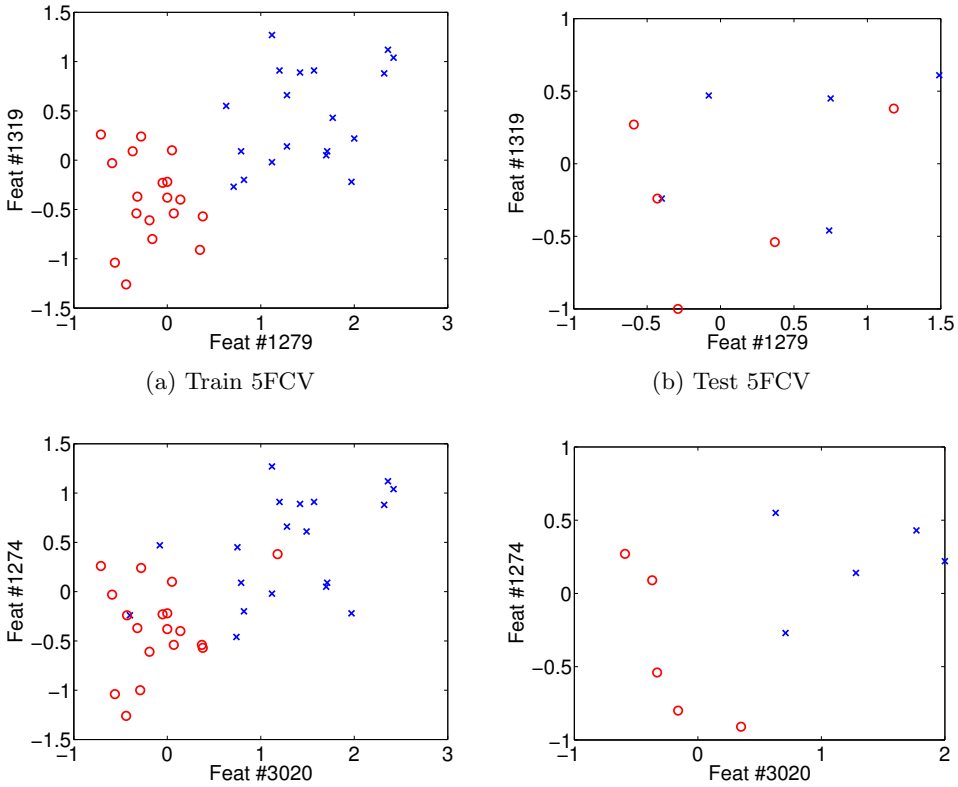
4.5.2.3 Analysis of datasets characteristics

As mentioned in Section 4.2, microarray datasets present several problematics such as the imbalance of the data, the overlapping between classes or the dataset shift. Tables 4.5 and 4.6 reported the imbalance ratio and F1 of the datasets studied in this section, which measure the imbalance of the data and the overlapping, respectively. GLI is the most unbalanced dataset, and its highest accuracy was obtained by SVM with a regular 5-fold cross validation and no feature selection (92%), although the Information Gain filter achieves a 91% of accuracy and this degradation can be equivalent to missclassify only one or two samples.

SMK is the dataset which presents a higher level of overlapping between classes, and its maximum classification accuracy is very poor, around 70%. A similar case happens with CNS dataset, which has also a low value of F1 (see Table 4.6).

Regarding the dataset shift problem and the adequacy of DOB-SCV to solve it, we will analyze in detail the case of DLBCL dataset. Figure 4.4 displays the 2-D representation of the first two features selected by mRMR in the first fold of a 5-fold cross-validation and a 5DOB-SCV for both train and test sets, where different colors stand for different classes. As can be seen, cross-validation can indeed introduce dataset shift. The two first features selected by mRMR obtain a linearly separable problem (see Figure 4.4a) in the train set, but these features are not so informative in the test set (see Figure 4.4b). However, the partitions created by DOB-SCV do not suffer from dataset shift. In Figure 4.4c, the two first features selected by mRMR in the training set make the problem almost linearly separable and this condition is maintained in the test set. In fact, it has been demonstrated in the previous section that DOB-SCV outperformed the regular cross-validation for this dataset.

In light of the above, it can be seen that the results displayed in this experimental study are very dependent of the classifier, the feature selection method, and specially the dataset. Although an analysis in detail of the results is out of the scope of this chapter, it is easy to realize that the large number of problematics present in this



(a) Train 5FCV (b) Test 5FCV

(c) Train 5DOB-SCV (d) Test 5DOB-SCV

Figure 4.4: Two first features selected by mRMR in the first fold for both 5-fold cross-validation and 5DOB-SCV

type of datasets make the classification task very arduous. In such a situation, it is recommended to study the particularities of each problem carefully, although it seems that the best results (in general) are obtained with SVM classifier, a subset filter for feature selection, and the DOB-SCV validation method.

4.6 Summary

This chapter has reviewed the up-to-date contributions of feature selection research applied to the field of DNA microarray data analysis. The advent of this type of data has posed a big challenge for machine learning researchers, because of the large input dimensionality and small sample size. Since the infancy of microarray data classification, feature selection became an imperative step, in order to reduce the number of features (genes).

Since the end of the nineties, when microarray datasets began to be dealt with, a large number of feature selection methods were applied. In the literature one can find both classical methods and methods developed especially for this kind of data. Due to the high computational resources that these datasets demand, wrapper and embedded methods have been mostly avoided, in favor of less expensive approaches such as filters.

The key point to understand all the attention devoted to microarray data is the challenge that their problematic poses. Besides the obvious disadvantage of having so much features for such a small number of samples, researchers have to deal also with classes which are very unbalanced, training and test datasets extracted under different conditions, dataset shift or the presence of outliers. This is the reason because new methods emerge every year, not only trying to improve previous results in terms of classification accuracy, but also aiming to help biologists to identify the underlying mechanism that relates gene expression to diseases.

The objective of this chapter is not in any way to evaluate feature selection methods for microarray data in terms of which one is the best, but to gather as much as possible up-to-date knowledge for the interested reader. Bearing this in mind, the recent literature has been analyzed in order to give the reader a brushstroke about the tendency in developing feature selection methods for microarray data. In order to have a complete picture on the topic, the most common validation techniques have been also mentioned. Since there is no consensus in the literature about this issue, some guidelines have been provided.

Finally, a framework for feature selection evaluation in microarray datasets and a practical evaluation have been provided, where the results obtained have been analyzed. This experimental study tries to show in practice the problematics explained in theory. For this sake, a suite of 9 widely-used binary datasets was chosen to apply over them 7 classical feature selection methods. For obtaining the final classification accuracy, 3 well-known classifiers were used. This large set of experiments aims also at facilitating future comparative studies when a researcher proposes a new method.

Regarding the opportunities for future feature selection research, the tendency is toward focusing on new combinations such as hybrid or ensemble methods. This type of methods are able to enhance the robustness of the final subset of selected features, which is also a trending topic in this domain. Another interesting line of future research might be to distribute the microarray data vertically (i.e. by features) in order to reduce the heavy computational burden when applying wrapper methods.

CHAPTER 5

Feature selection in other real applications

This chapter is devoted to prove the benefits of feature selection in other real applications apart from DNA microarray data. Feature selection may be very useful in real domains, since it allows to decrease the storage costs, to improve the performance of a classifier and to obtain a good understanding of the learned model. Two cases when feature selection has reported success will be presented: classification of the tear film lipid layer and the K-complex classification.

5.1 Tear film lipid layer classification

Dry eye is a symptomatic disease which affects a wide range of population and has a negative impact on their daily activities. Its diagnosis can be achieved by analyzing the interference patterns of the tear film lipid layer and by classifying them into one of the Guillon categories (Guillon, 1998): open meshwork, closed meshwork, wave, amorphous and color fringe. However, the classification into these grades is a difficult clinical task, especially with thinner lipid layers that lack color and/or morphological features. The subjective interpretation of the experts via visual inspection may affect the classification. This time-consuming task is very dependent on the training and experience of the optometrist(s), and so produces a high degree of inter- and also intra-observer variability (García-Resúa et al., 2013). The development of a systematic and objective computerized method for analysis and classification is thus highly desirable, allowing for homogeneous diagnosis and relieving the experts from this tedious task.

Remeseiro et al. (2011) proposed a wide set of feature vectors using different texture analysis methods in three color spaces and a 95% of classification accuracy was obtained. Nevertheless, the problem with this approach is that the time required to extract some of the textural features is too long (more than 1 minute). Interviews with optometrists

revealed that a scale of computation time over 10 seconds per image makes the system not usable. Therefore, the previous approach prevents the practical clinical use of an application developed to automatize the process, because it could not work in real time. So, the objective of our work in the field is to obtain a reduction in time that allows for the system to be used in daily clinical routine. In order to deal with this problem, feature selection can play a crucial role. Because the number of features to extract and process will be reduced, the time required will be also reduced in consonance, and most of the times, this can be achieved with a minimum degradation of performance.

In order to obtain an efficient method for automatic tear film lipid layer classification, a five-step methodology is applied as illustrated in Figure 5.1. First, feature extraction is performed, after which feature selection methods are applied to select the subset of relevant features, that allow for correct classification of the tear film lipid layer thickness. After that, several performance measures are computed to evaluate the performance of the system. Finally, a multiple-criteria decision-making method is carried out to obtain a final result.

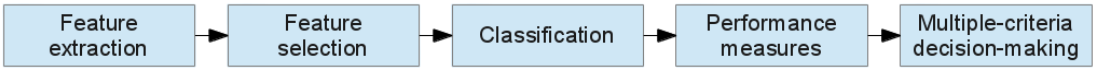


Figure 5.1: Steps of the research methodology.

The initial stage in this methodology is the processing of tear film images, in order to extract their features. Firstly, the region of interest of an input image in RGB is extracted. Then, this extracted region in RGB is converted to the Lab colour space and its channels L, a and b are obtained. After that, the texture features of each channel are extracted and three individual descriptors are generated. Finally, the three individual descriptors are concatenated in order to generate the final descriptor of the input image which contains its colour and texture features. Five different techniques for texture analysis are tested in this study:

- *Butterworth bandpass filters* (Gonzalez & Woods, 2008) are frequency domain filters that have an approximately flat response in the bandpass frequency, which gradually decays in the stopband. The order of the filter defines the slope of the decay; the higher the order, the faster the decay.

- The *discrete wavelet transform* (Mallat, 1989) generates a set of wavelets by scaling and translating a *mother wavelet*. The wavelet decomposition of an image consists of applying these wavelets horizontally and vertically, generating 4 im-

ages (LL, LH, HL, HH). The process is repeated iteratively on the LL subimage resulting in the standard pyramidal wavelet decomposition.

- *Co-occurrence features* method was introduced by Haralick, Shanmugam, and Dinstein (1973), which is based on the computation of the conditional joint probabilities of all pairwise combinations of gray levels. This method generates a set of grey level co-occurrence matrices and extracts several statistics from their elements. In general, the number of orientations and so of matrices for a distance d is $4d$.

- *Markov random fields* (Woods, 1972) generate a texture model by expressing the grey values of each pixel in an image as a function of the grey values in its neighborhood.

- *Gabor filters* (Gabor, 1946) are complex exponential functions modulated by Gaussian functions. The parameters of Gabor filters define their shape, and represent their location in the spatial and frequency domains.

Table 5.1 shows the arrangements for applying the texture analysis methods. Note that column *No. of features* contains the number of features generated by each method, in which they are always multiplied by 3 because of the use of Lab.

Two bank datasets are selected to test the proposed methodology. There is a first bank which contains 105 images from VOPTICAL_I1 dataset (VOPTICAL_I1, n.d.), all of them taken over the same illumination conditions, which are considered to be the optimum ones by practitioners. This dataset contains the samples that are expected to be obtained in a real case situation and will be used to compute the performance of algorithms. It includes 29 open meshwork, 29 closed meshwork, 25 wave and 22 color fringe images. The second bank contains 406 images from VOPTICAL_Is dataset (VOPTICAL_Is, n.d.), taken over different illumination conditions. This bank will be used only to evaluate the sensibility of algorithms to *noisy* data. It includes 159 open meshwork, 117 closed meshwork, 90 wave and 40 color fringe images.

The performance measures considered are the following:

- The *accuracy* is the percentage of correctly classified instances on a dataset with optimum illumination.

Table 5.1: Arrangements for texture analysis methods and number of features.

Texture analysis	Configuration (per component)	No. features
Butterworth filters	A bank of Butterworth bandpass filters composed of 9 second order filters was used, with bandpass frequencies covering the whole frequency spectrum. The filter bank maps each input image to 9 output images, one per frequency band. Each output image was normalised separately and then an uniform histogram with non-equidistant bins (Ramos et al., 2011) was computed. Since 16 bin histograms were used, the feature vectors contain 16 components per frequency band.	$9 \times 16 \times 3 = 432$
Discrete wavelet transform	A Haar algorithm (Mallat, 1989) was applied as the *mother wavelet*. The descriptor of an input image is constructed calculating the mean and the absolute average deviation of the input and LL images, and the energy of the LH, HL and HH images. Since 2 scales were used, the feature vectors contain 12 components.	$12 \times 3 = 36$
Co-occurrence features	A set of 14 statistics proposed by Haralick et al. (1973) was computed from each co-occurrence matrix. These statistics represent features such as homogeneity or contrast. The descriptor of an image consists of 2 properties, the mean and range across matrices of these statistics, thus obtaining a feature vector with 28 components per distance. Distances from 1 to 7 were considered.	$28 \times 7 \times 3 = 588$
Markov random fields	In this work, the neighborhood of a pixel is defined as the set of pixels within a Chebyshev distance d. To generate the descriptor, the directional variances proposed by Çesmeli and Wang (2001) were used. For a distance d, the descriptor comprises $4d$ features. Distances from 1 to 10 were considered.	$(\Sigma_{d=1}^{10} 4d) \times 3 = 660$
Gabor filters	A bank of 16 Gabor filters centered at 4 frequencies and 4 orientations was created. The filter bank maps each input image to 16 output images, one per frequency-orientation pair. Using the same idea as in *Butterworth Filters*, the descriptor of each output image is its uniform histogram with non-equidistant bins. Since 7 bin histograms were used, the feature vectors contain 7 components per filter.	$16 \times 7 \times 3 = 336$

- The *robustness* is the classification accuracy in a noisy dataset, i.e. its accuracy when the images in the dataset show illuminations outside the optimum range. This measure is related to the generalization ability of the method when handling noisy inputs. Notice that the higher the robustness the higher the generalization performance.
- The *feature extraction time* is the time that the texture analysis methods take to extract the selected features of a single image. Notice that this does not include the training time of the classifier, which is not relevant for practical applications because the classifier will be trained off-line. This also applies to FS, which is a pre-processing step that is performed off-line.

The experimental procedure is detailed as follows,

1. Apply the five texture analysis methods (see Table 5.1) to the two banks of images. Moreover, the concatenation of all the features extracted by these five methods is also considered. As a result, six datasets with optimum illumination (105 images) and six datasets with different illuminations (406 images) are available. Notice that the feature extraction method chosen determines the number of features of the datasets, as can be seen in the first column of Table 5.2.
2. Apply three feature subset selection methods (CFS, consistency-based and INTERACT, see Chapter 2) to the datasets with optimum illumination to provide the subset of features that describe properly the given problem.
3. Train a SVM classifier (see Appendix I) with radial basis kernel and automatic parameter estimation. A 10-fold cross validation is used, so the average error across all 10 trials is computed.
4. Evaluate the effectiveness of feature selection in terms of three performance measures (accuracy, robustness to noise and feature extraction time), by means of the multiple-criteria decision-making method TOPSIS (see Appendix I, Section I.6.1).

The results obtained with and without feature selection are compared in terms of the three performance measures described above. Bear in mind that the column *None* in the tables shows the results when no feature selection was performed, and *Concatenation* stands for the concatenation of all the features of the five texture analysis

methods. Note that the experimentation was performed on an Intel®Core^{TM}i5 CPU 760 @ 2.80GHz with RAM 4 GB.

The number of features selected by each of the three feature selection filters is summarized in Table 5.2. In average, CFS, Consistency-based filter (*Cons*) and INTERACT (*INT*) retain only the 4.9%, 1.6% and 3.2% of the features, respectively.

Table 5.2: Number of features.

Texture analysis	Feature selection filter			
	None	CFS	Cons	INT
Butterworth filters	432	26	6	14
Discrete wavelet transform	36	10	8	7
Co-occurrence features	588	27	6	21
Markov random fields	660	24	13	15
Gabor filters	336	29	7	18
Concatenation	2052	56	6	29

5.1.1 Classification accuracy

Table 5.3 shows the test accuracies for all pairwise texture analysis and feature selection methods after applying the SVM classifier over the VOPTICAL_I1 dataset. The best result for each is marked in bold face. As can be seen, all texture analysis techniques perform quite well providing results over 84% accuracy. The best result is generated by the concatenation of all methods. Individually, Gabor filters and co-occurrence features without feature selection outperform the other methods. In fact, feature selection outperforms primal results in three out of six methods (Butterworth filters, the discrete wavelet transform and Markov random fields), while accuracy is almost maintained in co-occurrence features and the concatenation of all methods when CFS is applied. As a conclusion, the best result is obtained by using the concatenation of all when no feature selection is performed (97.14%). Closely, the discrete wavelet transform with INTERACT filter and the concatenation of all with CFS obtain an accuracy of 96.19%. Notice that although these results improve slightly the previous ones in accuracy (Remeseiro et al., 2011) (95%), the goal of the work presented herein is to reduce the processing time whilst maintaining accuracy in order to be able to use the system in the daily clinical routine.

Table 5.3: Mean test classification accuracy (%), VOPTICAL_I1 dataset.

Texture analysis	Feature selection filter			
	None	CFS	Cons	INT
Butterworth filters	91.42	**93.33**	83.81	86.67
Discrete wavelet transform	88.57	91.43	94.29	**96.19**
Co-occurrence features	**95.24**	94.29	86.67	93.33
Markov random fields	84.76	**85.71**	83.81	75.24
Gabor filters	**95.24**	91.43	86.67	86.67
Concatenation	**97.14**	96.19	87.62	93.33

5.1.2 Robustness to noise

In some cases, data are taken over different illumination conditions which are not optimal. For this reason, it is also necessary to evaluate the sensibility of the proposed methodology to noisy data. Table 5.4 shows the robustness of the six different methods over VOPTICAL_IS dataset. Co-occurrence features and the concatenation of all obtain remarkable better results than the remainder methods. Both methods obtain values of robustness over 90% for some configurations. In particular, the best result is obtained by using the concatenation of all methods when CFS filter is used (93.84%). In relative terms, co-occurrence features and the concatenation of all methods deteriorate their mean classification accuracy by 2.66% and 4.59%, respectively (mean differences between the values contained in Tables 5.3 and 5.4).

Table 5.4: Robustness: mean test accuracy (%), VOPTICAL_IS dataset.

Texture analysis	Feature selection filter			
	None	CFS	Cons	INT
Butterworth filters	**88.18**	84.98	71.92	79.56
Discrete wavelet transform	82.27	**88.18**	86.21	86.70
Co-occurrence features	92.17	**92.36**	85.22	89.16
Markov random fields	**83.99**	76.35	70.94	70.69
Gabor filters	**89.90**	85.22	69.46	82.51
Concatenation	92.61	**93.84**	77.59	91.87

However, the remainder methods deteriorate their mean classification accuracy by between 6.78% and 8.23%. Note also that the illumination levels affect the robustness in different degrees. The brighter the illumination, the lower the robustness to noise. This also happens to practitioners when performing this task manually. For this reason, their experience to control the illumination level during the acquisition stage is cornerstone for ensuring good classification performance.

5.1.3 Feature extraction time

Tear film lipid layer classification is a real-time task so the time a method takes to process an image should not be a bottleneck. After applying feature selection and so reducing the number of input attributes, the time needed for analyzing a single image with any of the six methods was also reduced as can be seen in Table 5.5.

Table 5.5: Feature extraction time (s).

Texture analysis	Feature selection filter			
	None	CFS	Cons	INT
Butterworth filters	0.22	0.15	**0.04**	0.07
Discrete wavelet transform	0.03	**0.01**	**0.01**	**0.01**
Co-occurrence features	102.18	27.01	**0.05**	9.86
Markov random fields	13.83	0.50	**0.27**	0.31
Gabor filters	0.42	0.18	**0.06**	0.11
Concatenation	116.68	37.04	**0.05**	9.96

In general terms, Butterworth filters, the discrete wavelet transform and Gabor filters take a negligible lapse of time to extract the features of an image (regardless of whether or not feature selection is applied as preprocessing step). Moreover, Markov random fields takes a time which could be acceptable for practical applications, even when no feature selection is applied, although it could not work in real time. The co-occurrence features method has been known to be slow and, despite the authors implemented an optimization of the method (Clausi & Jernigan, 1998), it presents an unacceptable extraction time. Regarding the time of the concatenation of all methods, note that it is influenced by the time of co-occurrence features. Co-occurrence features and the concatenation of all methods are only acceptable for practical applications when consistency-based or INTERACT filters are used. Consistency-based filter selects fewer

features (see Table 5.2) and consequently the processing time when this filter is used is smaller. Co-occurrence features is the core behind the good performance of the concatenation of all methods. This is demonstrated by further experiments showing that the concatenation of the other four methods achieves a maximum accuracy of 93.33% and robustness of 88.91%. These results are significantly worse (around 4%) than the best results obtained by the concatenation of all methods.

5.1.4 Overall analysis

In general terms, we can assert that in a field with a very large number of features, feature selection filters play a significant role to reduce the cost of obtaining data and the complexity of the classifier. Consistency-based filter performed the most aggressive selection retaining only the 1.6% of the features (see Table 5.2). CFS retained three times more features (4.9%) than the former. Halfway, INTERACT selected in average 3.2% of features. Moreover, in most cases the test accuracy is improved or maintained with a remarkable reduction in the number of features when feature selection is used (see Table 5.3). The effectiveness of feature selection on tear film lipid layer classification was then demonstrated, paving the way for its use in daily clinical routine.

Evaluating the performance of the methods for texture analysis presented herein is a multi-objective problem defined in terms of accuracy, robustness, and feature extraction time. Butterworth filters, the discrete wavelet transform and Gabor filters obtain competitive classification accuracies in short spans of time (see Tables 5.3 and 5.5). However, these methods are very sensitive to noisy data (see Table 5.4) which make them inappropriate for practical applications. On the other hand, the co-occurrence features method presents competitive results in classification accuracy and generalisation (see Tables 5.3 and 5.5). However, the time the method takes to extract its features is an impediment (see Table 5.5). The concatenation of all methods improves the previous results but at the expense of an even longer feature extraction time.

Table 5.6 shows the TOPSIS values obtained for every method when the weights of each criteria are set equally. Note that the larger the value the better the method. The top 3 methods are marked in bold. As can be seen, those methods with the best balance among classification accuracy, robustness to noise and feature extraction time are ranked in higher positions. In particular, Gabor filters with no feature selection, the discrete wavelet transform with INTERACT filter, and the concatenation of all

methods with INTERACT filter rank in the top 3 positions of the ranking. However, those methods with good performance in accuracy and robustness but very long feature extraction time are penalized. e.g. co-occurrence features or the concatenation of all methods, with no feature selection in both cases.

Table 5.6: TOPSIS values obtained for every method when $w = [1/3, 1/3, 1/3]$

Texture analysis	Feature selection filter			
	None	CFS	Cons	INT
Butterworth filters	0.9774	0.9773	0.8159	0.9008
Discrete wavelet transform	0.9344	0.9775	0.9848	**0.9900**
Co-occurrence features	0.3431	0.9416	0.9281	0.9812
Markov random fields	0.8670	0.8691	0.8081	0.6923
Gabor filters	**0.9954**	0.9686	0.8295	0.9164
Concatenation	0.3066	0.8986	0.8988	**0.9853**

A more detailed look at the results contained in Tables 5.3, 5.4 and 5.5 reveals that the combination of all methods in both CFS filtering configuration and without feature selection obtain the best results in terms of accuracy and robustness. These two configurations are in the Pareto front (Teich, 2001) of accuracy versus robustness (see Figure 5.2). In multi-objective optimization, the Pareto front is defined as the border between the region of feasible points (not strictly dominated by any other), for which all constraints are satisfied, and the region of unfeasible points (dominated by others). In this case, solutions are constrained to maximize accuracy and robustness.

The suitability of these two solutions to the problem in question is also corroborated by TOPSIS. Table 5.7 shows the TOPSIS values when only accuracy and robustness are considered (note that the third term in the weight vector is considered to be the feature extraction time). The concatenation of all methods without feature selection and with CFS filtering rank first and second, respectively.

However, the time for extracting the features must be shortened for practical applications. Thus, a case of study, in which a deeper analysis for feature selection is carried out, is presented in the next section. Note that the number of features in the concatenation of all methods without feature selection is too large (2052 features) to be optimized by hand. Therefore, we will focus on the concatenation of all methods with CFS (56 features).

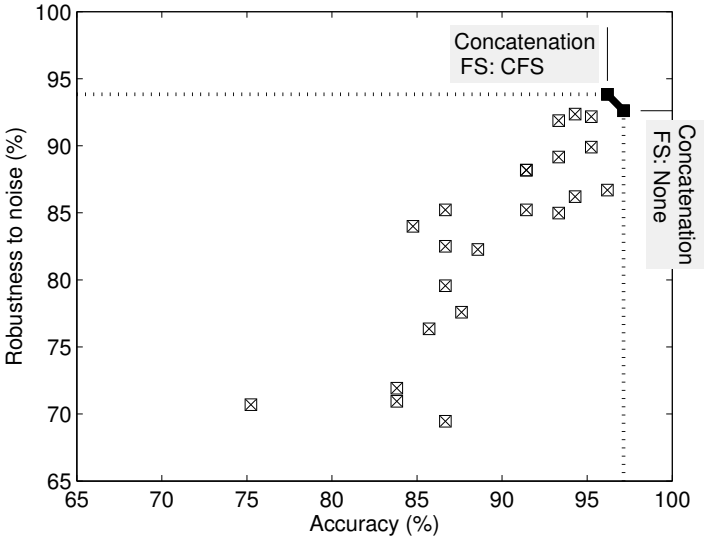


Figure 5.2: Pareto front of a multi-objective optimization problem based on accuracy and robustness to noise.

5.1.5 The concatenation of all methods with CFS: a case of study

When using feature selection, features are selected according to some specific criteria depending on the method. Filters remove features based on redundancy and relevance, but they do not take into account costs for obtaining them. Note that the cost for obtaining a feature depends on the procedures required to extract it. Therefore, each feature has an associated cost that can be related to financial cost, physical risk or computation demands. This is the case of co-occurrence features and, consequently, the concatenation of all methods. In co-occurrence features the cost for obtaining the 588 features is not homogeneous. Features are vectorized in groups of 28 related to distances and channels in the color space. Each group of 28 features corresponds with the mean and range of 14 statistics across the co-occurrence matrices.

If we focus on co-occurrence features when using CFS, the number of features was reduced by 95.41% (from 588 to 27) but the processing time was not reduced in the same proportion, being 27.01 instead of the initial 102.18 seconds (a reduction of 73.57%). This fact clearly shows that computing some of the 588 features takes longer than others. Some experimentation was performed on the time the method takes to compute each of the 14 statistics. Results disclosed that computing the 14^{th} statistic,

Table 5.7: TOPSIS values obtained for every method when $w = [1/2, 1/2, 0]$

Texture analysis	Feature selection filter			
	None	CFS	Cons	INT
Butterworth filters	0.8983	0.9017	0.1694	0.4722
Discrete wavelet transform	0.6567	0.8986	0.9377	0.9629
Co-occurrence features	**0.9923**	0.9846	0.6233	0.9539
Markov random fields	0.4777	0.3361	0.1601	0.0009
Gabor filters	0.9829	0.8526	0.2717	0.5526
Concatenation	**0.9991**	**0.9987**	0.4769	0.9706

which corresponds with the maximal correlation coefficient (Haralick et al., 1973), takes around 96% of the total time. So the time for obtaining a single matrix is negligible compared to the time for computing the 14^{th} statistic. Therefore, the key for reducing the feature extraction time is to reduce the number of 14^{th} statistics in the selection.

Table 5.8: Co-occurrence features selected by CFS over the concatenation of all methods, in which features corresponding with 14^{th} statistic are marked in bold.

Distance	Component in the colour space		
	L	a	b
1	–	29,50	66
2	**98**	121,133	–
3	193	–	230
4	267,268,275,276,277	–	321
5	**350**,359	–	–
6	**434**,443,446	–	492,502
7	**518**	**546**	576

In the case of the concatenation of all methods with CFS, the filter selects 56 features (see Table 5.2) distributed as follows: 17 features of Butterworth filters, 1 of the discrete wavelet transform, 24 of co-occurrence features, 1 of Markov random fields, and 13 of Gabor filters. Five of the features selected in co-occurrence features correspond with the 14^{th} statistic (see Table 5.8). In co-occurrence features, the cost of obtaining the statistics also depends on the distance and component in the color space. On the one hand, the longer the distance the larger the number of matrices to compute

(and so, the higher the processing time). On the other hand, the differences of color have little contrast so the colorimetric components of the Lab color space are minimal. As a consequence, the matrices within components a and b have smaller dimension than the matrices within component L. As expected, the smaller the dimension the shorter the time to compute a statistic.

Computing the five 14^{th} statistics in the different distances and components take: 3.12 s (feature 98), 8.23 s (feature 350), 9.61 s (feature 434), 11.49 s (feature 518), and 4.81 s (feature 546). As can be seen, avoiding computing some of them will entail saving a significant amount of time. The aim here is to explore the impact of removing some of the five 14^{th} statistics selected by CFS in terms of accuracy, robustness and time. There are 5 features within the 14^{th} statistic so only $2^5 = 32$ different configurations need to be explored. An empirical evaluation of brute force is acceptable. Table 5.9 shows the performance of the different configurations in terms of accuracy, robustness and time. Each configuration corresponds with those features selected by CFS removing some 14^{th} statistics. For purposes of simplicity, only the acceptable results are shown. It is assumed that one solution is unacceptable if it obtains a lower accuracy and robustness in a longer span of time than other.

Table 5.9: Performance measures for the concatenation of all methods with CFS when some of the five 14^{th} statistics are not selected. The best results are marked in bold.

Features removed	Acc (%)	Rob (%)	Time (s)
{}, *baseline performance*	96.19	93.84	37.04
{98, 434}	**97.14**	**94.09**	24.31
{98, 434, 546}	**97.14**	93.84	19.83
{98, 350, 518, 546}	**97.14**	93.60	9.72
{98, 434, 518, 546}	**97.14**	92.86	8.34
{98, 350, 434, 518, 546}	**97.14**	92.61	**0.11**

In terms of accuracy and robustness to noisy data, the best result is obtained when removing the features {98, 434} (results of 97.14% and 94.09%, respectively), but at the expense of a quite long lapse of time (24.31). Note that this result even improves the baseline performance. In the remainder results, the classification accuracy is maintained whilst the feature extraction time is reduced, only at the expense of a slightly deterioration in terms of robustness to noise (less than 2%).

It is also important to remark the effectiveness of CFS filter for selecting the most appropriate features. If we do not apply feature selection and we simply remove the 14^{th} statistics from the 588 features corresponding with co-occurrence features in the concatenation of all methods, the accuracy and the robustness are 92.86% for both of them. That is, the accuracy is worse than the results shown in Table 5.9 and the robustness is not significantly different. As expected, the time is also longer: 14.74 seconds.

To sum up, the manual process done by experts could be automatized with the benefits of being faster and unaffected by subjective factors, with maximum accuracy over 97% and processing time under 1 second. The clinical significance of these results should be highlighted, as the agreement between subjective observers is between 91%-100%.

5.2 K-complex classification

Sleep staging classification is one of the most important tasks within the context of sleep studies. For the characterization of patient's sleep macro structure, three non REM (Rapid Eye Movement) stages are identified. Then, to help in the sleep stage characterization, a micro structural analysis is also necessary. Transient events such as micro arousals, sleep spindles, K-complexes and other patterns should be analyzed.

According to the current American Academy of Sleep Medicine (AASM) (Iber, Ancoli-Israel, Chesson, & Quan, 2007), the K-complex is a "well-delineated negative sharp wave immediately followed by a positive component standing out from the background Electroencephalogram (EEG), with total duration $\geq$ 0.5 sec, usually maximal in amplitude when recorded using frontal derivations". The K-complex is one of the key features that contributes to sleep stages assessment, specifically is one of the hallmarks of stage 2. Unfortunately, their visual identification is very time-consuming –there are typically 1 to 3 K-complexes per minute in stage 2 of young adults (Kryger, Roth, & Dement, 2005)– and rather dependent on the knowledge and experience of the clinician since it cannot be performed on regular basis. This is the reason why automatic identification of K-complexes is of great interest. The main difficulty of the automated K-complex identification problem has been the lack of specific characterization of the wave and the similarity to other EEG waves as delta or vertex waves.

Based on shape analysis, one of the most relevant works in the K-complex detection so far has been the one by Bankman, Sigillito, Wise, and Smith (1992), where a feature-based detection approach is presented. However, analysis of the relevance of the different features has not been carried out, so classification of K-complexes based on shape analysis may take advantage of the application of feature selection methods. An automatic methodology is then proposed aiming at obtaining a method that achieves the best accuracy results with a low false positive rate in the K-complex classification task. Over the EEG signal of a set sleep recordings, after applying a band filter, a set of isolated waveforms are obtained. Using these patterns, two approaches were tested. The first one uses the set of 14 Bankman's features (Bankman et al., 1992) and over a set of classifiers, chooses the best one in terms of accuracy. The second approach uses feature selection over the 14 features previously mentioned to see if there exist irrelevant features and, again, to choose the best classifier in terms of accuracy with the selected features. The goal of this second approach is to achieve comparable or better results than the first one, and also check the existence of possible redundant or irrelevant features, that could be discarded so as to obtain a simpler final model. An outline of the proposed methodology is shown in Figure 5.3.

- **Data processing:** The first step is to process the available EEG signals, which are known to be very sensitive to noise. For this reason, the raw data was digitally filtered using two different criteria. In the first one, a band-pass filter is applied in the 0.5-2.3 Hz frequency band, being the resulting data identified as $Data_f$. In the second case, a less conservative approach was investigated, where only the very high frequency components were filtered out by means of a low-pass filter with cut-off at 18 Hz. The resulting data is referred as $Data_{f2}$. Then, the isolet waveforms are obtained. A set of positive and negative examples are identified by the medical expert, calculating the Bankman features for each recording over signal segments. Both datasets $Data_f$ and $Data_{f2}$ have 222 instances (111 positive and 111 negative examples), extracted from 32 different recordings scored at Brain Research Centre of the Medical University of Vienna.

- **Feature extraction:** A total of 14 features are extracted as defined by Bankman et al. (1992), based on several amplitude and duration measurements taken on significant points of the K-complex waveform.

- **Feature selection:** Three well-known feature selection methods (CFS, consistency-based and INTERACT, see Chapter 2) are applied to the two resulting data to provide a subset of features which could describe properly the given problem.

- **Classification:** Several approaches were considered as classifiers, three lineal models – a one-layer feedforward neural network (FNN), a logistic regression and a proximal support vector machine (pSVM)–, and two non linear ones – a multilayer feedforward neural network and a support vector machine (SVM)–. More details can be found in Appendix I.

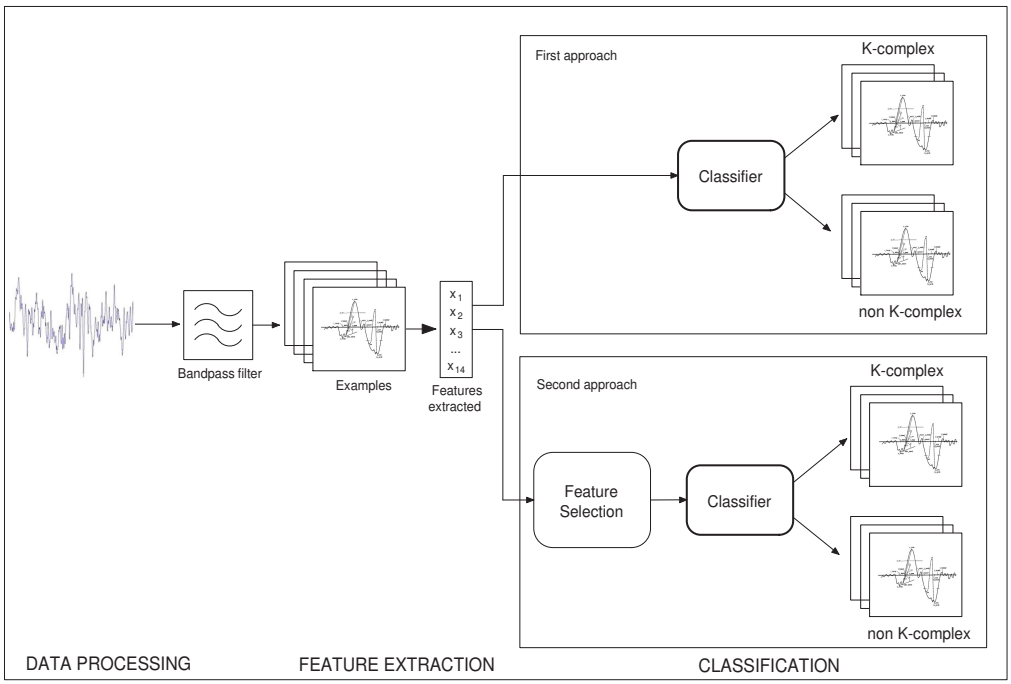


Figure 5.3: The K-complex classification methodology.

After the classifiers were trained, the performance of the system is evaluated in terms of different measures of relevance to the problem in question. The classification accuracy is computed as the percentage of correctly classified instances; the false positive rate is the proportion of normal patterns erroneously classified as K-complexes; and the sensitivity is the proportion of K-complexes which are correctly identified as such. In the context of K-complex classification, a false identification is more undesirable than missing a K-complex. Therefore, it is interesting to minimize the false positives while retaining a satisfactory level of sensitivity.

The experimental procedure is detailed as follows,

1. Extract the initial set of features to be used as inputs.

2. Apply the three feature subset selection methods (CFS, Consistency-based filter and INTERACT) to provide the subset of features that describe properly the given problem.

3. For each nonlinear classifier, establish its architecture/parameters

   (a) For FNNs, set the number of hidden units by training independently each of the FNNs. Bearing in mind that a FNN should never possess a number of hidden units more than twice plus one the number of its input units i, several topologies were trained using from 2 to 2 times i + 1. Then, for the multilayer model set the number of output units (1 vs. 2). Among these models, we try with a one hidden layer architecture and a two hidden layer architecture. Logistic transfer functions were used for each neuron in both the hidden and the output layers. The learning algorithm used was the conjugate gradient with the mean squared error cost function. A maximum number of 3000 epochs were performed on the training set.

   (b) For the SVM, set the kernel function. We try with linear, polynomial of degree 2, and gaussian (different sigma values were tried: 1, 100, 1000, 10000) kernel functions. The cost parameter value was set to values in the range $[100 - \infty]$.

4. Take the whole data set and generate 10-fold cross validation sets in order to better estimate the true error rate of each model.

5. Obtain the accuracy measures, and a decision threshold for the output of each model and select the best one. For the resulting model obtain false positive rate and sensitivity measures.

6. Apply the TOPSIS method (see Appendix I, Section I.6.1) to the performance measures previously obtained.

The results obtained with and without feature selection are presented, evaluated in terms of the three performance measures described above. For the sake of clarity, only the results that correspond with the models using the optimal number of hidden neurons and the optimal classifier parameters are shown.

5.2.1 Results without feature selection

Table 5.10 shows the accuracy, false positive rate and sensitivity measures obtained by the selected models over a 10-fold cross validation for the K-complex classification with all the Bankman features.

Table 5.10: K-complex classification results without feature selection. Mean test set accuracy, false positive and sensitivity (%) of a 10-fold CV. Best results marked in bold font

	Accuracy		False positive		Sensitivity	
	$Data_f$	$Data_{f2}$	$Data_f$	$Data_{f2}$	$Data_f$	$Data_{f2}$
One-lay. FNN	85.52	87.78	**1.61**	5.43	74.77	86.48
Log. Reg.	84.16	87.33	8.60	7.69	85.58	90.09
FNN 1out	85.07	87.33	8.14	8.14	86.48	**90.99**
FNN 2out	84.16	87.33	7.69	5.88	83.78	86.48
FNN 2lay-1out	**85.97**	87.33	8.60	**4.52**	**89.20**	83.78
SVM	84.61	**88.69**	9.95	5.88	**89.20**	89.20
pSVM	82.35	86.88	6.79	4.52	78.37	82.88

For the $Data_f$ data set, the best accuracy was 85.97% with a 14-8-6-1 FNN. For this model it was achieved a sensitivity of 89.20% and a FP rate of 8.60%. These values are similar to those obtained by Bankman et al. (1992) with an 14-3-1 ANN. Among the linear models tested (one-layer FNN, Logistic regression and pSVM), the one-layer FNN seems to be the best method, achieving even the lowest FP rate (1.61%) but with a sensitivity of 74.77%. As for the non-linear models, the two-layer FNN obtained the best results.

On the other side, the $Data_{f2}$ dataset presents an improvement over the previous dataset. This could be due to the fact that $Data_{f2}$ allows to obtain values more adjusted to the features because the K-complex wave derived is more similar to the real one. Besides, the 0.5-2.3 Hz band-pass filter from which $Data_f$ resulted, was too aggressive and some relevant features were missed. In this sense, the results improve for all the classifiers. The best behavior is shown by the SVM (RBF kernel, C=inf) with 88.69%, 5.88% and 89.19% values of accuracy, FP rate and sensitivity, respectively.

5.2.2 Results with feature selection

Tables 5.11, 5.12 and 5.13 show the accuracy, false positive rate and sensitivity measures obtained by the selected models over a 10-fold cross validation for the K-complex classification with the different feature selection methods used (indicating in brackets the number of features selected in average). Best values for each dataset and feature selection method are marked in bold font.

Table 5.11: K-complex classification results with feature selection. Mean test accuracy (%) of a 10-fold CV.

	CFS		Consistency		Interact	
	$Data_f$ (3.8)	$Data_{f2}$ (5)	$Data_f$ (4.5)	$Data_{f2}$ (4.5)	$Data_f$ (5.4)	$Data_{f2}$ (7.9)
One-lay. FNN	86.42	90.04	86.42	89.14	86.42	89.14
Log. Reg.	85.52	89.59	85.52	89.14	85.52	89.14
FNN 1out	86.88	90.95	86.88	**90.95**	**87.33**	**90.95**
FNN 2out	86.88	**91.40**	85.97	89.59	86.88	89.14
FNN 2lay-1out	**87.88**	**91.40**	**88.68**	90.50	86.42	90.50
SVM	85.97	90.49	85.06	90.04	85.97	90.04
pSVM	84.16	89.14	84.16	88.69	84.16	88.69

Table 5.12: K-complex classification results with feature selection. Mean test false positive rate (%) over a 10-fold cv.

	CFS		Consistency		Interact	
	$Data_f$ (3.8)	$Data_{f2}$ (5)	$Data_f$ (4.5)	$Data_{f2}$ (4.5)	$Data_f$ (5.4)	$Data_{f2}$ (7.9)
One-lay. FNN	7.24	4.98	7.24	6.33	7.24	6.33
Log. Reg.	9.95	3.62	5.43	**3.17**	7.69	3.17
FNN 1out	**1.81**	**3.17**	4.52	5.43	**1.36**	**2.71**
FNN 2out	4.98	4.52	5.43	4.07	4.07	4.52
FNN 2lay-1out	5.43	**3.17**	**2.71**	7.24	5.88	6.33
SVM	5.88	**3.17**	6.33	5.43	6.33	5.43
pSVM	6.33	4.07	6.33	3.62	6.33	3.62

Table 5.13: K-complex classification results with feature selection. Mean test sensitivity (%) over a 10-fold cv.

	CFS		Consistency		Interact	
	$Data_f$ (3.8)	$Data_{f2}$ (5)	$Data_f$ (4.5)	$Data_{f2}$ (4.5)	$Data_f$ (5.4)	$Data_{f2}$ (7.9)
One-lay. FNN	87.39	90.09	**87.39**	90.99	**87.39**	90.99
Log. Reg.	**90.99**	86.48	81.98	84.68	86.48	84.68
FNN 1out	74.77	88.29	82.88	92.79	74.77	87.39
FNN 2out	83.78	**91.89**	82.88	87.39	81.98	87.39
FNN 2lay-1out	86.49	89.19	82.88	**95.49**	84.68	**93.69**
SVM	83.78	87.39	82.88	90.99	84.68	90.99
pSVM	81.08	86.48	81.08	84.68	81.08	84.68

The feature selection procedure shows, in general, better performance than the classification made with all Bankman features. This fact can be observed in both data sets, $Data_f$ and $Data_{f2}$. In particular, the accuracy is improved after applying feature selection for any classifier. For $Data_f$, the two-layer FNN, with a 4-10-8-1 layer architecture, achieves the best results when combined with CFS and consistency-based filters. Notice that the reduction in features is also remarkable, since more than 50% of the features are discarded.

Specifically, the consistency-based filter obtains the highest accuracy (88.68%) with a FP rate of 2.71% and a sensitivity of 82.88%. This good result is repeated for the $Data_{f2}$ dataset. In this case, the CFS filter with a 5-10-8-1 FNN obtains the highest accuracy (91.40%), showing 3.17% and 89.19% as values for FP rate and sensitivity, respectively. The model with the smallest FP rate is chosen since in the K-complex classification task it is essential to minimize this value.

Among the linear models tested (one-layer FNN, Logistic regression and pSVM), the one-layer FNN outperforms the other methods by achieving the highest accuracy and sensitivity for the two datasets and filters tested. However, in this case the logistic regression and the pSVM are the ones with the lowest FP rate.

5.2.3 Overall analysis

In light of the results presented so far, it is not easy to conclude which method is the best. Having into account the accuracy, FP rate and sensitivity (SEN) measures for the model with and without feature selection, and for the two data sets (band-pass and low-pass filter approaches), the experimental results were evaluated in terms of the TOPSIS method. Due to the importance of avoiding false positives in the detection of K-complex, the highest weight was assigned to this measure (double than the other two measures). Among the different models evaluated, Table 5.14 displays the top ten ranking over a total of 56 combinations (7 classifier models, 3 feature selection methods, no feature selection and 2 filter approaches).

Table 5.14: Ten best results obtained from TOPSIS method.

TOPSIS	Filter	FS	Model	Accuracy	FP(%)	SEN
0.9343	low-pass	INT	8-15-1 FNN	90.95	2.71	87.39
0.9237	low-pass	CFS	5-10-8-1 FNN	91.40	3.17	89.19
0.9148	low-pass	CFS	5-15-1 FNN	90.95	3.17	88.29
0.9044	low-pass	CFS	SVM*	90.49	3.17	87.39
0.8714	band-pass	Cons.	4-8-6-1 FNN	88.68	2.71	82.88
0.8656	low-pass	INT	Log. Reg.	89.14	3.17	84.68
0.8656	low-pass	Cons.	Log. Reg.	89.14	3.17	84.68
0.8508	low-pass	CFS	Log. Reg.	89.59	3.62	86.48
0.8206	low-pass	INT	pSVM	88.69	3.62	84.68
0.8206	low-pass	Cons.	pSVM	88.69	3.62	84.68

* kernel=polynomial, $C = 10^3$.

These results confirm the fact that feature selection can play a crucial role in K-complex classification. To state which feature selection is the best for this problem, however, is not an easy-to-solve question. Regarding the filtering approach related to the data processing stage, low-pass band takes 9 out of the top 10 results and FNN seems to be the best classification model.

5.2.4 Comparative study with previous results

The results presented above are mainly focused on maximizing accuracy. However, aiming at comparing these results with the ones obtained by Bankman et al. (1992), the decision threshold has to be oriented to achieve a required sensitivity of 85%, 90% and 95%. Table 5.15 reports the sensitivity and false positive rate measures published by Bankman et al. (1992) for a FNN model with three hidden units and those obtained with the proposed methodology for $Data_f$ (band-pass filter approach) and $Data_{f2}$ (low-pass filter approach) datasets.

Table 5.15: False positive rate (%) for different sensitivity levels in the test set.

	Sensitivity		
	85%	90%	95%
Bankman	6.1	8.1	14.1
$Data_f$	5.4	7.7	9.9
$Data_{f2}$	4.5	3.2	8.6

The values obtained for $Data_f$ correspond to the 4-10-8-1 FNN model, using as inputs the features selected by the consistency-based filter. As for the $Data_{f2}$ dataset, the features selected by CFS were considered, combined with a 5-10-8-1 FNN model. It is easy to notice that the FP rate decreases for all the sensitivity levels considered, where the best performance was obtained on $Data_{f2}$. Obviously, this comparison has to be interpreted carefully as the datasets used in each case are different. Having said this, these results pave the way to important improvements in K-complex classification by using feature selection techniques.

5.3 Summary

Feature selection plays a crucial role in many real applications, since it reduces the number of input features and, most of the times, it improves performance. This chapter presented two real problems where feature selection has demonstrated to be useful to achieve better performance results.

The first real application considered was related with tear film lipid layer classification. The time required by existing approaches dealing with this issue prevented their clinical use because they could not work in real time. In this chapter, a methodology for improving this classification problem was proposed, which includes the application of feature subset selection methods: CFS, Consistency-based, and INTERACT. Results obtained with this methodology surpass previous results in terms of processing time whilst maintaining accuracy and robustness to noise. In clinical terms, the manual process done by experts can be automated with the benefits of being faster and unaffected by subjective factors, with maximum accuracy over 97% and processing time under 1 second. The clinical significance of these results should be highlighted, as the agreement between subjective observers is between 91%-100%.

The second real scenario was the K-complex classification, a key aspect in sleep studies. Three filter methods were applied combined with five different machine learning algorithms, trying to achieve a low false positive rate whilst maintaining the accuracy. When feature selection was applied, the results improved significantly for all the classifiers. It is remarkable the 91.40% of classification accuracy obtained by CFS, reducing in 64% the number of features.

CHAPTER 6

Scalability in feature selection

Continuous advances in computer-based technologies have enabled researchers and engineers to collect data at an increasingly fast pace (Z. A. Zhao & Liu, 2011). The proliferation of high-dimensional data brings new challenges to researchers, and scalability and efficiency are two critical issues in this new scenario.

Most algorithms were developed when data set sizes were much smaller, but nowadays distinct compromises are required for the case of small-scale and large-scale learning problems. Small-scale learning problems are subject to the usual approximation-estimation trade-off. In the case of large-scale learning problems, the trade-off is more complex because it involves not only the accuracy but also the computational complexity of the learning algorithm, as seen in tear film lipid layer classification (Chapter 5). Moreover, the problem here is that the majority of algorithms were designed under the assumption that the data set would be represented as a single memory-resident table. So if the entire data set does not fit in main memory, these algorithms are useless.

For all these reasons, scaling up learning algorithms is a trending issue. The organization of the workshop "PASCAL Large Scale Learning Challenge" at the 25th International Conference on Machine learning (ICML'08), and the workshop "Big Learning" at the conference of the Neural Information Processing Systems Foundation (NIPS2011) are cases in point. Scaling up is desirable because increasing the size of the training set often increases the accuracy of algorithms (Catlett, 1991). Scalability is defined as the effect that an increase in the size of the training set has on the computational performance of an algorithm: accuracy, training time and allocated memory. Thus the challenge is to find a deal among them or, in other words, getting "good enough" solutions as "fast" as possible and as "efficiently" as possible. This issue becomes critical in situations in which there exist temporal or spatial constraints like: real-time applications dealing with large data sets, unapproachable computational problems requiring learning, or initial prototyping requiring quickly-implemented solutions.

This chapter is devoted to scalability in feature selection and it is divided in two parts. The first part studies the influence of feature selection methods on the scalability of artificial neural networks (ANN) training algorithms by using the measures defined during the PASCAL workshop (Sonnenburg, Franc, Yom-Tov, & Sebag, 2008). These measures evaluate the scalability of algorithms in terms of error, computational effort, allocated memory and training time. Then, in the second part, the scalability of feature selection methods is studied, checking their performance in an artificial controlled experimental scenario, contrasting the ability of the algorithms to select the relevant features and to discard the irrelevant ones when the dimensionality increases and without permitting noise or redundancy to obstruct this process. For analyzing scalability, new evaluation measures are proposed, which need to be based not only in the accuracy of the selection, but also in other aspects such as the execution time or the stability of the features returned.

6.1 Scalability of neural networks through feature selection

The appearance of very large data sets is not sufficient to motivate scaling efforts. The most commonly cited reason for scaling up algorithms are based on (typically) increasing the accuracy of algorithms when increasing the size of the training data set (Catlett, 1991). In fact, learning from small data sets frequently decreases the accuracy of algorithms as a result of over-fitting.

For most scaling problems the limiting factor has been the number of samples and features describing each sample. The growth rate of the training time of an algorithm as the data set size increases is an outstanding question that arises. But temporal complexity does not reflect scaling in its entirety, and must be used in conjunction with other metrics. For scaling up learning algorithms the issue is not so much as one of speeding up a slow algorithm but as one of tuning an impracticable algorithm into a practical one. The crucial point in question is seldom how fast you can run on a certain problem but rather how large a problem can you deal with (Provost & Kolluri, 1999). More precisely, space considerations are critical to scale up learning algorithms. The absolute size of the main memory plays a key role in this matter. Almost all existing implementations of learning algorithms operate with the training set entirely in main memory. If the spatial complexity of the algorithm exceeds the main memory then

the algorithm will not scale well –regardless of its computational complexity– because page thrashing renders algorithms useless. Page thrashing is the consequence of many accesses to disk occurring in a short time, cutting drastically the performance of a system using virtual memory. Virtual memory is a technique for making a machine behave as if it had more memory than it really has, by using disk space to simulate RAM. But accessing to disk is much slower than accessing to RAM. In the worst case scenario, out of memory exceptions will make algorithms unfeasible in practice.

It has been shown that popular algorithms for ANNs are unable to deal with very large data sets (Peteiro-Barral, Guijarro-Berdiñas, Pérez-Sánchez, & Fontenla-Romero, 2013). For this reason, preprocessing methods may be desirable for reducing the input space size and improving scalability. This section aims to demonstrate that feature selection methods are an appropriate approach to improve scalability. By reducing the number of input features and, consequently, the dimensionality of the data set, we expect to reduce the computational time while maintaining the performance, as well as being able to apply certain algorithms which could not deal with large data sets.

Among the feature selection methods available, this research will be focused on the filter approach. The reason is that although wrappers and embedded methods tend to obtain better performances, they are very time consuming and they will be intractable in dealing with high dimensional data sets without compromising the time and memory requirements of machine learning algorithms.

6.1.1 Experimental study

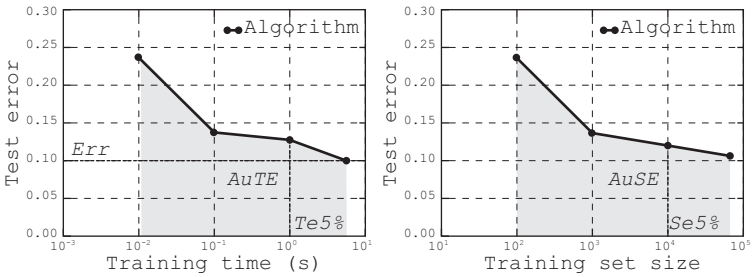
In order to check the effect of different feature selection methods on the scalability of machine learning algorithms, four of the most popular training algorithms for ANNs were selected. Two of these algorithms are gradient descent (GD) (Bishop, 2006) and gradient descent with momentum and adaptive learning rate (GDX) (Bishop, 2006), whose complexity is $O(n)$. The other algorithms are scaled conjugated gradient (SCG) (Moller, 1993) and Levenberg-Marquardt (LM) (More, 1978), whose complexities are $O(n^2)$ and $O(n^3)$, respectively, being n the input size.

Classification and regression are two of the most common tasks in machine learning. For classification, four datasets where chosen, Connect-4, KDD Cup 99, Forest and MNIST, whose characteristics can be consulted in Appendix I. Forest and MNIST

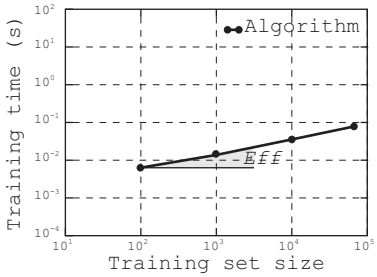
datasets, which are originally classification tasks, were also transformed into a regression task by predicting -1 for samples of class 1; and $+1$ for samples of class 2 (Collobert & Bengio, 2001). Notice that Friedman and Lorenz are artificial datasets. The goal of the network is to predict the current sample based on the four previous samples.

6.1.1.1 Performance measures

In order to assess the performance of learning algorithms, common measures as accuracy are insufficient since they do not take into account all aspects involved when dealing with large datasets. Accordingly, the goal for machine learning developers is to find a learning algorithm such that it achieves a low error in the shortest possible time using as few samples as possible. Since there are no standard measures of scalability, those defined in the *PASCAL Large Scale Learning Challenge* (Sonnenburg et al., 2008) are used:



(a) Training time vs Test error. (b) Training set size vs Test error.



(c) Training set size vs Training time.

Figure 6.1: Performance measures

- Figure 6.1a shows the relationship between *training time* and *test error*, computed on the largest dataset size the algorithm is able to deal with, aimed at

answering the question *"Which test error can we expect given limited training time resources?"*. Following the PASCAL Challenge, the different training time budgets are set to $10^{[\cdots -1,0,1,2\ldots]}$ seconds. We compute the following scalar measures based on this figure:

  - *Err*: minimum test error (standard class error for classification and mean squared error for regression (Weiss & Kulikowski, 1991)).
  - *AuTE*: area under Training time vs Test error curve (gray area).
  - *Te5%*: the time t for which the test error e falls below a threshold $\frac{e-Err}{e} < 0.05$.

- Figure 6.1b shows the relationship between different *training set sizes* and the *test error* of each one aimed at answering the question *"Which test error can be expected given limited training data resources?"*. Following the PASCAL Challenge, the different training set sizes (training samples) are set to $10^{[2,3,4\ldots]}$ up to the maximum size of the dataset. We compute the following scalar measures based on this figure:

  - *AuSE*: area under Training set size vs Test error curve (gray area).
  - *Se5%*: the size s for which the test error e falls below a threshold $\frac{e-Err}{e} < 0.05$

- Figure 6.1c shows the relationship between different *training set sizes* and the *training time* for each one aimed at answering the question *"Which training time can be expected given limited training data resources?"*. Again, the different training set sizes are set to $10^{[2,3,4\ldots]}$ and the maximum size of the dataset. We compute the following scalar measure based on this figure:

  - *Eff*: slope b of the curve using a least squares fit to ax^b.

In order to establish a general measure of scalability, the final *Score* of the algorithms is calculated as the average rank of its contribution with regard to the six scalar measures defined above.

6.1.1.2 Experimental procedure

As a preprocessing step, several feature selection methods were applied over the training set to obtain a subset of features. CFS, INTERACT and Consistency-based were used

for classification whilst CFS and ReliefF were employed for regression (see Chapter 2). The latter follows the individual evaluation framework, and therefore a threshold is required. In this research, we have opted for an aggressive (lower number of features to retain) and a soft reduction (higher number of features to retain).

After the preprocessing step, different simulations (N=10) were carried out over the training set for accurately estimating the scalability of algorithms on each dataset, as showed in the following procedure:

1. Select features over the training set using the feature selection methods mentioned above.
2. Set the number of hidden units of the ANN to $2 \times number\_of\_inputs + 1$ (Hecht-Nielsen, 1990) and train the network. It is important to remark that the aim here is not to investigate the optimal topology of an ANN for a given dataset, but to check the scalability of learning algorithms on large networks.
3. Compute the score of algorithms as the average rank of their contribution with regard to the six scalar measures defined in Section 6.1.1.1.
4. Apply a Kruskal-Wallis test (see Appendix I, Section I.4) to check if there are significant differences among the medians for each algorithm with and without feature selection for a level of significance $\alpha = 0.05$.
5. If there are differences among the medians, then apply a multiple comparison procedure (Tukey's, see Appendix I, Section I.4) to find the simplest approach whose score is not significantly different from the approach with the best score.

6.1.2 Experimental results

6.1.2.1 Classification

During the preprocessing step, the filters CFS, Consistency-based and INTERACT were applied over the training set in order to obtain a subset of features which will be employed in the classification stage. The number of features selected by each method, along with the time required for this task, are depicted in Table 6.1. It has to be noted that CFS is the filter which achieves the greatest reduction in the number of features

in the minimum time in 3 out of the 4 dataset tested. On the other hand, Consistency-based is the filter which requires more time to perform the selection (see that for Forest dataset, Consistency-based takes 2 hours to perform this task while CFS only needs 17 seconds).

Table 6.1: Features selected by each feature selection method along with the required time for classification datasets.

Data	Filter	Features	Time (s)	hh:mm:ss
Connect4	*None*	42	0.00	00:00:00.00
	CFS	6	8.00	00:00:08.00
	Consistency	40	2760.73	00:46:00.73
	INTERACT	38	173.32	00:02:53.33
Forest	*None*	54	0.00	00:00:00.00
	CFS	13	17.28	00:00:17.28
	Consistency	31	7702.12	02:08:22.12
	INTERACT	30	203.57	00:03:23.58
KDD Cup 99	*None*	42	0.00	00:00:00.00
	CFS	5	61.94	00:01:01.95
	Consistency	7	382.01	00:06:22.01
	INTERACT	7	106.71	00:01:46.72
MNIST	*None*	748	0.00	00:00:00.00
	CFS	55	805.27	00:13:25.27
	Consistency	18	13775.09	03:49:35.10
	INTERACT	36	652.96	00:10:52.96

Tables 6.2 and 6.3 present the average test results obtained by the learning algorithms after applying the three different filters compared with those where no feature selection was performed. Notice that N/A stands for *Not Applicable* and those algorithms of which *Score* average test results are not significantly worse than the best are labeled with a cross (†). Regarding the performance measures defined in Section 6.1.1.1, remind that *the lower the result, the higher the scalability.*

It has to be noted that not all the learning algorithms were able to deal with all available samples for every dataset, mostly due to the spatial complexity of the algorithms. In particular, on the MNIST dataset the Levenberg-Marquardt algorithm is not able to train even on the smallest subset when no feature selection is applied. If this occurs, the measures explained in Section 6.1.1.1 were computed on the largest dataset that the learning algorithms were able to process and this fact was specified along with the results.

Table 6.2: Performance measures for classification datasets Connect-4 and Forest.

(a) Connect-4.

Method	Filter	Score	Err	AuTE	AuSE	Te5%	Se5%	Eff
GD	*None*	8.67	0.38	5.16*e*1	0.97	1.08*e*2	1.00*e*2	0.43
	CFS	5.50†	0.51	1.01*e*1	1.24	1.39*e*1	1.00*e*2	0.26
	Consistency	7.83	0.40	4.32*e*1	0.94	8.44*e*1	1.00*e*2	0.40
	INTERACT	6.67†	0.35	3.37*e*1	0.90	8.14*e*1	1.00*e*2	0.40
GDX	*None*	7.00	0.31	3.71*e*1	0.92	7.98*e*1	6.00*e*4	0.40
	CFS	4.00†	0.32	9.44*e*0	0.89	1.70*e*1	1.00*e*4	0.25
	Consistency	4.83†	0.31	2.61*e*1	0.87	5.71*e*1	1.00*e*3	0.37
	INTERACT	4.83†	0.28	2.71*e*1	0.80	6.96*e*1	1.00*e*4	0.38
LM	*None**	8.83†	0.23	3.79*e*2	0.77	7.80*e*2	1.00*e*4	0.77
	CFS	6.67†	0.31	4.79*e*1	0.87	6.68*e*1	1.00*e*4	0.44
	Consistency	9.33†	0.27	2.31*e*2	0.85	5.01*e*2	1.00*e*4	0.71
	INTERACT	8.17†	0.24	1.60*e*2	0.79	3.50*e*2	1.00*e*4	0.68
SCG	*None*	7.17	0.21	7.01*e*1	0.77	2.62*e*2	1.00*e*4	0.50
	CFS	3.83†	0.29	9.97*e*0	0.82	2.28*e*1	1.00*e*4	0.31
	Consistency	6.83	0.23	5.34*e*1	0.72	1.44*e*2	6.00*e*4	0.47
	INTERACT	6.17†	0.23	4.95*e*1	0.70	1.41*e*2	6.00*e*4	0.47

* Largest training set it can deal with: 10^4 samples.

(b) Forest.

Method	Filter	Score	Err	AuTE	AuSE	Te5%	Se5%	Eff
GD	*None*	9.00	0.38	1.24*e*2	1.20	2.78*e*2	1.00*e*3	0.49
	CFS	5.67†	0.45	2.74*e*1	1.36	3.34*e*1	1.00*e*2	0.35
	Consistency	7.67†	0.41	5.67*e*1	1.30	1.12*e*2	1.00*e*3	0.42
	INTERACT	7.00†	0.38	4.97*e*1	1.28	1.08*e*2	1.00*e*5	0.41
GDX	*None*	7.33	0.42	4.74*e*1	1.32	1.01*e*2	1.00*e*4	0.41
	CFS	5.33†	0.51	6.81*e*0	1.41	0.43*e*0	1.00*e*2	0.24
	Consistency	5.67†	0.38	3.21*e*1	1.23	7.20*e*1	1.00*e*5	0.37
	INTERACT	4.33†	0.40	2.26*e*1	1.11	4.93*e*1	1.00*e*3	0.35
LM	*None**	9.33†	0.24	6.41*e*2	0.94	1.74*e*3	1.00*e*4	0.84
	CFS	9.17†	0.32	2.99*e*2	0.95	5.15*e*2	1.00*e*3	0.58
	Consistency	8.17†	0.26	6.71*e*1	0.96	1.72*e*2	1.00*e*4	0.59
	INTERACT	6.67†	0.25	5.72*e*1	0.93	1.55*e*2	1.00*e*4	0.58
SCG	*None*	7.50	0.20	1.64*e*2	0.81	5.80*e*2	1.00*e*5	0.55
	CFS	4.00†	0.29	3.51*e*1	0.86	5.97*e*1	1.00*e*3	0.40
	Consistency	6.50†	0.23	7.21*e*1	0.84	2.48*e*2	1.00*e*4	0.48
	INTERACT	5.67†	0.20	6.21*e*1	0.84	1.70*e*2	1.00*e*5	0.47

* Largest training set it can deal with: 10^4 samples.

Table 6.3: Performance measures for classification datasets KDD Cup 99 and MNIST.

(a) KDD Cup 99.

Method	Filter	Score	Err	AuTE	AuSE	Te5%	Se5%	Eff
GD	$None^{**}$	7.00	0.13	4.29e1	0.43	5.53e1	1.00e2	0.50
	CFS	$4.67^{\dagger}$	0.16	8.67e0	0.54	5.41e0	1.00e2	0.34
	$Consistency$	6.50	0.20	8.80e0	0.70	2.49e1	1.00e3	0.32
	$INTERACT$	$3.33^{\dagger}$	0.12	6.55e0	0.45	2.83e1	1.00e2	0.33
GDX	$None^{**}$	7.00	0.15	2.55e1	0.46	5.93e1	1.00e3	0.44
	CFS	$1.83^{\dagger}$	0.11	4.61e0	0.37	2.15e1	1.00e3	0.30
	$Consistency$	5.33	0.19	7.06e0	0.70	5.65e0	1.00e2	0.31
	$INTERACT$	$3.50^{\dagger}$	0.11	5.68e0	0.48	3.85e1	1.00e3	0.32
LM	$None^{*}$	$9.17^{\dagger}$	0.11	2.21e2	0.46	1.24e3	1.00e4	0.80
	CFS	$8.00^{\dagger}$	0.12	3.38e1	0.49	1.47e2	1.00e2	0.46
	$Consistency$	$9.33^{\dagger}$	0.17	3.11e1	0.63	1.10e2	1.00e4	0.42
	$INTERACT$	$8.00^{\dagger}$	0.12	2.89e1	0.55	6.32e1	1.00e5	0.44
SCG	$None^{**}$	9.67	0.14	1.10e2	0.51	3.54e2	1.00e4	0.55
	CFS	$4.17^{\dagger}$	0.08	1.13e1	0.31	4.40e1	1.00e4	0.38
	$Consistency$	7.83	0.18	2.06e1	0.70	4.88e1	1.00e3	0.39
	$INTERACT$	8.00	0.17	2.15e1	0.56	8.41e1	1.00e2	0.39

* Largest training set it can deal with: 10^4 samples.

** Largest training set it can deal with: 10^5 samples.

(b) MNIST.

Method	Filter	Score	Err	AuTE	AuSE	Te5%	Se5%	Eff
GD	$None^{*}$	9.17	0.36	1.41e2	0.85	2.26e2	1.00e2	0.65
	CFS	6.67	0.26	4.04e1	0.69	1.07e2	1.00e3	0.43
	$Consistency$	$5.00^{\dagger}$	0.37	1.72e1	1.07	3.99e1	1.00e3	0.33
	$INTERACT$	$5.50^{\dagger}$	0.31	2.92e1	0.79	7.46e1	1.00e3	0.39
GDX	$None^{*}$	9.00	0.22	2.30e2	0.66	6.91e2	1.00e3	0.72
	CFS	5.50	0.21	3.32e1	0.66	9.98e1	1.00e3	0.42
	$Consistency$	$3.50^{\dagger}$	0.22	1.50e1	0.68	3.98e1	1.00e4	0.33
	$INTERACT$	$3.83^{\dagger}$	0.21	2.33e1	0.64	6.80e1	1.00e3	0.38
LM	$None$	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	CFS	$9.17^{\dagger}$	0.13	3.22e2	0.56	1.10e3	1.00e4	0.80
	$Consistency$	$6.83^{\dagger}$	0.10	8.52e1	0.49	3.38e2	6.00e4	0.55
	$INTERACT$	$6.33^{\dagger}$	0.13	6.45e1	0.48	2.05e2	1.00e4	0.62
SCG	$None^{*}$	8.00	0.05	2.85e2	0.40	1.62e3	1.00e4	0.81
	CFS	$6.33^{\dagger}$	0.11	4.77e1	0.49	2.42e2	6.00e4	0.50
	$Consistency$	$3.83^{\dagger}$	0.11	1.73e1	0.51	8.62e1	6.00e4	0.40
	$INTERACT$	$5.50^{\dagger}$	0.11	3.14e1	0.54	1.61e2	6.00e4	0.46

* Largest training set it can deal with: 10^4 samples.

6.1.2.2 Regression

The filters CFS and ReliefF were selected for the regression task. It has to be reminded that ReliefF provides a ranking of features and a threshold is required. In this work, we have opted for two different arrangements: an aggressive and a soft reduction, represented in the tables as $ReliefF^{\vee}$ and $ReliefF^{\wedge}$, respectively.

Table 6.4: Features selected by each feature selection method along with the required time for regression datasets. $ReliefF^{\vee}$ and $ReliefF^{\wedge}$ stand for aggressive and soft reduction, respectively.

Data	Filter	Features	Time (s)	hh:mm:ss
Forest	$None$	54	0.00	00:00:00.00
	CFS	23	21.76	00:00:21.76
	$ReliefF^{\vee}$	5	11631.56	03:13:51.57
	$ReliefF^{\wedge}$	48	11631.56	03:13:51.57
Friedman	$None$	10	0.00	00:00:00.00
	CFS	5	17.39	00:00:17.39
	$ReliefF^{\vee}$	4	289191.43	80:19:51.44
	$ReliefF^{\wedge}$	5	289191.43	80:19:51.44
Lorenz	$None$	8	0.00	00:00:00.00
	CFS	1	11.83	00:00:11.83
	$ReliefF^{\vee}$	4	262575.40	72:56:15.40
	$ReliefF^{\wedge}$	6	262575.40	72:56:15.40
MNIST	$None$	748	0.00	00:00:00.00
	CFS	103	938.73	00:15:38.73
	$ReliefF^{\vee}$	31	57048.71	15:50:48.72
	$ReliefF^{\wedge}$	418	57048.71	15:50:48.72

As for classification, Table 6.4 shows the number of features selected by each method along with the time required. Note that the time required by ReliefF is the same for the two arrangements, since the construction of the ranking is a mutual step. Again, CFS is able to perform the selection mostly in the order of seconds whilst ReliefF needs in the order of hours.

Tables 6.5 and 6.6 depict the average test results achieved by the learning algorithms after applying CFS and ReliefF filters compared with those where no feature selection was performed. Notice that N/A stands for *Not Applicable*. Those algorithms whose *Score* average test results are not significantly worse than the best are labeled with a cross (†). $ReliefF^{\vee}$ and $ReliefF^{\wedge}$ stand for aggressive and soft reduction, respectively.

Table 6.5: Performance measures for regression datasets Forest and Friedman.

(a) Forest.

Method	Filter	Score	Err	AuTE	AuSE	Te5%	Se5%	Eff
GD	$None$	9.17	0.90	$1.26e3$	3.62	$5.38e2$	$1.00e4$	0.55
	CFS	7.67	0.99	$1.29e2$	3.15	$8.27e1$	$1.00e3$	0.38
	$ReliefF^{\vee}$	$4.67^{\dagger}$	0.95	$2.37e1$	2.96	$1.54e1$	$1.00e3$	0.26
	$ReliefF^{\wedge}$	8.67	0.90	$3.90e2$	2.98	$1.59e2$	$1.00e4$	0.44
GDX	$None$	8.50	0.68	$1.01e3$	4.17	$4.54e2$	$1.00e5$	0.53
	CFS	$7.00^{\dagger}$	1.13	$6.91e1$	3.32	$3.59e1$	$1.00e3$	0.32
	$ReliefF^{\vee}$	$4.67^{\dagger}$	0.99	$1.61e1$	3.00	$1.07e1$	$1.00e3$	0.21
	$ReliefF^{\wedge}$	9.33	1.16	$3.60e2$	4.63	$1.03e2$	$1.00e3$	0.41
LM	$None^{*}$	9.17	0.60	$1.02e4$	3.42	$1.35e3$	$1.00e4$	0.82
	CFS	$6.83^{\dagger}$	0.80	$4.20e2$	2.48	$8.83e1$	$1.00e3$	0.40
	$ReliefF^{\vee}$	$4.50^{\dagger}$	0.64	$4.71e1$	2.39	$4.07e1$	$1.00e4$	0.35
	$ReliefF^{\wedge}$	9.67	0.54	$2.36e3$	3.04	$2.83e2$	$1.00e4$	0.65
SCG	$None$	8.17	0.57	$1.64e3$	2.72	$9.86e2$	$1.00e5$	0.60
	CFS	7.00	0.81	$1.41e2$	2.66	$5.89e1$	$1.00e4$	0.42
	$ReliefF^{\vee}$	$3.67^{\dagger}$	0.67	$2.83e1$	2.29	$2.48e1$	$1.00e4$	0.31
	$ReliefF^{\wedge}$	8.00	0.61	$5.33e2$	2.60	$2.16e2$	$1.00e5$	0.50

* Largest training set it can deal with: 10^4 samples.

(b) Friedman.

Method	Filter	Score	Err	AuTE	AuSE	Te5%	Se5%	Eff
GD	$None^{*}$	$6.50^{\dagger}$	8.33	$2.19e3$	36.77	$7.51e1$	$1.00e3$	0.37
	CFS	$7.83^{\dagger}$	8.16	$6.72e3$	35.80	$1.71e2$	$1.00e3$	0.37
	$ReliefF^{\vee}$	$7.67^{\dagger}$	9.90	$6.30e3$	43.70	$1.36e2$	$1.00e4$	0.35
	$ReliefF^{\wedge}$	$8.67^{\dagger}$	9.71	$7.09e3$	36.80	$1.71e2$	$1.00e3$	0.37
GDX	$None^{*}$	$5.84^{\dagger}$	4.41	$1.83e3$	24.57	$7.20e1$	$1.00e5$	0.37
	CFS	$6.50^{\dagger}$	3.86	$6.36e3$	23.00	$1.69e2$	$1.00e5$	0.37
	$ReliefF^{\vee}$	$6.33^{\dagger}$	5.08	$5.49e3$	31.10	$1.36e2$	$1.00e6$	0.35
	$ReliefF^{\wedge}$	$6.67^{\dagger}$	4.00	$6.44e3$	23.71	$1.69e2$	$1.00e4$	0.37
LM	$None^{*}$	$5.00^{\dagger}$	0.11	$1.11e3$	8.57	$8.74e2$	$1.00e5$	0.59
	CFS	8.33	2.34	$1.87e4$	14.82	$1.03e3$	$1.00e4$	0.50
	$ReliefF^{\vee}$	7.50	2.35	$1.38e4$	14.24	$7.76e2$	$1.00e4$	0.48
	$ReliefF^{\wedge}$	$7.00^{\dagger}$	0.27	$1.79e4$	6.98	$1.03e3$	$1.00e4$	0.50
SCG	$None^{*}$	$5.00^{\dagger}$	0.79	$1.67e3$	10.33	$1.71e2$	$1.00e5$	0.44
	CFS	$6.33^{\dagger}$	2.66	$4.77e3$	16.10	$3.88e2$	$1.00e5$	0.43
	$ReliefF^{\vee}$	$6.17^{\dagger}$	2.85	$4.49e3$	17.81	$3.10e2$	$1.00e6$	0.41
	$ReliefF^{\wedge}$	$5.33^{\dagger}$	0.93	$5.58e3$	9.34	$3.87e2$	$1.00e4$	0.43

* Largest training set it can deal with: 10^5 samples.

Table 6.6: Performance measures for regression tasks Lorenz and MNIST.

(a) Lorenz.

Method	Filter	Score	Err	AuTE	AuSE	Te5%	Se5%	Eff
GD	$None^*$	$5.17^\dagger$	0.74	4.82e2	2.98	6.17e1	1.00e2	0.36
	CFS	$7.00^\dagger$	6.57	1.57e3	26.01	5.60e1	1.00e2	0.29
	$ReliefF^\vee$	7.67	1.44	2.16e3	6.37	1.37e2	1.00e3	0.35
	$ReliefF^\wedge$	8.17	0.94	2.26e3	4.30	2.02e2	1.00e6	0.38
GDX	$None^*$	$4.83^\dagger$	2.66	2.45e2	13.63	2.04e1	1.00e4	0.26
	CFS	$4.83^\dagger$	1.29	1.19e3	5.37	4.26e1	1.00e3	0.27
	$ReliefF^\vee$	$6.83^\dagger$	4.02	1.76e3	13.23	7.53e1	1.00e3	0.31
	$ReliefF^\wedge$	7.17	5.45	1.57e3	18.71	7.73e1	1.00e2	0.32
LM	$None^*$	$8.17^\dagger$	0.00	3.26e3	0.00	5.19e2	1.00e5	0.54
	CFS	$5.17^\dagger$	0.18	8.40e2	0.80	1.24e2	1.00e3	0.36
	$ReliefF^\vee$	$7.83^\dagger$	0.00	3.12e3	0.00	7.82e2	1.00e6	0.48
	$ReliefF^\wedge$	$8.50^\dagger$	0.00	8.97e3	0.00	1.44e3	1.00e6	0.52
SCG	$None^*$	$5.83^\dagger$	0.01	5.61e2	0.05	1.38e2	1.00e4	0.43
	CFS	$5.00^\dagger$	0.21	4.86e2	1.59	1.15e2	1.00e4	0.35
	$ReliefF^\vee$	$6.33^\dagger$	0.01	1.21e3	0.10	3.11e2	1.00e6	0.41
	$ReliefF^\wedge$	$7.17^\dagger$	0.01	2.00e3	0.12	4.53e2	1.00e4	0.44

* Largest training set it can deal with: 10^5 samples.

(b) MNIST.

Method	Filter	Score	Err	AuTE	AuSE	Te5%	Se5%	Eff
GD	$None^*$	8.17	303.12	1.66e3	903.14	6.60e0	1.00e2	0.44
	CFS	$5.17^\dagger$	37.14	1.44e2	143.11	3.23e0	6.00e4	0.23
	$ReliefF^\vee$	$3.50^\dagger$	1.33	3.86e1	3.95	1.40e1	1.00e3	0.28
	$ReliefF^\wedge$	$6.17^\dagger$	210.98	9.11e2	528.32	4.55e0	1.00e2	0.35
GDX	$None^*$	9.33	9.25	7.49e4	66.06	9.71e2	1.00e4	0.75
	CFS	6.33	1.52	1.34e3	6.02	1.62e2	1.00e3	0.46
	$ReliefF^\vee$	$2.83^\dagger$	0.85	3.73e1	3.86	1.26e1	1.00e4	0.27
	$ReliefF^\wedge$	8.83	12.92	1.51e4	76.51	3.06e2	1.00e4	0.62
LM	$None^*$	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	CFS	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	$ReliefF^\vee$	$4.83^\dagger$	0.59	3.80e2	3.08	8.68e1	1.00e4	0.51
	$ReliefF^\wedge$	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SCG	$None$	9.17	3.06	3.10e4	41.52	1.82e3	1.00e4	0.82
	CFS	5.83	0.44	1.09e3	3.62	4.20e2	6.00e4	0.55
	$ReliefF^\vee$	$3.00^\dagger$	0.55	3.87e1	2.12	1.56e1	1.00e4	0.32
	$ReliefF^\wedge$	8.50	2.51	1.12e4	36.71	6.68e2	1.00e4	0.71

* Largest training set it can deal with: 10^4 samples.

As well as in the classification case, when the learning algorithm is not able to train on all available samples, this fact is specified along with the results. LM was again not able to train on MNIST dataset (see Table 6.6b). Even when the number of weights of an ANN is lower in a regression task than in a classification task (as the number of outputs is also lower), the spatial complexity of the algorithm LM is still very high.

6.1.3 Discussion

The aim of the experiments carried out in this work is to assess the performance of ANN algorithms in terms of scalability and not simply in terms of error like the great majority of papers in the literature. All the six scalar measures defined in Section 6.1.1.1 are considered to evaluate the scalability of learning algorithms, trying to achieve a balance among them. When applying feature selection, it is expected that some measures are positively affected by the dimensionality reduction (such as *AuTe*, *Te5%* or *Eff*) because the algorithms can deal with a larger number of samples employing the same execution time. Nevertheless, the goal of this research is to demonstrate that this reduction does not always affect negatively the remaining measures and also to find a trade-off among all the measures which will be reflected on the final average *Score*.

6.1.3.1 Classification

In general lines, results without feature selection show a scarcely lower error (although in some cases it is maintained or even improved, depending on the filter) at the expense of a longer training time. On the other hand, the results after applying feature selection present a shorter training time.

As expected, the measures related to the training time (*AuTe*, *Te5%* and *Eff*) improve, since a shorter time is needed to train the same number of data. On the other hand, *AuSE* and *Se5%* deteriorate their results after applying feature selection. Although the error was expected to be higher after applying feature selection, INTERACT maintains or improves the classification error in most of the cases, obtaining also a good performance on the other scalability measures.

Since the assessment of the scalability of learning algorithms is a multi-objective problem and there is no chance of defining a single optimal order of importance among

measures, we have opted to focus on the general measure of scalability (*Score*). Tables 6.2 and 6.3 show that in most cases applying feature selection was significantly better than not applying it (13 out of 16 cases). In order to decide which filter is the best option, Table 6.7 depicts the average score for each filter on each dataset, as well as the average filtering time. Since the algorithm LM is not able to train over MNIST when no feature selection is applied, the results over this dataset are averaged on three learning algorithms, instead of four.

Table 6.7: Average of *Score* for each filter on each dataset for classification tasks along with the average time required by the filters.

Filter	Connect-4	Forest	KDD Cup 99	MNIST	Average	Time(s)
None	7.92	8.29	8.21	8.72	8.29	–
CFS	5.00	6.04	4.67	6.17	5.47	223.12
Consistency	7.21	7.00	7.25	4.11	6.39	6154.99
INTERACT	6.46	5.92	5.71	4.94	5.76	284.14

In light of the results showed in Table 6.7, it remains clear that applying feature selection is better than not doing it. Among the three filters tested, CFS exhibits the best *Score* in average, closely followed by INTERACT. Bearing in mind the average time required by each filter (see last column in Table 6.7), the Consistency-based filter does not seem to be a good option, due to the fact that it obtains the worst score along with the highest processing time. CFS tends to select the smallest number of features at the expense of a slightly higher error than INTERACT, therefore the decision of which one is more adequate for the scalability of ANNs depends on if the user is more interested in minimizing either the error or the other measures.

6.1.3.2 Regression

Over Forest and MNIST datasets, the performance measures follow the same trends as with classification tasks: those related with the training time (*AuTe*, *Te5%* and *Eff*) improve while *AuSE* and *Se5%* slightly worsen. However, this is not the case with Friedman and Lorenz datasets. This fact is explained because these datasets have only 10 and 8 features, respectively, and reducing the input dimensionality does not lead to a significant reduction in the training time, whilst it may remove some important information which affects accuracy.

Although in general it seems that feature selection methods do not achieve results as good as over classification tasks, an in-depth analysis reveals that for all the combinations between dataset and learning algorithm, using feature selection obtained a significantly better or equal *Score* (for all filters) than not using it.

Studying in detail the behavior of the filters, one may find that $ReliefF^{\vee}$ (aggressive reduction) is the best (or one of the best ones) method with a significant difference in all cases but two. Table 6.8 reinforces this fact by showing that this method obtains the best *Score* in average for all datasets and learning algorithms. On the other hand, $ReliefF^{\wedge}$ (soft reduction) presents the worst *Score* in average. This fact can be explained because $ReliefF^{\wedge}$ selects a higher number of features, which leads to a slightly better error, but at the expense of requiring more training time and hence getting worse results on the other measures.

Albeit $ReliefF^{\vee}$ is the best method according to *Score*, it has to be reminded that ReliefF requires a much higher computational time than CFS (see last column in Table 6.8). For this reason, in our opinion, CFS is the best option when looking for a feature selection method which could help on the scalability of ANNs on regression tasks. In fact, its *Score* shows that applying CFS is better than not doing it and the computational time required is not prohibitive, as happens with ReliefF.

Regarding the results for Friedman and Lorenz datasets, with 10 and 8 input features respectively, and bearing in mind the results depicted in Table 6.8, one may question the adequacy of feature selection on these datasets with such a small number of features. However, there is not a universal answer to this question, since it depends on the nature of the problem and on the presence of irrelevant features. In this work, no benefits were found after applying feature selection over Friedman dataset, but it was worth to do it over Lorenz. In fact, the filter CFS over the latter dataset only needs a couple of seconds to perform the selection and it retains one single feature. Further experimentation showed that this feature is highly correlated with the output and obtained results as good as with the whole set of features. It is remarkable the fact that for GDX algorithm, the lower error was achieved by CFS using only that one feature (see Table 6.6a).

Table 6.8: Average of *Score* for each filter on each dataset for regression tasks along with the average time required by the filters.

Filter	Forest	Friedman	Lorenz	MNIST	Average	Time(s)
None	7.88	5.59	6.00	8.89	7.09	–
CFS	7.13	7.25	5.50	5.78	6.42	247.43
$ReliefF^{\vee}$	4.38	6.92	7.17	3.11	5.39	155111.60
$ReliefF^{\wedge}$	8.92	6.92	7.75	7.83	7.86	155111.60

6.2 Scalability of feature selection methods

The previous section has demonstrated that feature selection can be helpful in scaling machine learning algorithms as it reduces the input dimensionality and therefore the run-time required by an algorithm. However, when dealing with a dataset which contains a huge number of features and samples, the scalability of a feature selection method also becomes of crucial importance. Since most of the existing feature selection techniques were designed to process small-scale data, their efficiency can be downgraded, if not totally inapplicable, with high-dimensional data.

In this scenario, feature selection researchers need to be focused not only on the accuracy of the selection but also on other aspects. Stability, that is the sensitivity of the results to training set variations, is one of such factors, with a few studies published regarding the behavior of filters in the case in which training set is small, but the number of features can be high (G. Brown, Pocock, Zhao, & Luján, 2012; Fahad, Tari, Khalil, Habib, & Alnuweiri, 2013; Gulgezen, Cataltepe, & Yu, 2009). The other important aspect, scalability, that is the behavior of feature selection methods in the case in which the training set is increasingly high, is still more scarce in the scientific literature (Peteiro-Barral, Bolón-Canedo, Alonso-Betanzos, Guijarro-Berdiñas, & Sánchez-Maroño, 2012). The studies are mainly concentrated in obtaining scalability in a particular application (Luo et al., 2012), modifying certain previously existing approaches (Sun, Todorovic, & Goodison, 2008b), or adopting on-line (Hoi, Wang, Zhao, & Jin, 2012) or parallel (Z. Zhao, Zhang, Cox, Duling, & Sarle, 2013) approaches. In general, one can say that most of the classical feature selection approaches that are univariate -that is each feature is considered separately- have an important advantage in scalability, but at the cost of ignoring feature dependencies, and thus perhaps leading to lower performances than other feature selection techniques. To improve performance, multivariate filter techniques are proposed, but at the cost of reducing scalability

(Alonso-Betanzos, Bolón-Canedo, Fernández-Francos, Porto-Díaz, & Sánchez-Maroño, 2013). In this situation, the scalability of a feature selection method becomes extremely important.

In this section, the scalability of feature selection methods is studied, checking their performance in an artificial controlled experimental scenario, contrasting the ability of the algorithms to select the relevant features and to discard the irrelevant ones when the dimensionality increases and without permitting noise or redundancy to obstruct this process. For analyzing scalability, new evaluation measures are proposed, which need to be based not only on the accuracy of the selection, but also on other aspects such as the execution time or the stability of the features returned.

6.2.1 Experimental study

The experimental study will test the scalability of the three types of feature selection methods: filters, embedded and wrappers (see Chapter 2 for consulting their descriptions). Some of the methods (Chi-Squared, ReliefF, Information Gain, mRMR, FS-P and SVM-RFE) follow the ranking approach, which consists of assessing individual features by assigning them weights according to their relevance. The remaining algorithms (CFS, FCBF, INTERACT, consistency-based and the wrapper) follow the subset evaluation approach, which consists of producing candidate feature subsets based on a certain search strategy. With regard to the computational cost, it can be noticed that some of the proposed filter techniques are univariate. This means that each feature is considered separately, thereby ignoring feature dependencies, which may lead to worse classification performance when compared to other types of feature selection techniques. However, they have the advantage, in theory, of being scalable. Multivariate techniques were introduced, aiming to incorporate feature dependencies to some degree, but at the cost of reducing their scalability.

A common problem when testing the effectiveness of a feature selection method on real data is that the relevant features are usually not known in advance. In these cases, the performance of the feature selection methods clearly rely on the performance of the learning method used afterwards and it can vary notably from one method to another. The objective of this research is to study the scalability of feature selection methods with independence of any other learning method (i.e. classifier, cluster method, etc.). For this reason, it has been chosen to use artificial datasets to perform this task. The

main advantage of these artificial scenarios is the knowledge of the set of optimal features that must be selected, thus the degree of closeness to any of these solutions can be assessed in a confident way.

The datasets chosen for this study try to cover different problems: increasing number of irrelevant features, redundancy, noise in the output, alteration of the inputs, non-linearity of the data, etc. These factors complicate the task of the feature selection methods, which are very affected by them. Besides, some of the datasets have a significantly higher number of features than samples, which implies an added difficulty for the correct selection of the relevant features. Details on the synthetic datasets employed in this chapter can be found in Appendix I.

For assessing the scalability of the methods, different configurations of these datasets were used. In particular, the number of features ranges from 2^3 to 2^7 whilst the number of samples ranges from 2^3 to 2^{14} (all pairwise combinations). In the case of the SD datasets, the number of features ranges from 2^6 to 2^{12} whilst the number of samples ranges from 3^2 to 3^5. Notice that the number of relevant features is fixed and it is the number of irrelevant features the one that varies, randomly generated. When the number of samples increases, the new instances are also generated from a random distribution.

Table 6.9 shows a summary of the ranges of features and samples tested for each dataset. The first seven datasets in the table are classical datasets, having more samples than features, so they will be studied together. SD1, SD2 and SD3 are datasets which represent the characteristics of microarray data, with more features than samples, and will be analyzed as a case of study in Section 6.2.3.2.

6.2.2 Evaluation metrics

The goal of this research is to assess the scalability of several feature selection methods. For this purpose, some evaluation measures need to be defined, covering different aspects that must be addressed. First, some metrics which take into account the accuracy of the selected features are presented, motivated by the measures proposed by M.-L. Zhang et al. (2009); Rokach, Schclar, and Itach (2013); Tsoumakas, Katakis, and Vlahavas (2010). However, it is also important to have measures which evaluate the stability of feature selection, i.e. the insensitivity of the result of a feature selection

Table 6.9: Summary of the synthetic datasets used

Dataset	No. of features	No. of samples
Corral	2^3 - 2^7	2^3 - 2^{14}
Led	2^3 - 2^7	2^3 - 2^{14}
Monk1	2^3 - 2^7	2^3 - 2^{14}
Monk2	2^3 - 2^7	2^3 - 2^{14}
Monk3	2^3 - 2^7	2^3 - 2^{14}
XOR	2^3 - 2^7	2^3 - 2^{14}
Parity	2^3 - 2^7	2^3 - 2^{14}
SD1	2^6 - 2^{12}	3^2 - 3^5
SD2	2^6 - 2^{12}	3^2 - 3^5
SD3	2^6 - 2^{12}	3^2 - 3^5

algorithm to variations in the training set. For this reason, some metrics for assessing this issue will be introduced, based on the works presented by Kendall (1938); Spearman (1904); Kumar and Vassilvitskii (2010). Last but not least, the training time is also a measure of success for the scalability of a feature selection method.

The evaluation measures used in this work are also divided in two types: the ones devoted to subset methods ($Hamming\_loss$, $F1-score$, $Tanimoto$ and $Jaccard$) and those devoted to ranking methods ($ranking\_loss$, $average\_error$, $Spearman$ and $Kendall$). The training time is also considered, for both types of methods, reported in seconds. For the sake of clarity, it has been decided that all the measures are desirable to be minimized. Therefore, the measures related to how accurate the selection is are focused on the error, whilst the metrics related to the stability are now considered as related to the distance between rankings.

For the subset methods, $feat\_sel$ stands for the subset of selected features, whilst in the case of rankers, it represents the ranking of features returned. Plus, $feats$ is the total number of features, $feat\_rel$ is the subset of relevant features and $feat\_irr$ represents the subset of irrelevant features (both of them known *a priori*). Notice that all measures mentioned below except training time are bounded between 0 and 1.

6.2.2.1 Measures for ranker methods

This section describes the evaluation measures applied to the feature selection methods which return an ordered ranking of the features.

- The *ranking_loss* (R) evaluates the number of irrelevant features that are better ranked than the relevant ones. The fewer irrelevant features are on the top of the ranking, the best classified are the relevant ones. Notice that *pos* stands for the position of the last irrelevant feature in the ranking.

$$R = \frac{pos\ -\ \#feat\_rel}{\#feats\ -\ \#feat\_rel}$$

- The *average_error* (E) evaluates the mean of average fraction of relevant features ranked above a particular feature of the ranking.

$$E = \frac{\sum j; feat\_sel(j)\ \in\ feat\_rel\ \cap\ j<i\ -\ \frac{\#feat\_rel\ \times\ (\#feat\_rel\ -\ 1)}{2}}{\#feat\_irr\ \times\ \#feat\_rel}$$

- The *Spearman-distance* (S) is a metric which measures dissimilarity between rankings of features. It is complimentary of the Spearman correlation coefficient (ρ), which is defined as the Pearson correlation coefficient between the ranked variables. Therefore, the Spearman-distance between two rankings, A and B, is obtained by subtracting the Spearman correlation coefficient from 1, where d is the distance between the same elements in both rankings.

$$S(A,B) = 1 - \rho = 1 - \left(1 - \frac{6\sum d^2}{\#feats(\#feats^2-1)}\right)$$

- The *Kendall-distance* (K) is a metric that counts the number of pairwise disagreements between two ranking lists A and B. The larger the distance, the more dissimilar the two lists are.

$$K(A,B) = \sum_{\{i,j\}\in P} \bar{K}_{i,j}(A,B)$$

where
P is the set of unordered pairs of distinct elements in A and B
$\bar{K}_{i,j}(A,B) = 0$ if i and j are in the same order in A and B
$\bar{K}_{i,j}(A,B) = 1$ if i and j are in the opposite order in A and B

6.2.2.2 Measures for subset methods

This section describes the evaluation measures applied to the feature selection methods which return a subset of selected features.

- The *Hamming_loss* (H) measure evaluates how many times a feature is misclassified (selected when is irrelevant or not selected when is relevant)

$$H = \frac{\#(feat\_sel \cap feat\_irr) + \#(feat\_not\_sel \cap feat\_rel)}{\#(feat\_rel \cup feat\_irr)}$$

- The *F1-score* is defined as the harmonic mean between precision and recall. *Precision* is computed as the number of relevant features selected divided by the number of features selected; and *recall* is the number of relevant features selected divided by the total number of relevant features. Therefore, the F1-score can be interpreted as a weighted average of the precision and recall. Considered 1 − F1-score, it reaches its best value at 0 and worst score at 1.

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

- The *Tanimoto-distance* (T) is a metric which measures dissimilarity between sets of features. It is complimentary of the Tanimoto-coefficient (T_C) of the sets A and B.

$$T(A,B) = 1 - T_C = 1 - \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

- The *Jaccard-distance* (J) is a metric which measures dissimilarity between sets of samples (in this case, sets of features). It is complimentary of the *Jaccard-index* (JI), which is defined as the cardinality of the intersection divided by the cardinality of the union of the sets A and B. Therefore, the Jaccard-distance is obtained by subtracting the Jaccard-index from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union.

$$J(A,B) = 1 - JI = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

6.2.2.3 Summary of measures

As mentioned before, the measures described above are divided in three types of measures: those related with the *error*, the *distance* and the *time*. For visualizing the graphs which will be obtained in the experimental section, it is necessary to have a single measure per group. Therefore, since all the measures are bounded between 0 and 1 and are desired to be minimized, the arithmetic mean of the measures of the same type is computed.

- Ranker methods:
  - $error = mean(ranking\_loss\ ,\ average\_error)$
  - $distance = mean(spearman\ ,\ kendall)$
  - $time$
- Subset methods:
  - $error = mean(hamming\_loss\ ,\ F1)$
  - $distance = mean(tanimoto\ ,\ jaccard)$
  - $time$

Motivated by the methodology proposed by Sonnenburg et al. (2008), we define three figures from which eight scalar measures are extracted. Note that the evaluation of feature selection algorithms relies on the bi-dimensional features-samples space (X - Y -axes). So, these evaluation measures shape a surface (Z-axis) in a three-dimensional space.

- Error surface: *Feature size* vs *Sample size* vs *Error*. It is obtained by displaying the evolution of the error measure across the feature-sample space. The following scalar measures are computed:
  1. *MinEr*: minimum error.
  2. *Er5%*: the minimum amount of data (features x samples) for which the error drops below a threshold (5% of error).
  3. *VuEr*: volume under the error surface.

- Distance surface: *Feature size* vs *Sample size* vs *Coverage*. It is obtained by displaying the evolution of the distance measure across the feature-sample space.

    4. *MinDi*: minimum distance.
    5. *Di5%*: the minimum amount of data (features x samples) for which the distance drops below a threshold (5% of distance).
    6. *VuDi*: volume under the distance surface.

- Training time surface: *Feature size* vs *Sample size* vs *Traning time*. It is obtained by displaying the evolution of the average_precision across the feature-sample space.

    7. *MaxTt*: training time in seconds for the maximum amount of data tested.
    8. *VuTt*: volume under the training time surface.

6.2.3 Experimental results

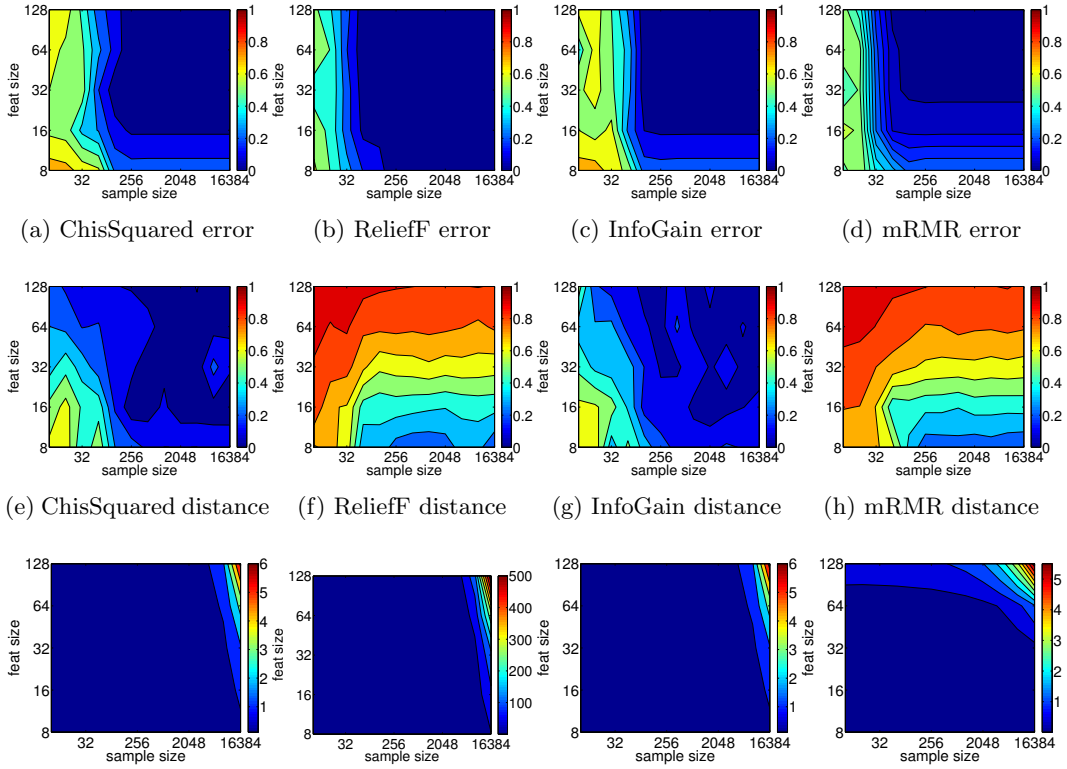
This section shows the scalability results according to the measures explained above after applying a 10-fold cross validation, where all the metrics are desirable to be minimized. Section 6.2.3.1 is devoted to show the scalability of filter methods, whilst Sections 6.2.3.3 and 6.2.3.4 are dedicated to wrapper and embedded methods, respectively.

6.2.3.1 Scalability of filters

This section studies the scalability properties shown by the eight filter methods considered (FCBF, CFS, consistency-based, INTERACT, InfoGain, ReliefF, Chi-Squared and mRMR, see Chapter 2). Figures 6.2 and 6.3 plot an scalar metric of scalability per row (error, distance and time) of ranker and subset filters, respectively, for Corral dataset as an example. In general, the error is more affected by the number of samples than of features. Nevertheless, it is easy to see that Chi-Squared, InfoGain and mRMR require a larger number of features to achieve the lowest error than the remaining methods.

In terms of distance, which is a manner of measuring the stability of the features selected by the algorithms (see Section 6.2.2), there is not a clear trend. The subset

filters (FCBF, CFS, Cons and INTERACT, Figure 6.3) seem to be more affected by the sample size, although as expected with high numbers of samples, the more features, the more difficult is to select stable subsets of features. The exception to this behavior is the Consistency-based filter, which is not affected by the number of features when having more than 200 samples. On the other hand, the ranker filters (Figure 6.2) have two types of behaviors. ReliefF and mRMR are mainly affected by the number of features, worsening their performance as the number of features increases. However, Chi-Squared and InfoGain are more affected by the number of samples, showing better results than their counterparts.



(a) ChisSquared error (b) ReliefF error (c) InfoGain error (d) mRMR error

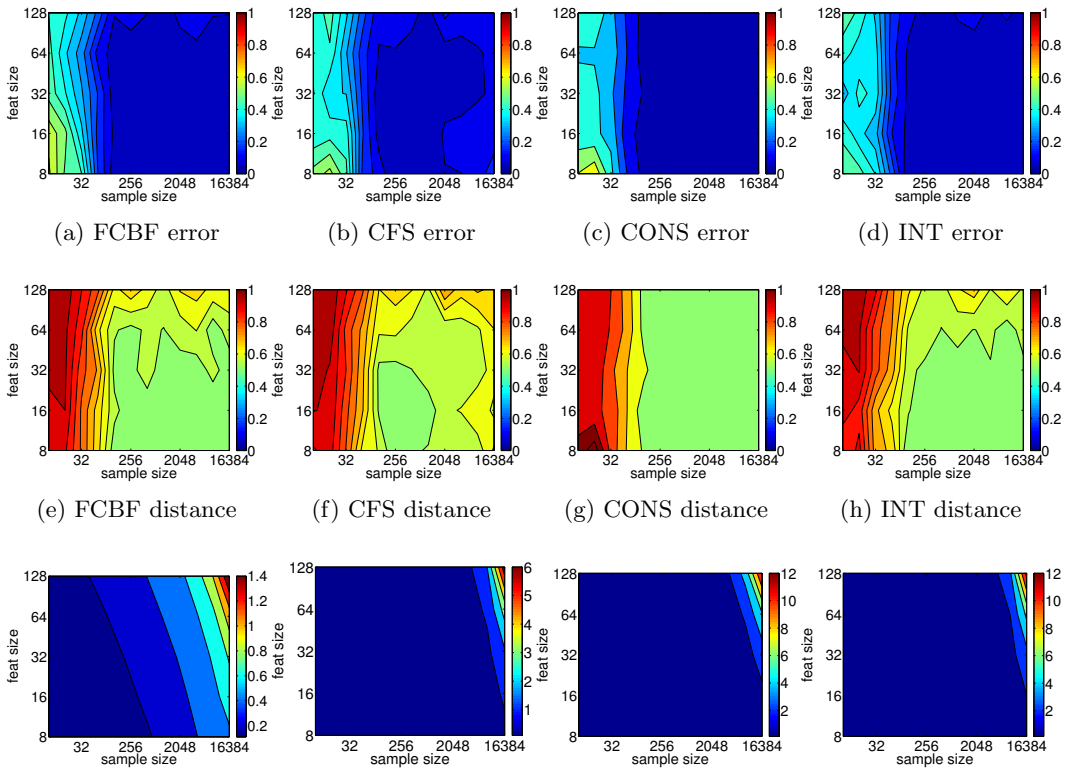
(e) ChisSquared distance (f) ReliefF distance (g) InfoGain distance (h) mRMR distance

(i) ChisSquared time (j) ReliefF time (k) InfoGain time (l) mRMR time

Figure 6.2: Measures of scalability of ranker selection methods in the Corral dataset, showing feature size vs. sample size

Regarding the training time, it has to be noted that these plots are not bounded between 0 and 1, since the time has not been normalized. For this reason, the reader has to bear in mind that in some cases the time is up to 500 seconds (ReliefF, figure 6.2j), whilst in other cases the time only raises until 1 second (FCBF, figure 6.3i). Having said that, the training time raises exponentially with large amounts of both samples and features, although it seems to be more affected by the number of samples. On the

contrary, mRMR is more influenced by the number of features, as it is a multivariate filter which takes into account relationships between features.



(a) FCBF error (b) CFS error (c) CONS error (d) INT error

(e) FCBF distance (f) CFS distance (g) CONS distance (h) INT distance

(i) FCBF time (j) CFS time (k) CONS time (l) INT time

Figure 6.3: Measures of scalability of subset methods in the Corral dataset, showing feature size vs. sample size

For the sake of brevity, it is not possible to plot all the measures for all the datasets considered in this study, therefore Tables 6.10 and 6.11 depict the eight scalar measures defined in Section 6.2.2.3 for the ranker and subset methods, respectively, along with all the pairwise combinations between filter and dataset for the first seven datasets of Table 6.9. Notice that the lower the value, the better the performance of the feature selection method. Since it is difficult to draw conclusions from such amount of information, we will apply some statistical tests to determine which feature selection performs better in terms of the three groups of measures: error, distance and time.

The Friedman test (M. Friedman, 1937) is a non-parametric equivalent of the repeated-measures ANOVA. It ranks the algorithms for each data set separately, the best performing algorithm getting the rank of 1, the second best rank 2, etc. If the null-hypothesis is rejected, we can proceed with a post-hoc test. The Nemenyi test

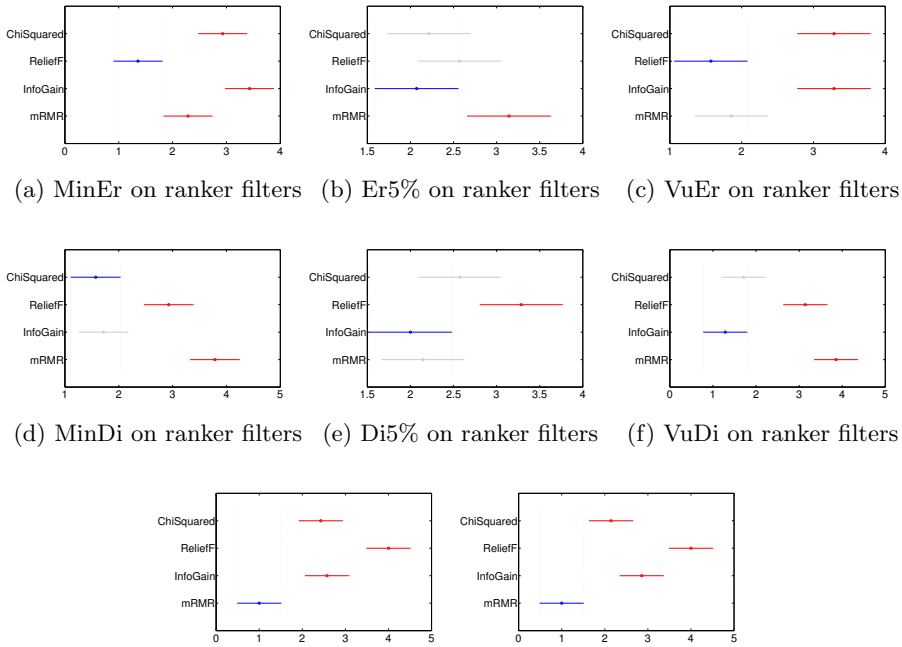
Table 6.10: Precision, stability and time measures for ranker filters on classical datasets

	Dataset	MinEr	Er5%	VuEr	MinDi	Di5%	VuDi	MaxTt	VuTt
Chi-Squared	Corral	0.0080	32768	9.0826	0.0145	16384	7.8864	6.2475	16.9977
	Led	0.0000	256	2.1907	0.0000	256	4.2786	6.4141	16.9912
	Monk1	0.5925	128	30.4160	0.0000	512	6.3249	6.4752	16.6834
	Monk2	0.0000	262144	23.6574	0.0551	262144	11.3443	6.4105	16.8058
	Monk3	0.5142	512	29.0216	0.0534	1048576	8.6334	6.5255	17.9424
	XOR	0.7650	128	36.6948	0.0000	256	5.6573	6.3941	16.8869
	Parity	0.6822	512	34.3876	0.0000	512	5.7985	6.3602	16.8149
ReliefF	Corral	0.0000	2048	4.1311	0.2148	131072	28.7959	547.3706	606.7580
	Led	0.0000	256	2.1828	0.0000	256	22.9613	545.3251	610.5523
	Monk1	0.0000	32768	25.0318	0.0317	131072	33.1593	621.0799	683.9469
	Monk2	0.0000	2048	12.3433	0.2918	16384	31.5688	621.1729	683.3628
	Monk3	0.0000	32768	26.1069	0.1587	131072	35.4930	608.1903	676.2464
	XOR	0.0000	512	3.2857	0.0169	131072	34.0864	538.0400	592.2760
	Parity	0.0000	1024	8.3501	0.0794	65536	32.8342	539.6729	604.9484
InfoGain	Corral	0.0081	65536	9.1260	0.0303	32768	8.8550	6.4418	17.1277
	Led	0.0000	128	2.1755	0.0000	128	2.8284	6.2209	17.0493
	Monk1	0.5881	512	30.5219	0.0000	512	5.9126	6.3224	16.8259
	Monk2	0.0192	131072	23.7973	0.0858	16384	11.8515	6.4306	16.9691
	Monk3	0.5467	256	28.9641	0.0293	2048	8.3751	6.5071	19.3314
	XOR	0.6440	512	36.7136	0.0000	2048	5.5588	6.3952	16.4992
	Parity	0.7197	128	34.2097	0.0000	1024	4.9880	6.3817	16.8673
mRMR	Corral	0.0081	32768	6.3044	0.2151	131072	29.2121	5.8034	15.9463
	Led	0.0000	256	2.8337	0.0000	256	22.6035	5.4714	15.7171
	Monk1	0.1809	4096	14.2597	0.6635	128	37.0434	4.3368	14.5079
	Monk2	0.0000	32768	12.2265	0.3335	131072	35.1601	4.2203	14.3078
	Monk3	0.1427	524288	10.8967	0.6087	32768	35.8379	4.2913	14.5225
	XOR	0.4087	1024	24.8969	0.6802	256	37.7154	6.2138	16.1184
	Parity	0.5071	1024	28.0068	0.6966	128	37.9024	5.6746	15.9854

Table 6.11: Precision, stability and time measures for subset filters on classical datasets

	Dataset	MinEr	Er5%	VuEr	MinDi	Di5%	VuDi	MaxTt	VuTt
FCBF	Corral	0.0556	2048	6.5688	0.5000	1024	27.9232	1.5979	14.0284
	Led	0.0424	8192	4.4397	0.5000	64	23.3853	1.5778	14.1738
	Monk1	0.3792	32768	21.7247	0.5000	512	29.7338	1.6233	13.7766
	Monk2	0.0216	131072	19.9815	0.5000	4096	35.5311	1.6215	13.8563
	Monk3	0.1983	262144	15.4796	0.5000	512	28.9344	1.5939	13.7829
	XOR	0.4883	2048	24.8144	0.5000	512	34.7336	1.5812	13.8722
	Parity	0.4896	512	25.9331	0.5000	256	33.7904	1.6173	13.9016
CFS	Corral	0.0556	2048	7.2957	0.5000	2048	29.1369	6.6850	17.7719
	Led	0.0424	8192	4.3782	0.5000	256	23.3411	6.7841	18.7223
	Monk1	0.3765	8192	21.5832	0.5000	512	29.9757	6.7498	18.0770
	Monk2	0.0794	2097152	18.0169	0.5941	256	35.9813	6.8811	18.6108
	Monk3	0.2009	1048576	15.3117	0.5000	65536	29.3408	7.0857	25.1294
	XOR	0.1979	1024	14.1757	0.5000	128	33.7684	6.6487	17.6479
	Parity	0.3094	512	18.0365	0.5000	256	34.9355	6.5489	18.1507
Cons	Corral	0.0466	8192	6.0037	0.5000	2048	27.0504	12.1497	26.9764
	Led	0.0911	4096	6.3158	0.5000	128	22.5916	11.5639	26.8334
	Monk1	0.3767	16384	21.5503	0.5000	512	26.3205	9.1498	22.6751
	Monk2	0.4739	2048	28.0390	0.9667	256	59.8062	8.6256	23.0369
	Monk3	0.1784	524288	15.0830	0.5000	1024	26.1744	10.4860	24.3033
	XOR	0.4375	256	23.7986	0.9880	256	52.9460	8.8971	22.2011
	Parity	0.4895	512	25.0176	0.9852	128	53.4165	9.2617	22.5169
INTERACT	Corral	0.0556	1024	6.3456	0.5000	1024	27.6555	12.6862	26.2053
	Led	0.0424	8192	4.4217	0.5000	256	23.4841	12.3948	26.5851
	Monk1	0.3729	32768	21.4582	0.5000	512	29.6188	12.5901	26.3971
	Monk2	0.0061	524288	17.7676	0.5167	131072	35.7957	12.7796	27.5088
	Monk3	0.1916	524288	15.2224	0.5000	1024	28.4826	12.5099	26.0087
	XOR	0.1979	32768	14.4977	0.5000	1024	34.4921	12.3733	25.8947
	Parity	0.3094	4096	17.7334	0.5000	512	34.1927	12.4180	26.6287

(Nemenyi, 1963) is similar to the Tukey test for ANOVA and is used when all algorithms are compared to each other. The performance of two algorithms is significantly different if the corresponding average ranks differ by at least a critical difference.



(a) MinEr on ranker filters (b) Er5% on ranker filters (c) VuEr on ranker filters

(d) MinDi on ranker filters (e) Di5% on ranker filters (f) VuDi on ranker filters

(g) MaxTt on ranker filters (h) VuTt on ranker filters

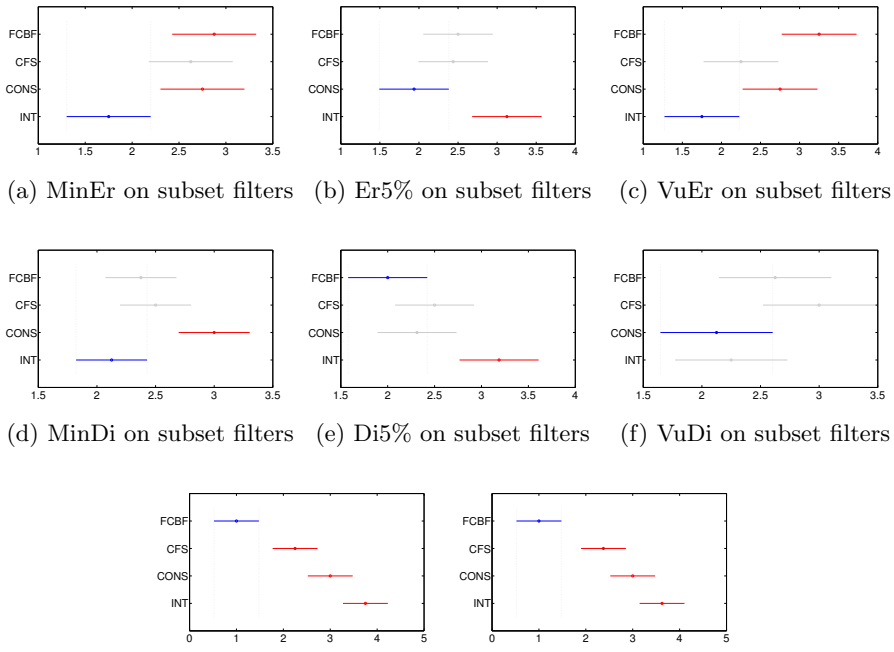
Figure 6.4: Comparison of scalability measures for ranker filters (ChiSquared, InfoGain, ReliefF and mRMR)

Figure 6.4 plots the results of the statistical tests carried out to compare the ranker filters according to their performance on all datasets. In terms of error, ReliefF clearly outperforms the other methods according to the minimum error achieved. In fact, in Table 6.10, one can see that the minimum error obtained by this algorithm is zero for all datasets. As expected, this is reflected also in the volume under the curve, where ReliefF is better than ChiSquared and InfoGain with significant differences. Focusing on the minimum amount of data for which the error drops below the 5% of its minimum value (Er5%), InfoGain is the only one which improves significantly its performance with respect to mRMR, whilst among the other methods there are no significant differences.

However, when examining the results related with the distance measure, InfoGain and ChiSquared are significantly better than ReliefF and mRMR in two metrics (MinDi and VuDi). On the other hand, InfoGain and mRMR are significantly better than

ReliefF in terms of Di5%, which means that although mRMR is not the method which achieves the minimum distance, it stabilizes rapidly. Finally, with regard to the training time, mRMR clearly beats the other methods. Remark that ReliefF obtains very good results in terms of error (0%) even when for XOR and Parity the minimum error obtained by other methods ranges from 40.87% to 76.50%. Nevertheless, this is accomplished at the expense of a much longer training time than the other rankers (see Table 6.10, around 100 times longer).

Figure 6.5, in turn, displays the results of the statistical tests executed to compare the different subset filters based on their scalability properties on all datasets. With regard to the error, the results obtained by INTERACT outperform significantly those achieved by FCBF and Consistency-based in terms of both MinEr and VuEr. However, regarding Er5%, the performance of the filter Consistency-based is significantly better than INTERACT, which means that the former requires a significantly smaller amount of data than the latter to achieve a low error.



(a) MinEr on subset filters (b) Er5% on subset filters (c) VuEr on subset filters

(d) MinDi on subset filters (e) Di5% on subset filters (f) VuDi on subset filters

(g) MaxTt on subset filters (h) VuTt on subset filters

Figure 6.5: Comparison of scalability measures for subset filters (FCBF, CFS, Cons and INTERACT)

In terms of distance, FCBF achieves a low value in a significantly smaller amount of data than INTERACT (see Figure 6.5e), as well as outperforming the filter Consistency-

Table 6.12: Overview of the behavior regarding scalability of filter methods.

Method	Error	Distance	Training time
Chi-Squared	••	•••	••••
ReliefF	•••••	•••	•
InfoGain	••	•••••	••••
mRMR	•••	••	•••••
FCBF	•••	••••	•••••
CFS	•••	•••	•••
Consistency	•••	••	••
INTERACT	••••	•••	••

based significantly regarding the minimum distance. Regarding the training time, FCBF outperforms significantly the remaining algorithms.

In general terms, one can say that FCBF shows a good behavior in terms of stability and training time at the expense of a slightly degradation in error. In fact, only INTERACT is significantly better than FCBF in terms of minimum error (actually, in 3 out of the 7 datasets considered). However, FCBF requires a maximum training time around 1 second whilst InfoGain needs more than 12 seconds. Moreover, FCBF becomes stable with the smallest amount of data.

In light of those results, Table 6.12 provides an overview regarding the specific scalability aspects considered for classical datasets. Notice that the larger the number of bullets, the better the behavior. It remains clear the predominance of ReliefF in terms of accuracy, although InfoGain shows better performance according to stability. The methods that require a smaller training time are mRMR and FCBF.

6.2.3.2 Case of study: SD datasets

These synthetic datasets (Appendix I, Section I.2.1.6) have a small ratio between number of samples and features, which makes difficult the task of feature selection. This is the problematic present in microarray data, a hard challenge for machine learning researchers. Besides these particularities of the data, there is a high number of irrelevant features for the task of gene classification and also the presence of redundant variables is a critical issue.

Table 6.13: Precision, stability and time measures for ranker filters on SD datasets

	Dataset	MinEr	Er5%	VuEr	MinDi	Di5%	VuDi	MaxTt	VuTt
Chi-Sq	SD1	0.0010	995328	2.9980	0.0069	995328	2.0839	1.5609	7.6786
	SD2	0.0287	497664	4.4338	0.0125	995328	2.6706	1.5655	7.7935
	SD3	0.0288	995328	5.3982	0.0150	995328	3.5776	1.6365	8.0069
ReliefF	SD1	0.0019	995328	2.4878	0.4578	15552	15.5683	6.1160	11.2931
	SD2	0.0212	995328	5.1582	0.4425	15552	15.4352	6.1757	11.2726
	SD3	0.0332	995328	6.9056	0.5372	31104	15.7661	5.9989	11.4583
IG	SD1	0.0012	995328	2.8732	0.0065	995328	2.0854	1.5320	7.7193
	SD2	0.0173	248832	4.4797	0.0085	124416	2.7332	1.6691	7.9059
	SD3	0.0073	995328	5.3365	0.0201	497664	3.6923	1.4625	7.8284
mRMR	SD1	0.0009	995328	3.2350	0.5121	15552	15.1389	1178.9798	3732.9157
	SD2	0.0185	497664	6.3140	0.5065	5184	14.6257	1143.7379	3742.0801
	SD3	0.0599	15552	7.1948	0.4999	10368	14.9588	1161.7751	3751.4456

Tables 6.13 and 6.14 report the eight scalar measures defined in Section 6.2.2.3 for the ranker and subset methods, respectively, along with all the pairwise combinations between filter and dataset for the SD datasets (three last rows of Table 6.9). Focusing on the ranker methods, one can see that in terms of error (MinEr, Er5% and VuEr), there are not clear differences. In turn, with regard to the distance, Chi-Squared and InfoGain return stable rankings (low distance) but they need a significant amount of data to do it, whilst ReliefF and mRMR show high distances (around 50%). Finally, the training time required by mRMR is in the order of thousands of seconds while the remaining methods require in the order of seconds. This is due to the fact that mRMR is a multivariate filter and so the time raises exponentially when the number of features increases (in these experiments, up to 4096 features). In this scenario, the best ranker seems to be InfoGain, since it achieves a low error, it becomes stable although requires certain amount of data, and the computational cost is acceptable (low training time).

Table 6.14 displays the results devoted to the subset filters. First of all, it is worth mentioning that the high distance results are due to the fact that the SD datasets have an extremely high number of features (up to 4096), so the more features, the more difficult is to select stable subsets of features. Having said that, FCBF seems to be the best subset method, since it gets the lowest errors, for the three datasets, and the maximum training time consumed is around 1 second whilst the remaining methods takes in the order of tens of seconds. To sum up, Table 6.15 provides some guidelines for the specific scalability aspects considered for the SD datasets. Note that the larger the number of bullets, the better the behavior.

Table 6.14: Precision, stability and time measures for subset filters on SD datasets

	Dataset	MinEr	Er5%	VuEr	MinDi	Di5%	VuDi	MaxTt	VuTt
FCBF	SD1	0.0000	15552	5.5601	0.9407	1728	17.7571	1.7655	8.1437
	SD2	0.0206	15552	5.6811	0.9546	576	17.8004	1.6179	8.1407
	SD3	0.0053	15552	5.7787	0.9704	576	17.8314	1.5242	8.1623
CFS	SD1	0.1995	5184	6.4969	0.9080	1728	17.4210	31.8743	57.5217
	SD2	0.2301	31104	6.5341	0.9121	3456	17.4970	36.2336	64.3233
	SD3	0.2162	62208	6.4938	0.9253	576	17.5632	37.6943	67.9754
Cons	SD1	0.1701	10368	6.7155	0.8450	15552	17.1870	9.4680	16.8951
	SD2	0.1833	10368	6.6635	0.8750	15552	17.3662	10.3833	18.3579
	SD3	0.2461	5184	6.6344	0.8914	15552	17.4645	11.8190	19.3837
INT	SD1	0.1570	20736	6.6934	0.8425	15552	17.2295	8.1040	18.6182
	SD2	0.2353	5184	6.5664	0.8985	15552	17.4977	10.8081	19.9816
	SD3	0.2404	124416	6.5725	0.9095	576	17.5437	16.0517	22.1883

Table 6.15: Overview of the behavior regarding scalability of filters on SD datasets.

Method	Error	Distance	Training time
Chi-Squared	●●●●	●●●●	●●●●●
ReliefF	●●●●	●●	●●●●
InfoGain	●●●●	●●●●	●●●●●
mRMR	●●●●	●●	●
FCBF	●●●●●	●	●●●●●
CFS	●●●	●	●●
Consistency	●●●	●●	●●●
INTERACT	●●●	●●	●●●

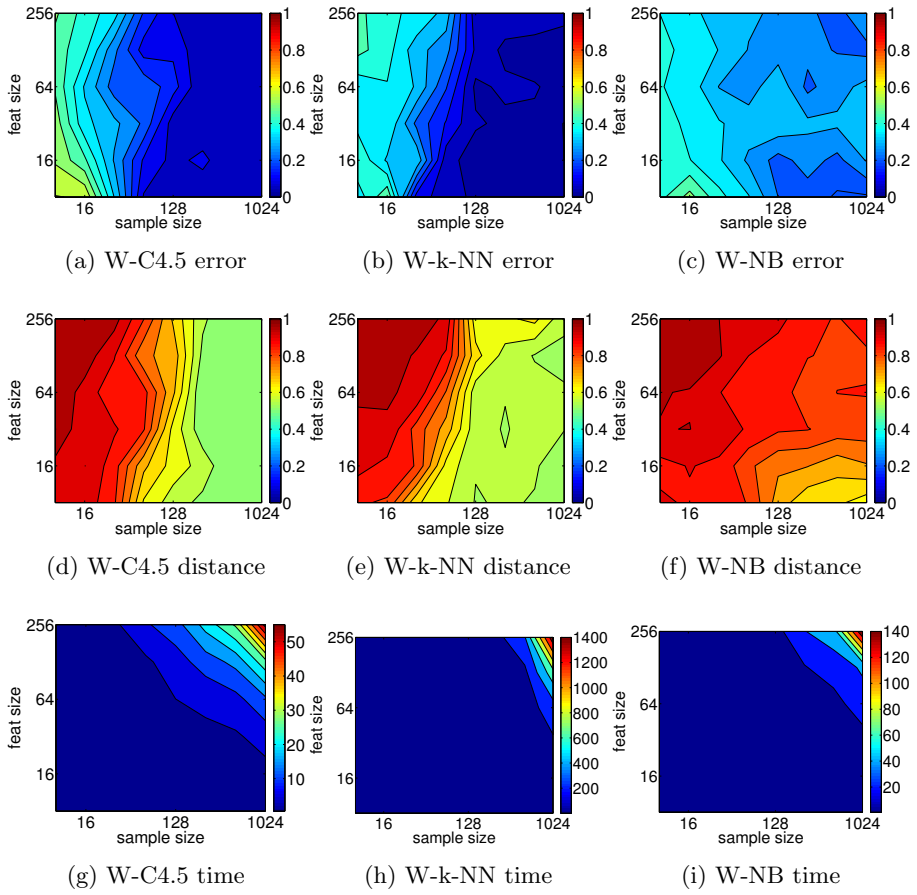
6.2.3.3 Scalability of wrappers

This section reveals the scalability of wrapper methods. For this sake, we have chosen three representative classifiers (C4.5, k-NN and naive Bayes, see Appendix I) which will be used to assess the relative usefulness of the subsets of variables. Notice that the search strategy is *best first*, starting with the empty set of attributes and searching forward. Figure 6.6 displays the results achieved by the wrapper combined with these three classifiers applied on the Corral dataset. It can be seen that, in terms of error, C4.5 and k-NN reported an acceptable behavior, although it seems that they are more affected by the sample dimension and the minimum error is achieved with around 128 samples. The minimum error obtained by the wrapper using naive-Bayes is slightly higher, but this method appears to be more stable. A similar behavior was also shown on the remaining datasets (see Table 6.16). The wrapper combined with C4.5 and k-NN obtained similar results for MinEr, whilst when combined with naive Bayes, it requires more data to reach Er5%.

Table 6.16: Precision, stability and time measures for wrappers on classical datasets

	Dataset	MinEr	Er5%	VuEr	MinDi	Di5%	VuDi	MaxTt	VuTt
W-C4.5	Corral	0.0548	8192	7.0565	0.5000	4096	25.4117	56.8844	181.8961
	Led	0.0439	32768	6.4084	0.5000	4096	23.0851	64.8711	200.3963
	Led3	0.1725	131072	11.8689	0.5000	8192	24.2955	50.9306	168.4447
	Parity	0.2104	8192	16.4408	0.7742	4096	33.4075	394.2772	616.2489
	XOR	0.0975	1024	10.3626	0.6274	1024	30.6118	183.1688	420.8220
W-k-NN	Corral	0.0167	65536	6.1080	0.5200	8192	25.0587	1539.2690	1720.1047
	Led	0.0470	16384	5.2805	0.5167	512	21.2260	919.9601	1125.8321
	Monk3	0.2217	16384	12.6873	0.5672	2048	28.0120	1567.8623	1670.3355
	Parity	0.1993	1024	17.1441	0.7222	512	32.0195	10073.3259	8702.4207
	XOR	0.1712	256	15.1980	0.7046	256	31.8317	11076.9449	7814.4760
W-NB	Corral	0.1926	4096	10.9833	0.6230	8192	30.0811	157.3330	262.6080
	Led	0.0504	32768	5.2008	0.5000	1024	22.5974	110.8897	302.8381
	Monk3	0.3083	8192	13.7294	0.6484	256	29.0853	83.4932	186.0066
	Parity	0.4385	2048	18.1368	0.9171	128	34.1900	72.6553	194.4969
	XOR	0.3838	256	17.1055	0.9052	256	33.9530	71.8749	197.2576

In terms of distance, a similar behavior to that of the error is reported in Figure 6.6. The wrapper combined with C4.5 and k-NN obtained the minimum distance using around 128 samples, whereas naive Bayes, although being more stable, achieved a higher minimum distance. Focusing on Table 6.16, it is easy to note that this circumstance also happened with the rest of the datasets. The most different results were found



(a) W-C4.5 error (b) W-k-NN error (c) W-NB error

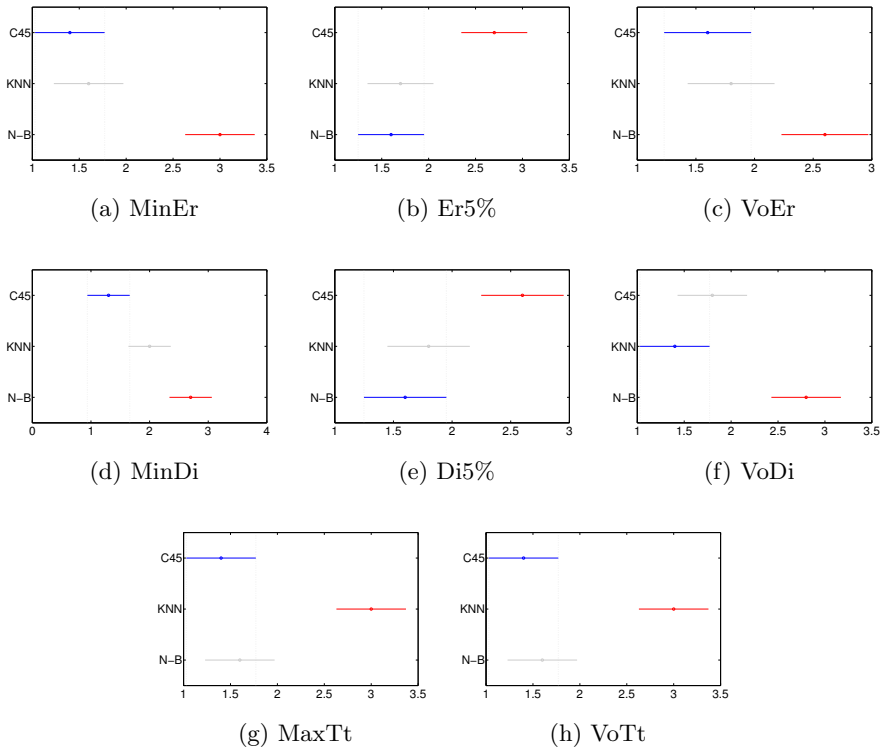
(d) W-C4.5 distance (e) W-k-NN distance (f) W-NB distance

(g) W-C4.5 time (h) W-k-NN time (i) W-NB time

Figure 6.6: Measures of scalability of wrappers in the Corral dataset, showing feature size vs. sample size

on the training time. According to Table 6.16, the wrapper using C4.5 employed the lowest time on datasets Corral, Led and Monk3. When combining the wrapper with naive Bayes, the lowest values were achieved on Parity and XOR, whereas the k-NN classifier caused the wrapper to consume more training time (around 139 times higher when comparing the MaxTt value on Parity dataset with those of the k-NN and naive Bayes learning methods).

Figure 6.7 shows the results of the statistical tests executed to compare the different wrapper methods based on their scalability properties on all datasets. For MinEr and MinDi, the use of C4.5 as learning algorithm outperformed those results achieved by naive Bayes, although the latter presented the best results in terms of Er5% and Di5%. As for the training time, C4.5 and naive Bayes required significantly less time than k-NN.



(a) MinEr (b) Er5% (c) VoEr

(d) MinDi (e) Di5% (f) VoDi

(g) MaxTt (h) VoTt

Figure 6.7: Comparison of scalability measures for wrappers (combined with C4.5, k-NN and naive Bayes)

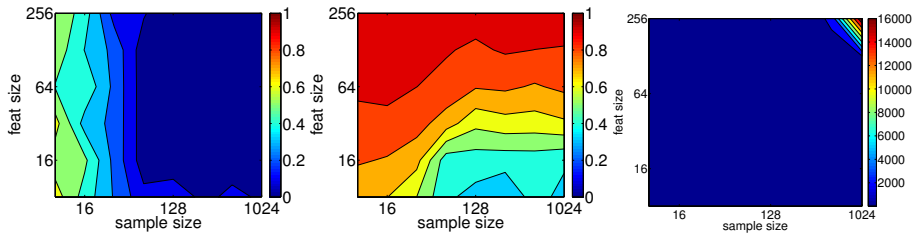
In light of the above, the best learning algorithm to be combined with the wrapper seems to be C4.5. Although k-NN shows a good behavior in terms of error and distance, the training time is quite high. C4.5 achieves a lower error and distance in a shorter time. To sum up, Table 6.17 provides some guidelines for the specific scalability aspects considered for wrapper methods. Notice that the larger the number of bullets, the better the behavior.

Table 6.17: Overview of the behavior regarding scalability of wrappers.

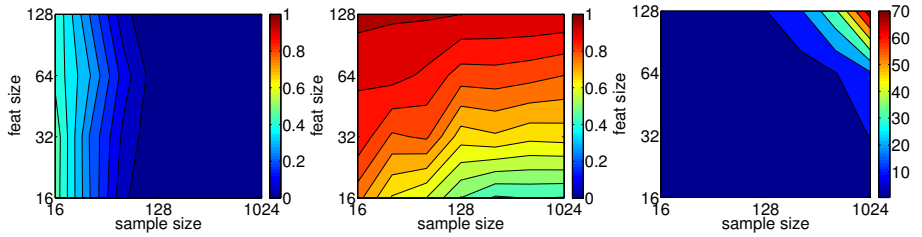
Method	Error	Distance	Training time
W-C45	●●●●	●●●	●●●
W-k-NN	●●●●	●●●	●
W-NB	●●●	●●	●●

6.2.3.4 Scalability of embedded methods

Finally, this section studies the scalability of two embedded methods: FS-P and SVM-RFE (see Chapter 2, Section 2.2.2). Figure 6.8c shows the results obtained with Corral dataset. As can be seen, the maximum training time required by SVM-RFE is around 16000 seconds. This high time is due to the recursive nature of the method so it prevented its application to the remaining datasets.



(a) SVM-RFE error (b) SVM-RFE distance (c) SVM-RFE time



(d) FS-P error (e) FS-P distance (f) FS-P time

Figure 6.8: Measures of scalability of embedded methods in the Corral dataset, showing feature size vs. sample size

In terms of error (Figure 6.8 and Table 6.18), the embedded methods seem to be more affected by the number of samples than of features. Both of them achieved the minimum error with around 128 samples, but FS-P performed better. Regarding the distance, their performance is comparable to those of the ranker filters ReliefF and mRMR, being more affected by the number of features than of samples.

Table 6.18: Precision, stability and time measures for embedded methods on Corral dataset

MEthod	MinEr	Er5%	VoEr	MinDi	Di5%	VoDi	MaxTt	VoTt
SVM-RFE	0.0017	16384	5.5982	0.3138	2048	26.0390	17959.8755	7808.8825
FS-P	0.0000	4096	1.6481	0.4422	4096	13.8323	76.1025	116.2191

Focusing on the training time, SVM-REF requires a much longer time than FS-P. For this reason, FS-P seems to be a better option since in the rest of the measures the performance is similar. However, only one dataset is not enough to draw strong conclusions. For this reason, FS-P was applied to the remaining datasets.

Tables 6.19 and 6.20 report the eight scalar measures defined in section 6.2.2.3 after applying the FS-P embedded method to classical and SD datasets, respectively. In terms of error, FS-P obtains the best results with Corral and Led datasets whilst it shows a poor performance with non-linear datasets, such as XOR or the Monk problems. This result may be caused because FS-P uses a linear perceptron so it can only solve linear problems (see Chapter 3). Regarding stability, FS-P obtains poor results, except for the case of Led dataset, but at the expense of needing a larger amount of data. Finally, the highest training times were required by the SD datasets, since they have the largest number of features. Among the classical datasets (Table 6.19), it is surprising the maximum training time obtained with Led dataset, which is almost four times that of the remaining datasets. It can be due because of the fact that Led dataset is the only multiclass dataset, which complicates the learning process.

Table 6.19: Precision, stability and time measures for embedded method FS-P on classical datasets

Method	MinEr	Er5%	VoEr	MinDi	Di5%	VoDi	MaxTt	VoTt
Corral	0.0000	4096	1.6481	0.4422	4096	13.8323	76.1025	116.2191
Led	0.0000	1024	0.7920	0.2765	16384	11.9760	287.0183	435.4861
Monk1	0.5395	2048	11.2919	0.7919	256	16.1382	77.8572	117.4911
Monk2	0.5776	16384	11.9769	0.7718	512	16.0206	77.1672	117.0610
Monk3	0.4983	2048	10.5079	0.7917	2048	15.9776	77.0355	116.9248
XOR	0.3054	2048	9.2897	0.8003	256	16.0837	78.9096	117.9691
Parity	0.2362	32768	10.5824	0.8108	512	16.0578	77.0948	117.3677

Table 6.20: Precision, stability and time measures for embedded method FS-P on SD datasets

Method	MinEr	Er5%	VoEr	MinDi	Di5%	VoDi	MaxTt	VoTt
SD1	0.0135	995328	4.6949	0.7888	5184	11.2110	582.1999	695.6317
SD2	0.0884	995328	7.0557	0.6103	1728	10.9029	581.9035	694.7263
SD3	0.1054	995328	7.4857	0.7308	10368	11.1356	578.1531	695.1090

Table 6.21 provides a summary of the behavior of the two embedded methods tested, in which the larger the number of bullets, the better the behavior. Notice that the comparative is not complete, since SVM-RFE only was applied to Corral dataset.

Table 6.21: Overview of the behavior regarding scalability of embedded methods. SVM-RFE has only been tested with Corral dataset.

Method	Error	Distance	Training time
SVM-RFE	●●●	●●●	●
FS-P	●●●●	●●	●●●

6.2.3.5 Comparative among filters, wrappers and embedded

This section compares the three different types of feature selection methods according to their scalability properties. Since it was not feasible to test all the methods on the same datasets and settings, an in-depth comparative study cannot be accomplished. A brief analysis focusing on Corral dataset (see Appendix I, Section I.2.1.1) is shown in Table 6.22.

Table 6.22: Comparative of the scalability properties of filters, embedded and wrappers on Corral dataset

Method	MinEr	Er5%	VoEr	MinDi	Di5%	VoDi	MaxTt	VoTt
Chi-Squared	0.0080	32768	9.0826	0.0145	16384	7.8864	6.2475	16.9977
ReliefF	0.0000	2048	4.1311	0.2148	131072	28.7959	547.3706	606.7580
InfoGain	0.0081	65536	9.1260	0.0303	32768	8.8550	6.4418	17.1277
mRMR	0.0081	32768	6.3044	0.2151	131072	29.2121	5.8034	15.9463
FCBF	0.0556	2048	6.5688	0.5000	1024	27.9232	1.5979	14.0284
CFS	0.0556	2048	7.2957	0.5000	2048	29.1369	6.6850	17.7719
Cons	0.0466	8192	6.0037	0.5000	2048	27.0504	12.1497	26.9764
INTERACT	0.0556	1024	6.3456	0.5000	1024	27.6555	12.6862	26.2053
W-C4.5	0.0548	8192	7.0565	0.5000	4096	25.4117	56.8844	181.8961
W-k-NN	0.0167	65536	6.1080	0.5200	8192	25.0587	1539.2690	1720.1047
W-NB	0.1926	4096	10.9833	0.6230	8192	30.0811	157.3330	262.6080
SVM-RFE	0.0017	16384	5.5982	0.3138	2048	26.0390	17959.8755	7808.8825
FS-P	0.0000	4096	1.6481	0.4422	4096	13.8323	76.1025	116.2191

In terms of error, the embedded methods achieved lower values than wrappers and filters, except for ReliefF, which obtained a minimum error of 0%. With regard to the distance, the three types of methods obtained similar results, although is worth highlighting the good performance of Chi-Squared and InfoGain, being both of them univariate filters. As for the training time, all the filters except for ReliefF required lower times than wrappers and embedded, as it was expected. If comparing the wrapper and the embedded model, the former combined with C4.5 needed less time than embedded methods. SVM-RFE is the method which presents the higher training times, followed by k-NN and naive Bayes combined with the wrapper.

As expected, filters seem to be the best option since, in general, showed a better performance than embedded and wrappers. However, this comparative was only over one dataset and there exist some variations in the number of features and samples tested for each method.

6.3 Summary

When dealing with the performance of machine learning algorithms, most studies are focused on the accuracy obtained by the algorithm. However, with the advent of high dimensionality problems, researchers must study not only accuracy but also scalability. Aiming at dealing with a problem as large as possible, feature selection can be helpful as it reduces the input dimensionality and therefore the run-time required by an algorithm.

In this chapter, the effectiveness of feature selection on the scalability of training algorithms for ANNs was evaluated, both for classification and regression tasks. Since there are no standard measures of scalability, those defined in the *PASCAL Large Scale Learning Challenge* were used to assess the scalability of the algorithms in terms of error, computational effort, allocated memory and training time. Results showed that feature selection as a preprocessing step is beneficial for the scalability of ANNs, even allowing certain algorithms to be able to train on some datasets in cases where it was impossible due to the spatial complexity. Moreover, some conclusions about the adequacy of the different feature selection methods over this problem were extracted.

Then, an analysis of the scalability of feature selection methods was presented, an important issue that however has not received much consideration in the literature. Eight well-known filter-based feature selection algorithms were evaluated, covering both

ranking and subset methods. Moreover, some representatives of embedded and wrapper methods were also considered in this study. A suite of ten artificial datasets was chosen, so as to be able to assess the degree of closeness to the optimal solution in a confident way. For determining the scalability of the methods, several new measures were proposed, based not only in accuracy but also in execution time and stability, and their adequacy was demonstrated. In light of the experimental results, filters seem to be the most scalable feature selection methods. Specifically, FCBF obtained the best performance in terms of scalability. As for the ranker methods, ReliefF is a good choice when having a small number of features (up to 128) at the expense of a long training time. For this reason, when dealing with extremely-high datasets, Information Gain demonstrated better scalability properties.

PART II

Novel feature selection methods

CHAPTER 7

Combination of discretization and feature selection methods

After analyzing the particularities of feature selection methods and demonstrating their adequacy on several scenarios, the second part of this thesis is devoted to developing novel feature selection methods capable of being applied to high dimensional datasets. The first step is to analyze how preprocessing techniques prior to feature selection can affect the accuracy of the selection and subsequent classification. Therefore, this chapter is committed to propose novel methods which take into account the discretization of the features.

As mentioned in previous chapters, in complex classification domains features or attributes may contain false correlations, which hinder the underlying process and in general, the learning task to be carried out. Furthermore, some features may be irrelevant and some others may be redundant since the information they add is contained in other features. These extra features can increase computation time, and can have an impact on the accuracy of the classifier built. For this reason, these classification domains seem to be suitable for the application of feature selection methods. Feature selection methods are adequate for both classification and regression tasks, although most of the research done is related to classification. In the feature selection field, it is difficult to find methods that can deal directly with multiple class problems, as very little research has been done in this aspect (Chidlovskii & Lecerf, 2008; Bruzzone & Serpico, 2000). In these studies, the multiple class approaches are shown to suffer from the so-called accumulative effect, which becomes more visible when the number of classes grows and results in removing relevant and unredundant features. The main difficulties to be taken into account in multiple class algorithms are the following:

- The data set presents one or several classes that contain a considerable higher number of samples than the data of the other classes (unbalanced classes).

- Determining which features are appropriate for each class is complicated, because feature selection results in a set of attributes that could represent only the majority classes (Bosin, Dessì, & Pes, 2007).

There are two main approaches for dealing with classification problems that involve more than two classes. One tries to transform the multiple class problem in several binary problems, while the other deals directly with the multiple class problem. This last strategy can present overtraining for those classes that are easily separable or those classes that are the majority in the dataset, i.e. those classes with a much higher number of representative samples than the other classes (Forman, 2003). However, the first strategy also presents some drawbacks, such as how to integrate the information that comes from each of the binary classifiers, or the fact that there may not exist enough representation of a specific class in the training set generated from the original one to train the binary classifier.

On top of this, many filter algorithms are shown to work on discrete data (Liu & Setiono, 1997). In order to deal with numeric attributes, a common practice for those algorithms is to discretize the data before conducting feature selection. For both users and experts, discrete features are easier to understand, use, and explain and discretization can make learning more accurate and faster (Liu, Hussain, Tan, & Dash, 2002). In general, the results obtained (decision trees, induction rules) by using discrete features are usually more compact, shorter and more accurate than by using continuous ones, hence the results can be more closely examined, compared, used and reused. In addition to the many advantages of having discrete data over continuous one, a suite of classification learning algorithms can only deal with discrete data. Besides, the well-known Weka tool (Witten & Frank, 2005), commonly used for the filtering process, uses discretization by default.

In essence, the process of discretization (Janssens, Brijs, Vanhoof, & Wets, 2006) involves the grouping of continuous values into a number of discrete intervals. However, the decision of which continuous values group together, how many intervals to generate, and thus where to position the interval cutpoints on the continuous scale of attribute values is not always identical for the different discretization methods.

In this chapter, a method consisting of a combination of discretizers, filters and classifiers is presented. The main goal of this method is to significantly reduce the number of features while maintaining the performance of the classifiers, or even improving it.

Filters were selected because they are fast and more suitable for large data sets. The first discretization step is required for some filters and its election may influence the selection done by the filter and thus the performance results achieved by the classifier.

The proposed method will be firstly applied over the KDD (Knowledge Discovery and Data Mining Tools Conference) Cup 99 benchmark dataset (*KDD Cup 99 Dataset*, n.d.) to test its effectiveness and to compare the results achieved with those available in the literature, including the KDD Cup 99 winner. The KDD Cup problem will be considered in its two aspects: binary classification (attack or non-attack) and multiple class classification (four types of attack and normal patterns). For the sake of simplicity, the binary problem will be considered first. Some of the classifiers employed in this research only can work with numerical features, so a conversion technique is required to transform symbolic features into numerical ones. Then, in order to reinforce this study, several techniques will be applied. Later, the combinations (discretizer+filter+classifier) exhibiting better performance results will be applied to the multiple class approach. As stated before, this problem can be considered by dealing directly with its multiple class version or by subdividing it into several binary problems. It will be shown that splitting the problem into several binary problems lead to better performance results and besides, features needed for each class are easily reflected. It has to be noticed that different subdivision techniques have been adopted (one versus rest, one versus one) and also different strategies to unify the results obtained by the binary classifiers have been used. Besides, to prevent the problem of unbalanced classes, clearly present in the KDD Cup dataset, an undersampling technique was employed, i.e., some samples for those classes with a high number of samples were discarded.

The results achieved by the proposed method will illustrate its adequacy because it outperforms, in most cases, the results existing in the literature, even the KDD Cup winner, in both its binary and multiclass form. Moreover, the number of features employed is considerably reduced: 17% in both approaches. Once the effectiveness of the proposed method is tested on the KDD Cup 99 benchmark dataset, it will be tested over other challenging scenarios, such as DNA microarray data (see Chapter 4) and several datasets with multiple classes, which have not received the same amount of attention as the binary problems.

7.1 The proposed methodology

In this chapter a combination method for binary and multiclass classification problems is proposed. The method is divided in three steps, which will be following described.

- First, a discretizer method is applied over the input data, with the aim of solving problems of unbalanced values, and preparing the attributes of the sample to be processed by the feature selection algorithm of the next step. Several discretizers have been chosen to test their influence on the classification problem.
- After discretization, feature selection is carried out using filters.
- Finally, a classifier is applied.

Figure 7.1 illustrates the proposed methodology, showing the specific methods used in each step throughout this chapter.

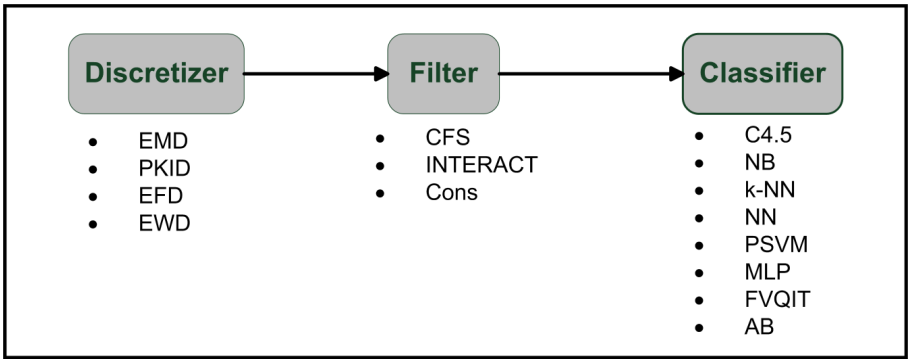


Figure 7.1: The proposed methodology.

7.1.1 Discretization

There exist many discretizators in the literature, but in this research the most suitable for large datasets have been chosen (Y. Yang & Webb, 2002). In this manner, EMD (Entropy Minimization Discretization), EWD (Equal Width Discretization) and EFD (Equal Frequency Discretization) were chosen because they are classic algorithms, and PKID because it is a new approach that works well with large datasets. All of them are described in the following subsections. Previously, some notation considerations

are given: suppose a numeric attribute X_i and n training instances for which the value of X_i is known, the minimum and the maximum value are v_{min} and v_{max} respectively. All the discretization methods first sort the values into ascending order.

7.1.1.1 Entropy Minimization Discretization, EMD

This popular method was created by Fayyad and Irani (1993). EMD evaluates as a candidate cut point the midpoint between each successive pair of the sorted values. For evaluating each candidate cut point, the data are discretized into two intervals and the resulting class information entropy is calculated. A binary discretization is determined by selecting the cut point for which the entropy is minimal amongst all candidates. The binary discretization is applied recursively, always selecting the best cut point. A minimum description length criterion (MDL) is applied to decide when to stop discretization.

7.1.1.2 Proportional k-Interval Discretization, PKID

PKID is a method created by Y. Yang and Webb (2001). The idea behind PKID is that discretization bias and variance relate to interval size and interval number. This strategy seeks an appropriate trade-off between the bias and variance of the probability estimation by adjusting the number and size of intervals to the number of training instances. The following compromise is adopted: given a numeric attribute, supposing we have n training instances with known values for the attribute, we discretize it into $\sqrt{n}$ intervals, with $\sqrt{n}$ instances in each interval. Thus we give equal weight to both bias and variance management. Further, with n increasing, both the number and size of intervals increase correspondingly, which means discretization can decrease both the bias and variance of the probability estimation. This is very desirable, because if a numeric attribute has more instances available, there is more information about it. PKID has greater capacity to take advantage of the additional information inherent to large volumes of training data.

7.1.1.3 Equal Width Discretization, EWD

EWD divides the number line between v_{min} and v_{max} into k intervals of equal width; k is a user predefined parameter and usually is set as 10.

A variant of this method is the Bin-log l. In this case, the number of intervals, k, is established as $k = \max\{1, 2 * \log l\}$, where l is the number of distinct observed values for each attribute (J. Dougherty, Kohavi, & Sahami, 1995).

7.1.1.4 Equal Frequency Discretization, EFD

EFD divides the sorted values into k intervals so that each interval contains approximately the same number of training instances. Thus each interval contains n/k (possibly duplicated) adjacent values; k is a user predefined parameter and usually is set as 10.

7.1.2 Feature selection

Correlation-based Feature Selection (CFS), INTERACT and Consistency-based filters were chosen for this study, with the aim of employing filters that use different performance measurements to select the final features. CFS is one of the most well-known and used filters, INTERACT is a new approach based in the interaction between features and finally, Consistency-based is a classical algorithm. A short description of each one can be found in Chapter 2.

7.1.3 Classification

Each combination of discretizer and filter may lead to different performance results depending on the classifier used, then several classifiers have to be checked. Many datasets, such as the KDD Cup 99, contain different types of attributes (symbolic, numerical, binary) and it is well-known that not all classifiers may deal with symbolic attributes. Therefore, in some cases, a symbolic-conversion is required. Throughout this chapter, different classifiers will be used. Some of them do not need the symbolic-

conversion (C4.5, naive Bayes, k-NN, AdaBoost, FVQIT) whereas others do demand it (one-layer NN, PSVM, MLP). A detailed description of all of them is available in Appendix I, Section I.5.

7.2 The KDD Cup 99 Dataset

The KDD Cup 99 dataset, which derived from the DARPA intrusion detection system (IDS) evaluation dataset (Lippmann et al., 2000), was used for the KDD Cup 99 Competition (*KDD Cup 99 Dataset*, n.d.). The complete dataset has almost 5 million input patterns and each record represents a TCP/IP connection that is composed of 41 features that are both qualitative and quantitative in nature (Stolfo, Fan, Lee, Prodromidis, & Chan, 2000). The dataset used in our study is a smaller subset (10% of the original training set), that contains 494 021 instances and it was already employed as the training set in the competition. For the test set, we used the original KDD Cup 99 dataset containing 331 029 patterns. Around 20% of the two datasets are normal patterns (no attacks). As for attack patterns, the 39 types of attacks are grouped into four categories (Mukkamala, Sung, & Abraham, 2005):

- Denial of Service (DoS) attacks, where an attacker makes some computing or memory resource too busy or too full to handle legitimate requests, thus denying legitimate users access to a machine.
- Probe attacks, where an attacker scans a network to gather information or find known vulnerabilities.
- Remote-to-Local (R2L) attacks, where an attacker sends packets to a machine over a network, then exploits machines vulnerability to illegally gain local access as a user.
- User-to-Root (U2R) attacks, where an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system.

The training and test set percentages for normal activities and for the four attack types are shown in Table 7.1. The KDD Cup 99 problem can be treated under two approaches: the binary case, that consists of distinguishing between attack and no

attack, and the multiple class case, that consists of distinguishing which kind of attack it is.

Table 7.1: Percentages of distribution of normal activities and different kinds of attacks in the KDD Cup 99

Type	% Train Set	% Test Set
Normal	19.69	19.48
DoS	79.24	73.90
Probe	0.83	1.34
R2L	0.23	5.21
U2R	0.01	0.07

As Table 7.1 shows, the percentage of attacks in both datasets is very high, overcoming 80 %, where most of the attacks belong to type DoS. Furthermore, it is a very unbalanced dataset, with some classes (such as U2R and R2L) containing very few samples, which will make their classification difficult during the learning stage.

Table 7.2: Unbalanced continuous attributes of KDD Cup 99 dataset

Feature	Min.	Max.	Mean	StdDev	Distinct
duration	0	58 329	47.98	707.75	2495
src_byes	0	693 375 640	3025.61	988 218.10	3300
dst_bytes	0	5 155 468	868.53	33 040.00	10 725

The KDD Cup 99 training dataset (494 021 instances) is a good candidate to feature selection because of the characteristics of its input attributes. There are two features that are constant (*num_outbound_cmds* and *is_host_login*) and some that are almost constant (*land*, *root_shell*, *num_shells* ...). Apart from constant features, KDD Cup 99 dataset has continuous features that are very skewed and for which a possible solution can be to discretize numeric data. Some examples of these attributes can be viewed in Table 7.2, where it is shown the minimum and maximum value of each feature, as well as its mean, standard deviation and the number of distinct examples.

7.2.1 Results on the binary case

In general, the users of an intrusion detection system (IDS) are interested in differentiating between normal connections and attack situations at a first stage. Later, they would like to distinguish the type of attack that they have suffered. In this sense, the KDD Cup 99 dataset can be considered as a binary problem, detecting normal versus attack patterns, or a multiple class problem, classifying different types of attacks.

The goal of this research is to reduce the number of features of the KDD Cup 99 dataset in both cases, binary and multiple class, but maintaining the performance results. Therefore, the proposed method, consisting of a combination of discretizator, filter and classifier, was applied first, for the sake of simplicity, to the binary case. Later, it will be applied to the multiple class case.

The distribution of the classes for the binary case can be seen in Table 7.3. It can be checked that classes are clearly unbalanced, although there exist enough samples of the minority class (97 278) for the adequate training of the classifiers.

Table 7.3: Percentages of distribution of normal activities and attacks in the KDD Cup 99 training and test datasets for the binary case

Type	% Train Set	% Test set
Normal	19.69	19.48
Attack	80.31	80.52

Table 7.4 illustrates the results achieved in a previous work by the KDD Cup winner and other authors in the literature (Bolón-Canedo, Sánchez-Maroño, & Alonso-Betanzos, 2009) using the whole set of features (41). The performance measures used are the error (percentage of samples incorrectly classified), true positive rate and false positive rate (see Appendix I, Section I.6).

Different filters (CFS, consistency-based and INTERACT, see Chapter 2) were applied to achieve a feature reduction. Moreover, some filters require a discretization step. So, different analysis were carried out to check the influence of the distinct discretizators and filters. The results obtained show that both discretizators and filters have a great influence in the results of the classifiers, as can be seen in a previous work (Bolón-Canedo et al., 2009). In fact, each combination of discretizator and filter leads

Table 7.4: Results obtained in the binary case over the test set by other authors (in %).

Method	Error	True Positives	False Positives
KDD winner	6.70	91.80	0.55
5FNs_poly	6.48	92.45	0.86
5FNs_fourier	6.69	92.72	0.75
5FNs_exp	6.70	92.75	0.75
SVM Linear	6.89	91.83	1.62
SVM 2poly	6.95	91.79	1.74
SVM 3poly	7.10	91.67	1.94
SVM RBF	6.86	91.83	1.43
ANOVA ens.	6.88	91.67	0.90
Pocket 2cl.	6.90	91.80	1.52
Pocket mcl.	6.93	91.86	1.96

to a different subset of features. Therefore, it was not possible to determine which combination was the best and several combinations were tested over the KDD Cup 99 binary dataset.

The classifier stage also affects the performance results, therefore it is necessary to check several classifiers. Two types of classifiers will be tested: those which can deal with symbolic attributes and those which cannot. For the latter, notice that a symbolic-conversion will be required. Next sections will described the experiments related with this issue.

7.2.1.1 Classifiers without symbolic conversion

This subsection shows the results obtained with two classifiers that can deal with both numerical and symbolic attributes and so no conversion is needed: C4.5 and naive Bayes (see Appendix I). In the case of C4.5, the confidence factor parameter was tuned in order to obtain the best results, which has been set to 0.25 and 0.50.

Table 7.5 shows the best results obtained in a previous work (Bolón-Canedo et al., 2009), where C4.5 results outperform the KDD winner score. However, in general, none of the results obtained with naive Bayes improved it and thus only the best result

achieved is included. The measures employed are again the error, true positive's rate and false positive's rate, computed over the same test set, besides of the number of features used. The method applied achieved good results over the binary case, outperforming the KDD winner results in terms of error and true positive rate, while false positive rate maintains reasonable values (see Table 7.4) with only 17% of the total features. It is remarkable the result obtained with the combinaton EMD+INT+C4.5(0.50), since none of the other authors (Table 7.4) have achieved better results than the KDD winner in all three measures. It must be highlighted the low FP rate, while maintaining the Error and TP rates. Notice that the FP rate is a measure of immense importance in determining the quality of an IDS system (Axelsson, 1999).

Table 7.5: Results obtained in the binary case over the test set (in %).

Method	Error	True Positives	False Positives	No. of Features
PKID+Cons+C4.5(0.25)	5.15	94.07	1.90	6
PKID+Cons+C4.5(0.50)	**5.14**	**94.08**	1.92	6
EMD+INT+C4.5(0.25)	6.74	91.73	0.44	7
EMD+INT+C4.5(0.50)	6.69	91.81	0.49	7
PKID+Cons+NB	7.99	90.18	**0.42**	6

7.2.1.2 Classifiers with symbolic conversion

Another three classifiers have been tested over the binary KDD Cup 99: one-layer NN, PSVM and MLP (see Appendix I). One-layer NN and PSVM were chosen due to their good performance and reduced time costs while MLP was selected because it is one of the most employed neural network architectures. Unlike the methods tested before (C4.5 and naive Bayes), these classifiers need a previous conversion of the symbolic attributes of the dataset into numerical labels. For this task, the Separability of Split Value (SSV) criterion based method (Grkabczewski & Jankowski, 2003) was chosen due to the fact that it is a successful tool for symbolic to real-valued feature mapping.

The results obtained with these methods following the process illustrated in Figure 7.2 (i.e., only the symbolic features selected by filters are converted) are shown in Table 7.6 and compared with the best results obtained by (Bolón-Canedo et al., 2009) with C4.5 and naive Bayes.

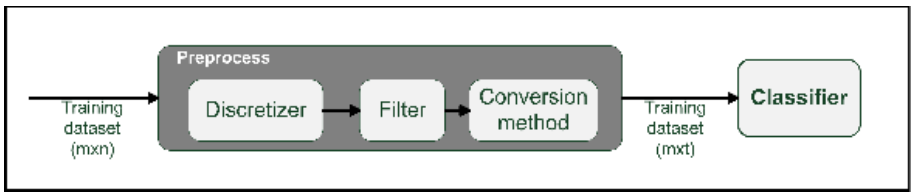


Figure 7.2: Preprocessing for the classifiers with symbolic conversion.

Table 7.6: Comparison of the results (in %) obtained in the binary case with several classifiers over the test set.

Method	Error	True Positives	False Positives	No. of Features
EMD+Cons+One-layer	7.78	90.52	0.77	6
EMD+Cons+PSVM	7.78	90.53	0.78	6
EMD+Cons+MLP	8.01	90.18	0.54	6
EMD+INT+C4.5	6.69	91.81	0.49	7
PKID+Cons+NB	7.99	90.18	0.42	6

As can be seen, the results obtained with the three first classifiers of Table 7.6 did not improve those results shown in the previous subsection with naive Bayes and, specially, with C4.5 (see Table 7.5). Hence, based on these results, it seems better to use classifiers that do not require symbolic conversion methods, that in fact might cause information loss.

7.2.2 Results on the multiple class case

The simplest way to classify a dataset with more than two classes is to use a multiple class classifier. However it is not a very extended choice, because not all the machine learning algorithms have this capacity. C4.5 and naive Bayes classifiers have turned to give good results in the binary case and they can deal with multiple classes. This approach, however, has the risk of focusing on the majority classes (see Table 7.1) and good results are not expected (Forman, 2003).

Therefore, a more used approach is the one based on multiple binary classifiers. This approach consists of employing class binarization techniques which reduce the multiple class problem into a series of binary problems which are solved individually.

Then, the resultant predictions are combined to obtain a final solution (Khoshgoftaar, Gao, & Ibrahim, 2005). There are various class binarization approaches proposed in the literature, and all of them can be summarized using a coding matrix $M_{b\times c}$, being b the number of classifiers and c the number of classes. A simple example for a problem with 4 classes is shown in Figure 7.3. White and black boxes are positive and negative examples, respectively, while gray boxes represent samples that must be ignored for the corresponding classifier.

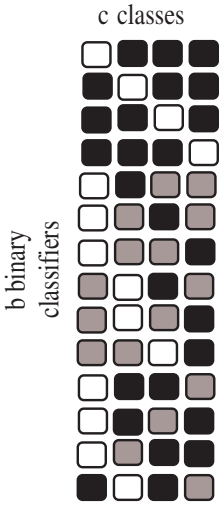


Figure 7.3: Code matrix for a four-class problem

Once each classifier is individually trained, the global performance must be checked using a test data set. Then, a test sample x is fed to each learning algorithm that finds a hypothesis for it. The vector of predictions of these classifiers on an instance x is denoted as $\mathbf{h}(\mathbf{x}) = (h_1(x), \ldots, h_b(x))$ and the y^{th} column of the matrix $M_{b\times c}$ is denoted by M_y. Given an instance x, the label y for which the column M_y is the *closest* to $\mathbf{h}(\mathbf{x})$ is predicted. In other words, it is predicted the class y that minimizes the $d(M(y), \mathbf{h}(\mathbf{x}))$ for some distance d. Several distance measures can be considered turning to different results.

Several class binarization techniques (Khoshgoftaar et al., 2005) have been proposed in the literature. This research, however, is focused on the two more popular ones, namely *One vs Rest* and *One vs One*.

- **One vs Rest:** This is the most popular class binarization technique, where one takes each class in turn and learns binary concepts that discriminate that class

from all other classes. This technique transforms a c-class problem into c two-class problems. These two-class problems are constructed by using the examples of class i as the positive examples and the examples of the rest of the classes as the negative examples. Notice that this technique corresponds to the first four rows of Figure 7.3.

- **One vs One:** This class binarization technique consists of learning one classifier for each pair of classes. The *One vs One* class binarization transforms a c-class problem into $\frac{c(c-1)}{2}$ two-class problems, one for each set of classes $\{i, j\}$. The binary classifier for a problem is trained with examples of its corresponding classes i, j, whereas examples of the rest of classes are ignored for this problem. This technique is represented by rows 5-10 in Figure 7.3.

A c-class problem is subdivided into b binary problems according to the matrix $M_{b\times c}$, then each problem is solved by using a different classifier that is trained individually. Given a new sample x, one classifier obtains a probability (p) of assigning that pattern to a given class and the rest ($1-p$) to the opposite class. However, in some cases, a given test sample may not be assigned to any of those classes, because the classifier was not trained with them (consider for example a pattern of class 3 or 4 in the fifth classifier (fifth row) in Figure 7.3). Therefore, there are three possible outputs: *positive (+1), negative (-1)* or *ignored (0)*. Then, to assign a pattern to the positive or negative classes, their corresponding probability must be higher than a determined threshold ($p > \sigma$), otherwise the considered output is *ignored*. Given a test pattern x, each individual classifier will obtain its corresponding output, $\mathbf{h}(\mathbf{x}) = (h_1(x), \ldots, h_b(x))$. However, a global result must be finally reached. There exist different techniques to obtain the proper output, two of them use a measure distance between the vector $\mathbf{h}(\mathbf{x})$ and the columns of matrix $M_{b\times c}$, M_y and they are subsequently illustrated. Finally, three different measures were designed *ad hoc* based on the probabilities.

- **Hamming decoding:** This technique (Allwein, Schapire, & Singer, 2001) counts up the number of positions in which the sign of the vector of predictions $\mathbf{h}(\mathbf{x})$ differs from the matrix column M_y.

- **Loss-based decoding:** Unlike Hamming decoding, this method (Allwein et al., 2001) takes into account the magnitude of the predictions which can often be an indication of a level of *confidence* as well as the relevant *loss* function L. The idea is to choose the label y that is most consistent with the predictions $h_s(x)$ in

the sense that, if instance x were labeled y, the total *loss* on instance (x, y) would be minimized over choices of $y \in \{1, \ldots, c\}$.

- **Accumulative sum:** Considering the learning methods used (NB and C4.5), for each sample, the binary classifier obtains a probability (p) for the winning class (positive or negative) while the probability for the remaining class is $1 - p$. Therefore, instead of calculating distances to determine the class from the b different results, the accumulative probability sum of each class is computed. Then, the desired output is the one with the highest value.

- **Accumulative sum with threshold:** This method is a modification of the previous one and takes under consideration the fact that test patterns include *ignored* classes, i.e., classes not used for the learning of the classifier. Then, this technique only computes those probabilities that are over an established threshold to guarantee that only clearly winning classes are computed.

- **Probability accumulative sum:** Previous techniques are more adequate for one-versus-one binarization because of the existence of *ignored* outputs. In the *One vs Rest* approach, a different technique was employed. Notice that, if all the classifiers work properly, each pattern of a given class will be recognized by the classifier of the class, while it will be assigned to class *rest* by the remaining classifiers. However, if one of the classifiers does not work well, a conflict appears, and the same pattern can be assigned to several classes, one for classifier. Again, the accumulate sum of probabilities was selected to determine the adequate class.

The aim of the experiments that will be presented in this section is to overcome the KDD winner score, that was measured according to the cost matrix that is shown in Table 7.7 (Elkan, 2000). The KDD winner obtained a score of 0.2331 and non-winning entries obtained an average cost per test example ranging from 0.2356 to 0.9414.

As was stated in Table 7.1, the classes are clearly unbalanced in both training and test set. Normal and DoS classes have enough samples, while the number of instances available for U2R and R2L is small and, moreover, it is different than in the test set. In fact, they may suffer from the data shift phenomenon (see Chapter 4, Section 4.2.4). Then, some classes are clearly difficult to learn, and for that reason it is also required to compute the detection for each class (true positives).

Table 7.8 displays the best results for the multiple class case, in which the best score is highlighted in bold face. Notice that the score is calculated according to

Table 7.7: Cost matrix used in KDD Cup 99 competition

		Prediction				
		Normal	Probe	DoS	U2R	R2L
Real	Normal	0	1	2	2	2
	Probe	1	0	2	2	2
	DoS	2	1	0	2	2
	U2R	3	2	2	0	2
	R2L	4	2	2	2	0

the competition cost matrix seen in Table 7.7 and the lower the score, the better the behavior. First row shows the result obtained by the KDD Winner, second row displays the best result obtained by a multiple class algorithm (Multi) and third row reports the result for the one vs rest technique (1vsR). Finally, the last four rows reveal the best results for the one vs one approach (1vs1) combined with the four different decoding techniques.

Table 7.8: Best test results obtained with the multiple class algorithms

Combination	Decoding technique	Score	No. of features
KDD winner	–	0.2331	41
EMD + INT + C4.5 (Multi)	–	0.2344	11
EMD + INT + C4.5 (1vsR)	Accumulative sum	0.2324	15
PKID + Cons + C4.5 (1vs1)	Sum	**0.2132**	13
PKID + Cons + C4.5 (1vs1)	Sum-with-threshold	0.2209	13
PKID + Cons + C4.5 (1vs1)	Loss-Based	0.2167	13
PKID + Cons + C4.5 (1vs1)	Hamming	0.2242	13

For the multiclass algorithm, several combinations of discretizators, filters and classifiers were tested. A total of 13 experiments were conducted, but none of them improved the KDD winner score. However, the best result achieved is close to the KDD winner result but using a significant reduction in the number of features (see Table 7.8). As for the results achieved with the one vs rest approach (third row of Table 7.8), one of the combinations improved the KDD winner score with an important reduction (37 %) of the total number of features.

The results for the one vs one approach shown in Table 7.8 were obtained employing the PKID discretizer, the consistency-based filter and the C4.5 classifier, combined with the four decoding techniques mentioned above. They manage to improve the KDD winner score using only 32 % of the total features. It is very remarkable the first result, achieved with the *Accumulative Sum* decoding technique, since the difference with the KDD winner score is very wide and exactly the same that the one existing between the winner and the tenth entry of the competition (Elkan, 2000).

7.2.2.1 Comparison with other authors

There exist several works in the literature (Fugate & Gattiker, 2003; Alonso-Betanzos, Sánchez-Maroño, Carballal-Fortes, Suárez-Romero, & Pérez-Sánchez, 2007; Mukkamala & Sung, 2002) that deal with the KDD Cup 99 problem, so in this subsection a comparative study will be carried out. In order to compare our results with those obtained by other authors, it is not possible to use the score employed in the prior comparisons with the KDD winner (according to the cost matrix seen in Table 7.7), due to the fact that the confusion matrix is not available for those results and only the detection percentage of attacks for each class is provided (see Table 7.9). Then, the same performance measure will be used to make fair comparisons although it is not possible to perform statistical tests in order to determine if there are significant differences.

Table 7.9 shows the detection percentage of each class. The six first rows correspond with results achieved in this work and are compared with those results obtained by the KDD winner and other authors. The best result for each class is emphasized in bold font.

The first six results were obtained with the proposed methodology. Comparing our results with those obtained by the KDD winner, our approach improves the detection in the minority classes: U2R, R2L and Probe, that are the more difficult classes to detect (see Table 7.1). Regarding the results obtained by other authors, it can be seen again that we achieve the best detections in classes R2L and Probe. Note that in any case, the number of features used by our proposed approach is considerably lower than those of other authors. This is very important because this reduction in the number of features causes an important decrement in the data processing and the data storage costs besides obtaining an interesting improvement in performance.

Table 7.9: Best results obtained over the test dataset. Comparison with other authors.

Combination	Normal	U2R	DoS	R2L	Probe
PKID + Cons + C4.5	97.08	25.00	96.08	8.12	73.62
EMD + INT + C4.5	96.36	19.30	94.07	**32.87**	79.48
EMD + CFS + C4.5 (1)	96.76	21.05	92.54	26.29	86.08
EMD + CFS + C4.5 (2)	96.56	32.89	93.35	5.32	78.18
PKID + Cons + NB (1)	96.63	24.12	90.20	29.98	89.94
PKID + Cons + NB (2)	96.13	11.40	95.12	13.85	**96.67**
KDD winner	**99.45**	13.16	97.12	8.40	83.32
5FNs_poly	92.45	8.33	96.77	29.43	85.96
5FNs_fourier	92.72	10.97	96.86	23.75	85.74
5FNs_exp	92.75	13.60	96.85	23.77	85.60
SVM Linear	91.83	21.93	97.38	16.55	81.76
SVM 2poly	91.79	1.75	97.41	14.74	86.44
SVM 3poly	91.67	2.63	97.62	9.35	88.45
SVM RBF	91.83	25.88	97.30	18.29	79.26
ANOVA ens.	91.67	53.94	97.64	8.51	87.52
Pocket 2cl.	91.80	29.82	97.40	14.77	85.84
Pocket mcl.	91.86	**54.38**	**97.65**	11.45	86.79

7.3 DNA microarray data

As mentioned in Chapter 4, microarray data classification is a serious challenge for machine learning researchers because of its high dimensionality and the small sample size. Typical values are around 10 000 gene expressions and a hundred or less tissue samples. Theoretically, having more genes should give more discriminating power. However, this fact can cause several problems, such as increasing computational complexity and cost, too many redundant or irrelevant genes and estimation degradation in the classification error. Having much higher number of attributes than instances causes difficulties for most of machine learning methods, since they cannot generalize adequately and therefore, they obtain very poor test performances. To deal with this problem, the need to reduce dimensionality was soon recognized and several works have used methods of feature (gene) selection (see Chapter 4).

Besides of the need for feature selection, this research introduce the necessity of a previous discretization of the data for two main reasons: the first one is to help the filtering process and the second one is related to the high number of genes with very

unbalanced values in microarray data. In fact, it will be shown that this preprocessing discretization step will improve the performance of the filter+classifier approaches employed. Also, and compared with other authors, better or similar performance results will be achieved by our combination method, that uses simpler classifiers and less computational resources than those of other authors.

In an attempt to discover features with unbalanced values, a study of ten microarray datasets was executed, in which unbalanced features were found for all of them. Figures 7.4-7.6 show some examples of unbalanced features and some values of those attributes can be seen in Table 7.10.

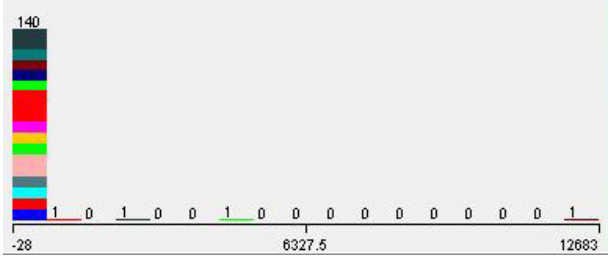


Figure 7.4: An unbalanced gene of GCM dataset

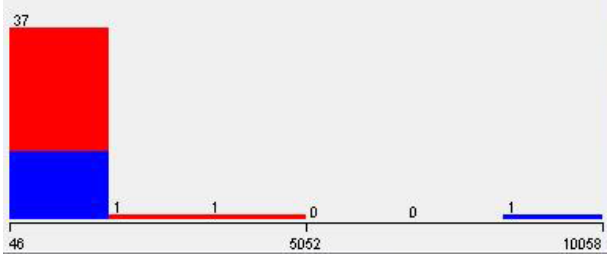


Figure 7.5: An unbalanced gene of CNS dataset

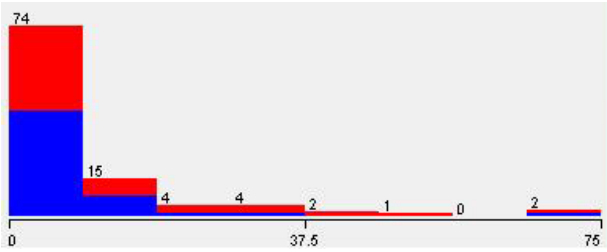


Figure 7.6: An unbalanced gene of Prostate dataset

The proposed method consists of applying a discretizer, a filter and a classifier (see Section 7.1). Two different discretizers (EMD and PKID, Section 7.1.1), three filters (CFS, consistency-based and INTERACT, Chapter 2) and three classifiers (C4.5, naive

Table 7.10: Unbalanced attributes of GMC, CNS and Prostate datasets

Dataset	Min.	Max.	Mean	StdDev	Distinct
GCM (at. 164)	-28	12 683	307.04	1141.79	122
CNS (at. 4503)	46	10 028	854.70	1673.27	39
Prostate (at. 52)	0	75	9.63	12.25	31

Bayes and k-NN, Appendix I) were tested, so as to be able to compare their results, over these ten well-known microarray datasets with unbalanced features.

A total of ten datasets were chosen for this study. Two of them are multiclass (Lymphoma and GCM) and their characteristics can be consulted in Appendix I, Table I.9. The class distribution of GCM dataset is balanced and the percentages in the training and the test sets are roughly maintained. However, for Lymphoma dataset, one of the nine classes corresponds with half of the samples and thus, there are some classes without representation in the training set or in the test set, making more difficult the adequate classification of the data. The properties of the eight remaining binary datasets (Leukemia, CNS, DLBCL, Colon, Prostate, Lung, Ovarian and Breast) are summarized in Appendix I, Table I.8.

As mentioned in Chapter 4, not all the datasets included in this study were divided to training and test sets. For the sake of comparison, the datasets with only training set (CNS, DLBCL, Colon, Lymphoma, Ovarian and Breast) were randomly divided using the common rule 2/3 for training and 1/3 for testing.

7.3.1 Experimental results

After demonstrating the adequacy of the proposed method over the KDD Cup 99 dataset, the aim of these experiments is to prove that it is also adequate for microarray datasets, which have a much larger number of input features and a considerably lower number of samples available. For this purpose, the results obtained with the proposed method will be compared with the performance achieved by the classifier without preprocessing.

Table 7.11: Best results for each binary data set.

Dataset	Method	Validation Accuracy	Test Accuracy	No. of genes
Leukemia	C4.5	84.21	91.18	7129
	PKID+Cons+C4.5	94.74	**94.12**	**2**
	NB	94.74	88.24	7129
	PKID+CFS+NB	100.00	94.12	18
	k-NN	89.47	70.59	7129
	EMD+Cons+k-NN	100.00	91.18	1
CNS	C4.5	55.00	60.00	7129
	EMD+INT+C4.5	82.50	65.00	47
	NB	65.00	60.00	7129
	PKID+INT+NB	90.00	**75.00**	**4**
	k-NN	50.00	55.00	7129
	PKID+INT+k-NN	85.00	65.00	4
DLBCL	C4.5	87.50	86.67	4026
	EMD+Cons+C4.5	96.88	86.67	2
	NB	84.38	93.33	4026
	EMD+INT+NB	100.00	**93.33**	**36**
	k-NN	71.88	73.33	4026
	EMD+INT+k-NN	96.88	66.67	36
Colon	C4.5	83.33	90.00	2000
	EMD+Cons+C4.5	97.62	85.00	3
	NB	54.14	70.00	2000
	EMD+Cons+NB	100.00	85.00	3
	k-NN	66.67	**95.00**	**2000**
	EMD+Cons+k-NN	97.62	85.00	3
Prostate	C4.5	85.29	26.47	12 600
	PKID+Cons+C4.5	88.24	**73.53**	**2**
	NB	63.73	26.47	12 600
	PKID+Cons+NB	85.29	**73.53**	**2**
	k-NN	84.31	52.94	12 600
	PKID+Cons+k-NN	88.24	**73.53**	**2**
Lung	C4.5	71.88	81.88	12 533
	EMD+Cons+C4.5	100.00	81.88	1
	NB	96.88	95.30	12 533
	EMD+Cons+NB	96.88	95.30	1
	k-NN	100.00	97.99	12 533
	PKID+INT+k-NN	100.00	**100.00**	**40**
Ovarian	C4.5	91.72	98.81	15 154
	EMD+Cons+C4.5	97.04	98.81	3
	NB	92.31	88.10	15 154
	EMD+Cons+NB	98.22	**100.00**	**3**
	k-NN	93.49	92.86	15 154
	PKID+CFS+k-NN	95.86	100.00	17
Breast	C4.5	51.28	73.68	24 481
	PKID+INT+C4.5	67.95	**78.95**	**3**
	NB	57.69	36.84	24 481
	EMD+Cons+NB	96.15	73.68	5
	k-NN	62.82	68.42	24 481
	EMD+Cons+k-NN	96.15	73.68	5

In Tables 7.11 and 7.12 one can see the results over ten microarray datasets obtained after comparing the performance achieved with a classifier (without preprocessing) and the best performance obtained by our proposed method grouped by dataset and classifier. The tables show the validation accuracy (achieved from a 10-fold cross-validation over the training dataset), the test accuracy (result of applying the model to the independent test dataset) and the number of genes required for each combination tested. Table 7.11 shows the results achieved for the binary datasets, while Table 7.12 is devoted to multiclass problems. In both tables, the best test accuracy obtained for each dataset is emphasized in bold font. When some methods achieve the same test accuracy, it is highlighted the one which uses the smaller number of features/genes.

Table 7.12: Best results for each multiclass data set.

Dataset	Method	Validation Accuracy	Test Accuracy	No. of genes
GCM	C4.5	50.00	52.17	16 063
	EMD+Cons+C4.5	61.81	41.30	9
	NB	67.36	52.17	16 063
	EMD+Cons+NB	68.06	**54.35**	**9**
	k-NN	58.33	45.65	16 063
	PKID+CFS+k-NN	86.11	52.17	1431
Lymphoma	C4.5	70.31	75.00	4026
	EMD+Cons+C4.5	85.94	59.38	3
	NB	65.63	68.75	4026
	EMD+INT+NB	98.44	81.25	160
	k-NN	95.31	**87.50**	**4026**
	PKID+CFS+k-NN	98.44	81.25	160

In most datasets, we can see an improvement in the accuracy over the test dataset with a drastic reduction in the number of features employed over all the 10 datasets studied. Colon and Leukemia are broadly studied, but not the remaining datasets, perhaps due to their difficulties at classification tasks (in Table I.8 we can see that Prostate and Lung have very different class distributions in training and test sets).

It is important to notice that while binary classification is heavily studied, only a small amount of work has been made on multiclass classification (Li et al., 2004). Multiclass classification problem is much more complicated than the binary one for the DNA microarray datasets. The difficulty lies in the fact that the data are of high dimensionality and the sample size is very small. Therefore, the classification accuracy appears to degrade very rapidly as the number of classes increases. In this

work there are two multiclass datasets: GCM (14 classes) and Lymphoma (9 classes), and their results can be seen in Table 7.12. This first attempt towards multiclass classification over microarray datasets has obtained satisfactory results. For GCM, one of our combinations (Table 7.12, EMD+Cons+NB) achieved the best test accuracy. On the other hand, Leukemia seems to suffer overfitting (it is important to remind that some of the classes have no representation in the training or test sets). However, for the naive Bayes classifier a improvement of 12% is achieved with our approach over the test set (see Table 7.12).

Table 7.13 shows a summary in order to see clearly the performance of the proposed method focused in each classifier. "Wins" indicates in how many datasets our proposed method is better than the original classifier, "Loses" symbolizes in how many datasets our proposed method is worse than the original classifier and "Draws" shows the number of times that the performances are the same.

Table 7.13: Summary of Table 7.11

Classifier	Wins	Loses	Draws
C4.5	4	3	3
NB	8	0	2
k-NN	7	3	0

The worse scores obtained by our method are the ones related with C4.5 classifier. This fact can be explained because of the embedded capacity of C4.5 to select features to construct its predictive model. Without a gene selection process, the classification trees built by C4.5 have in most of the cases, the same number of features than the filter provides. Anyway, our method overcomes or maintains the results in 7 of 10 datasets, in some cases with a drastic improvement (see Breast and Prostate in Table 7.11).

However, for naive Bayes and k-NN classifiers our results are clearly better than the ones accomplished by the original classifier. For NB there are some remarkable improvements (see Ovarian and Breast in Table 7.11) and in this case, the drastic reduction in the number of genes is real, since the method does not have a embedded capacity to select features. Something similar happens for k-NN classifier, emphasizing the results obtained over Leukemia and Prostate datasets (see Table 7.11), although they are hard to classify due to the unbalanced distribution of their classes.

Regarding the number of features employed, the reduction is very drastic for all the classifiers. This is a very interesting fact, because it can facilitate the identification of the underlying mechanisms relating gene expression to diseases. As the biologists have much less genes to consider, their problem could be greatly facilitated.

7.3.1.1 Comparison with other authors

In order to see the adequacy of the proposed method over DNA microarray datasets we compare our results with the results achieved by other authors in the literature. Nevertheless, it is impossible to do a equivalent comparison, because different methods were employed by the different authors to test the effectiveness of their techniques. Tables I.8 and I.9 in Appendix I indicate the number of samples and class distribution of the train and test set used in this research (such as they appear in their original references or, when only the training set was provided, using the common rule 2/3 for training and 1/3 for testing). In other works, new train and test datasets were generated by joining the original ones and randomly dividing them using 90%-10% proportion. Therefore, the training and test sets employed by other authors are notoriously different from our sets; apart from the number of samples considered for the training (for example, Lung data set has a 18%-82% train-test division), the class distribution problem may be partially settled by the union of both sets (see Prostate data set in Table I.8). Obviously, it is much more difficult to achieve good performance results over the original test set. This fact can explain the low values of test accuracy obtained by the proposed method in some data sets. On the contrary, showing the validation accuracy is advantageous for our method because our validation samples have been seen by the discretization and filtering steps, although not by the classifier. For these reasons, both values (validation and test) are to be displayed in Tables 7.14 and 7.15. Table 7.14 shows a comparative study among the test results obtained by Ruiz et al. (2006) and the validation and test results obtained by our proposed approach. They performed a 10-fold cross validation with the datasets Colon, Leukemia, Lymphoma and GCM to test the efficacy of their wrapper method, called BIRS (Best Incremental Ranked Subset).

For each dataset, the results achieved by all three classifiers are shown. For each classifier, a comparison between the score obtained by BIRS (first row) and the best accomplished by our proposed method (second row) can be seen. For both binary class data sets, our performance results improve or maintain those obtained by Ruiz et al. (2006), using a smaller or similar number of genes, except in the Leukemia data set

Table 7.14: Comparison with Ruiz et al.

Dataset	Method	Accuracy		No.genes
		Val.	Test	
Leukemia	BIRS+C4.5	88.57		1.20
	PKID+Cons+C4.5	94.74	94.12	2.00
	BIRS+NB	93.04		2.50
	PKID+CFS+NB	100.00	94.12	18.00
	BIRS+k-NN	93.04		3.30
	EMD+Cons+k-NN	100.00	91.18	1.00
Colon	BIRS+C4.5	83.81		2.90
	EMD+Cons+C4.5	97.62	85.00	3.00
	BIRS+NB	85.48		3.50
	EMD+Cons+NB	100.00	85.00	3.00
	BIRS+k-NN	79.76		6.30
	EMD+Cons+k-NN	97.62	85.00	3.00
GCM	BIRS+C4.5	46.84		9.80
	EMD+Cons+C4.5	61.81	41.30	9.00
	BIRS+NB	67.37		44.00
	EMD+Cons+NB	68.06	54.35	9.00
	BIRS+k-NN	58.95		37.00
	PKID+CFS+k-NN	86.81	52.17	1431.00
Lymphoma	BIRS+C4.5	80.00		8.80
	EMD+Cons+C4.5	85.94	59.38	3.00
	BIRS+NB	82.14		10.30
	EMD+INT+NB	98.44	81.25	160.00
	BIRS+k-NN	85.56		16.40
	PKID+CFS+k-NN	98.44	81.25	160.00

using PKID+CFS. Multiple class data sets are more difficult to deal with and, although the validation results were promising, the accuracy decreases in test set. It is important to note that the GCM data set has 14 classes and only 146 samples were used to train our model (171 by BIRS method). Lymphoma data set has very different representation of classes in the original train and test sets. It has a class with no representation in the train set, besides two classes with 1 and 2 samples, respectively. However, these three classes have 9 samples in the test set. So, considering the original data distribution, there are 27 samples that are biased towards being erroneously classified.

These results represent an important achievement, because wrappers tend to obtain better performances than filters (at the expense of a higher computational cost). Nevertheless, our proposed method achieves similar or even better scores for binary class data sets, demanding less computational resources. The multiple class problems present heterogeneous results. However, it must be borne in mind that the validation and test results of our method are not direly comparable with those of Ruiz et al., being the first results (validation) advantageous for our method, while the second (test) results are clearly advantageous for their method, due to the different divisions made in both studies.

Table 7.15 shows a comparison with the results obtained by Alonso-González, Moro, Prieto, and Simón (2010), who executed 15 repetitions where 90% randomly selected instances were used for training and the remaining 10% for testing. Again, their results will be compared with our validation and test results. They combined feature selection techniques with several classifiers and chose the combination with best accuracy for several binary data sets. For each data set, the first row corresponds with the best result achieved by Alonso-González et al. (2010) and the second row corresponds with the best result obtained with the method proposed in this work. For the sake of brevity, only the best combination for each data set is shown.

The method proposed herein reduces considerably the number of genes required, obtaining the best accuracy for some data sets, although decreasing it for others. In those cases in which the accuracy is lower (see Table 7.15, Leukemia data set) the reduction in the number of genes is considerable (from 90 to 2), and also the number of samples used for train and test is very different for both methods (52% - 48% vs. 90% - 10%, 38 samples used for training with our method vs. 65 samples used by Alonso-González et al. (2010)). This data distribution together with the high number of features has an influence on the performance results. The most noteworthy improvement is the one achieved with Breast data set, that surpasses the one obtained by Alonso-González

et al. (2010) in more than 8% and moreover reducing the genes needed from 50 to 3. Therefore, our method presents similar accuracy results that the one proposed by Alonso-González et al. (2010), which does not include discretization as a previous step.

Table 7.15: Comparison with Alonso-González et al.

Dataset	Method	Accuracy		No.genes
		Val.	Test	
Leukemia	SVM-RFE+3-NN	100.00		90
	PKID+Cons+C4.5	94.74	94.12	2
Breast	ReliefF+SVM	70.52		50
	PKID+INT+C4.5	67.95	78.95	3
CNS	SVM-RFE+SVM	75.49		100
	PKID+INT+NB	90.00	75.00	4
Colon	Random+SVM	88.41		60
	EMD+Cons+NB	100.00	85.00	3
DLBCL	ReliefF+NB	98.67		120
	EMD+INT+NB	100.00	93.33	36
Lung	SVM-RFE+NB	99.63		100
	PKID+INT+k-NN	100	100	40
Ovarian	SVM-RFE+3-NN	100		10
	EMD+Cons+NB	98.22	100	3
Prostate	SVM-RFE+SVM	95.39		180
	PKID+Cons+k-NN	88.24	73.53	2

7.3.1.2 Results obtained by a classifier based on information theoretic learning

Local modeling classifiers have not been prodigally applied to microarray-based datasets. For this reason, the study presented in this section is devoted to the application of the FVQIT (Frontier Vector Quantization using Information Theory) local modeling classifier (Martínez-Rego, Fontenla-Romero, Porto-Díaz, & Alonso-Betanzos, 2009) combined with discretization and feature selection methods to several DNA microarray datasets.

The classification method used is a supervised algorithm for binary classification. It is based on local modeling and information theoretic learning (Martínez-Rego et al., 2009). The algorithm works in two stages. First, a set of nodes or PEs (Processing Elements) is set on the frontier between classes in such a way that each of these nodes defines a local model. Then, a one-layer neural network is trained in each local model. Therefore, the final system consists of a set of local experts, each of which is trained to solve a subproblem of the original. In this manner, the method improves its generalization ability benefiting from a finer adaptation to the characteristics of the training set.

The proposed method has been tested extensively over twelve binary DNA microarray data sets (Brain, Breast, CNS, Colon, DLBCL, GLI, Leukemia, Lung, Myeloma, Ovarian, Prostate and SMK). The number of features and samples for each data set can be consulted in Appendix I, Table I.8. Some of these data sets are originally divided in training and test, but some are not. So first, and similar to what has been previously done in this section, Brain, Breast, CNS, Colon, DLBCL, GLI, Ovarian, Myeloma and SMK data sets have been divided using 2/3 for training and 1/3 for testing. A 10-fold cross-validation is performed upon the training sets, in order to estimate the validation error to choose a good configuration of parameters for the classification algorithms.

In the following experiments the FVQIT method is compared with other classifiers with the objective of finding out which classifier obtains the best performance. Thus, six well-known machine learning classifiers which can be consulted in Appendix I – naive Bayes (NB), k-Nearest Neighbor (k-NN), C4.5, Support Vector Machines (SVM), Multi-Layer Perceptron (MLP) and AdaBoost (AB) – are applied over the filtered datasets. Two discretizers will be used – Entropy Minimization Discretization (EMD) and Proportional k-Interval Discretization (PKID) – in combination with three filters – Correlation-based Feature Selection (CFS), consistency-based and INTERACT.

Table 7.16 shows the estimated test errors (TE in the table) as well as the sensitivity (Se) and specificity (Sp) rates – in percentage – and the number of features (NF) selected by each method tested (see Appendix I, Section I.6). Also, the best result obtained for each dataset is emphasized in bold font. Despite having executed all six combinations of discretizer + filter, only the best result for each classifier in each data set is shown.

As can be seen in Table 7.16, the FVQIT method obtains good performance on all data sets, with an adequate number of selected features. Specially remarkable are the results obtained on the datasets DLBCL and Leukemia, where the FVQIT classifier is

Table 7.16: Best results on microarray datasets.

Dataset		FVQIT	SVM	NB	MLP	AB	k-NN	C4.5
Brain	TE	**0.00**	14.29	14.29	28.57	**0.00**	**0.00**	**0.00**
	Se	100.00	100.00	100.00	0.00	100.00	100.00	100.00
	Sp	100.00	83.33	83.33	71.43	100.00	100.00	100.00
	NF	1	45	45	1	1	1	45
Breast	TE	**21.05**	**21.05**	26.32	**21.05**	36.84	26.32	**21.05**
	Se	75.00	83.33	83.30	83.33	85.71	83.30	66.70
	Sp	85.71	71.43	57.10	71.43	50.00	57.10	100
	NF	17	119	5	17	17	5	3
CNS	TE	**25.00**	35.00	**25.00**	35.00	35.00	35.00	35.00
	Se	69.20	71.43	69.20	68.75	68.75	69.20	76.90
	Sp	85.70	50	85.70	50.00	50.00	57.10	42.90
	NF	4	60	4	60	60	4	47
Colon	TE	**10.00**	**10.00**	15.00	40.00	15.00	15.00	15.00
	Se	80.00	80.00	87.50	50.00	77.78	87.50	87.50
	Sp	100.00	100.00	83.30	61.11	90.91	83.30	83.30
	NF	12	12	3	12	3	3	3
DLBCL	TE	**0.00**	6.67	6.67	6.67	13.33	6.67	13.33
	Se	100.00	100.00	85.70	100.00	100.00	85.70	85.70
	Sp	100.00	88.89	100.00	88.89	80.00	100.00	87.50
	NF	36	36	36	47	47	36	2
GLI	TE	**10.71**	14.29	**10.71**	17.86	17.86	14.29	21.43
	Se	85.71	85.00	85.71	78.26	80.95	81.82	75.00
	Sp	100.00	87.50	100.00	100.00	85.71	100.00	100.00
	NF	113	23	23	23	3	122	3
Leukemia	TE	**0.00**	2.94	5.88	5.88	8.82	8.82	5.88
	Se	100.00	100.00	100.00	92.86	86.67	92.90	92.86
	Sp	100.00	95.24	90.00	95.00	94.74	90.00	95.00
	NF	2	18	18	2	1	1	2
Lung	TE	0.67	1.34	4.70	0.67	18.12	**0.00**	18.12
	Se	100.00	99.26	94.80	99.26	96.52	100.00	82.80
	Sp	93.75	93.33	100.00	100.00	32.35	100.00	73.30
	NF	40	40	1	40	1	40	1
Myeloma	TE	21.05	21.05	21.05	21.05	24.56	29.82	**19.3**
	Se	84.00	81.48	81.48	80.36	80.77	82.20	80.70
	Sp	42.86	33.33	33.33	0.00	20.00	25.00	0.00
	NF	2	40	2	2	7	2	2
Ovarian	TE	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	1.19
	Se	100.00	100.00	100.00	100.00	100.00	100.00	98.10
	Sp	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	NF	3	3	3	37	17	3	
Prostate	TE	**20.59**	73.53	26.47	23.53	23.53	26.47	26.47
	Se	56.25	26.47	0.00	100.00	100.00	0.00	0.00
	Sp	100.00	0.00	100.00	75.76	75.76	100.00	100.00
	NF	64	3	2	3	11	2	2
SMK	TE	25.81	33.87	40.32	32.26	**24.19**	33.87	33.87
	Se	78.79	71.88	67.85	89.47	79.41	75.00	68.42
	Sp	68.97	60.00	52.94	58.14	71.43	58.82	62.50
	NF	21	3	3	21	93	21	3

Table 7.17: Ranking of test errors for each method in the comparative study.

Dataset	FVQIT	SVM	MLP	NB	k-NN	C4.5	AB
Brain	**1st**	5th	7th	5th	**1st**	**1st**	**1st**
Breast	**1st**	**1st**	**1st**	5th	5th	**1st**	7th
CNS	**1st**	3rd	3rd	**1st**	3rd	3rd	3rd
Colon	**1st**	**1st**	7th	3rd	3rd	3rd	3rd
DLBCL	**1st**	2nd	2nd	2nd	2nd	6th	6th
GLI	**1st**	3rd	5th	**1st**	3rd	7th	5th
Leukemia	**1st**	2nd	3rd	3rd	6th	3rd	6th
Lung	2nd	4th	2nd	5th	**1st**	6th	6th
Myeloma	2nd	2nd	2nd	2nd	7th	**1st**	6th
Ovarian	**1st**	**1st**	**1st**	**1st**	**1st**	7th	**1st**
Prostate	**1st**	7th	2nd	4th	4th	4th	2nd
SMK	2nd	4th	3rd	7th	4th	4th	**1st**
Average	**1.25**	2.92	3.17	3.25	3.33	3.83	3.92

the only method able to achieve 0% of test error. The result obtained on the Prostate data set is also important. Its test set is unbalanced (26% of one class and 74% of the other). C4.5, naive Bayes and k-NN are assigning all the samples to the majority class and SVM is assigning all the samples to the minority class, whereas FVQIT is able to do something different and better, which results in a lower test error.

In Tables 7.17, 7.18 and 7.19 a ranking of the test errors, sensitivities and specificities, respectively, is shown. The ranking assigns a position between 1 and 7 to each method in each data set, taking into account the ties among them.

In Table 7.17 the proposed method obtains the best results for all data sets except for three of them, where it gets the second best results. In average, the proposed method is clearly preferable above the other methods studied. Tables 7.18 and 7.19 show that our method is the most specific (it correctly identifies most of the negatives) and the most sensitive (it correctly identifies most of the positives). Therefore, in light of the above, we can conclude that the FVQIT classifier is suitable to be combined with discretizers and filters to deal with problems with a much higher number of features than instances, such as DNA microarray gene-expression problems.

Table 7.18: Ranking of sensitivity rates for each method in the comparative study.

Dataset	FVQIT	SVM	k-NN	NB	AB	MLP	C4.5
Brain	**1st**	**1st**	**1st**	**1st**	**1st**	7th	**1st**
Breast	6th	2nd	2nd	2nd	**1st**	2nd	7th
CNS	3rd	2nd	3rd	3rd	6th	6th	**1st**
Colon	4th	4th	**1st**	**1st**	6th	7th	**1st**
DLBCL	**1st**	**1st**	5th	5th	**1st**	**1st**	5th
GLI	**1st**	3rd	4th	**1st**	5th	6th	7th
Leukemia	**1st**	**1st**	**1st**	**1st**	7th	5th	5th
Lung	**1st**	3rd	6th	6th	5th	3rd	7th
Myeloma	**1st**	3rd	3rd	3rd	5th	7th	6th
Ovarian	**1st**	**1st**	**1st**	**1st**	**1st**	**1st**	7th
Prostate	3rd	4th	5th	5th	**1st**	**1st**	5th
SMK	3rd	5th	4th	7th	2nd	**1st**	6th
Average	**2.17**	2.50	2.75	3.00	3.42	3.92	4.83

Table 7.19: Ranking of specificity rates for each method in the comparative study.

Dataset	FVQIT	NB	k-NN	C4.5	SVM	MLP	AB
Brain	**1st**	5th	**1st**	**1st**	5th	7th	**1st**
Breast	2nd	5th	5th	**1st**	3rd	3rd	7th
CNS	**1st**	**1st**	3rd	7th	4th	4th	4th
Colon	**1st**	4th	4th	4th	**1st**	7th	3rd
DLBCL	**1st**	**1st**	**1st**	6th	4th	4th	7th
GLI	**1st**	**1st**	**1st**	**1st**	6th	**1st**	7th
Leukemia	**1st**	6th	6th	3rd	2nd	3rd	5th
Lung	4th	**1st**	**1st**	6th	5th	**1st**	7th
Myeloma	**1st**	2nd	4th	6th	2nd	6th	5th
Ovarian	**1st**	**1st**	7th	**1st**	**1st**	**1st**	**1st**
Prostate	**1st**	**1st**	**1st**	**1st**	7th	5th	5th
SMK	2nd	7th	5th	3rd	4th	6th	**1st**
Average	**1.42**	2.92	3.25	3.33	3.67	4.00	4.42

7.4 Multiple class datasets

In the previous sections, the results obtained with the proposed combination method have been presented, both over binary and multiclass datasets. However, while binary classification problems are studied intensively in general, only very few works deal with multiclass classification. In order to broaden the scope of this research and present more reliable conclusions, a exhaustive study over 21 different multiclass datasets will be carried out.

The direct multiple approach and the two multiple binary approaches, with and without feature selection, and with the different methods of discretizers, filters and classifiers are tested. A suite of 21 different datasets were selected (see Appendix I, Table I.4), attempting to include several aspects such as different number of classes, different number of samples, different ratios features/samples, unbalanced data sets, etc. Specifically, there are 4 data sets with clearly unbalanced classes: Glass, Connect-4, Dermatology and Thyroid (see details in Appendix I, Section I.2). Since the Glass dataset has 3 out of 6 classes with very reduced number of samples (lower than 20), an oversampling technique has been applied to it to ensure an adequate learning of the classifiers. Oversampling techniques, as explained in Chapter 4, Section 4.2.2, creates a superset of the original dataset by replicating some instances or creating new instances from existing ones.

A 10-fold cross-validation was used to obtain results in percentage of correct classification and in features selected by the methods that use filters. For each data set, in order to check if the several methods used exhibit statistically significant differences in performance, statistical tests were applied (see Appendix I, Section I.4). The prefix *MFeat* employed in some data sets of the tables of this section means *Multi-feature*, whereas the prefix MLL in *Leukemia* data set denotes the type of leukemia being tackled. Both prefixes will be ignored in the rest of this study.

The methodology for the experiments is the one explained in Section 7.1. Four different discretizers (EWD, EFD, EMD and PKID, Section 7.1.1), two filters (CFS and consistency-based filter, Chapter 2) and two classifiers (C4.5 and naive Bayes, Appendix I) were tested. These classifieres were selected because both of them can be used for the direct multiclass approach. Besides, their use of computer resources is affordable, an important factor in our study due to the high dimensionality of some of the data sets employed.

7.4.1 Experimental results

For each data set, 16 different combinations with feature selection and 8 without filtering were used for each approach considered. Then, there are 24 performance results for both multiclass and *One vs Rest* approach and, moreover, 96 (24×4 union techniques, see Section 7.2.2) *One vs One* results. Trying to show all results becomes intractable, so the best results for each scheme with and without feature selection are shown. It turns to 12 different results for each data set. As an example, in Figure 7.7 the results obtained for the *Thyroid* data set are shown. As it can be seen, the best performance with the smallest set of features is obtained by the *One vs One* approach using the sum method to make the union of the multiple binary classifiers and the combination EM+Cons+C4.5. The precision obtained was 99.52 ± 0.34 using 5 features (the number of attributes used by this approach is 23.81% of the total).

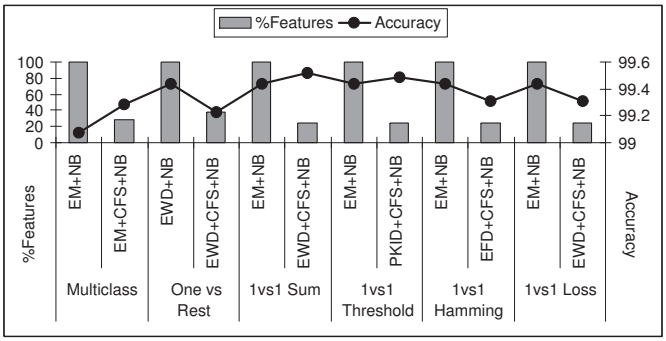


Figure 7.7: Best results obtained for the *Thyroid* data set. Percentage of features selected is represented at the left y axis, while on the right side, the accuracy obtained by the approach is drawn. On the x axis, the best results for each approach without and with feature selection are displayed.

In any case, Figure 7.7 shows that all approaches in this data set obtained good results when using feature selection. Both multiclass and *One vs One* approaches (using sum and sum with threshold as union techniques) achieved better accuracy values using feature selection than not using it. On the other hand, two of the *One vs One* versions (using Hamming and loss-based) and the *One vs Rest* approach obtained lower values for the accuracy by using feature selection. This latter case is the one with the lowest accuracy value when feature selection is applied (99.23 ± 0.36), a slightly lower value than using the same approach without feature selection (99.44 ± 0.34). However, the difference is not statistically significant and the reduction in the number of features used is important (21 vs 8).

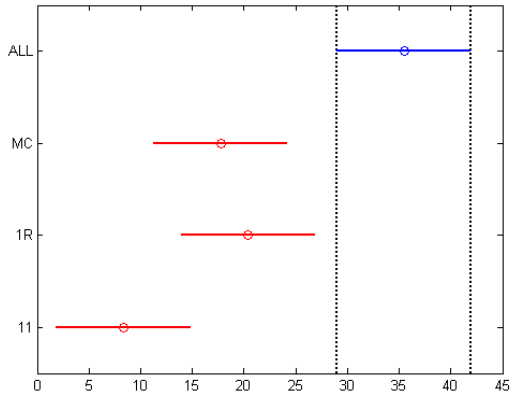


Figure 7.8: Multiple comparison results using all features (ALL) and the features selected by the multiple classes (MC), the *One vs Rest*(1R) and *One vs One* (11) approaches for the *Thyroid* data set

Figure 7.8 shows the result of applying a multiple comparison function (0.05 significance level) between the number of features selected by the approaches *multiclass*, *One vs Rest* and *One vs One Sum* (for the sake of completeness also the set containing all features was included) for the *Thyroid* data set. The graph displays each group mean represented by a symbol and an interval around the symbol. Two means are significantly different if their intervals are disjoint, and are not significantly different if their intervals overlap. Therefore, it can be seen that there is a significant statistical difference among the approaches with and without feature selection. Notice that these differences are not so notorious as bars in figure 7.7 may suggest because X-axis denotes average group ranks. Ranks are found by ordering the data from smallest to largest across all groups, and taking the numeric index of this ordering. The rank for a tied observation is equal to the average rank of all observations tied with it. So, in figure 7.8, 40 different values were ranked (10 values, one per fold, for each one of the 4 approaches considered). Clearly, there are 10 tied values at the last positions of this rank, one per fold of the approach with all features. On the other hand, the feature subset returned by the *One vs One* approach is most of the times at the top.

The important reduction in the number of necessary attributes achieved by the feature selection methods make their use worthwhile, specially in those cases in which the elimination of features can contribute to better explanation of clinical situations, such as it is the case in some data sets used in this work (i.e., *Leukemia*, *Thyroid*,...). So, the results of the *Leukemia* dataset constitute another interesting example, since it has a much higher number of features (12582) than samples (57). The results obtained

for the Leukemia data set can be seen in Figure 7.9. Again, better results are obtained by the feature selection versions of the methods, and the best is that of the *One vs Rest* approach. Although accuracy is the same for multiclass with feature selection, and *One vs Rest* with and without feature selection, the drastic reduction in the number of features needed (108 out of 12582) makes it worthwhile to use EWD+CFS+NB.

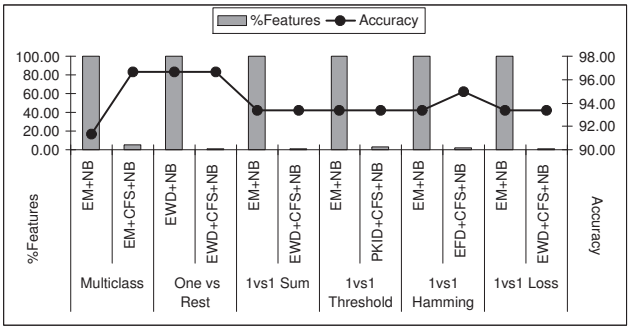


Figure 7.9: Best results obtained for the *Leukemia* data set. Percentage of features selected is represented at the left y axis, while on the right side, the accuracy obtained by the approach is drawn. On the x axis, the best results for each approach without and with feature selection are displayed.

7.4.1.1 Analysis of multiclass versus multiple binary approaches

In order to be able to establish a general comparative picture between all methods and both alternatives, in Table 7.20 the results obtained for all 21 data sets by the Multiclass, *One vs Rest* and *One vs One* approaches with (FS) and without (without FS) feature selection are shown. For the latter approach, only the results achieved by the sum as union method are shown, because it is the one obtaining the best results in average. A column labeled *Rk* has been added with the idea of detecting which approximation is the best in average. To do this, the 12 different approaches are listed in order of percentage obtained for each data set. Subsequently, the average ranking position is computed for each of the 12 approaches, so as to compare the different methods. This value is shown in the last row of the table. Also, the average in accuracy is displayed for each method in the previous row. It is necessary to note that the ranks are computed over all the 12 approaches, although only the best 6 of them are shown in the table. From results in Table 7.20 it can be concluded that, although the multiclass approach appears to behave better in number of features selected and with similar accuracy than the alternative multiple binary classifiers, the approach with the best average results, in both ranking and accuracy, is *One vs One* using sum and

feature selection. The ranking value of the latter, 3.9, is clearly separated from the rest, that obtain values higher than 5. The same behavior can be observed for the accuracy, in which the best value is obtained again by *One vs One* using sum and feature selection, although in this case the difference with the other methods is smaller. Notice that the scheme *One vs One* was not carried out for the *Vowel* data set, because its 10 different classes will imply the training of 45 classifiers in this approximation. A deeper analysis the properties of the datasets used (see Appendix I) suggests that the *One vs One* approach clearly surpasses the others when there exists a high ratio number of samples/number of features and enough samples per class. In general, the *One vs Rest* approximation exhibits a poor behavior, even when using feature selection, but surprisingly it achieves the best results when leading with the *Leukemia* data set that has a very low ratio number of samples/number of features. Besides, the results obtained by this approach worsen as the number of classes increases. This is due to the consequent unbalanced division into positive and negative classes for each of its classifiers.

The good results achieved by the multiclass approach were unexpected and a further analysis was done in order to get more insight. It is important to remember that different combinations of discretizer, filter and classifier were tested for each approach and the best combination was selected for each one. An example showing all the combinations for the *Factor* data set is depicted in Table 7.21. The first block of rows of this table shows the results achieved when using the C4.5 classifier, while the last block is devoted to the naive Bayes classifier. The best accuracy obtained for each combination is emphasized in bold font. The last row indicates the average accuracy achieved by each approach. It can be seen that the best accuracy is obtained by the combination of EMD + CFS + C4.5 using the multiclass approach. However, the last row denotes that the *One vs One* scheme achieves the best result in average, and moreover, this scheme obtains the best accuracy in 12 of 16 combinations, whereas the multiclass approach only gets the best values in 3 of them. Table 7.22 shows similar results for *Dermatology* and *Karhounen* data sets, but summarized. For each data set, the first row shows the values for each approach when the best accuracy is achieved (EFD+CFS+NB combination for *Dermatoloy* data set and EWD+CFS+NB combination for *Karhounen* data set). The second row indicates the mean average. Again, the multiclass approach gets the best performance result in a combination, however it is clearly surpassed by the *One vs One* scheme when focusing on averages.

Table 7.20: Best results for each data set.

	Multiclass					1VsRest					1vs1 Sum				
	Without FS		FS			Without FS		FS			Without FS		FS		
	Acc	Rk	Acc	Rk	% Feat	Acc	Rk	Acc	Rk	% Feat	Acc	Rk	Acc	Rk	% Feat
Iris	95.99±3.44	6	97.99±3.22	1	50.00	96.00±6.44	3	96.65±3.51	2	75.00	94.00±7.34	9	96.00±4.66	3	50.00
Vehicle	69.73±6.02	8	71.62±4.33	2	38.89	72.11±4.83	1	67.60±6.12	12	100	71.27±4.70	3	71.05±6.23	4	100
Wine	98.88±2.34	2	98.33±2.68	3	84.62	98.3±2.74	6	97.71±4.83	10	100	97.77±3.88	7	97.19±2.96	11	53.85
Waveform	80.98±1.72	10	80.86±1.82	11	71.42	79.78±1.78	12	82.36±1.76	6	95.23	81.02±1.72	8	82.64±1.50	4	90.47
Segmentation	91.38±1.99	12	92.42±2.07	9	63.15	92.12±1.96	11	92.42±2.25	9	94.73	94.31±1.61	5	94.37±1.54	4	84.21
Glass	73.83±8.56	9	70.15±9.06	11	77.77	72.01±8.20	10	68.83±8.51	12	100	85.81±4.58	3	85.34±8.51	4	100
Connect 4	80.94±0.74	2	81.16±0.51	1	83.33	80.61±0.50	3	80.61±0.51	3	97.62	80.38±0.56	5	80.26±0.55	8	97.62
Dermat.	98.35±2.31	4	98.91±1.89	1	67.64	97.79±2.87	11	97.00±2.36	12	76.47	98.07±2.28	8	98.34±1.93	7	94.11
Vowel	74.04±5.15	2	74.64±3.67	1	84.61	54.04±5.37	4	64.44±4.82	3	84.61	- ± -	0	- ± -	0	-
KDDSC	96.99±2.19	9	96.16±2.83	10	91.66	95.00±1.76	11	82.33±19.11	12	96.66	97.00±2.19	6	98.33±1.11	1	95.00
Splice	95.74±0.94	9	96.08±1.57	6	55.74	96.14±1.00	3	95.89±1.63	8	55.73	96.21±0.82	1	96.14±1.12	3	63.93
Thyroid	99.28±0.60	10	99.44±0.56	3	28.57	99.23±0.34	11	99.44±0.36	3	38.09	99.52±0.44	1	99.44±0.34	3	23.81
Optdigits	92.67±1.05	8	92.62±0.69	10	57.81	92.62±1.91	10	91.68±1.55	12	85.93	93.83±1.94	5	94.19±1.04	3	84.37
Pendigits	89.61±0.93	11	89.32±0.93	12	62.50	93.50±1.21	8	93.40±1.10	9	100	95.78±0.65	1	94.15±1.40	6	100
Landsat	85.27±1.40	8	84.75±1.45	12	88.88	84.82±2.14	11	84.98±1.57	9	100	86.89±1.21	3	87.10±1.36	1	100
Karhounen	92.50±1.90	4	93.10±1.64	1	92.18	92.50±1.56	4	90.95±1.78	8	100	89.75±2.58	10	92.50±2.27	4	100
Factor	93.30±1.70	6	95.90±1.32	1	49.53	88.70±2.2	8	92.90±2.52	7	84.72	88.00±2.51	10	95.65±1.85	2	99.07
Fourier	77.40±2.75	9	78.60±1.76	3	69.73	77.10±2.54	10	76.25±2.40	12	98.68	78.40±2.36	4	78.90±3.71	1	88.15
Pixel	93.50±1.24	4	93.30±1.18	5	57.50	91.70±1.33	11	91.15±1.39	12	90.00	92.95±1.42	7	94.15±1.05	2	97.91
Zernike	72.05±1.97	11	72.35±3.68	9	68.08	71.65±2.40	12	72.15±3.44	10	95.74	75.20±2.62	1	74.90±4.21	2	100
Leukemia	91.33±12.09	12	96.67±7.03	1	4.79	96.67±7.02	1	96.67±7.03	1	0.86	93.33±8.61	5	93.33±11.65	5	0.70
Accuracy	87.80		88.30			86.78		86.45			89.47		**90.20**		
Ranking		7.43		5.48			7.67		8.19			5.10		**3.90**	

Table 7.21: Results in accuracy obtained for *Factor* data set. Average accuracies are shown in last row of the table.

C4.5 Classifier			
	Multiclass	1vsRest	1vs1 Sum
EWD+CFS	**95.00 ± 1.81**	92.150 ± 2.30	94.65 ± 2.07
EFD+CFS	**94.65 ± 1.47**	91.50± 1.83	94.65 ± 1.55
PKID+CFS	92.50 ± 2.12	90.85 ± 1.93	**94.25 ± 1.21**
EMD+CFS	**95.90 ± 1.33**	92.90 ± 2.53	95.65 ± 1.86
EWD+CBF	84.90 ± 3.00	90.45± 2.73	**92.35 ± 1.29**
EFD+CBF	82.10 ± 1.90	88.65 ± 1.68	**90.90 ± 1.88**
PKID+CBF	69.75 ± 3.84	84.35 ± 1.89	**90.95 ± 2.78**
EMD+CBF	84.90 ± 3.34	**92.55 ± 1.74**	92.15 ± 1.97
NB Classifier			
	Multiclass	OnevsRest	1vs1 Sum
EWD+CFS	79.20 ± 1.89	84.60 ± 2.85	**89.75 ± 2.53**
EFD+CFS	74.10 ± 2.53	85.45 ± 1.76	**88.95 ± 2.06**
PKID+CFS	52.75 ± 3.07	71.00 ± 4.60	**88.05 ± 2.31**
EMD+CFS	80.65 ± 3.07	87.75 ± 3.16	**88.95 ± 2.60**
EWD+CBF	78.60 ± 3.70	85.60 ± 2.39	**89.20 ± 1.89**
EFD+CBF	76.05 ± 3.13	85.00 ± 2.20	**88.75 ± 2.23**
PKID+CBF	53.95 ± 2.99	71.50 ± 2.59	**89.40 ± 2.13**
EMD+CBF	80.80 ± 3.15	87.30 ± 2.39	**89.55± 2.24**
Average	79.74±2.65	86.35±2.41	91.13±2.04

Table 7.22: Best and average accuracy obtained for multiclass and both multiple binary classes approaches for the *Dermatology* and *Karhounen* datasets.

Dataset	Multiclass	1vsRest	1vs1 Sum
Dermat.	**98.91 ± 1.90**	96.70 ± 3.61	98.08 ± 3.43
	94.34 ± 4.06	94.66 ± 4.16	**96.62 ± 3.52**
Karhounen	**93.10 ± 1.64**	89.60 ± 2.20	91.60 ± 1.43
	71.72 ± 3.16	73.73 ± 2.85	**87.12 ± 2.33**

7.4.1.2 Best discretizer, filter and classifier combination

In this study, 16 different combinations of discretizer, filter and classifier were tried over 21 data sets. Moreover, for reasons of completeness, the filtering step was considered optional.

Table 7.23: Number of times a combination gets the best results. W, F, K and M stands for EWD, EFD, PKID and EMD discretizers, respectively.

Naive Bayes Classifier											
CFS				CBF				Without FS			
W	F	K	M	W	F	K	M	W	F	K	M
4	1	0	4	1	0	0	2	1	0	0	2
C4.5 Classifier											
CFS				CBF				Without FS			
W	F	K	M	W	F	K	M	W	F	K	M
0	0	1	0	2	0	0	2	1	0	0	3

Table 7.23 attempts to determine which combination gets the best accuracy values. If two combinations obtain identical accuracy, both are computed in this table, and so all values in table 7.23 add up to 24, not to 21. Several conclusions can be extracted from this table attending to different issues:

- Feature selection (with or without): the number of combinations using feature selection and reaching the best performance values are 17 from 24, which denotes the adequacy of its use.
- Discretizer: Entropy minimization discretizer forms part of the "best" combination in 13 occasions. On the other hand, PKID is only included in a "best" combination using C4.5, although it is suited for naive Bayes classifier, but it is suboptimal when learning from training data of small size (Y. Yang & Webb, 2001).
- Filter: CFS seems to be a good filter combined with NB classifier either using EWD or EMD discretizers. However, this filter does not achieve the best results

when it is applied together with C4.5 classifier; in this case, the consistency based filter is preferred.

- Classifier: Naive Bayes obtains better results in more data sets than C4.5. Specifically, naive Bayes gets the best values for 12 different data sets, while C4.5 only for 8 of them (both classifiers achieve the same accuracy for the *Dermatology* dataset).

7.5 Summary

This chapter proposed a method based on the combination of discretization, filtering and classification methods to be applied to different datasets with the goal of maintaining or even improving the performance while using a reduced set of features. Up to four discretizers, three filters and nine classifiers have been used to demonstrate the adequacy of the combination method.

First, the proposed method was applied to the KDD Cup 99 dataset, a high dimensional benchmark in the intrusion detection field with some features highly dispersed. It improved the results obtained by the KDD Cup 99 Competition winner and also by other authors with a significant reduction in the number of features used in two different approaches. The simplest one considered a binary classification and the second one a multiple class classification.

Then, the method was tested on the crucial task of accurate gene selection in class prediction problems over DNA microarray datasets. The combination method was then compared with the approaches of other authors (using wrappers and filters). Our results outperform those obtained by them, in some cases with improvements in the accuracy and descents in the number of genes needed. This result is very interesting, especially when comparing with the wrapper approach, for two reasons: first, wrappers theoretically obtain better performances, but at the expense of higher time and computational requirements than the filter approach, which in turn has the advantage of independence of the evaluation function used. Second, for very high-dimensional data sets, wrapper methods might be computationally unfeasible. So, our proposed method turns out to be a fast technique, independent of any evaluation function used, and provides good performance in prediction accuracy. The benefit of including a discretization step previous to the filter+classification step was so proved.

Among the broad suite of classifiers existing in the literature, local modeling classifiers have not been prodigally applied to microarray data. For this sake, the proposed method was also combined with a local classifier called FVQIT (Frontier Vector Quantization using Information Theory). This classifier is able to successfully classify high dimensional data sets with few instances. Some methods can not even confront those data sets due to the very high number of features (tens of thousands) and other suffer from overfitting problems when the ratio number of samples / number of features is small. The results obtained when combining FVQIT with a discretizer and a filter obtained better performance than combinations with other classifiers. FVQIT turned out to be the best option when dealing with microarray data, obtaining an average of 1.25 position in the ranking of classifiers, clearly improving the 2.92 of the second classified.

Finally, a further study on multiclass datasets was presented. Multiple class problems can be dealt with by means of two different approaches: using a multiple class algorithm and using multiple binary classifiers. For the latter approach, two class binarization techniques were utilized, namely *One vs Rest* and *One vs One*. For the *One vs One* scheme, the results provided for each classifier were gathered using 4 decoding techniques. The experimental results supported the hypothesis that using feature selection leads to better performance results than not using it. Comparing the different approaches to deal with multiple class problems, the One vs One scheme obtains better accuracy results in average than the others, although using a higher number of features. This approach is also more computational demanding than the others, because each sample must be tested for different classifiers, unifying their results later to obtain the desired output. From the experimental results achieved, the *One vs One* scheme should be used when there are enough samples per class and features, while, on the contrary, the *One vs Rest* can be used with data sets with large number of features and reduced number of samples.

CHAPTER 8

An ensemble of filters and classifiers

Classically, machine learning methods have used a single learning model to solve a given problem. However, the technique of using multiple prediction models for solving the same problem, known as *ensemble learning*, has proven its effectiveness over the last few years (Kuncheva, 2004). The idea builds on the assumption that combining the output of multiple experts is better than the output of any single expert. Typically, ensemble learning has been applied to classification, where the most popular methods are bagging (Breiman, 1996) and boosting (Schapire, 1990) due to their theoretical performance guarantees and robust experimental results. However, it can be also thought as a means of improving other machine learning disciplines such as *feature selection*.

Feature subset selection was employed as a useful technique for creating diversity in classification ensembles. In this case, diversity was incorporated as an objective in the search for obtaining the best collection of feature subsets. While traditional feature selection algorithms have as goal to find the best subset for both the learning task and the selected inductive learning algorithms, the aim of this ensembled feature selection was additionally finding a set of feature subsets that promote disagreement among the base classifiers (Tsymbal, Pechenizkiy, & Cunningham, 2005). T. Ho (1998) has shown that simple random selection of feature subsets may be an effective technique for ensemble feature selection because the lack of accuracy in the ensemble members is compensated by their diversity. Opitz (1999) describes an ensemble feature selection technique for neural networks called *Genetic Ensemble Feature Selection* and another ensemble method for decision trees is called *Stochastic Attribute Selection Committees* (Zheng & Webb, 1998). More recently, Aly and Atiya (2006) proposed several novel variations to the basic feature subset ensembles present in the literature, trying to improve their results. Finally, Abeel et al. (2010) conducted a large-scale analysis of ensemble feature selection in order to show their adequacy over biomarker selection.

However, the idea of ensemble feature selection presented herein is a little different. Real life datasets come in diverse flavors and sizes, and so their nature imposes several

substantial restrictions for both learning models and feature selection algorithms (Tuv et al., 2009). Datasets may be very large in samples and number of features, and also there might be problems with redundant, noisy, multivariate and non-linear scenarios. Thus, most existing methods alone are not capable of confronting these problems, and something like "the best feature selection method" simply does not exist in general, making it difficult for users to select one method over another. In order to make a correct choice, a user not only needs to know the domain well and the characteristics of each dataset, but also is expected to understand technical details of available algorithms (Liu & Yu, 2005).

So, the idea is to use an ensemble of filters to induce diversity, instead of a single method, which is the approach that has been used in the previous chapters. In this chapter, an in-depth research in ensembles of filters is presented, exploring novel ensembles that could improve performance. The objective is to introduce diversity and increase the stability of the feature selection process, since it takes advantage of the strengths of the single selectors and overcomes their weak points. A total of five configurations for the ensemble of filters are proposed, which are tested using four different classifiers. Experimental validation of the methodology on synthetic data shows the adequacy of the proposed ensembles, paving the way to their application on real and DNA microarray data obtaining high level performance results.

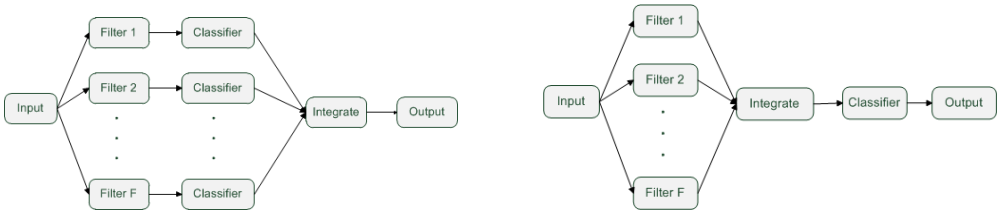
8.1 The rationale of the approach

In some of the examples mentioned above, the disturbances in the training set due to resampling cause diverse base classifiers to be built. In other cases, the technique employed was to use different features for each of the base classifiers. Usually, the ensembles found in the literature involving feature selection are based on the idea of applying several feature selection methods in order to distribute the whole set of features into the instances of the classifier (Pradhananga, 2007). It has to be noted that this method implies that all the features in the training set are exhaustively used.

Nevertheless, the purpose of the proposed ensemble is different. As stated before, one of the problems of choosing a filter is its variability of results over different data sets. That is, a filter can obtain excellent classification results in a given data set while performing poorly in another data set, even in the same domain, depending on the specific properties of the different data sets. Our goal is to achieve a method that

reduces the variability of the features selected by the filters in the different classification domains. Therefore, this research will be based on the idea of combining several filters, employing different metrics and performing a feature reduction.

Two distinct general approaches are proposed: Ensemble1 and Ensemble2 (see Figure 8.1). The main difference between them is that the former uses several filters and classifies once for each filter, thus an integration method for the outputs of the classifier is necessary; whilst the later uses several filters, combines the different subsets returned by each filter, and finally obtains a classification output for this unique subset of features.



(a) Ensemble 1 (b) Ensemble 2

Figure 8.1: Implementations of the ensemble

8.2 The process of selecting the methods for the ensemble

As stated in Chapter 2, feature selection algorithms designed with different evaluation criteria broadly fall into three categories: the filter model, the embedded model and the wrapper model. The objective of the proposed ensemble is that it can be applied to high-dimensional data, such as DNA microarray, so the wrapper model is discarded because it could not generalize adequately. Therefore, in a first stage, filters and embedded methods were chosen to perform a previous study, paving the way for its application to the ensemble.

As the goal is to choose methods based on different metrics, five filters and two embedded methods were tested over five synthetic data sets under different situations: increasing number of irrelevant features and the insertion of noise in the inputs, as well as the inclusion of correlated features. Both filter and embedded methods were described in detail in Chapter 2 and it has to be noted that three of them (CFS, consistency-based and INTERACT) provide a subset of features, whereas the remain-

ing four (Information Gain, ReliefF, SVM-RFE and FS-P) provide features ordered according to their relevance (a ranking of features).

In order to determine the effectiveness of each one of the feature selection methods mentioned above at different situations, several widely-used synthetic datasets were employed (see Appendix I): the LED dataset, the CorrAL dataset and the XOR-100 dataset.

Table 8.1 shows the score for each feature selection method over each scenario and also an overall score for each method (last column). This score is defined as:

$$score = \left[\frac{R_s}{R_t} - \frac{I_s}{I_t}\right] \times 100,$$

where R_s is the number of relevant features selected, R_t is the total number of relevant features, I_s is the number of irrelevant features selected and I_t is the total number of irrelevant features. Notice that 100 is the desired value for this score and negative values indicate a high selection of irrelevant features.

Table 8.1: Score for every feature selection method tested

Method	CorrAL	CorrAL-100	XOR-100	Led-25	Led-100	Average
CFS	50.00	94.00	46.00	71.50	71.33	66.57
Consistency	50.00	94.00	46.00	68.00	64.00	64.40
INTERACT	25.00	92.00	47.00	66.67	73.50	60.83
InfoGain	0.00	88.00	-1.00	66.33	70.00	44.67
ReliefF	50.00	88.00	95.00	78.17	82.50	78.73
SVM-RFE	50.00	59.00	-15.00	22.83	25.33	27.93
FS-P	0.00	43.00	-9.00	72.00	70.67	35.33

As can be seen in Table 8.1, the two embedded methods (SVM-RFE and FS-P) achieve the poorest scores. As SVM-RFE achieved the worst result, we decided not to use it in the proposed ensemble. Focusing on the filters, although ReliefF obtained the best average, CFS, Consistency and INTERACT also showed a good performance. Information Gain obtained the poorest results of the filters methods, and similar to those obtained by FS-P. However, since Information Gain performs better than FS-P and bearing in mind the higher computational cost of the embedded methods, FS-P is discarded. Thus, all the five filters were selected to conform the proposed ensemble.

8.3 The proposed filter ensemble approaches

As has been said before, when dealing with ensemble feature selection, a typical practice is to use different features for each of the base classifiers. However, with the ensemble proposed herein, not all the features have to be necessarily employed, since the idea is to apply several filters based on different metrics so as to have a diverse set of selections. By using this ensemble of filters, the user is released from the task of choosing an adequate filter for each scenario, because this approach obtains acceptable results independently of the characteristics of the data. Among the broad suite of filters available in the literature, five filters were selected according to a study performed in Section 8.2, all of them based on different metrics. Two distinct general approaches are proposed: Ensemble1 and Ensemble2 (see Figure 8.1). The main difference between them are that the former uses several filters and classifies once for each filter, as an integration method for the outputs of the classifier is necessary, whilst the later uses several filters, combines the different subsets returned by each filter, and finally obtains a classification output for this unique subset of features.

8.3.1 Ensemble 1

This is a more classic approach, consisting of an ensemble of classifiers including a previous stage of feature selection (see Figure 8.1a). Particularly, each one of the F filters selects a subset of features and this subset is used for training a given classifier. Therefore, there will be as many outputs as filters were employed in the ensemble (F). Due to the different metric the filters are based on, they select different sets of features leading to classifier outputs that could be contradictory, so an integration method becomes necessary. Note that in each execution F filters and only one classifier are used, but the classifier is trained F times (once for each filter). The pseudo-code is shown in Algorithm 8.1. Different variants of this philosophy will be implemented regarding the combination of the F outputs. Two different methods are considered, producing two implementations of Ensemble1. The first uses the well-known simple voting (E1-sv), where for a particular instance, each classifier votes for a class and the class with the greatest number of votes is considered the output class. The second implementation (E1-cp) stores the probability with which an instance has been assigned to a class. The class with the highest cumulative probability is considered the output class.

Algorithm 8.1: Pseudo-code for *Ensemble1*

Data: $F \leftarrow$ number of filters

Result: P $\leftarrow$ classification prediction

1 **for** *each f from 1 to F* **do**

2 Select attributes A using filter f

3 Build classifier C_f with the selected attributes A

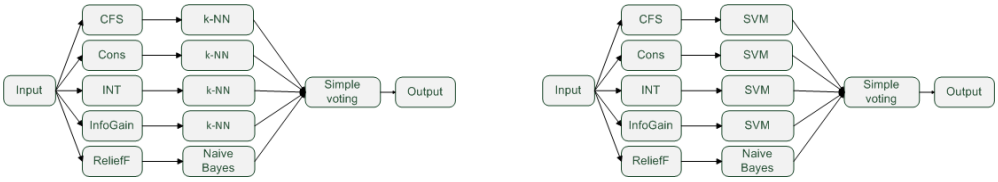
4 Obtain prediction P_f from classifier

end

5 Apply a combination method over predictions $P_1 \dots P_f$

6 Obtain prediction P

Another variation in the basic scheme of Ensemble1 comes from thinking that instead of using the same classifier for all five filters, there might be classifiers more suitable for certain feature selection methods. In fact, in Chapter 3 it was stated that CFS, Consistency-based, INTERACT and InfoGain select a small number of relevant features, whilst ReliefF is very effective at removing redundancy. On the other hand, k-NN and SVM deteriorate their performance when irrelevant features are present whereas naive Bayes is robust with respect to irrelevant features but deteriorates with redundant ones. In this situation, we propose to try an ensemble which uses naive Bayes together with ReliefF and k-NN with the remaining filters (E1-nk) and another which uses again naive Bayes together with ReliefF and SVM with the remaining filters (E1-ns). Both these configurations can be seen in Figure 8.2.



(a) E1-nk (b) E1-ns

Figure 8.2: Configurations of Ensemble1: E1-nk and E1-ns

8.3.2 Ensemble 2

This approach consists of combining the subsets selected by each one of the F filters obtaining only one subset of features. This method has the advantage of not requiring

a combiner method in order to obtain the class prediction. On the contrary, it needs a method to combine the features returned by each F filter, as can be seen in Figure 8.1b, since it only employs one classifier. In preliminary studies, strategies such as the union or the intersection of the subsets were explored, but they led to poor results due to the redundancy induced by the union or the extremely aggressive reduction in the set of features produced by the intersection. Thus, as can be seen in Algorithm 8.2, in this research we propose to combine the subsets of features so as to add to the final subset only those subsets capable of outperforming the classification accuracy in the training set.

Algorithm 8.2: Pseudo-code for *Ensemble2*

Data: $F \leftarrow$ number of filters

Result: P $\leftarrow$ classification prediction

```
for each f from 1 to F do
    Select attributes A_f using filter f
end
A = A_f
baseline = classifying subset A with classifier C
for each f from 2 to F do
    A_aux = A ∪ A_f
    accuracy = classifying subset A_aux with classifier C
    if accuracy > baseline then
        A = A_aux
        baseline = accuracy
    end
end
Build classifier C with the selected attributes A
Obtain prediction P
```

The complexity of the proposed ensembles depends on the machine learning algorithms used. Let K and J be the complexities of the feature selection and the data mining algorithms, respectively, and F the number of filters used in the ensembles. The complexity will be $F \max(K, J)$. Since the idea of both Ensemble1 and Ensemble2 is to use a small number of filters, compared with the number of features or samples of the datasets, F can be considered negligible and it can be said that the complexity of these ensembles is determined by the method with the higher complexity (either K or J). Therefore, it is not more computationally complex than the filters employed alone.

8.4 Experimental setup

Although the final goal of a feature selection method is to test its effectiveness over a real dataset, a first step showing performance over synthetic data follows. The main advantage of artificial scenarios is the knowledge of the set of optimal features that must be selected, thus the degree of closeness to any of these solutions can be assessed in a confident way. The LED problem (see Appendix I) has been chosen as the synthetic dataset to test the proposed ensembles. It is a simple classification task that consists of, given the active leds on a seven segments display, identifying the digit that the display is representing. Therefore, the classification task to be solved is described by 7 binary attributes and 10 possible classes available. In particular, it will be used the dataset Led100, which consists of 50 samples and 100 features, where 92 irrelevant attributes (with random binary values) have been added to the 7 relevant ones.

Then, to check if the behavior displayed by the ensembles over the synthetic dataset can be extrapolated to the real world, 5 real classical datasets were chosen, which can be consulted in Appendix I, Table I.2. This suite of datasets represents different problematic that can appear in real data, such as non-linearity (Madelon) or high imbalance of the classes (Ozone). These datasets have only available a training dataset, so a 10-fold cross-validation will be performed. Finally, and in order to widen the scope of this research, the proposed ensemble will be also tested over a challenging scenario: DNA microarray data. These type of datasets poses an enormous challenge for feature selection researchers due to their high number of gene expression and the small sample size. Seven well-known binary microarray datasets (see Chapter 4) are considered: Colon, DLBCL, CNS, Leukemia, Prostate, Lung and Ovarian. Those datasets originally divided into training and test sets were maintained, whereas, for the sake of comparison, datasets with only training set were randomly divided using the common rule 2/3 for training and 1/3 for testing, as done in previous chapters. This division introduces a more challenging scenario, since in some datasets, the distribution of the classes in the training set differs from the one in the test set. Table I.8 (Appendix I) depicts the number of attributes and samples and also the distribution of the binary classes, i.e. the percentage of binary labels in the datasets, showing if the data is unbalanced.

While three of the filters which form part of the ensemble return a feature subset (CFS, Consistency-base and INTERACT), the other two (ReliefF and Information Gain) are ranker methods, so it is necessary to establish a threshold in order to obtain a subset of features. Initial experiments on microarray data showed that for most of

the data sets, the subset filters selected a number of features between 25 and 50. For the sake of fairness, the rankers were forced to select a number of features similar to the cardinality obtained by the other type of filters. Several experiments were carried out with 25 and 50 features. As performance did not improve using 50 features with respect to 25, we have decided to force these ranker methods to obtain subsets with 25 features. Finally, to test the performance of the different ensembles of filters proposed it is necessary to use a classifier which provides classification accuracy as a measure of adequacy of the method. For this purpose, four well-known classifiers were chosen: C4.5, naive Bayes, k-NN and SVM.

8.4.1 The stability of the selected filters

With the advent of high-dimensionality data sets in classification problems, a variety of feature selection methods have been developed to tackle them. The major challenge in feature selection methods is to extract a set of features, as small as possible, that accurately classifies the learning examples (Kalousis, Prados, & Hilario, 2007). But a relatively neglected issue in the work on high-dimensionality problems is the stability of the feature selection methods used, which is defined as the sensitivity of a method to variations in the training set.

In this study, we checked the stability of the different filters used in the proposed ensemble. For this purpose, we measured similarity between two subsets of features $\{s, s'\}$, using an adaptation of the Tanimoto distance between two sets (Kalousis et al., 2007):

$$S(s, s') = 1 - \frac{|s| + |s'| - 2|s \cap s'|}{|s| + |s'| - |s \cap s'|}$$

Table 8.2 shows the stability of the microarray datasets in terms of average. For each data set and algorithm, the stability was measured comparing the subset selected in each fold of a 10-fold cross-validation. Then, an average of each one of these results was performed and is shown in Table 8.2. It has to be noted that according to Tanimoto measure, 0 means the minimum stability and 1 the maximum. The best result for each data set is emphasized in bold font.

Table 8.2: Stability of the filters selected.

Algorithm	Leuk.	CNS	DLBCL	Lung	Ovar.	Prost.	Colon	AVG
CFS	0.213	0.208	0.274	0.247	0.386	0.340	0.319	0.284
Cons	0.170	0.151	0.470	0.126	0.486	0.077	0.138	0.231
INT	0.246	0.182	0.232	0.221	0.262	0.207	0.264	0.231
InfoGain	0.654	0.252	0.488	**0.721**	**0.875**	0.322	0.529	0.549
ReliefF	**0.684**	**0.307**	**0.621**	0.605	0.688	**0.395**	**0.675**	**0.568**

As one can see in the table above, the best stabilities are achieved by the ReliefF filter, which also obtained the best performance in the previous study (see Table 8.1). Therefore, this filter is expected to obtain good classification results. On the other hand, Information Gain obtained very good stability results, although this filter performed poorly in the previous study over synthetic data (see Table 8.1), but this result suggests that this can be a useful filter since its stability is very high. It has to be noted that both ReliefF and Information Gain are ranker methods. Interact, CFS and Consistency are the least stable, however they obtained very good results in terms of score. Thus, this study reaffirms the choice of these 5 filters to comprise the proposed ensemble.

8.5 Experimental results

In this section the results obtained after applying the different proposed ensembles will be shown. To sum up, five ensemble approaches will be tested: *E1-sv*, which is Ensemble1 using simple voting as combination method; *E1-cp*, which is Ensemble1 using cumulative probabilities as combination method; *E1-nk*, which is Ensemble1 with specific classifiers naive Bayes and k-NN; *E1-ns*, which is Ensemble1 with specific classifiers naive Bayes and SVM and *E2*, which is Ensemble2.

8.5.1 Results on synthetic data

Table 8.3 shows the results obtained by the ensembles and the filters alone over the synthetic dataset Led100. The number of relevant features selected (where the optimal is 7), the number of irrelevant features selected (where the maximum is 92) and the test classification error is exhibited, after randomly dividing the dataset using the common

rule 2/3 for training and 1/3 for testing. Please note that in this concrete case, the number of features the rankers were forced to select was 7, corresponding with the optimal number of relevant features.

Table 8.3: Error results over the synthetic dataset Led100 as well as number of relevant and irrelevant features selected.

		No. rel.	No. irrel.	C4.5	NB	k-NN	SVM
Ensembles	E1-sv	6	2	6.25	0.00	0.00	6.25
	E1-cp	6	2	6.25	0.00	0.00	6.25
	E1-nk	6	2	0.00	0.00	0.00	0.00
	E1-ns	6	2	6.25	6.25	6.25	6.25
	E2	6	0	6.25	0.00	0.00	6.25
Filters	CFS	6	0	6.25	0.00	0.00	6.25
	Cons	5	0	6.25	0.00	0.00	6.25
	INT	6	0	6.25	0.00	0.00	6.25
	IG	6	1	6.25	12.50	12.50	0.00
	ReliefF	5	2	18.75	31.25	31.25	18.75

The results demonstrate the adequacy of the proposed ensembles, since they matched or improved upon the results achieved by the filters alone. Focusing on the features selected, it is important to note that although the theoretical number of relevant features is 7 (one for each led segment), there are two segments that are not relevant for distinguishing among the 10 numbers. For this reason, the consistency filter was able to correctly classify all the instances using only 5 out of the 7 theoretical relevant features. The reader should also notice that the features selected by the four Ensemble1 approaches are the union of the features selected by each one of the filters. Therefore it is more informative to focus on the classification error. According to this measure, it is easy to see that the ensembles take advantage of the filters which work correctly on a dataset and discard the influence of those which do not (IG and ReliefF, in the dataset at hand).

8.5.2 Results on classical datasets

After verifying on synthetic data that our proposed ensembles behave in a confident way, the next step is to evaluate their performance on real classical datasets. Five datasets were chosen for this task (Ozone, Spambase, Mushrooms, Splice and Madelon), which

can be consulted in Table I.2 (Appendix I). In this case, a 10-fold cross validation will be used and the results obtained are depicted in Table 8.4: average test classification error along with the average number of features required to train the model. In the case of the classifier alone, it uses the whole set of features. When using this type of validation with several repetitions, the use of statistical inference for analyzing the results is a crucial and necessary task in an investigation.

For this reason, a Kruskal-Wallis test was applied to check if there are significant differences among the medians for each method for a level of significance $\alpha = 0.05$. If differences among the medians were found, a multiple comparison procedure (Tukey's) was applied to find the simplest approach whose classification error is not significantly different for the approach with the lowest error (labeled with a cross in the tables).

For all datasets and classifiers, one of the five ensembles proposed in this research obtains the lowest error, showing the adequacy of the ensemble approach in these standard datasets. It is necessary to note that Ozone is an extremely unbalanced dataset (see Table I.2, Appendix I) and by assigning all the samples to the majority class, an error of 2.88% could be obtained. None of the methods tested was able to improve this result, although some methods matched it. An oversampling technique was applied over this difficult dataset but due to its extremely high imbalance, no improvement was obtained. On the other hand, it is worth mentioning that E1-nk achieves a promising result over Madelon dataset. In fact, it reduces the test error up to 33% compared with the classifier alone and up to 24% compared with CFS filter, using only 8% of the total features (see results for SVM).

These results are not easy to analyze since the classifier plays a crucial role and provides a very different classification error even with the same set of features. There are several cases found in Table 8.4 that confirm this fact, for example: ReliefF over Ozone dataset achieves an error of 2.88% according to SVM whilst naive Bayes classifier raises the error up to 29.06%; and the consistency filter over Madelon dataset increases its error from 9.08% to 33.42% using k-NN and SVM, respectively.

Table 8.5 displays the average of test error for each dataset and method, independently of the classifier, which should help to clarify which one is the best method for a given dataset. E1-nk is the method which is significantly better in the maximum number of datasets (3), followed by E2 and the consistency filter.

Table 8.4: Test classification error after 10 fold cross-validation for classical datasets, the number in parenthesis is the number of features selected by the method. Those methods whose average test classification results are not significantly worse than the best are labeled with a cross (†).

		Method	Ozone	Spambase	Mushrooms	Splice	Madelon
C4.5	Ensembles	E1-sv	3.35 (43)†	5.76 (44)†	0.00 (42)†	4.10 (32)†	16.17 (38)
		E1-cp	3.23 (43)†	5.69 (44)†	0.00 (42)†	4.10 (32)†	15.42 (38)
		E1-nk	4.18 (43)	8.22 (44)	0.00 (42)†	15.10 (32)	9.08 (38)†
		E1-ns	2.88 (43)†	11.50 (44)	1.13 (42)	20.70 (32)	33.25 (38)
		E2	3.71 (21)†	6.74 (31)†	0.00 (22)†	5.80 (18)†	14.21 (25)†
		C4.5	4.61	6.67†	0.00†	6.00†	19.88
	Filters	CFS	3.71 (17)†	7.43 (15)	1.48 (8)	6.00 (12)†	19.33 (8)
		Cons	3.08 (4)†	7.35 (25)†	0.15 (10)	5.50 (12)†	17.04 (13)
		INT	3.82 (19)†	7.32 (29)†	0.00 (10)†	4.90 (15)†	17.04 (13)
		IG	3.67 (25)†	7.43 (25)	0.00 (25)†	5.90 (25)†	18.33 (25)
		ReliefF	3.86 (25)†	8.80 (25)	0.00 (25)†	5.90 (25)†	14.08 (25)†
Naive Bayes	Ensembles	E1-sv	21.06 (43)	14.39 (44)	6.84 (42)	17.60 (32)†	30.17 (38)
		E1-cp	21.14 (43)	13.39 (44)	6.87 (42)	16.70 (32)†	29.92 (38)
		E1-nk	4.18 (43)†	8.22 (44)†	0.00 (42)†	15.10 (32)†	9.08 (38)†
		E1-ns	2.88 (43)†	11.50 (44)†	1.13 (42)†	20.70 (32)†	33.25 (38)
		E2	20.98 (17)	15.24 (32)	1.23 (14)†	16.50 (27)†	30.08 (15)
		NB	28.98	20.47	6.81	16.40†	31.38
	Filters	CFS	20.98 (17)	21.23 (15)	1.48 (8)†	17.90 (12)†	30.29 (8)
		Cons	6.94 (4)†	11.82 (25)†	4.89 (10)	17.50 (12)†	29.92 (13)
		INT	20.58 (19)	16.82 (29)	5.45 (10)	16.00 (15)†	29.92 (13)
		IG	26.10 (25)	11.63 (25)†	6.95 (25)	17.10 (25)†	30.42 (25)
		ReliefF	29.06 (25)	29.41 (25)	6.60 (25)	16.30 (25)†	30.08 (25)
k-NN	Ensembles	E1-sv	3.51 (43)†	7.93 (44)†	0.00 (42)†	15.50 (32)†	8.92 (38)†
		E1-cp	3.51 (43)†	7.93 (44)†	0.00 (42)†	15.50 (32)†	8.92 (38)†
		E1-nk	4.18 (43)	8.22 (44)†	0.00 (42)†	15.10 (32)†	9.08 (38)†
		E1-ns	2.88 (43)†	11.50 (44)	1.13 (42)	20.70 (32)	33.25 (38)
		E2	4.85 (28)	10.76 (19)	0.00 (19)†	20.20 (15)†	9.08 (13)†
		k-NN	4.73	9.09†	0.00†	30.80	41.04
	Filters	CFS	4.97 (17)	11.50 (15)	1.75 (8)	20.50 (12)	13.38 (8)
		Cons	3.23 (4)†	10.43 (25)	0.00 (10)†	19.40 (12)†	9.08 (13)†
		INT	4.85 (19)	10.61 (29)	0.00 (10)†	19.10 (15)†	9.08 (13)†
		IG	4.61 (25)	10.32 (25)	0.00 (25)†	24.70 (25)	23.08 (25)
		ReliefF	4.30 (25)	12.54 (25)	0.00 (25)†	21.60 (25)	10.96 (25)†
SVM	Ensembles	E1-sv	2.88 (43)†	12.00 (44)	1.08 (42)	20.80 (32)	33.46 (38)
		E1-cp	2.88 (43)†	12.00 (44)	1.08 (42)	20.80 (32)	33.46 (38)
		E1-nk	4.18 (43)	8.22 (44)†	0.00 (42)†	15.10 (32)†	9.08 (38)†
		E1-ns	2.88 (43)†	11.50 (44)	1.13 (42)	20.70 (32)†	33.25 (38)
		E2	2.88 (17)†	10.28 (41)†	0.00 (19)†	19.70 (25)†	33.58 (16)
		SVM	2.88†	9.59†	0.00†	20.40†	42.25
	Filters	CFS	2.88 (17)†	13.17 (15)	1.66 (8)	21.00 (12)	34.04 (8)
		Cons	2.88 (4)†	12.50 (25)	0.00 (10)†	21.90 (12)	33.42 (13)
		INT	2.88 (19)†	11.76 (29)	2.26 (10)	21.60 (15)	33.42 (13)
		IG	2.88 (25)†	12.08 (25)	1.86 (25)	20.00 (25)†	33.50 (25)
		ReliefF	2.88 (25)†	13.50 (25)	0.10 (25)†	19.30 (25)†	33.67 (25)

Table 8.5: Average of test error for classical datasets focusing on the dataset. Those methods whose average test classification results are not significantly worse than the best are labeled with a cross (†).

		Ozone	Spambase	Mushrooms	Splice	Madelon
Ensembles	E1-sv	7.70	10.02	1.98	14.50†	22.18
	E1-cp	7.69	9.75†	1.99	14.28†	21.93
	E1-nk	4.18	8.22†	0.00†	15.10	9.08†
	E1-ns	2.88†	11.50	1.13	20.70	33.25
	E2	8.10	10.75	0.31†	15.55†	21.74
	Classif	10.30	11.45	1.70	18.40†	33.64
Filters	CFS	8.13	13.33	1.59	16.35†	24.26
	Cons	4.03†	10.52	1.26	16.07†	22.36
	INT	8.03	11.63	1.93	15.40†	22.36
	IG	9.32	10.37	2.20	16.93†	26.33
	ReliefF	10.03	16.06	1.67	15.78†	22.20

Table 8.6: Average of test error for classical datasets focusing on the classifier.

		C4.5	NB	k-NN	SVM
Ensembles	E1-sv	5.88	18.01	7.17	14.04
	E1-cp	5.69	17.60	7.17	14.04
	E1-nk	7.32	7.32	7.32	7.32
	E1-ns	13.89	13.89	13.89	13.89
	E2	6.09	16.81	8.98	13.29
	Classif	7.43	20.81	17.13	15.02
Filters	CFS	7.59	18.38	10.42	14.55
	Cons	6.62	14.21	8.43	14.14
	INT	6.62	17.76	8.73	14.38
	IG	7.07	18.44	12.54	14.06
	ReliefF	6.53	22.29	9.88	13.89

Table 8.6 depicts the average of test error for each method and classifier, independently of the dataset. In this case it makes no sense to perform a statistical study since the results achieved by the classifiers over different datasets are very different. We can see that for these kinds of datasets, the best option is to use E1-cp combined with C4.5 classifier. It is also worth noting that for the remainder of classifiers tested, it is always one of the ensembles which achieves the best results, outperforming the results obtained by the filters alone. In the next section we will see if the methods tested exhibit the same behavior when dealing with an extremely complex scenario: DNA microarray data classification.

8.5.3 Results on microarray data

The last step for testing the proposed methods is to evaluate them on a difficult scenario such as DNA microarray classification, where the number of features is much higher than the number of samples. Remind that, in this case, a division in training and test sets is assumed (see Section 8.4).

Table 8.7 exhibits the results over all the seven microarray datasets considered for the classifiers included in this research. Along with the error test achieved, one can see the number of features required to train the model. In the case of the classifier alone, it uses the whole set of features. For all the four classifiers employed, one of the five ensemble approaches proposed achieves the lowest error, except for Colon dataset with C4.5 and k-NN. Although the number of features is higher using ensembles than filters alone, it is insignificant when compared with the difference in feature number regarding the complete original feature set.

In order to summarize, Table 8.8 shows the results on average. The best result on average for all datasets and classifiers is obtained by E1-sv and E1-cp combined with SVM classifier, which happens to be a frequently used and appropriate classifier for DNA microarray classification (Mukherjee et al., 1999; M. P. Brown et al., 2000; Furey et al., 2000). As in the classical datasets case, again one of the ensembles achieves always the best result for each classifier. It also should to be noted that there is a slight difference between using cumulative probabilities (E1-cp) or simple voting (E1-sv) as union method in Ensemble1. E1-cp only appears to be better for naive Bayes classifier, but as it does not produce deterioration for any classifier, it is considered to be a better choice than E1-sv. For all these reasons, the authors recommend to use E1-cp combined with SVM classifier when dealing with DNA microarray data.

Table 8.7: Test classification error for microarray datasets, the number in parenthesis is the number of features selected by the method. Best error for each dataset is highlighted in bold face.

		Method	Colon	DLBCL	CNS	Leukemia	Prostate	Lung	Ovarian
C4.5	Ensembles	E1-sv	15.00 (58)	13.33 (73)	50.00 (95)	8.82 (63)	73.53 (126)	18.12 (55)	**0.00** (67)
		E1-cp	15.00 (58)	13.33 (73)	50.00 (95)	8.82 (63)	73.53 (126)	18.12 (55)	**0.00** (67)
		E1-nk	20.00 (58)	13.33 (73)	40.00 (95)	**5.88** (63)	67.65 (126)	**0.00** (55)	**0.00** (67)
		E1-ns	20.00 (58)	**6.67** (73)	**30.00** (95)	11.76 (63)	**29.41** (126)	1.34 (55)	**0.00** (67)
		E2	15.00 (34)	13.33 (47)	50.00 (60)	8.82 (36)	73.53 (89)	18.12 (40)	**0.00** (37)
		C4.5	**10.00**	13.33	40.00	8.82	73.53	18.12	1.19
	Filters	CFS	15.00 (19)	13.33 (47)	50.00 (60)	8.82 (36)	73.53 (89)	18.12 (40)	**0.00** (37)
		Cons	15.00 (3)	13.33 (2)	50.00 (3)	8.82 (1)	76.47 (4)	18.12 (1)	**0.00** (3)
		INT	15.00 (16)	13.33 (36)	45.00 (47)	8.82 (36)	73.53 (73)	18.12 (40)	1.19 (27)
		IG	30.00 (25)	13.33 (25)	50.00 (25)	8.82 (25)	70.59 (25)	10.07 (25)	**0.00** (25)
		ReliefF	15.00 (25)	13.33 (25)	35.00 (25)	8.82 (25)	67.65 (25)	2.68 (25)	1.19 (25)
Naive Bayes	Ensembles	E1-sv	20.00 (58)	**6.67** (73)	45.00 (95)	8.82 (63)	73.53 (126)	**0.00** (55)	1.19 (67)
		E1-cp	20.00 (58)	**6.67** (73)	40.00 (95)	8.82 (63)	73.53 (126)	**0.00** (55)	1.19 (67)
		E1-nk	20.00 (58)	13.33 (73)	40.00 (95)	**5.88** (63)	67.65 (126)	**0.00** (55)	**0.00** (67)
		E1-ns	20.00 (58)	**6.67** (73)	**30.00** (95)	11.76 (63)	**29.41** (126)	1.34 (55)	**0.00** (67)
		E2	25.00 (34)	**6.67** (47)	35.00 (60)	**5.88** (36)	73.53 (89)	**0.00** (40)	2.38 (37)
		NB	30.00	**6.67**	40.00	11.76	73.53	4.70	11.90
	Filters	CFS	**10.00** (19)	**6.67** (47)	**30.00** (60)	**5.88** (36)	73.53 (89)	**0.00** (40)	2.38 (37)
		Cons	15.00 (3)	13.33 (2)	45.00 (3)	8.82 (1)	67.65 (4)	14.09 (1)	**0.00** (3)
		INT	15.00 (16)	**6.67** (36)	35.00 (47)	**5.88** (36)	73.53 (73)	**0.00** (40)	**0.00** (27)
		IG	15.00 (25)	**6.67** (25)	45.00 (25)	8.82 (25)	73.53 (25)	0.67 (25)	2.38 (25)
		ReliefF	20.00 (25)	**6.67** (25)	40.00 (25)	8.82 (25)	79.41 (25)	**0.00** (25)	2.38 (25)
k-NN	Ensembles	E1-sv	20.00 (58)	13.33 (73)	35.00 (95)	14.71 (63)	67.65 (126)	**0.00** (55)	**0.00** (67)
		E1-cp	20.00 (58)	13.33 (73)	35.00 (95)	14.71 (63)	67.65 (126)	**0.00** (55)	**0.00** (67)
		E1-nk	20.00 (58)	13.33 (73)	40.00 (95)	**5.88** (63)	67.65 (126)	**0.00** (55)	**0.00** (67)
		E1-ns	20.00 (58)	**6.67** (73)	**30.00** (95)	11.76 (63)	**29.41** (126)	1.34 (55)	**0.00** (67)
		E2	50.00 (34)	13.33 (47)	35.00 (60)	14.71 (36)	67.65 (89)	**0.00** (40)	**0.00** (37)
		k-NN	**5.00**	26.67	45.00	29.41	47.06	2.01	7.14
	Filters	CFS	20.00 (19)	13.33 (47)	35.00 (60)	14.71 (36)	67.65 (89)	**0.00** (40)	**0.00** (37)
		Cons	15.00 (3)	26.67 (2)	35.00 (3)	8.82 (1)	73.53 (4)	18.12 (1)	**0.00** (3)
		INT	20.00 (16)	13.33 (36)	40.00 (47)	14.71 (36)	67.65 (73)	**0.00** (40)	**0.00** (27)
		IG	15.00 (25)	**6.67** (25)	30.00 (25)	**5.88** (25)	58.82 (25)	1.34 (25)	3.57 (25)
		ReliefF	15.00 (25)	**6.67** (25)	40.00 (25)	17.65 (25)	70.59 (25)	1.34 (25)	**0.00** (25)
SVM	Ensembles	E1-sv	**15.00** (58)	**6.67** (73)	**30.00** (95)	14.71 (63)	**2.94** (126)	1.34 (55)	**0.00** (67)
		E1-cp	**15.00** (58)	**6.67** (73)	**30.00** (95)	14.71 (63)	**2.94** (126)	1.34 (55)	**0.00** (67)
		E1-nk	20.00 (58)	13.33 (73)	40.00 (95)	**5.88** (63)	67.65 (126)	**0.00** (55)	**0.00** (67)
		E1-ns	20.00 (58)	**6.67** (73)	**30.00** (95)	11.76 (63)	29.41 (126)	1.34 (55)	**0.00** (67)
		E2	20.00 (34)	**6.67** (47)	35.00 (60)	11.76 (36)	**2.94** (89)	1.34 (40)	**0.00** (37)
		SVM	25.00	13.33	**30.00**	14.71	47.06	0.67	**0.00**
	Filters	CFS	**15.00** (19)	**6.67** (47)	35.00 (60)	11.76 (36)	**2.94** (89)	1.34 (40)	**0.00** (37)
		Cons	20.00 (3)	13.33 (2)	35.00 (3)	20.59 (1)	73.53 (4)	18.12 (1)	**0.00** (3)
		INT	**15.00** (16)	13.33 (36)	40.00 (47)	11.76 (36)	29.41 (73)	1.34 (40)	**0.00** (27)
		IG	**15.00** (25)	**6.67** (25)	35.00 (25)	11.76 (25)	**2.94** (25)	0.67 (25)	1.19 (25)
		ReliefF	**15.00** (25)	**6.67** (25)	**30.00** (25)	17.65 (25)	5.88 (25)	2.01 (25)	**0.00** (25)

Table 8.8: Average of test error

		C4.5	NB	k-NN	SVM
Ensembles	E1-sv	25.54	22.17	21.53	**10.09**
	E1-cp	25.54	21.46	21.53	**10.09**
	E1-nk	20.98	20.98	20.98	20.98
	E1-ns	**14.17**	**14.17**	**14.17**	14.17
	E2	25.54	20.49	25.81	11.10
	Classifier	23.57	25.51	23.18	18.68
Filters	CFS	25.54	18.35	21.53	10.39
	Cons	25.96	23.41	25.31	25.80
	INT	25.00	19.44	22.24	15.83
	IG	26.12	21.72	17.33	10.46
	ReliefF	20.52	22.47	21.61	11.03

8.5.4 The imbalance problem

Four of the microarray datasets considered in this work presented the so-called imbalance problem (Colon, CNS, Leukemia and Ovarian; see Table I.8 in Appendix I). A dataset is considered imbalanced when the classification categories are not approximately equally represented (see Chapter 4, Section 4.2.2).

Table 8.9: Average of test error after applying SMOTE for datasets Colon, CNS, Leukemia and Ovarian.

		C4.5	NB	k-NN	SVM
Ensembles	E1-sv	**13.46**	17.50	**13.68**	**15.96**
	E1-cp	**13.46**	17.50	**13.68**	**15.96**
	E1-nk	16.47	16.47	16.47	16.47
	E1-ns	18.46	18.46	18.46	18.46
	E2	16.25	**15.82**	24.93	20.44
	Classifier	17.09	26.36	25.98	17.43
Filters	CFS	16.25	**15.82**	24.93	20.44
	Cons	17.50	17.21	15.96	18.46
	INT	14.71	16.47	21.18	20.44
	IG	15.00	17.80	14.86	18.54
	ReliefF	16.25	18.32	17.21	18.16

To overcome this issue, the SMOTE method is applied after the feature selection process in the datasets that show imbalance in the training set. For the sake of brevity, only the average of test error will be shown in Table 8.9. E1-sv and E1-cp obtained again the lowest error combined with C4.5 classifier. As the results obtained are better using SMOTE, the adequacy of this oversampling technique when combined with ensemble techniques is confirmed.

8.6 Summary

Real life datasets may be very large in number of samples or features, and their classification can be hindered by phenomena such as redundancy, noise or non-linearity of the data. For these reasons, some feature selection methods are not able to confront these problems so it is the responsibility of the user to decide which one to use in a particular situation. In this chapter, a framework for feature selection which can be applicable to different scenarios with promising results was presented. The idea was to use an ensemble of filters rather than a single method, in order to take advantage of their individual strengths and overcome their weak points at the same time.

Two general approaches were presented, based on the role the classifier plays. Ensemble1 classifies as many times as there are filters, whereas Ensemble2 classifies only once with the result of joining the different subsets selected by the filters. For Ensemble1, two methods for combining the outputs of the classifiers were studied (simple voting and cumulative probabilities), as well as the possibility of using an adequate specific classifier for each filter. A total of five different implementations of the two approaches of ensemble were proposed, tested in the first place over synthetic data. Results showed the adequacy of the proposed methods on this controlled scenario since they selected the correct features. The next step was to apply these approaches over 5 UCI classical datasets. Experimental results demonstrated that one of the ensembles (E1-cp) combined with C4.5 classifier was the best option when dealing with this type of dataset. Finally, the ensemble configurations were tested over 7 DNA microarray data. These are extremely challenging datasets because of their high number of input features and small sample size, where feature selection becomes indispensable. It turned out that using a ensemble was again the best option. Specifically, the best performance was achieved again with E1-cp but this time combined with SVM classifier. It should be noted that some of these datasets presented a high imbalance of the data. To overcome this problem, an oversampling method was applied after the feature selection process.

The result was that once again one of the ensembles achieved the best performance, and that this was even better than the one obtained with no preprocessing, showing the adequacy of the ensemble combined with over-sampling methods. Thus, the appropriateness of using an ensemble instead of a single filter remained demonstrated, considering that for all scenarios tested, the ensemble was always the more successful solution.

Regarding the different implementations of the ensemble tested, several conclusions can be drawn. There is a slight difference between the two combiner methods employed with Ensemble1 (simple voting and cumulative probability), although the second one obtained the best performance. Among the different classifiers chosen for this study, it appeared that the type of data to be classified determines significantly the error achieved, so it is responsibility of the user to know which classifier is more suitable for a given type of data. The authors recommend using E1-cp with C4.5 when classifying classical datasets (with more samples than features) and E1-cp with SVM when dealing with microarray dataset (with more features than samples). In complete ignorance of the particulars of the data, we suggest using E1-ns, which releases the user from the task of choosing a specific classifier.

CHAPTER 9

Cost-based feature selection

The most common approaches followed by feature selection methods are to find either a subset of features that maximizes a given metric or either an ordered ranking of the features based on this metric. Two of the most popular filter metrics for classification problems are correlation and mutual information, although other common filter metrics include error probability, probabilistic distance, entropy or consistency (see Chapter 2).

There are some situations where a user is not only interested in maximizing the merit of a subset of features, but also in reducing costs that may be associated to features. For example, for medical diagnosis, symptoms observed with the naked eye are costless, but each diagnostic value extracted by a clinical test is associated with its own cost and risk. In other fields, such as image analysis, the computational expense of features refers to the time and space complexities of the feature acquisition process (Feddema, Lee, & Mitchell, 1991). This is a critical issue, specifically in real-time applications, where the computational time required to deal with one or another feature is crucial, and also in the medical domain, where it is important to save economic costs and to also improve the comfort of a patient by preventing risky or unpleasant clinical tests (variables that can be also treated as costs).

Our goal is to obtain a trade-off between a filter metric and the cost associated to the selected features, in order to select relevant features with low cost. A general framework to be applied together with the filter approach is introduced. In this manner, any filter metric can be modified to have into account the cost associated to the input features. In this chapter three implementations of this framework will be presented as an example of use, choosing three representative and widely-used filters: Correlation-based Feature Selection (CFS), Minimal-Redundancy-Maximal-Relevance (mRMR) and ReliefF. All these filters were described in Chapter 2 and are based on different metrics. The results obtained with these three filters are promising, showing that the approach is sound. Finally, the framework was applied to a real problem aiming at reducing the time required to automatically classify the tear film lipid layer.

9.1 Background

New feature selection methods are constantly appearing, however, the great majority of them only focus on removing irrelevant and redundant features but not on the costs for obtaining the input features. The cost associated to a feature can be related to different concepts. For example, in medical diagnosis, a pattern consists of observable symptoms (such as age, sex, etc.) along with the results of some diagnostic tests. Contrary to observable symptoms, which have no cost, diagnostic tests have associated costs and risks. For example, an invasive exploratory surgery is much more expensive and risky than a blood test (J. Yang & Honavar, 1998). Another example of the risk of extracting a feature can be found in the work presented by Bahamonde et al. (2004), in which for evaluating the merits of beef cattle as meat producers is necessary to carry out zoometry on living animals.

On the other hand, the cost can also be related to computational issues. In the medical imaging field, extracting a feature from a medical image can have a high computational cost. For example, in the texture analysis technique known as co-occurrence features (Haralick et al., 1973), the computational cost for extracting each feature is not the same, which implies different computational times. In other cases, such as real-time applications, the space complexity is negligible, but the time complexity is very important (Feddema et al., 1991).

As one may notice, features with an associated cost can be found in many real-life applications. However, this has not been the focus of much attention for machine learning researchers. As mentioned above, the purpose of this research is to propose a general framework to the problem of cost-based feature selection, trying to balance the correlation of the features with the class and their cost. There have been similar attempts to balance the contribution of different terms in other areas. For instance, in classification, J. H. Friedman (1989) included a regularization term to the traditional Linear Discriminant Analysis (LDA). The left side term of their cost function evaluates the error and the right side term would be the regularization one, which is weighted with λ. This provides a framework in which, according to the λ value, different regularized solutions can be obtained. Related to feature extraction, You, Hamsici, and Martinez (2011) proposed a criterion to select kernel parameters based on maximizing between-class scattering and minimizing within-class scattering. Applied to face recognition, Wright, Yang, Ganesh, Sastry, and Ma (2009) proposed a general classification framework to study feature extraction and robustness to occlusion via obtaining

a sparse representation. Instead of measuring the correlation between a feature and the class, this method evaluates the representation error. However, our objective is completely different, as it is to provide a framework for feature selection where features with an inherent cost could be dealt with. Thus, the practitioners could decide on the balance on performance/cost that they prefer.

Despite the previous attempts in classification and feature extraction, to the best knowledge of the authors, there are only a few attempts to deal with this issue in feature selection. In the early 90s, Feddema et al. (1991) were developing methodologies for the automatic selection of image features to be used by a robot. For this selection process, they employed a weighted criterion that took into account the computational expense of features, i.e. the time and space complexities of the feature extraction process. Several years later, J. Yang and Honavar (1998) proposed a genetic algorithm to perform feature subset selection where the fitness function combined two criteria: the accuracy of the classification function realized by the neural network and the cost of performing the classification (defined by the cost of measuring the value of a particular feature needed for classification, the risk involved, etc.). A similar approach was presented by Huang and Wang (2006), in which a genetic algorithm is used for feature selection and parameters optimization for a support vector machine. In this case, classification accuracy, the number of selected features and the feature cost were the three criteria used to design the fitness function. Sivagaminathan and Ramakrishnan (2007) presented a hybrid method for feature subset selection based on ant colony optimization and artificial neural networks. The heuristic that enables ants to select features is the inverse of the cost parameter.

The methods found in the literature that deal with cost associated to the features, which were described above, have the disadvantage of being computationally expensive by having interaction with a classifier, which prevents their use in large databases, a trending topic in recent years (Jiawei & Kamber, 2001). However, the general framework proposed herein is applied together with the filter model, which is known to have a low computational cost and be independent of any classifier. By being fast and with a good generalization ability, filters using this cost-based feature selection framework will be suitable for application to databases with a great number of input features like microarray DNA data.

In light of the above, the novelty of this proposal lies in that there does not exist too much research in cost-based feature selection methods. As a matter of fact, no cost methods can be found in the most popular machine learning and data mining tools. For

instance, in Weka (M. Hall et al., 2009) one can only find some methods that address the problem of cost associated to the instances (not to the features), and they were incorporated in the latest release. RapidMiner (Mierswa, Wurst, Klinkenberg, Scholz, & Euler, 2006) does in fact include some methods that take cost into account, but they are quite simple. One of them selects the attributes that have a cost value which satisfies a given condition and another one just selects the k attributes with the lower cost. Therefore, the general framework for cost-based feature selection proposed in this chapter intends to cover this necessity.

9.2 Description of the method

The proposed method intends to be a framework applicable to any filter. However, in this chapter three representative filters were chosen to implement the idea and carry out an experimentation to discover whether the approach is sound. The filters chosen are CFS, which is a subset filter, and mRMR and ReliefF, which are ranker filters. For the sake of clarity, the modifications we have implemented in CFS, mRMR and ReliefF will be presented and only then the approach will be generalized. Notice that the basic descriptions of these methods can be found in Chapter 2.

9.2.1 minimum cost CFS, mC-CFS

CFS is a multivariate subset filter algorithm. It uses a search algorithm combined with an evaluation function to estimate the merit of feature subsets (see Chapter 2). The implementation of CFS utilized in this research uses forward best first search (Rich & Knight, 1991) as its search algorithm. Best first search is an artificial intelligence search strategy that allows backtracking along the search path. It moves through the search space by making local changes to the current feature subset. If the explored path looks uninteresting, the algorithm can backtrack to a previous subset and continue the search from there on. As a stopping criterion, the search terminates if five consecutive fully expanded (all possible local changes considered) subsets show no improvement over the current best subset.

The evaluation function takes into account the usefulness of individual features for predicting the class label as well as the level of correlation among them. It is assumed that good feature subsets contain features highly correlated with the class and uncorrelated with each other. The evaluation function can be seen in (9.1).

$$M_S = \frac{k\overline{r_{ci}}}{\sqrt{k + k(k-1)\overline{r_{ii}}}} \tag{9.1}$$

where M_S is the merit of a feature subset S that contains k features, $\overline{r_{ci}}$ is the average correlation between the features of S and the class, and $\overline{r_{ii}}$ is the average intercorrelation between the features of S. In fact, this function is Pearson's correlation with all variables standardized. The numerator estimates how S can predict the class and the denominator quantifies the redundancy among the features in S.

The modification of CFS proposed consists of adding a term to the evaluation function to take into account the cost of the features, as can be seen in equation (9.2).

$$MC_S = \frac{k\overline{r_{ci}}}{\sqrt{k + k(k-1)\overline{r_{ii}}}} - \lambda\frac{\sum_{i=1}^{k} C_i}{k} \tag{9.2}$$

where MC_S is the merit of the subset S affected by the cost of the features, C_i is the cost of the feature i, and λ is a parameter introduced to weight the influence of the cost in the evaluation function.

The parameter λ is a positive real number. If λ is 0, the cost is ignored and the method works as the regular CFS. If λ is between 0 and 1, the influence of the cost is smaller than the other term. If $\lambda = 1$ both terms have the same influence and if $\lambda > 1$, the influence of the cost is greater than the influence of the other term.

9.2.2 minimum cost mRMR, mC-mRMR

mRMR is a multivariate ranker filter algorithm. As mRMR is a ranker, the search algorithm is simpler than that of CFS.

The evaluation function combines two constraints (as the name of the method indicates), maximal relevance and minimal redundancy. The former is denoted by the letter D, it corresponds with the mean value of all mutual information values between each feature x_i and class c, and has the expression shown in equation (9.3).

$$D(S, c) = \frac{1}{|S|}\sum_{x_i \in S} I(x_i; c) \tag{9.3}$$

where S is a subset of features and $I(x_i; c)$ is the mutual information between the feature x_i and the class c. The expression of $I(x; y)$ is shown in equation (9.4).

$$I(x;y) = \int\int p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy \tag{9.4}$$

The constraint of minimal redundancy is denoted by the letter R, and has the expression shown in (9.5).

$$R(S) = \frac{1}{|S|^2} \sum_{x_i, x_j \in S} I(x_i, x_j) \tag{9.5}$$

The evaluation function to be maximized combines the two constraints (9.3) and (9.5). It is called *minimal-redundancy-maximal-relevance* (mRMR) and has the expression shown in (9.6).

$$\Phi(D, R) = \frac{1}{|S|} \sum_{x_i \in S} I(x_i; c) - \frac{1}{|S|^2} \sum_{x_i, x_j \in S} I(x_i, x_j) = D(S, c) - R(S) \tag{9.6}$$

In practice, this is an incremental search method that selects on each iteration the feature that maximizes the evaluation function. Suppose we already have S_{m-1}, the feature set with m - 1 features, the m^{th} selected feature will optimize the following condition:

$$\max_{x_j \in X - S_{m-1}} \left[I(x_j; c) - \frac{1}{m-1} \sum_{x_i \in S_{m-1}} I(x_j; x_i) \right] \tag{9.7}$$

The modification of mRMR which is proposed in this research consists of adding a term to the condition to be maximized so as to take into account the cost of the feature to be selected, as can be seen in (9.8)

$$\max_{x_j \in X - S_{m-1}} \left[\left(I(x_j; c) - \frac{1}{m-1} \sum_{x_i \in S_{m-1}} I(x_j; x_i) \right) - \lambda C_j \right] \tag{9.8}$$

where C_j is the cost of the feature j, and λ is a parameter introduced to weight the influence of the cost in the evaluation function, as explained in the previous subsection.

9.2.3 minimum cost ReliefF, mC-ReliefF

ReliefF is a multivariate ranker filter algorithm. It randomly selects an instance R_i, but then searches for k of its nearest neighbors from the same class c, nearest hits H_j, and also k nearest neighbors from each one of the different classes, nearest misses $M_j(c)$. It updates the quality estimation $W[A]$ for all attributes A depending on their values for R_i, hits H_j and misses $M_j(c)$. If instances R_i and H_j have different values of the attribute A, then this attribute separates instances of the same class, which clearly is not desirable, and thus the quality estimation $W[A]$ has to be decreased. On the contrary, if instances R_i and M_j have different values of the attribute A for a class then the attribute A separates two instances with different class values which is desirable so the quality estimation $W[A]$ is increased. Since ReliefF considers multiclass problems, the contribution of all the hits and all the misses is averaged. Besides, the contribution for each class of the misses is weighted with the prior probability of that class $P(c)$ (estimated from the training set). The whole process is repeated m times (where m is a user-defined parameter) and can be seen in Algorithm 9.1.

Algorithm 9.1: Pseudo-code of ReliefF algorithm

Data: training set D, iterations m, attributes a

Result: the vector W of estimations of the qualities of attributes

```
1 set all weights W[A] := 0
2 for i ← 1 to m do
3 |   randomly select an instance R_i
4 |   find k nearest hits H_j
5 |   for each class c ≠ class(R_i) do
6 |   |   from class c find k nearest misses M_j(c)
  |   end
  end
7 for f ← 1 to a do
8 |   W[f] := W[f] − Σ_{j=1}^{k} diff(f,R_i,H_j) / (m·k)
  |          + Σ_{c≠class(R_i)} [ P(c)/(1−P(class(R_i))) Σ_{j=1}^{k} diff(f,R_i,M_j(c)) ] / (m·k)
  end
```

The function $diff(A, I_1, I_2)$ calculates the difference between the values of the attribute A for two instances, I_1 and I_2. If the attributes are nominal, it is defined as:

$$diff(A, I_1, I_2) = \begin{cases} 0; & value(A, I_1) = value(A, I_2) \\ 1; & otherwise \end{cases}$$

The modification of ReliefF we propose in this research, mC-ReliefF, consists of adding a term to the quality estimation $W[f]$ to take into account the cost of the features, as can be seen in (9.9).

$$\begin{aligned} W[f] := W[f] - \frac{\sum_{j=1}^{k} diff(f, R_i, H_j)}{(m \cdot k)} + \\ \frac{\sum_{c \neq class(R_i)} \left[\frac{P(c)}{1 - P(class(R_i))} \sum_{j=1}^{k} diff(f, R_i, M_j(c)) \right]}{(m \cdot k)} - \lambda \cdot C_f, \end{aligned} \tag{9.9}$$

where C_f is the cost of the feature f, and λ is a free parameter introduced to weight the influence of the cost in the quality estimation of the attributes. When $\lambda > 0$, the greater the λ the greater the influence of the cost.

9.2.4 Generalization

Ultimately, the general idea consists on adding a term to the evaluation function of the filter to take into account the cost of the features. Since, to our best knowledge, all filters use an evaluation function, this evaluation function could be modified to contemplate costs in the following manner. Let M_S be the merit of the set of k features S, that is, the value originally returned by the function.

$$M_S = EvF(S) \tag{9.10}$$

where EvF is the evaluation function. Let C_S be the average cost of S.

$$C_S = \frac{\sum_{i=1}^{k} C_i}{k} \tag{9.11}$$

where C_i is the cost of feature i. The evaluation function can be modified to become:

$$MC_S = M_S - \lambda C_S \tag{9.12}$$

where λ is a parameter introduced in order to weight the influence of the cost in the evaluation. If λ is 0, the cost is ignored and the method works as the regular feature

selection method that is being modified. If the merit of the set of features M_S is bounded between 0 and 1, and λ is between 0 and 1, the influence of the cost is smaller than the other term. If $\lambda = 1$ both terms have the same influence and if $\lambda > 1$, the influence of the cost is greater than the influence of the other term. Notice that when using a ranker, which selects or evaluates one feature at a time, the cardinality of S is one and then C_S in (9.11) results in the cost of that single feature.

Note that the parameter λ needs to be left as a free parameter because determining the importance of the cost is highly dependent of the domain. For example, in a medical diagnosis, the accuracy cannot been sacrificed in favor of reducing economical costs. On the contrary, in some real-time applications, a slight decrease in classification accuracy is allowed in order to reduce the processing time significantly. An example of this behavior will be shown on a real scenario in Section 9.5.

9.3 Experimental study

The experiment is performed over three blocks of datasets. The main feature of the first block of datasets is that they have intrinsic cost associated to the input features (Hepatitis, Liver, Pima and Thyroid, see Appendix I). Since this type of datasets is not common in the literature, we have opted for including dataset with no cost and generate it randomly. Thus, the second block of datasets consists of four classical datasets selected from the UCI repository (Letter, Magic04, Optdigits, Pendigits, Sat, Segmentation, Wafeform and Yeast, see Appendix I) with a larger number of samples than of features. Finally, the third block consists of five microarray datasets (Brain, CNS, Colon, DLBCL and Leukemia), which are characterized for having a much larger number of features than samples (see Appendix I). Since datasets from second and third block do not have intrinsic cost associated, random cost for their input attributes has been generated. For each feature, the cost was generated as a random number between 0 and 1. As an example, Table 9.1 displays the random costs generated for each feature of *Magic04* dataset.

Overall, the chosen classification datasets are very heterogeneous. They present a variable number of classes, ranging from two to twenty six. The number of samples and features range from single digits to the tens of thousands. Notice that datasets in the first and second blocks have a larger number of samples than features, whilst datasets in the third block have a much larger number of features than samples, which poses

a big challenge for feature selection researchers. This variety of datasets allows for a better understanding of the behavior of the proposed method.

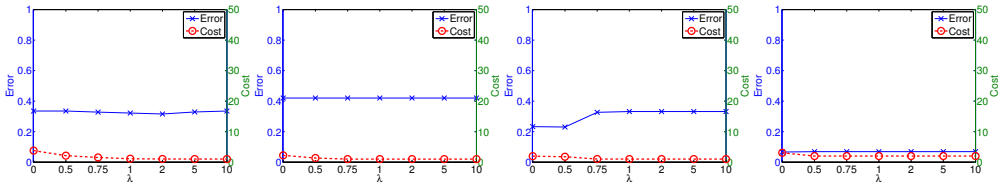
Table 9.1: Random cost for the features of Magic04 dataset.

Feature	Cost
1	0.3555
2	0.2519
3	0.0175
4	0.9678
5	0.6751
6	0.4465
7	0.8329
8	0.1711
9	0.6077
10	0.7329

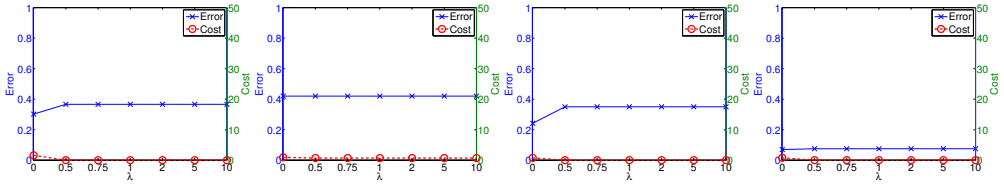
For the sake of brevity, mC-CFS and mC-mRMR will be tested in the following section on these datasets whilst mC-ReliefF will be applied on a real problem in Section 9.5. The goal of these experiments is to study the behavior of the proposed framework under the influence of λ parameter. The performance is evaluated in terms of both the total cost of the selected features and the classification error by a SVM classifier estimated under a 10-fold cross-validation (see Appendix I). It is expected that the larger the λ the lower the cost and the higher the error, because increasing λ gives more weight to the cost at the expense of correlation between features. Moreover, a Kruskal-Wallis statistical test and a multiple comparison test (based on Tukey's honestly significant difference criterion) have been run on the errors obtained. The results of the tests could help the user to choose the adequate value of the λ parameter.

9.4 Experimental results

Figures 9.1, 9.3 and 9.6 show the average cost and error for several values of λ. The solid line with 'x' represents the error (referenced on the left Y axis) and the dashed line with 'o' represents the cost (referenced on the right Y axis). Notice than when $\lambda = 0$, the cost has no influence on the behavior of the method and it behaves as if it was the non-cost version.



(a) Hepatitis CFS (b) Liver CFS (c) Pima CFS (d) Thyroid CFS



(e) Hepatitis mRMR (f) Liver mRMR (g) Pima mRMR (h) Thyroid mRMR

Figure 9.1: Error / cost plots of first block of datasets for cost feature selection with CFS and mRMR.

Figure 9.1 plots the error/cost of the four datasets with cost associated found at the UCI repository. The behavior expected when applying cost feature selection is that the higher the λ, the lower the cost and the higher the error. The results obtained for the first block of datasets, in fact, show that cost value behaves as expected (although the magnitude of the cost does not change too much because these datasets have few features and the set of selected ones is often very similar). The error, however, remains constant in most of the cases. This may happen because these datasets are quite simple and the same set of features is often chosen. The Kruskal-Wallis statistical test run on the results displayed that the errors are not significantly different, except for Pima dataset. This fact can be caused because this dataset has very few expensive features (which are often associated with a higher predictive power), as can be seen on Table 9.2. Therefore, removing them has a greater effect on the classification accuracy.

Fig. 9.2 displays the results of the Kruskal-Wallis statistical test for Pima dataset. The entries in the ANOVA (ANalysis Of VAriance) Table (Figs. 9.2a and 9.2c) are the usual sums of squares (SS), degrees of freedom (df), mean square estimator (MS), chi-square statistic (Chi-sq) and the p value that determines the significance of the chi-square statistic (Prob>Chi-sq). As can be seen, the p value is 9×10^{-6} for the Cost CFS and 2×10^{-4} for the Cost mRMR, as displayed in Figs. 9.2a and 9.2c. This indicates that there exist values significantly different than others. In Figs. 9.2b and 9.2d it is shown which groups of errors are significantly different, information that can be helpful for the user to decide which value of λ utilize. When using Cost CFS,

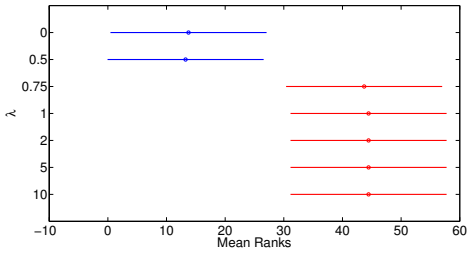
Table 9.2: Cost of the features for Pima dataset (normalized to 1).

Feature	Cost
1	0.0100
2	0.7574
3	0.0100
4	0.0100
5	0.9900
6	0.0100
7	0.0100
8	0.0100

Kruskal-Wallis ANOVA Table

Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	13.557,8	6	2.259,63	33,28	9,28E-06
Error	14.554,7	63	231,03		
Total	28.112,5	69			

(a) ANOVA Table (mC-CFS).

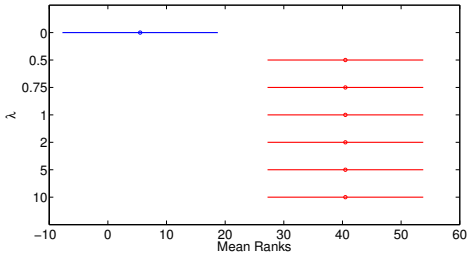


(b) Graph of multiple comparison (mC-CFS).

Kruskal-Wallis ANOVA Table

Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	10.500,0	6	1.750,00	25,91	2,00E-04
Error	17.467,5	63	277,26		
Total	27.967,5	69			

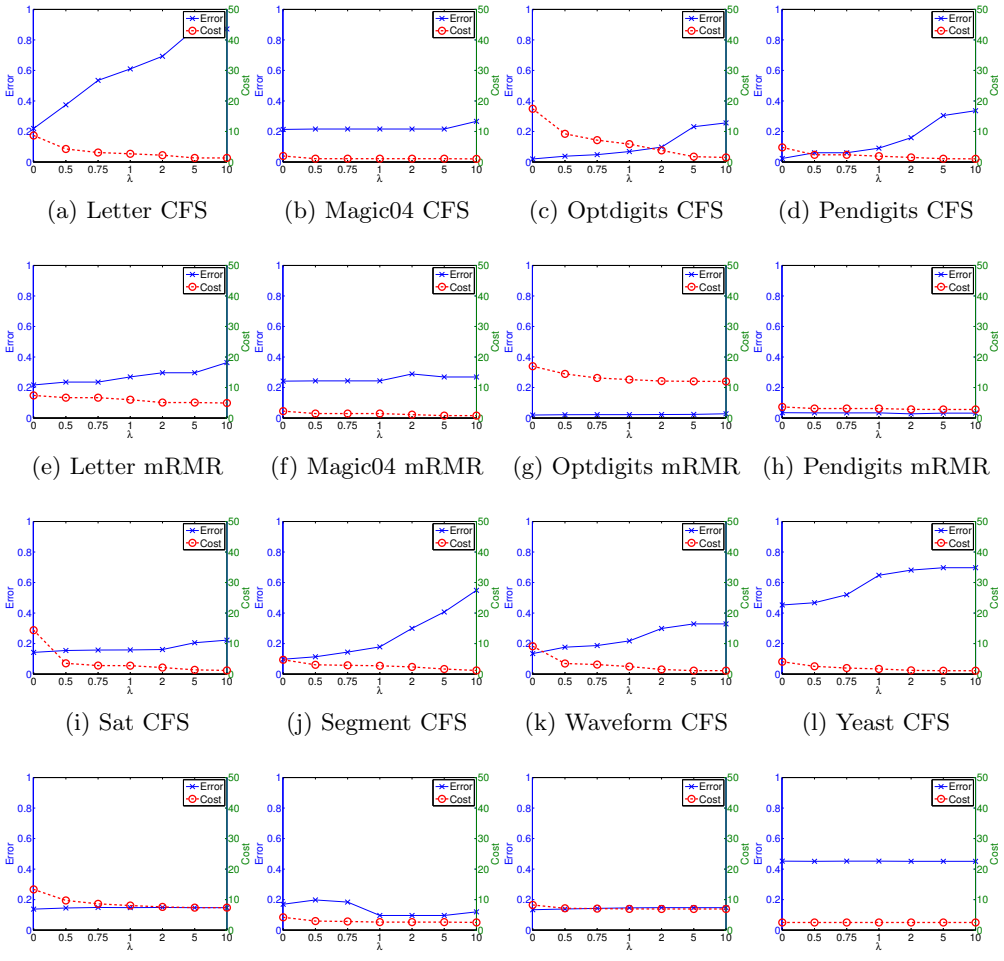
(c) ANOVA Table (mC-mRMR).



(d) Graph of multiple comparison (mC-mRMR).

Figure 9.2: Kruskal-Wallis statistical test results of Pima dataset.

λ can be 0.5 (which means decreasing the cost) without significantly increasing the error, whilst when using Cost mRMR, a reduction in cost can not be achieved without worsening the error measure. For Cost mRMR, when λ is 0 (and hence, the cost is not taken into account), the second feature is selected, which has a high cost (see Table 9.2). However, when the method is forced to decrease the cost (by increasing the value of λ), this feature is not selected anymore and prevents the classifier to obtain a high prediction accuracy. Something similar happens with Cost CFS, where the second feature is selected for λ values 0 and 0.5 and removed for the remaining values due to its high cost.



(a) Letter CFS (b) Magic04 CFS (c) Optdigits CFS (d) Pendigits CFS

(e) Letter mRMR (f) Magic04 mRMR (g) Optdigits mRMR (h) Pendigits mRMR

(i) Sat CFS (j) Segment CFS (k) Waveform CFS (l) Yeast CFS

(m) Sat mRMR (n) Segment mRMR (o) Waveform mRMR (p) Yeast mRMR

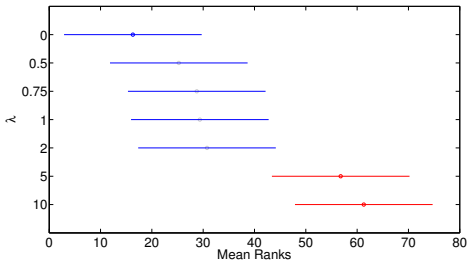
Figure 9.3: Error / cost plots of second block of datasets for cost feature selection with CFS and mRMR.

The error/cost graphs of the second block of datasets are displayed in Fig. 9.3. It can be seen how cost decreases, according to expected, and how, contrary to first block, error usually raises when λ increases. In the cases when error raises monotonically (see Figures 9.3a or 9.3j, for example), there exist significant error changes (p-values are close to zero), therefore the user has to make a choice to find an appropriate trade-off between cost and error. On the other hand, there are some other cases, such as Sat dataset with Cost CFS (see Fig. 9.4) where with $\lambda = 2$, the error is not significantly worse but a significant reduction in cost (Fig. 9.5) is achieved.

Kruskal-Wallis ANOVA Table

Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	16.989,8	6	2.831,63	41,07	2,80E-07
Error	11.553,7	63	183,39		
Total	28.543,5	69			

(a) ANOVA Table.



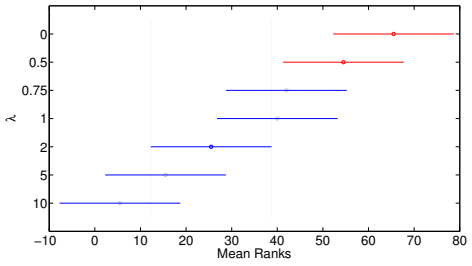
(b) Graph of multiple comparison.

Figure 9.4: Kruskal-Wallis error statistical test of Sat dataset with mC-CFS.

Kruskal-Wallis ANOVA Table

Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	27.235	6	4539.17	67.56	1,03E-06
Error	580	63	9.21		
Total	27.815	69			

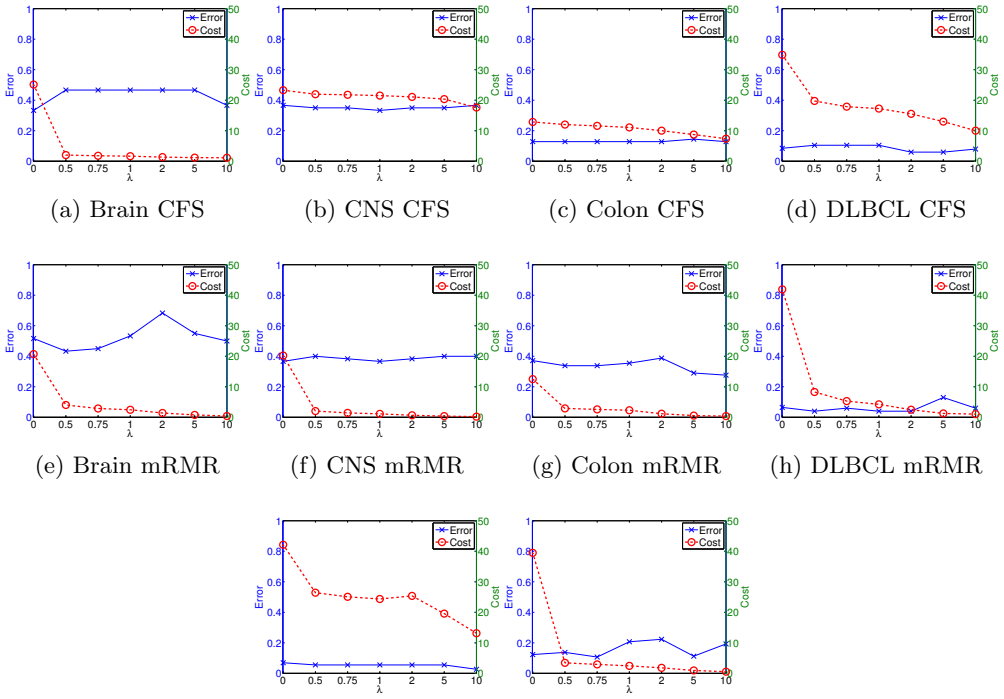
(a) ANOVA Table.



(b) Graph of multiple comparison.

Figure 9.5: Kruskal-Wallis cost statistical test results of Sat dataset with mC-CFS.

Finally, Fig. 9.6 presents the results for the third block of datasets, corresponding with the well-known DNA microarray domain, with much more features than samples. As expected, the cost decreases as λ increases, and since these datasets have a larger number of input attributes than the ones in previous blocks, the cost experiments also a larger variability (see, for example, Figs. 9.6h, 9.6j). For instance, for the DLBCL dataset with Cost mRMR, we can choose $\lambda = 10$, as the errors are not significantly different (see Fig. 9.7) and the cost for $\lambda = 10$ is significantly lower than the one for the four first λ (0, 0.5, 0.75 and 1) (see Fig. 9.8).



(a) Brain CFS (b) CNS CFS (c) Colon CFS (d) DLBCL CFS

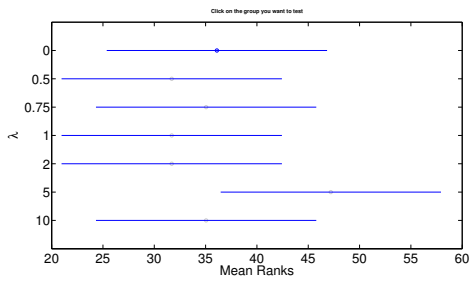
(e) Brain mRMR (f) CNS mRMR (g) Colon mRMR (h) DLBCL mRMR

(i) Leukemia CFS (j) Leukemia mRMR

Figure 9.6: Error / cost plots on third block of datasets for cost feature selection with CFS and mRMR.

Kruskal-Wallis ANOVA Table					
Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	1.809,8	6	301,63	6,83	0,3371
Error	16.481,3	63	261,61		
Total	18.291,0	69			

(a) ANOVA Table.

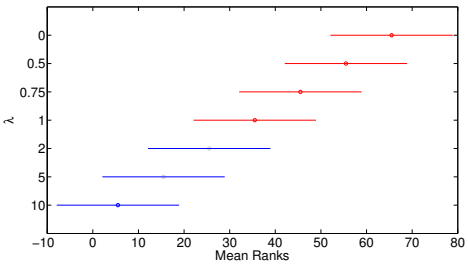


(b) Graph of multiple comparison.

Figure 9.7: Kruskal-Wallis error statistical test of DLBCL dataset with mC-mRMR.

Kruskal-Wallis ANOVA Table					
Source	SS	df	MS	Chi-sq	Prob>Chi-sq
Columns	28.000,0	6	4.666,67	67,61	1,27E-12
Error	577,5	63	9,17		
Total	28.577,5	69			

(a) ANOVA Table.



(b) Graph of multiple comparison.

Figure 9.8: Kruskal-Wallis cost statistical test of DLBCL dataset with mC-mRMR.

Notwithstanding, the behavior of the error, in some cases, and contrary to expected, remains almost constant (see, for instance, Figs. 9.6c or 9.6f). The reason why the error is not raising can be two-fold:

- On the one hand, it is necessary to remind that the proposed framework is being tested using filter feature selection methods. This approach has the benefit of being fast and computationally inexpensive. This characteristic of filters can cause that the selected features, according to particular criteria, would not be the more suitable for a given classifier to obtain the highest accuracy. Therefore, forcing a filter to select features according to another criterion rather than correlation (or the one used for each particular filter) may cause the selection of features to be more suitable for minimizing classification error. For example, in Chapter 3, a synthetic dataset called Monk3 is dealt with. Among others, this dataset contains three relevant features. However, some classifiers obtain a better classification accuracy when filters only had selected two relevant features than when selecting the three relevant ones. This fact demonstrates that the behavior of some filters is somewhat unpredictable and not always the one expected.

- On the other hand, it has to be noted that DNA microarray datasets are a difficult challenge for feature selection methods, due to the enormous amount of features they present. In fact, the filters evaluated in these experiments are usually retaining a maximum of 2% of the original features. Therefore, irregular results are expected with such an important reduction in number of features.

It is also worth reflecting on the behavior of CFS and mRMR filters. Studying in detail graphs in Figure 9.6, one can see that in general, mRMR achieves the lowest

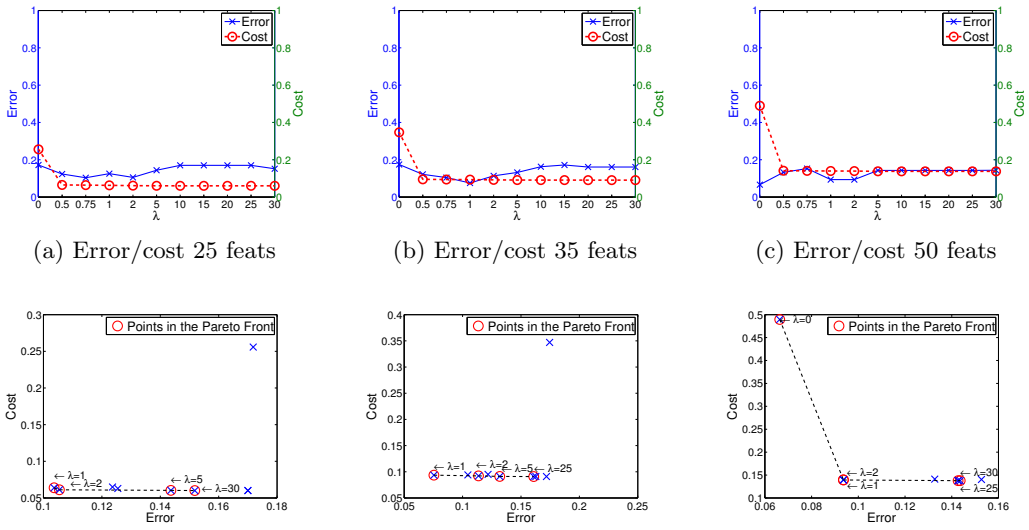
cost at the expense of a higher error than CFS. Therefore, it is the user who has to decide which filter to use depending on the degradation in the error he/she is willing to assume.

9.5 Case of study: a real life problem

In this section it is presented a real-life problem where the cost, in the form of computational time, needs to be reduced. *Evaporative dry eye* (EDE) is a symptomatic disease which affects a wide range of population and has a negative impact on their daily activities, such as driving or working with computers (see Chapter 5). Its diagnosis can be achieved by several clinical tests, one of which is the analysis of the interference pattern and its classification into one of the four categories defined by Guillon (1998) for this purpose. A methodology for automatic *tear film lipid layer* (TFLL) classification into one of these categories has been developed (Remeseiro et al., 2011), based on color texture analysis. The co-occurrence features technique (Haralick et al., 1973), as a texture extraction method, and the Lab color space provide the highest discriminative power from a wide range of methods analyzed. This methodology needs to be fast in order to be applied as a clinical routine. However, the best accuracy results were obtained at the expense of a too long processing time (38 seconds) because many features had to be computed. This fact makes the methodology unfeasible for practical applications and prevents its clinical use. Reducing processing time is a critical issue in this application which should work in real-time in order to be used in the clinical routine. Therefore, the proposed mC-ReliefF is applied in an attempt to decrease the number of features and, consequently, the computational time without compromising the classification performance. Note that this processing time was recorded on an Intel®Core^{TM}i5 CPU 760 @ 2.80GHz with RAM 20 GB.

So, the adequacy of the proposed framework using ReliefF is tested on the real problem of TFLL classification using the dataset VOPTICAL_I1 (VOPTICAL_I1, n.d.). This dataset consists of 105 images (samples) belonging to the four Guillon's categories (classes). The methodology for TFLL classification proposed by Remeseiro et al. (2011) consists of extracting the *region of interest* (ROI) of an input image, and analyzing it based on color and texture information. Thus, the ROI in the RGB color space is transformed to the Lab color space and the texture of its three components of color (L, a and b) is analyzed. For texture analysis, the co-occurrence features method generates a set of *grey level co-occurrence matrices* (GLCM) for an specific distance and extracts

14 statistical measures from their elements. Then, the mean and the range of these 14 statistical measures are calculated across matrices and so a set of 28 features is obtained. Distances from 1 to 7 in the co-occurrence features method and the 3 components of the Lab color space are considered, so the size of the final descriptor obtained from an input image is: 28 features $\times$ 7 distances $\times$ 3 components = 588 features. Notice that the cost for obtaining these features is not homogeneous. Features are vectorized in groups of 28 related to distances and components in the color space, where the higher the distance, the higher the cost. Plus, each group of 28 features corresponds with the mean and range of the 14 statistical measures calculated across the GLCMs. Among these statistical measures, it was shown that computing the so-called 14^{th} statistic takes around 75% of the total time. Therefore, we have to deal with a dataset with a very variable cost (in this case, computational time) associated to the input features.



(a) Error/cost 25 feats (b) Error/cost 35 feats (c) Error/cost 50 feats

(d) Pareto front 25 feats (e) Pareto front 35 feats (f) Pareto front 50 feats

Figure 9.9: Error / cost plots (top) and Pareto front (bottom) of VOPTICAL_I1 dataset for different values of λ, and different number of selected features (25, 35 and 50)

Figure 9.9 (top) shows the average error and cost after performing a 10-fold cross-validation for VOPTICAL_I1 dataset for different values of λ, for three different sets of features. As expected, when λ increases, the cost decreases and the error either raises or is maintained (see Section 9.2.4). Regarding the different subsets of features, the larger the number of features, the higher the cost. The Kruskal-Wallis statistical test run on the results demonstrated that there are no significant differences among the errors achieved using different values of λ, whilst using a $\lambda > 0$ decreases significantly the cost. This situation happens when retaining 25, 35 and 50 features.

Trying to shed light on the issue of which value of λ is better for the problem at hand, the Pareto front (Teich, 2001) for each alternative is showed in Figure 9.9 (bottom). In multi-objective optimization, the Pareto front is defined as the border between the region of feasible points, for which all constraints are satisfied, and the region of infeasible points. In this case, solutions are constrained to minimize classification error and cost. In Figure 9.9 (bottom), points (values of λ) in the Pareto front are marked with a red circle. All those points are equally satisfying the constraints, and it is decision of the users if they prefer to minimize either the cost or the classification error. On the other hand, choosing a value of λ outside the Pareto front would imply to chose a worse solution than any in the Pareto front.

Table 9.3: Mean classification error(%), time (milliseconds), and number of features in the union of the 10 folds for the Pareto front points. Best error and time are marked in bold face.

Feats	λ	Error	Time	Feats union
25	0.75	10.36	208.68	30
	2	10.55	206.46	30
	5	14.36	197.22	29
	30	15.18	**174.35**	26
35	1	7.55	306.53	43
	2	11.36	328.24	46
	5	13.18	273.11	39
	25	16.09	249.92	36
50	0	**6.64**	1377.04	82
	1	9.36	397.70	55
	2	9.36	412.14	57
	25	14.27	364.45	51
	30	14.36	364.45	51

Table 9.3 reports the classification error and cost (in the form of time) for all the Pareto front points. Notice that as a 10-fold cross-validation was performed, the final subset of selected features is the union of the features selected in each fold, and that is why the number of features in column 5 differs from the one in the first column. Even so, the reduction in the number of features is considerable. As expected, the higher the λ, the higher the error and the lower the time. The best result in terms of classification error was obtained with $\lambda = 0$ when retaining 50 features per fold. In turn, the lowest time was obtained with $\lambda = 30$ when retaining 25 features per fold,

but at the expense of increasing the error in 8.54%. In this situation, the authors think that it is better to choose a trade-off between cost and error. The error obtained with $\lambda = 1$ when retaining 35 features is 7.55%, which is slightly higher than the best one but no significant differences were found between them. With this combination the time required is 306.53 milliseconds, which although is not the lowest time, it is still under 1 second. The time required by previous approaches which deal with TFLL classification prevented their clinical use because they could not work in real time, since extracting the whole set of features took 38 seconds. Thus, since this is a real-time scenario where reducing the computing time is a crucial issue, having a processing time under 1 second leads to a significant improvement. In this manner, the methodology for TFLL classification could be used in the clinical routine as a support tool to diagnose EDE.

9.6 Summary

In this chapter a new framework for cost-based feature selection was proposed. The objective was to solve problems where it is not only interesting to minimize the classification error, but also to reduce costs that may be associated to input features. This framework consists of adding a new term to the evaluation function of any filter feature selection method so that it is possible to reach a trade-off between a filter metric (e.g. correlation or mutual information) and the cost associated to the selected features. A new parameter, called λ, is introduced in order to adjust the influence of the cost into the evaluation function, allowing the users fine control of the process according to their needs.

In order to test the adequacy of the proposed framework, three well-known and representative filters are chosen: CFS, mRMR and ReliefF. Experimentation is executed over a broad suite of different datasets including one real-life problem. Results after performing classification with a SVM displayed that the approach is sound and allows the user to reduce the cost without compromising the classification error significantly, which can be very useful in fields such as medical diagnosis or real-time applications.

CHAPTER 10

Distributed and parallel feature selection

As can be seen in previous chapters, feature selection is usually applied in a centralized manner, i.e. a single learning model to solve a given problem. However, if the data is distributed, feature selection may take advantage of processing multiple subsets in sequence or concurrently. There are several ways in which a feature selection task could be distributed (Bramer, 2007):

(a) If all the data is together in one very large dataset, we can distribute it on several processors, run an identical feature selection algorithm on each one and combine the results.

(b) The data may inherently be in different datasets on different locations, for example in different parts of a company or even in different cooperating organizations. As for the previous case, we could run an identical feature selection on each one and combine the results.

(c) An extreme case of a large data volume is *streaming data* arriving in effectively a continuous infinite stream in real time. If the data is all coming to a single source, different parts of it could be processed by different processors acting in parallel. If it is coming into several different processors, it could be handled in a similar way to (b).

(d) An entirely different situation arises where we have a dataset that is not particularly large, but we wish to generate several or many different feature selection methods from it and then combine the results by some kind of voting system in order to learn unseen instances. In this case we might have the whole dataset on a single processor, accessed by different feature selection methods (possibly identical or possibly different) accessing all or part of the data. This approach is known as *ensemble learning* and it has been discussed in Chapter 8.

The first part of this chapter will be focused on the first category of application, i.e. all the data is together in one large dataset, a part of which can be distributed in different processors, then run an identical feature selection algorithm on each one and combine the results. This approach can be useful since most existing feature selection algorithms do not scale well and their efficiency significantly deteriorates or even becomes inapplicable when dealing with large-scale data (see Chapter 6). There are two main techniques for partitioning and distributing data: vertically, i.e. by features, and horizontally, i.e. by samples. Distributed learning has been used to scale up datasets that are too large for batch learning in terms of samples (Chan & Stolfo, 1993; Ananthanarayana, Subramanian, & Murty, 2000; Tsoumakas & Vlahavas, 2002). While not common, there are some other developments that distribute the data by features (McConnell & Skillicorn, 2004; Skillicorn & McConnell, 2008). When the data come distributed in origin, vertical distribution is solely useful where the representation of data could vary along time by adding new attributes.

Both alternatives will be explored in this chapter. Horizontal partitioning is especially suitable when dealing with datasets with a high number of samples whilst vertical partitioning is more appropriate to datasets with a high number of features. The experimental results on several different datasets demonstrate that our proposal can improve the performance of original feature selection methods and show important savings in running times.

Then, this chapter will also attempt to deal with the third category of application (the (c) type above). In this manner, the vertical partitioning will be modified so as to combine the learned models obtained from each node in an incremental manner, paving the way to the application of the methodology to streaming data.

10.1 General methodology

As stated above, parallel and distributed feature selection have not been deeply explored yet. A method is said to be *parallel* when the processes are executed on different cores of one or several nodes connected by a high-speed network at the same location (e.g. a cluster). On the other hand, a method is said to be *distributed* if the data are allocated in different locations and there is very low interaction among the processes (e.g. grid computing). The method that we present in this chapter can be seen as potential distributed, since it was designed to be applied in a distributed manner although, for technical reasons, the experiments were executed on a single machine.

So, in this chapter we present a distributed filter approach trying to improve standard accuracy results as well as reducing the running time. Our proposal consists of performing several fast filters over several partitions of the data, combined afterwards into a single subset of features. Thus, we divide each dataset D into several small disjoint subsets D_i. The feature selection algorithm is applied to each one of these subsets, and a selection S_i is generated for each subset of data. After all the small datasets D_i were used, (which could be done in parallel, as all of them are independent from each other), the combination method builds the final selection S as the result of the feature selection process. To sum up, there are three main steps in this methodology:

1. Partition of the datasets.
2. Application of feature selection to the subsets.
3. Combination of the results.

The partition of the dataset consists of dividing the original dataset into several disjoint subsets of approximately the same size that cover the full dataset. As mentioned in the introduction of this chapter, the partition can be done vertically or horizontally. Therefore, the data is split by assigning groups of k data to each subset, where the number of data k in each subset needs to be determined. After having several small disjoint datasets D_i, the feature selection method will be applied to each of them, returning a selection S_i for each subset of data. Finally, to combine the results, a merging procedure using a classifier will be executed. At the end, the final selection S is applied to the training and test sets in order to obtain the ultimate classification accuracies.

10.2 Horizontal partitioning

In this section we present a parallel filter approach by partitioning the data horizontally. The methodology consists of applying filters over several partitions of the data, obtaining several subsets that are combined in the final step into a single subset of features. The feature selection algorithm (see pseudo-code in Algorithm 10.1) is applied to each dataset in several iterations or rounds. This repetition ensures capturing enough information for the combination stage. At each round, the first step is the partition of the dataset, which consists of randomly dividing the original training dataset into

several disjoint subsets of approximately the same size that cover the full dataset (see Algorithm 10.1, line 3). As mentioned above, the partition will be done horizontally (i.e. by samples). Then, the chosen filter algorithm is applied to each subset separately and the features selected to be removed receive a vote (Algorithm 10.1, lines 5 - 8). At that point, a new partition is performed and another round of votes is accomplished until reaching the predefined number of rounds. Finally, the features that have received a number of votes above a certain threshold are removed. Therefore, a unique set of features is obtained to train a classifier C and to test its performance over a new set of samples (test dataset).

Determining the threshold of votes required to remove a feature is not an easy-to-solve question, since it depends on the given dataset. Therefore, we have developed our own automatic method which calculates this threshold, outlined in Algorithm 10.1, lines 9-19. The best value for the number of votes is estimated from its effect on the training set, but due to the large size of the datasets, the complete training set was not used, only 10% was employed.

Following the recommendations exposed by de Haro García (2011), the selection of the number of votes must take into account two different criteria: the training error and the percentage of features retained. Both values must be minimized to the maximum possible extent, by minimizing the fitness criterion $e[v]$:

$$e[v] \leftarrow \alpha \times error + (1 - \alpha) \times featPercentage \tag{10.1}$$

where α is a parameter in [0,1] which measures the relative relevance of both values and v is any possible value for the threshold. This parameter was set to $\alpha = 0.75$ as suggested by de Haro García (2011), giving more influence to the classification error. Because of performing a horizontally partition of the data, the maximum number of votes is the number of rounds r times the number of subsets s. Since in some cases this number is in the order of thousands, instead of evaluating all the possible values for the number of votes we have opted for delimiting it into an interval $[minVote, maxVote]$ computed used the mean and standard deviation (see lines 9-12 in Algorithm 10.1). In order to reduce the searching time, the increment within the interval was set to 5.

Algorithm 10.1: Pseudo-code for horizontal partitioning

Data: $\mathbf{D}_{(m \times n+1)} \leftarrow$ labeled training dataset with m samples and n input features

X $\leftarrow$set of features, X $= \{x_1, \ldots, x_n\}$
$s \leftarrow$ number of submatrices of $\mathbf{D}$ with p samples
$r \leftarrow$ number of rounds
$\alpha \leftarrow 0.75$

Result: S $\leftarrow$ subset of features \S $\subset$ X

`/* Obtaining a vector of votes for discarding features */`

initialize the vector votes to 0, |vector|=n
for *each round* **do**
    Split $\mathbf{D}$ randomly into s disjoint submatrices
    for *each submatrix* **do**
        apply a feature selection algorithm
        F $\leftarrow$ features selected by the algorithm
        E $\leftarrow$ features eliminated by the algorithm \E $\cup$ F = X
        increment one vote for each feature in E
    end
end

`/* Obtain threshold of votes, Th, to remove a feature */`

$avg \leftarrow$ compute the average of the vector votes
$std \leftarrow$ compute the standard deviation of the vector votes
$minVote \leftarrow$ minimum threshold considered (computed as $avg - 1/2std$)
$maxVote \leftarrow$ maximum threshold considered (computed as $avg + 1/2std$)
$\mathbf{z} \leftarrow$ submatrix of $\mathbf{D}$ with only 10% of samples
for $v \leftarrow$ *minVote to maxVote with increment 5* **do**
    $\mathrm{F}_{th} \leftarrow$ subset of selected features (number of votes $< v$)
    $error \leftarrow$ classification error after training $\mathbf{z}$ using only features in F_{th}
    $featPercentage \leftarrow$ percentage of features retained $\left(\frac{|\mathrm{F}_{th}|}{|\mathrm{X}|} \times 100\right)$
    $e[v] \leftarrow \alpha \times error + (1 - \alpha) \times featPercentage$
end
$Th \leftarrow min(e)$, Th is the value which minimizes the error e
S $\leftarrow$ subset of features after removing from X all features with a number of votes $\geq Th$

10.2.1 Experimental setup

In order to test the proposed distributed filter approach, we have selected six benchmark datasets which can be consulted in Appendix I, Section I.2.2: Connect4, Isolet, Madelon, Ozone, Spambase and MNIST. These datasets can be considered representative of problems from medium to large size, since the horizontal distribution is not suitable for small-sample datasets. Those datasets originally divided to training and test sets were maintained, whereas, for the sake of comparison, datasets with only training set were randomly divided using the common rule 2/3 for training and 1/3 for testing. For calculating the number of packets s, the proportion between number of samples and number of features is computed, with the constraint of having, at least, three packets per datasets. The following rules are considered, trying to have a enough number of samples in each packet:

1. If $proportion \geq 10\,000$, $p = 10\,000$ samples per packet.

2. If $proportion \geq 1000$, $p = 1000$ samples per packet.

3. If $proportion \geq t$ being $t \in [0.5, 1000)$, $p = n \times t$ in which n is the number of features.

According to these rules, the number of packets (s) to partition the dataset in each round is displayed in Table 10.1.

Table 10.1: Number of packets (s) for the datasets used with the horizontal partition

Dataset	Packets
Connect4	45
Isolet	5
Madelon	3
Ozone	11
Spambase	5
Mnist	5

10.2.2 Experimental results

In this section the experimental results over the six benchmark datasets described above will be presented and discussed. The distributed approach proposed in this chapter can be used with any filter method. In these experiments, five well-known filters, based on different metrics where chosen (all of them available in the Weka tool). While three of the filters return a feature subset (CFS, Consistency-based and INTERACT), the other two (ReliefF and Information Gain) are ranker methods, so it is necessary to establish a threshold in order to obtain a subset of features. In this case we have opted for retaining the c top features, being c the number of features selected by CFS. It is also worth noting that although most of the filters work only over nominal features, the discretization step is done by default by Weka, working as a black box for the user. A detailed description of the filters can be found in Chapter 2.

Our distributed approach is compared with the centralized standard approach for each method. To distinguish between both approaches, a "C" (centralized) or a "D" (distributed) was added to the name of the filter. In the case of the distributed approach, five rounds (r in Algorithm 10.1) have been executed. To ensure reliable results, a hold-out validation was applied and repeated 5 times, using the common partition 2/3 for training and 1/3 for testing.

10.2.2.1 Number of features selected

Tables 10.2 and 10.3 show the number of features selected (in average) by the centralized and distributed approach, respectively. Notice that the number of features selected by the distributed approach (Table 10.3) depends on the classifier, because of the stage devoted to finding the threshold of votes (see Algorithm 10.1, lines 14-18).

As can be seen in the tables, there are no significant differences between the number of features selected by both approaches. Therefore, we can affirm that applying the distributed approach does not imply a larger selection of features.

Table 10.2: Number of features selected by the centralized approach

	Connect4	Isolet	Madelon	Ozone	Spambase	Mnist
Full set	42	617	500	72	57	717
CFS-C	7	186	18	20	19	61
IG-C	7	186	18	20	19	61
ReliefF-C	7	186	18	20	19	61
INT-C	36	56	23	16	26	40
Cons-C	39	11	22	16	20	18

Table 10.3: Number of features selected by the distributed approach

		Connect4	Isolet	Madelon	Ozone	Spambase	Mnist
C4.5	CFS-D	9	132	14	14	19	90
	IG-D	9	142	15	13	19	79
	ReliefF-D	9	146	18	12	17	74
	INT-D	9	75	14	12	19	72
	Cons-D	9	32	15	6	18	50
NB	CFS-D	9	132	14	14	20	90
	IG-D	9	142	15	13	19	79
	ReliefF-D	9	146	18	13	19	74
	INT-D	9	75	14	12	19	72
	Cons-D	10	32	15	6	20	50
k-NN	CFS-D	9	131	14	14	19	90
	IG-D	10	142	15	13	19	79
	ReliefF-D	11	145	18	12	17	74
	INT-D	10	78	14	12	19	72
	Cons-D	10	34	15	6	18	50
SVM	CFS-D	9	137	14	14	19	90
	IG-D	9	142	15	13	20	79
	ReliefF-D	9	149	18	12	17	74
	INT-D	9	70	14	12	19	72
	Cons-D	9	28	15	6	18	50

10.2.2.2 Classification accuracy results

This section presents the classification accuracy obtained by C4.5, naive Bayes, k-NN and SVM classifiers both with the centralized and distributed approach. After the 5 repetitions, a Kruskal-Wallis test (see Appendix I, Section I.4) was applied to check if there were significant differences for a level of significance $\alpha = 0.05$. Then, a multiple comparison procedure (Tukey's) was applied to find the simplest approach in which the accuracy is not significantly different from the approach with the best accuracy (labeled with a cross in Table 10.4). Notice that for each one of the 5 repetitions, 5 rounds of the partitioning and filtering steps are executed. Table 10.4 reports the test classification accuracies obtained with C4.5, naive Bayes, k-NN and SVM (see Appendix I, Section I.5). The best result for each dataset and classifier is highlighted in bold face.

As expected, the results are very variable depending on the dataset and the classifier. However, in terms of average (last column), the best result for each classifier is obtained by a distributed approach. In particular, ReliefF-D combined with C4.5 achieves the highest accuracy, outperforming in at least 3.5% the best results for the remaining classifiers. For some datasets (Connect4, Isolet and Madelon) the highest accuracies are obtained by centralized approaches, whereas for others (Spambase and MNIST) the best results are accomplished by a distributed method. The important conclusion, however, is that by distributing the data there is not a significant degradation in classification accuracy. In fact, in some cases the accuracy is improved. It is worth mentioning, for example, the case of MNIST, in which distributed consistency-filter combined with k-NN classifier reports 94.08% accuracy whilst the same filter method in the standard centralized approach degrades significantly its performance (85.21% accuracy).

10.2.2.3 Runtime

Table 10.5 reports the runtime of the feature selection algorithms, both in centralized and distributed manners. In the distributed approach, considering that all the subsets can be processed at the same time, the time displayed in the table is the maximum of the times required by the filter in each subset generated in the partitioning stage. In these experiments, all the subsets were processed in the same machine, but the proposed algorithm could be executed in multiple processors. Please note that this filtering time is independent of the classifier chosen. The lowest time for each dataset is emphasized in bold font.

Table 10.4: Test classification accuracy for horizontal partitioning.

		Connect4	Isolet	Madelon	Ozone	Spambase	Mnist	Average
C4.5	CFS-C	70.38	**80.61**†	75.58	95.63†	91.91†	85.78	83.31
	CFS-D	72.66†	80.05†	77.74†	95.42†	91.80†	87.28 †	84.16
	IG-C	70.70	78.90†	80.15†	95.59†	91.52†	86.56†	83.90
	IG-D	72.60†	76.93†	77.80†	95.74†	91.46†	86.70†	83.53
	ReliefF-C	70.98	78.31†	**84.49**†	95.52†	88.53	85.61	83.90
	ReliefF-D	72.08†	77.63†	83.64†	95.45†	90.63†	86.18†	**84.26**
	INT-C	74.79†	76.61†	77.98†	95.59†	91.40†	85.08	83.57
	INT-D	72.44†	77.18†	75.63	95.93†	**91.97**†	87.40†	83.57
	Cons-C	**74.89**†	57.03	79.20†	95.59†	90.89†	84.47	80.34
	Cons-D	72.45†	69.54	78.74†	**96.56**†	90.98†	**88.25**†	80.34
NB	CFS-C	65.74†	72.74†	69.05†	79.85†	78.54†	72.13†	73.00
	CFS-D	**66.55**†	**73.63**†	69.36†	81.45†	80.64†	73.51†	74.09
	IG-C	65.75†	71.19†	69.23†	75.78	87.97†	69.80	73.15
	IG-D	66.22†	66.59†	69.63†	79.70†	**87.99**†	69.51	73.27
	ReliefF-C	65.72†	66.75†	**69.57**†	67.37	71.65	69.96	68.50
	ReliefF-D	66.04†	60.35	69.54†	68.84	80.53†	70.58†	69.31
	INT-C	59.38	66.39†	68.92†	81.04†	82.14†	68.44†	71.05
	INT-D	66.21†	68.35†	69.44†	84.33†	82.02†	71.26†	73.53
	Cons-C	59.07	42.68	69.00†	81.09†	84.19†	73.26†	68.21
	Cons-D	66.19†	60.70	69.34†	**91.36**†	85.79†	**75.17**†	**74.76**
k-NN	CFS-C	62.31†	55.31†	68.41	95.28†	87.71†	87.38	76.06
	CFS-D	65.38†	53.94†	75.44†	94.97†	88.11†	90.41†	78.04
	IG-C	62.91†	54.27†	80.36†	95.16†	88.15†	89.46†	78.38
	IG-D	65.09†	**60.52**†	78.34†	**95.43**†	88.33†	90.37†	79.68
	ReliefF-C	60.99	57.07†	**91.16**†	95.17†	84.07	89.49†	79.66
	ReliefF-D	**66.66**†	54.07†	89.89†	94.60†	87.28†	90.21†	**80.45**
	INT-C	61.74	48.85	74.82	95.22†	88.43†	85.49	75.76
	INT-D	65.16†	49.24	77.18†	95.10†	**88.74**†	90.69†	77.68
	Cons-C	61.28	54.16†	76.12†	95.14†	86.81†	85.21	76.45
	Cons-D	65.21†	60.35†	79.03†	93.05†	87.13†	**94.08**†	79.80
SVM	CFS-C	**65.72**†	**82.04**†	64.16†	**97.16**†	87.67†	80.76†	79.58
	CFS-D	**65.72**†	81.10†	64.82†	**97.16**†	87.80†	**81.75**†	**79.73**
	IG-C	**65.72**†	81.48†	64.68†	**97.16**†	87.64†	78.64†	79.22
	IG-D	**65.72**†	77.34†	65.35†	**97.16**†	87.99†	79.06†	78.80
	ReliefF-C	**65.72**†	82.00†	65.84†	**97.16**†	85.23	74.86	78.47
	ReliefF-D	**65.72**†	79.03†	**65.85**†	**97.16**†	86.91†	75.15	78.30
	INT-C	**65.72**†	71.92†	63.70†	**97.16**†	**88.22**†	78.25†	77.50
	INT-D	**65.72**†	73.64†	65.01†	**97.16**†	88.07†	80.67†	78.37
	Cons-C	**65.72**†	28.22	64.22†	**97.16**†	86.99†	75.19	69.54
	Cons-D	**65.72**†	53.53	64.81†	**97.16**†	86.70†	80.02†	74.65

Table 10.5: Runtime (hh:mm:ss) for the feature selection methods tested

	Connect4	Isolet	Madelon	Ozone	Spambase	Mnist
CFS-C	00:01:40	00:04:10	00:00:36	00:00:10	00:00:12	00:29:47
CFS-D	00:00:10	00:01:17	**00:00:25**	00:00:08	00:00:06	00:04:17
IG-C	00:01:37	00:02:51	00:00:41	00:00:09	00:00:11	00:24:11
IG-D	**00:00:04**	**00:00:54**	00:00:29	00:00:09	00:00:05	00:03:55
ReliefF-C	00:28:00	00:09:13	00:01:02	00:00:14	00:00:21	08:26:53
ReliefF-D	00:00:11	00:01:43	00:00:40	00:00:08	00:00:04	00:22:26
INT-C	00:01:52	00:03:16	00:00:40	00:00:09	00:00:13	00:52:25
INT-D	00:00:11	00:01:10	00:00:31	00:00:08	00:00:04	00:03:19
Cons-C	00:06:08	00:04:05	00:00:52	00:00:11	00:00:14	01:42:43
Cons-D	00:00:10	00:01:20	**00:00:25**	**00:00:06**	**00:00:02**	**00:03:17**

As expected, the advantage of the distributed approach in terms of execution time over the standard method is significant. The time is reduced for all datasets and filters, except for Ozone with Information Gain filter, in which it is maintained. It is worth mentioning the important reductions when the dimensionality of the dataset grows. For Mnist dataset, which has 717 features and 40000 training samples, the reduction is more than notable. For ReliefF filter, the processing time is reduced from more than 8 hours to 22 minutes, proving the adequacy of the distributed approach when dealing with large datasets.

For the distributed approach, it is necessary to take into account the time required to find the threshold to build the final subset of features. This time depends highly on the classifier, as can be seen in Table 10.6. In the table it is visualized the average runtime for each filter, classifier and dataset. It is easy to note that the classifier which requires more execution time is SVM whilst the one which requires the shortest time is naive Bayes. Except for Mnist dataset with SVM classifier, this time is usually in the order of seconds or a couple of minutes, so it is insignificant when compared with the time required by any of the centralized algorithms showed above. Moreover, if the user would prefer to save this time, it is possible to establish a fixed threshold in an attempt to avoid this specific calculation. In light of these results, we can conclude that our horizontal distributed proposal performed successfully, since the running time was considerably reduced and the accuracy did not drop to inadmissible values. In fact, our approach is able to match and in some cases even improve the standard algorithms applied to the non-partitioned datasets.

Table 10.6: Runtime (hh:mm:ss) for obtaining the threshold of votes.

		Connect4	Isolet	Madelon	Ozone	Spambase	Mnist
C4.5	CFS-D	00:01:46	00:00:28	00:00:02	00:00:02	00:00:07	00:05:01
	IG-D	00:01:42	00:00:24	00:00:02	00:00:02	00:00:06	00:00:29
	ReliefF-D	00:01:42	00:00:28	00:00:02	00:00:02	00:00:06	00:04:49
	INT-D	00:02:03	00:00:30	00:00:02	00:00:03	00:00:06	00:05:03
	Cons-D	00:01:54	00:00:05	00:00:02	00:00:02	00:00:06	00:00:19
NB	CFS-D	00:01:21	00:00:23	00:00:02	00:00:02	00:00:08	00:02:57
	IG-D	00:01:16	00:00:21	00:00:02	00:00:02	00:00:07	00:00:20
	ReliefF-D	00:01:18	00:00:22	00:00:03	00:00:03	00:00:07	00:03:01
	INT-D	00:01:32	00:00:25	00:00:02	00:00:04	00:00:07	00:02:53
	Cons-D	00:01:22	00:00:04	00:00:02	00:00:02	00:00:07	00:00:14
IB1	CFS-D	00:02:37	00:00:18	00:00:02	00:00:02	00:00:06	00:10:39
	IG-D	00:02:33	00:00:16	00:00:02	00:00:02	00:00:06	00:01:06
	ReliefF-D	00:02:33	00:00:18	00:00:02	00:00:02	00:00:06	00:10:28
	INT-D	00:03:04	00:00:20	00:00:02	00.00:03	00:00:06	00:10:29
	Cons-D	00:02:50	00:00:04	00:00:02	00:00:02	00:00:06	00:0040
SVM	CFS-D	00:03:11	00:01:49	00:00:02	00:00:02	00:00:07	00:56:58
	IG-D	00:03:09	00:01:39	00:00:02	00:00:02	00:00:06	00:03:40
	ReliefF-D	00:03:09	00:01:35	00:00:02	00:00:04	00:00:06	01:04:51
	INT-D	00:03:53	00:01:41	00:00:02	00:00:03	00:00:06	00:57:46
	Cons-D	00:03:24	00:00:20	00:00:02	00:00:02	00:00:06	00:02:54

10.3 Vertical partitioning

When having a large number of features, it is more appropriate to distribute the data by features. Thus, we will deal with subsets with a more balanced features/samples ratio and overfitting problems will be avoided. The approach that will be tested in this section for the vertical partitioning is the same than the one applied for the horizontal partitioning. Analogously to Algorithm 10.1, the pseudocode for the vertical partitioning can be seen in Algorithm 10.2. Notice that, in this case, and since there are not duplicated features in different packets, the maximum number of votes corresponds to the number of rounds r, so only the threshold values from 1 to r are evaluated.

Algorithm 10.2: Pseudo-code for vertical partitioning

Data: $\mathbf{D}_{(m \times n+1)} \leftarrow$ labeled training dataset with m samples and n input features

X $\leftarrow$set of features, X $= \{x_1, \ldots, x_n\}$
$s \leftarrow$ number of submatrices of $\mathbf{D}$ with p features
$r \leftarrow$ number of rounds
$\alpha \leftarrow 0.75$

Result: S $\leftarrow$ subset of features \S $\subset$ X

`/* Obtaining a vector of votes for discarding features */`

initialize the vector votes to 0, |vector|=n
for *each round* **do**
  Split $\mathbf{D}$ randomly into s disjoint submatrices
  for *each submatrix* **do**
    apply a feature selection algorithm
    F $\leftarrow$ features selected by the algorithm
    E $\leftarrow$ features eliminated by the algorithm \E $\cup$ F = X
    increment one vote for each feature in E
  end
end

`/* Obtain threshold of votes, Th, to remove a feature */`

$avg \leftarrow$ compute the average of the vector votes
$std \leftarrow$ compute the standard deviation of the vector votes
$minVote \leftarrow$ minimum threshold considered (1)
$maxVote \leftarrow$ maximum threshold considered (number of rounds, r)
$\mathbf{z} \leftarrow$ submatrix of $\mathbf{D}$ with only 10% of samples
for $v \leftarrow$ *minVote to maxVote with increment 1* **do**
  $\mathrm{F}_{th} \leftarrow$ subset of selected features (number of votes $< v$)
  $error \leftarrow$ classification error after training $\mathbf{z}$ using only features in F_{th}
  $featPercentage \leftarrow$ percentage of features retained $\left(\frac{|\mathrm{F}_{th}|}{|\mathrm{X}|} \times 100\right)$
  $e[v] \leftarrow \alpha \times error + (1 - \alpha) \times featPercentage$
end
$Th \leftarrow min(e)$, Th is the value which minimizes the error e
S $\leftarrow$ subset of features after removing from X all features with a number of votes $\geq Th$

10.3.1 Experimental results

For these experiments, only the datasets with the largest number of features were considered, i.e. Isolet, Madelon and MNIST (see Appendix I, Section I.2). We have opted for dividing the datasets in five packets, so that each packet will contain 20% of features, without replacement. In this manner, the considered datasets will have enough features to lead to a correct learning.

This approach has been tested using filter methods (CFS, Information Gain, ReliefF, INTERACT and consistency-based, see Chapter 2. The wrapper model has been also studied in a previous work (Bolón-Canedo, Sánchez-Maroño, & Alonso-Betanzos, 2013b). However, this chapter will be focused on the filter approach, since the results happened to be more interesting.

10.3.1.1 Number of features selected

This section reports the average number of features selected by the centralized and distributed approaches. First, Table 10.7 displays the results for the standard centralized method, as well as the full number of features for each dataset. Then, Table 10.8 shows the number of features selected by the distributed approach, which depends on the classifier employed to find the optimal threshold of votes.

Contrary to the horizontal distribution case, the number of features selected by the distributed methods is larger than those selected by the centralized approaches. Moreover, the number of features selected by the distributed approach with the vertical partition is also larger than in the case of the horizontal distribution (see Table 10.3). This is caused by the fact that, with the vertical partition, the features are distributed across the packets and it is more difficult to detect redundancy between features if they are in different partitions. Even so, our distributed approach is using (in the worst case) 53%, 5% and 16% of the total features for Isolet, Madelon and Mnist, respectively.

Table 10.7: Number of features selected by the centralized approach

	Isolet	Madelon	Mnist
Full set	617	500	717
CFS-C	189	17	60
IG-C	189	17	60
ReliefF-C	189	17	60
INT-C	58	23	38
Cons-C	11	22	17

Table 10.8: Number of features selected by the distributed approach

		Isolet	Madelon	Mnist
C4.5	CFS-D	229	20	60
	IG-D	323	17	111
	ReliefF-D	307	13	75
	INT-D	103	23	49
	Cons-D	78	20	28
NB	CFS-D	229	17	90
	IG-D	328	20	99
	ReliefF-D	293	16	42
	INT-D	57	23	85
	Cons-D	36	20	28
k-NN	CFS-D	242	23	114
	IG-D	330	18	111
	ReliefF-D	307	16	113
	INT-D	219	23	85
	Cons-D	19	21	76
SVM	CFS-D	229	20	90
	IG-D	325	2	99
	ReliefF-D	304	11	75
	INT-D	187	23	49
	Cons-D	162	19	76

10.3.1.2 Classification accuracy results

To test the vertical approach, the same filters and classifiers used with the horizontal approach were selected. Again, for the vertical partitioning, the number of rounds r was set to 5 and a hold-out validation was applied and repeated 5 times. Statistical tests were performed, labeled with a cross in Table 10.9, which shows the average test classification accuracy obtained with C4.5, naive Bayes, k-NN and SVM. Notice that in the table, the best result for each dataset and classifier is highlighted in bold face.

In terms of classification accuracy, the best results were obtained by the distributed approach for Isolet and Madelon, as well as in average for all datasets and classifiers. However, as can be seen in Table 10.9, in general there are no significant differences in terms of accuracy, as expected. When proposing a distributed approach, the goal is to maintain the classification performance whilst reducing the running time.

10.3.1.3 Runtime

Table 10.10 shows the runtime required by the feature selection methods. In the case of the distributed approach, the time displayed is the maximum time among those obtained in the different packets. As happened with the horizontal distribution, this filtering time is independent of the classifier chosen. The lowest time for each dataset is marked in bold.

Again, the execution time is drastically shortened by applying the distributed approach. It is specially notable the case of Mnist with ReliefF, in which the runtime is reduced from six hours to one hour. Notice that the larger the dataset, the larger the reduction in time. Table 10.11 depicts the time required to find the threshold in order to obtain the final subset of features. Analogously to the horizontal partition, the classifier that takes the longest time is SVM, particularly when it is combined with the Mnist dataset. In the remaining cases, however, the runtime is in the order of seconds or a few minutes. If the main objective is to reduce notably the time, the user must select another classifier instead of SVM.

Our proposal was able to reduce the running time significantly with respect to the standard (centralized) filtering algorithms. In terms of execution time, the behavior is excellent, being this fact the most important advantage of our method. Moreover,

Table 10.9: Test classification accuracy for the first approach of vertical partitioning.

		Isolet	Madelon	Mnist	Average
C4.5	CFS-C	80.61 †	75.34	85.36	80.44
	CFS-D	**80.85** †	75.08 †	86.16 †	80.70
	IG-C	79.00 †	79.63 †	86.69 †	81.77
	IG-D	79.54 †	78.27 †	**88.85** †	82.22
	ReliefF-C	79.02 †	**82.93** †	85.59 †	82.51
	ReliefF-D	80.04 †	81.33 †	88.57 †	**83.31**
	INT-C	76.76	77.94 †	84.77 †	79.82
	INT-D	77.59 †	78.09 †	87.06 †	80.91
	Cons-C	54.04	78.39 †	85.40	72.61
	Cons-D	74.99	77.3 †	85.01	79.10
NB	CFS-C	72.44 †	**69.94** †	72.18 †	71.52
	CFS-D	**74.07** †	69.57 †	73.26 †	**72.30**
	IG-C	69.62 †	69.78 †	68.80	69.40
	IG-D	70.93 †	69.55 †	71.38 †	70.62
	ReliefF-C	64.67	69.64 †	69.58	67.96
	ReliefF-D	69.53 †	69.32 †	69.27	69.37
	INT-C	67.17 †	69.73 †	69.87	68.92
	INT-D	65.21 †	69.46 †	**76.08** †	70.25
	Cons-C	40.19	69.81 †	71.78 †	60.59
	Cons-D	60.22	69.74 †	73.84 †	67.93
k-NN	CFS-C	55.63 †	68.86	86.34	70.28
	CFS-D	57.90 †	71.75	91.25 †	73.63
	IG-C	53.94	78.07 †	89.35 †	73.79
	IG-D	55.62 †	78.76 †	94.02 †	76.13
	ReliefF-C	56.69 †	**89.32** †	89.27 †	78.43
	ReliefF-D	**59.36** †	88.76 †	95.80 †	**81.31**
	INT-C	46.88	72.95 †	85.04	68.29
	INT-D	53.10 †	72.64 †	94.36 †	73.37
	Cons-C	53.05	75.04 †	85.42	71.17
	Cons-D	57.39 †	73.91 †	**96.03** †	75.78
SVM	CFS-C	82.36 †	65.23 †	80.15 †	75.91
	CFS-D	83.55 †	64.92 †	**81.58** †	**76.68**
	IG-C	81.09 †	65.9 †	78.6 †	75.20
	IG-D	82.61 †	65.16 †	80.02 †	75.93
	ReliefF-C	81.92 †	66.4 †	74.93	74.42
	ReliefF-D	**83.98** †	**66.33** †	76.82 †	75.71
	INT-C	72.53	65.01 †	78.44 †	71.99
	INT-D	80.38 †	64.97 †	78.46 †	74.60
	Cons-C	29.29	65.04 †	74.42	56.25
	Cons-D	76.45	65.73 †	80.17 †	74.12

Table 10.10: Runtime (hh:mm:ss) for the feature selection methods tested.

	Isolet	Madelon	Mnist
CFS-C	00:03:52	00:00:49	00:32:39
CFS-D	**00:00:40**	00:00:18	00:04:47
IG-C	00:02:34	00:00:47	00:26:24
IG-D	00:00:42	**00:00:15**	00:04:33
ReliefF-C	00:08:55	00:01:15	06:22:00
ReliefF-D	00:03:10	00:00:26	01:32:02
INT-C	00:03:06	00:00:49	00:31:01
INT-D	00:00:46	00:00:16	**00:03:45**
Cons-C	00:03:57	00:01:05	01:21:16
Cons-D	00:00:58	00:00:16	**00:03:45**

Table 10.11: Runtime (hh:mm:ss) for obtaining the threshold of votes.

		Isolet	Madelon	Mnist
C4.5	CFS-D	00:02:14	00:00:14	00:03:01
	IG-D	00:01:59	00:00:14	00:03:08
	ReliefF-D	00:01:10	00:00:13	00:03:59
	INT-D	00:01:11	00:00:13	00:03:58
	Cons-D	00:00:33	00:00:14	00:03:56
NB	CFS-D	00:01:30	00:00:16	00:02:03
	IG-D	00:01:40	00:00:14	00:02:04
	ReliefF-D	00:01:43	00:00:14	00:03:30
	INT-D	00:00:56	00:00:16	00:02:29
	Cons-D	00:00:29	00:00:16	00:02:16
IB1	CFS-D	00:01:15	00:00:13	00:06:14
	IG-D	00:01:26	00:00:13	00:06:03
	ReliefF-D	00:01:27	00:00:12	00:07:25
	INT-D	00:00:51	00:00:13	00:07:35
	Cons-D	00:00:26	00:00:13	00:05:34
SVM	CFS-D	00:06:41	00:00:14	00:19:06
	IG-D	00:06:59	00:00:13	00:26:38
	ReliefF-D	00:07:06	00:00:14	01:02:20
	INT-D	00:04:00	00:00:14	00:36:08
	Cons-D	00:02:01	00:00:15	00:37:02

regarding the classification accuracy, our vertical distributed approach was able to match and in some cases even improve the standard algorithms applied to the non-partitioned datasets.

10.4 Case of study: vertical partitioning applied to DNA microarray data

As could be seen in Chapter 4, many filter approaches were applied to successfully classify microarray data. However, up to the authors' knowledge, there is no attempt in the literature to tackle this problem with distributed feature selection, apart from the method proposed by Sharma et al. (2012). They introduced an algorithm that first distributes genes into relative small subsets, then selects informative smaller subsets of genes from a subset and merges the chosen genes with another gene subset to update the final gene subset.

Still, DNA microarray data prevents the use of horizontal partition because of the small sample size. The distributed methods found in the literature which are based on vertical partition (McConnell & Skillicorn, 2004; Skillicorn & McConnell, 2008; Rokach, 2009) have not been designed specifically for dealing with microarray data, so they do not tackle the particularities of this data, such as the high redundancy present among the features. The method proposed by Sharma et al. (2012) address these issues, but it has the disadvantage of being computationally expensive by having interaction with a classifier to select the genes in each subset. In this case of study a distributed filter method will be applied, suitable to microarray data and with a low computational cost.

The general methodology is the same explained in Section 10.1, which consists of the partition of the datasets, the application of feature selection techniques fo the subsets and, finally, the combination of the results. The partition of the dataset is performed by dividing the original dataset into several disjoint vertical subsets of approximately the same size that cover the full dataset. Two different methods were used for partitioning the data: (a) performing a randomly partition and (b) ranking the original features before generating the subsets. The second option was introduced trying to improve the performance results obtained by the first one. By having an ordered ranking, features with similar relevance to the class will be in the same subset, which will facilitate the task of the subset filter which will be applied later. These two techniques for

partitioning the data will generate two different approaches for the distributed method: Distributed Filter (DF) with the randomly partition and Distributed Ranking Filter (DRF) associated to the ranking partition.

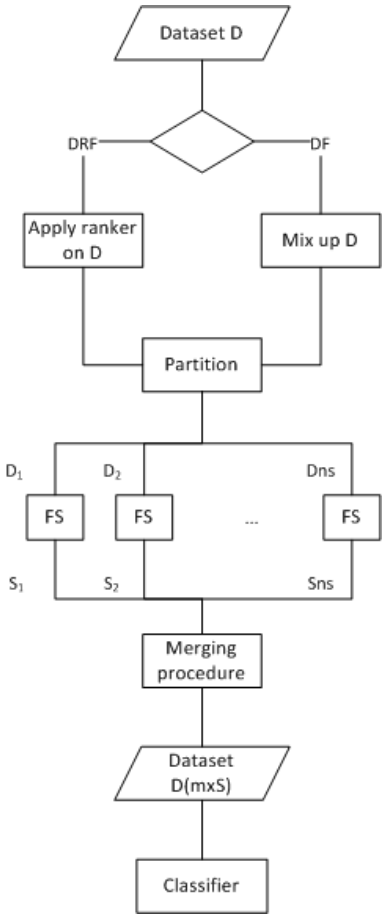


Figure 10.1: Flow chart of proposed algorithm

After this step, the data is split by assigning groups of k features to each subset, where the number of features k in each subset is half the number of samples, to avoid overfitting. Notice that, since we are working with microarray data, this division generates a large number of subsets. When opting for the randomly partition (DF), the groups of k features are constructed randomly, having into account that the subsets have to be disjoint. In the case of the ranking partition (DRF), the groups of k features are generated sequentially over the ranking, so features with a similar ranking position will be in the same group. Notice that the random partition is equivalent to obtain a random ranking of the features and then follow the same steps as with the ordered ranking. Figure 10.1 shows a flow chart which reflects the two algorithms proposed, DF and DRF.

Algorithm 10.3: Pseudo-code for vertical partitioning for DNA microarray data

Data: $\mathbf{D}_{(m \times n+1)} \leftarrow$ labeled training dataset with m samples and n input features

X $\leftarrow$set of features, X $= \{x_1, \ldots, x_n\}$

$s \leftarrow$ number of submatrices of $\mathbf{D}$ with p features

Result: S $\leftarrow$ subset of features \S $\subset$ X

Select the partition method:

DF $\leftarrow$ Mix up dataset $\mathbf{D}$ to obtain a random ranking R or

DRF $\leftarrow$ Apply a ranker method to $\mathbf{D}$ to obtain an ordered ranking R

`/* Splitting the dataset */`

for *i* $\leftarrow$ *1 to s with increment 1* **do**

  $R_i =$ first p features in R

  $R = R \setminus R_i$

  $D_i = D_{(m \times R_i)}$

end

`/* Application of feature selection methods to the subsets */`

for *i* $\leftarrow$ *1 to s with increment 1* **do**

  $S_i =$ subset of features obtained after applying the chosen filter over D_i

end

`/* Merging procedure */`

$S = S_1$

baseline = classification accuracy after training subset $D_{(m \times S_i)}$

for *i* $\leftarrow$ *2 to s with increment 1* **do**

  $S_{aux} = S \cup S_i$

  accuracy = classification accuracy after training subset $D_{(m \times S_{aux})}$

  if *accuracy* $>$ *baseline* **then**

    $S = S_{aux}$

    baseline = *accuracy*

  end

end

S $\leftarrow$ final subset of features

After having several small disjoint datasets D_i, the filter method will be applied to each of them, returning a selection S_i for each subset of data. Finally, to combine the results, a merging procedure using a classifier will be executed. The first selection S_1 is taken to calculate the classification accuracy, which will be the *baseline*, and the features in S_1 will become part of the final selection S. Then, the features in the remaining selections S_i will be merged with the candidate feature subset. If the classification accuracy using them improves the baseline accuracy, these features will be part of the final selection and this accuracy will become the baseline accuracy, as can be seen in more detail in Algorithm 10.3. By combining the features in this manner, it is expected to remove redundancy and solve the problem detected with the previous approach (see Section 10.3), since a redundant feature will not improve the accuracy and hence will not be added to the final selection. At the end, this final selection S is applied to the training and test sets in order to obtain the ultimate classification accuracies. It has to be noted that this algorithm can be used with any feature subset filter.

10.4.1 Experimental setup

In order to test the proposed distributed filter approach, we have selected eight microarray benchmark datasets which can be consulted in Appendix I, Table I.8: Colon, DLBCL, CNS, Leukemia, Prostate, Lung, Ovarian and Breast. Those datasets originally divided to training and test sets were maintained, whereas, for a fair comparison, datasets with only training set were randomly divided using the common rule 2/3 for training and 1/3 for testing. For calculating the number of packets s, it was considered that each packet would have as many features as twice the number of samples, to avoid overfitting.

As can be seen in Table I.8, Leukemia dataset presents the so-called imbalance problem (see Chapter 4, Section 4.2.2). A dataset is considered imbalanced when the classification categories are not approximately equally represented in the training set. This situation is very common in microarray datasets, when most of the instances correspond with "normal" patterns while the standard interest consists of identifying the "abnormal" patterns. There exist some techniques to overcome the imbalance problem, however some of them are not appropriate to microarray data (Blagus & Lusa, 2012). For this sake, and following the recommendations given by Blagus and Lusa (2012), we will use random oversampling to deal with Leukemia dataset, which consists of replicating, at random, elements of the under-sized class until it matches the size of the other classes.

10.4.2 Election of the ranker method

There are two different types of ranker feature selection: univariate and multivariate (see Chapter 2). Univariate methods are fast and scalable, but ignore feature dependencies. On the other hand, multivariate filters model feature dependencies, but at the cost of being slower and less scalable than univariate techniques. As a representative of univariate ranker filters we have chosen the well known Information Gain (M. Hall & Smith, 1998). As for multivariate ranker filter, ReliefF (Kira & Rendell, 1992) and minimum-Redundancy-maximum-Relevance (mRMR) were chosen. The detailed descriptions of these filters can be consulted in Chapter 2.

Because of their nature, the ranking returned by Information Gain and ReliefF places the more relevant features on the top of the ranking (even when they are redundant) whilst mRMR is able to detect the redundant features and move them to lower positions in the ranking. Feature/gene redundancy is an important problem in microarray data classification. In the presence of thousands of features, researchers noticed that it is common that a large number of features are not informative because they are redundant. Empirical evidence showed that along with irrelevant features, redundant features also affect the speed and accuracy of mining algorithms (Yu & Liu, 2004b).

In light of the above, the behavior of mRMR might seem an advantage in removing redundancy, but it is not the case in these experiments. Preliminary tests with these three ranker filters (not included for the sake of brevity) revealed that it was better to have the redundant features in the same packet of features, so the filter applied afterwards could remove the redundant features at once. On the contrary, if the redundant features are split in different packets, there are more chances to keep them and degrade the performance. Moreover, multivariate ranker filters are slower than univariate ones, and to deal with the entire set of features (thousands of them) it is important to reduce this processing time. For these reasons, Information Gain was chosen for this study. This univariate filter provides an ordered ranking of all the features in which the worth of an attribute is evaluated by measuring the information gain with respect to the class.

One of the particularities of the Information Gain filter is that, if a feature is not relevant to the prediction task, its information gain with respect to the class would be zero. For this reason, we will also try an approach which consists of eliminating the features with information gain zero from the initial ranking.

10.4.3 Experimental results

In this section the experimental results over the eight datasets mentioned above will be presented and discussed. The distributed approach proposed in this chapter can be used with any filter method. In these experiments, five well-known filters, based on different metrics where chosen. While three of the filters return a feature subset (CFS, Consistency-based and INTERACT), the other two (ReliefF and Information Gain) are ranker methods, so it is necessary to establish a threshold in order to obtain a subset of features. In this case we have opted for retaining the 10% and 25% top features. Notice that although most of the filters work only over nominal features, the discretization step is done by default by Weka, working as a black box for the user. A detailed description of the filters can be found in Chapter 2.

Four different approaches will be compared in the tables of this section: the centralized filter approach (CF), the distributed filter approach (DF), the distributed ranking filter approach (DRF) and the distributed ranking filter approach removing the features with information gain zero from the ranking (DRF0). Notice that with the DF method, the subsets of features are randomly generated so they are different in each iteration. For this reason, the experiments are run 5 times and the average of them is shown in the tables.

10.4.3.1 Number of features selected

Table 10.12 displays the number of features selected by the centralized approach, which is independent of the classifier, as well as the original number of features for each dataset. Notice that the number of features selected by the ranker methods (Information Gain and ReliefF) are notably higher than those of the subset filters, because it is necessary to establish a percentage of features to retain. Tables 10.13 and 10.14 show the number of features selected by the distributed approaches on the eight datasets. Note that the features selected by the distributed approaches depend on the classifier because of the merging procedure.

From these tables it can be observed that the number of features selected by any method is considerably much smaller than that of the full features. Therefore, all the feature selection algorithms tested herein are able to reduce significantly the number of features as well as the storage requirements and the classification runtime.

Table 10.12: Number of features selected by the centralized approach

	Colon	Leukemia	CNS	DLBCL	Prostate	Lung	Ovarian	Breast
Full set	2000	7129	7129	4026	12600	12533	15154	24481
CFS	19	36	60	47	89	40	37	130
Cons	3	1	3	2	4	1	3	5
INT	16	36	47	36	73	40	27	102
IG10%	200	713	713	403	1260	1254	1516	2449
IG25%	500	1783	1783	1007	3150	3134	3789	6121
ReliefF10%	200	713	713	403	1260	1254	1516	2449
ReliefF25%	500	1783	1783	1007	3150	3134	3789	6121

In general, the feature selection method that selects the lowest number of features for any dataset is the consistency-based filter, especially in its centralized approach. The subset filters (CFS, consistency-based and INTERACT) tend to select a larger number of features when they are applied in a distributed approach. On the contrary, the ranker methods (Information Gain and ReliefF) reduce the number of features in the final subset when the process is distributed. This can be explained because with the centralized approach, they have no option but to select the established percentage of features, and this fixed number is smaller when it is applied to a subset of the data.

It is worth noticing that the centralized version of consistency-based, as well as some distributed approaches, select a single feature for Leukemia and Lung datasets. These two datasets were originally divided to training and test datasets, with the data extracted under different conditions, which may hinder the process of feature selection and classification. This phenomenon is known as dataset shift and it has been discussed in Chapter 4, Section 4.2.4. In fact, the single feature selected by the consistency-based filter on the Lung dataset was the feature #1136. This feature in the training set can be used to distinguish clearly the target concept values, as shown in Figure 4.2a. Nevertheless, this feature is not so informative in the test set and the class is not linearly separable, as displayed in Figure 4.2b. For this reason, although 100% classification accuracy was obtained with the training set of this dataset using this feature (not included in the tables for the sake of brevity), the precision decreases on the test set (see Tables 10.15 - 10.18). A similar situation also happens with the Leukemia dataset.

Table 10.13: Number of features selected by the distributed approaches with C4.5 and naive Bayes classifiers

		Colon	Leukemia	CNS	DLBCL	Prostate	Lung	Ovarian	Breast
C4.5	DF-CFS	8	11	8	6	50	11	95	20
	DRF-CFS	25	13	28	11	121	17	16	31
	DRF0-CFS	22	13	28	11	129	17	16	27
	DF-Cons	5	8	7	4	39	9	62	16
	DRF-Cons	5	1	12	2	21	1	22	24
	DRF0-Cons	10	1	11	2	13	1	22	6
	DF-INT	8	10	8	6	53	11	75	19
	DRF-INT	20	13	16	9	87	17	33	30
	DRF0-INT	18	13	26	9	88	17	33	27
	DF-IG10%	14	9	14	9	53	9	85	48
	DRF-IG10%	9	2	6	2	30	2	36	44
	DRF0-IG10%	9	2	4	2	30	2	27	16
	DF-IG25%	34	22	28	19	125	17	194	118
	DRF-IG25%	6	5	15	4	91	4	44	100
	DRF0-IG25%	6	5	5	4	65	4	44	50
	DF-ReliefF10%	17	11	14	11	49	9	70	41
	DRF-ReliefF10%	9	4	8	2	30	2	18	20
	DRF0-ReliefF10%	9	4	6	2	24	2	18	8
	DF-ReliefF25%	34	22	34	21	99	18	176	142
	DRF-ReliefF25%	12	5	20	4	52	4	66	80
	DRF0-ReliefF25%	12	5	10	4	26	4	66	10
NB	DF-CFS	11	12	12	10	58	9	97	24
	DRF-CFS	25	13	50	26	20	17	38	152
	DRF0-CFS	10	13	50	26	20	17	34	150
	DF-Cons	5	10	10	6	56	8	66	16
	DRF-Cons	12	4	28	5	5	3	3	33
	DRF0-Cons	5	4	17	5	5	3	3	33
	DF-INT	11	11	12	10	60	9	87	24
	DRF-INT	18	13	41	9	22	17	22	126
	DRF0-INT	15	13	34	9	21	17	22	123
	DF-IG10%	16	11	22	14	48	9	108	32
	DRF-IG10%	6	6	14	4	12	2	45	28
	DRF0-IG10%	3	6	8	4	12	2	36	20
	DF-IG25%	37	22	47	33	109	20	172	76
	DRF-IG25%	12	5	15	8	26	4	66	100
	DRF0-IG25%	6	5	5	8	39	4	22	30
	DF-ReliefF10%	19	11	16	14	50	10	99	44
	DRF-ReliefF10%	6	4	14	4	24	4	27	40
	DRF0-ReliefF10%	6	4	4	4	24	4	18	24
	DF-ReliefF25%	35	26	38	30	114	17	202	116
	DRF-ReliefF25%	6	10	25	4	13	8	88	40
	DRF0-ReliefF25%	12	10	10	4	13	8	66	20

Table 10.14: Number of features selected by the distributed approaches with k-NN and SVM classifiers

		Colon	Leukemia	CNS	DLBCL	Prostate	Lung	Ovarian	Breast
k-NN	DF-CFS	12	16	9	8	70	11	77	22
	DRF-CFS	27	13	37	11	48	17	33	34
	DRF0-CFS	23	13	32	11	35	17	33	27
	DF-Cons	7	13	8	7	54	10	56	15
	DRF-Cons	12	4	19	6	48	3	22	44
	DRF0-Cons	9	4	18	2	31	3	23	54
	DF-INT	12	18	9	8	66	11	77	24
	DRF-INT	20	13	31	9	58	17	11	35
	DRF0-INT	7	13	27	9	51	17	11	27
	DF-IG10%	15	12	16	14	55	9	61	60
	DRF-IG10%	6	4	12	4	24	2	18	52
	DRF0-IG10%	6	4	6	4	30	2	18	32
	DF-IG25%	41	25	43	33	135	20	158	136
	DRF-IG25%	18	10	10	8	65	4	66	40
	DRF0-IG25%	6	10	5	8	65	4	44	10
	DF-ReliefF10%	16	12	16	11	46	10	68	42
	DRF-ReliefF10%	9	6	8	6	30	2	18	40
	DRF0-ReliefF10%	6	6	6	6	24	2	18	20
	DF-ReliefF25%	35	27	39	24	101	18	172	100
	DRF-ReliefF25%	12	15	30	12	65	4	22	50
	DRF0-ReliefF25%	6	15	15	8	52	4	22	40
SVM	DF-CFS	8	17	10	9	59	14	51	21
	DRF-CFS	12	13	52	11	62	17	16	82
	DRF0-CFS	10	13	64	11	113	17	16	82
	DF-Cons	7	15	5	7	54	10	41	21
	DRF-Cons	5	10	18	5	59	5	8	42
	DRF0-Cons	5	10	17	5	51	5	8	51
	DF-INT	8	18	10	9	68	13	47	18
	DRF-INT	7	13	78	9	65	17	11	116
	DRF0-INT	7	13	27	9	61	17	11	97
	DF-IG10%	14	14	13	16	52	9	49	56
	DRF-IG10%	3	8	2	4	30	2	45	44
	DRF0-IG10%	3	8	2	2	36	2	45	24
	DF-IG25%	35	34	57	30	117	18	84	130
	DRF-IG25%	12	15	30	12	26	4	66	100
	DRF0-IG25%	6	15	15	12	39	4	44	40
	DF-ReliefF10%	18	14	16	13	54	10	47	47
	DRF-ReliefF10%	6	6	12	6	30	2	18	44
	DRF0-ReliefF10%	6	6	8	6	18	2	18	24
	DF-ReliefF25%	32	26	43	22	99	18	84	102
	DRF-ReliefF25%	12	10	40	8	52	4	22	100
	DRF0-ReliefF25%	18	10	20	4	52	4	22	60

This situation reflects the enormous problematic of microarray data (see Chapter 4), in which in some cases the training and test samples are recorded under completely different situations. In the case of Leukemia dataset, the training samples were extracted from adult patients, whereas the test samples were obtained mainly from children.

10.4.3.2 Classification accuracy results

In this section we discuss the test classification accuracy obtained by C4.5, naive Bayes, k-NN and SVM classifiers with the centralized and distributed approaches. Notice that in these tables, the best results are marked in bold.

Table 10.15 reports the classification accuracy obtained by C4.5 for all datasets and approaches tested. The centralized approach achieves the best result for two datasets (Lung and Breast), whilst for the remaining datasets, the distributed approach DRF outperforms the other methods with Colon, CNS and Prostate datasets. It is worth mentioning the case of the CNS datasets, in which DRF and DRF0 combined with the consistency-based filter surpassed the best centralized results by 20%. As for the Prostate dataset, the results obtained with DRF combined with ReliefF retaining 25% of the features outperformed the best centralized results by 32.35%. In terms of average accuracy for all datasets, the best option is DRF0 combined with the consistency-based filter. It seems that, in general, the features with no information gain with respect to the class are not relevant to the prediction task.

From Table 10.16 we can observe the classification accuracy reported by naive Bayes for the eight datasets at hand. For some datasets the best choice is the centralized approach (Leukemia, Prostate and Lung), whereas for others it is better to apply a distributed method (DLBCL and Breast). The differences between the centralized and the distributed approach are not so prominent as with C4.5 classifier. Even though, the case of Breast dataset can be emphasized, in which DRF combined with ReliefF outperforms the best centralized result in more than 10%. In fact, the best result in average for all datasets was achieved by DRF together with ReliefF when retaining the top 25% of features in each partition.

Table 10.17 shows the results obtained by k-NN. On the one hand, the highest accuracy for the datasets Colon and Lung was reported by a centralized approach. On the other hand, the distributed approach performed better with Leukemia, CNS,

Table 10.15: Test classification accuracy of C4.5

		Colon	Leukemia	CNS	DLBCL	Prostate	Lung	Ovarian	Breast	Average
CFS	CF	85.00	**91.18**	50.00	**86.67**	26.47	81.88	**100.00**	68.42	73.70
	DF	76.00	74.71	69.00	80.00	35.88	91.28	95.24	58.95	72.63
	DRF	85.00	**91.18**	45.00	**86.67**	29.41	89.26	**100.00**	57.89	73.05
	DRF0	85.00	**91.18**	45.00	**86.67**	29.41	89.26	**100.00**	52.63	72.39
Cons	CF	85.00	**91.18**	50.00	**86.67**	23.53	81.88	**100.00**	68.42	73.33
	DF	80.00	70.00	62.00	84.00	36.47	90.60	95.95	65.26	73.04
	DRF	70.00	**91.18**	**80.00**	**86.67**	58.82	89.26	96.43	68.42	80.10
	DRF0	85.00	**91.18**	**80.00**	**86.67**	58.82	89.26	96.43	73.68	**82.63**
INT	CF	85.00	**91.18**	55.00	**86.67**	26.47	81.88	98.81	**78.95**	75.49
	DF	76.00	78.24	69.00	78.67	37.06	91.28	94.76	54.74	72.47
	DRF	85.00	**91.18**	55.00	**86.67**	29.41	89.26	98.81	52.63	73.49
	DRF0	85.00	**91.18**	60.00	**86.67**	29.41	89.26	98.81	52.63	74.12
IG 10%	CF	85.00	**91.18**	45.00	**86.67**	26.47	89.26	98.81	73.68	74.51
	DF	80.00	74.71	54.00	82.67	35.88	91.41	93.81	68.42	72.61
	DRF	**90.00**	**91.18**	55.00	**86.67**	23.53	89.26	**100.00**	63.16	74.85
	DRF0	85.00	91.18	40.00	**86.67**	38.24	89.26	**100.00**	73.68	75.50
IG 25%	CF	85.00	**91.18**	60.00	**86.67**	26.47	89.26	98.81	73.68	76.38
	DF	80.00	82.35	61.00	81.33	29.41	91.28	96.67	62.11	73.02
	DRF	85.00	**91.18**	65.00	**86.67**	23.53	89.26	98.81	47.37	73.35
	DRF0	80.00	**91.18**	65.00	**86.67**	26.47	89.26	98.81	52.63	73.75
ReliefF 10%	CF	85.00	**91.18**	45.00	**86.67**	29.41	**97.32**	98.81	63.16	74.57
	DF	75.00	78.24	60.00	**86.67**	25.88	91.28	96.90	65.26	72.40
	DRF	85.00	67.65	65.00	**86.67**	26.47	89.93	98.81	52.63	71.52
	DRF0	85.00	67.65	65.00	**86.67**	26.47	89.93	98.81	57.89	72.18
ReliefF 25%	CF	85.00	**91.18**	45.00	**86.67**	29.41	**97.32**	98.81	73.68	75.88
	DF	80.00	74.71	54.00	84.00	40.59	90.60	95.95	64.21	73.01
	DRF	**90.00**	**91.18**	65.00	**86.67**	**61.76**	89.93	98.81	47.37	78.84
	DRF0	85.00	**91.18**	65.00	**86.67**	41.18	89.93	98.81	42.11	74.98

DLBCL and Prostate datasets. In general, focusing on the distributed approach, the methods which include a ranking as a first step (DRF and DRF0) obtain better results than DF, which divides the data randomly. As a matter of fact, the method with the best average accuracy for all datasets was DRF0 combined with CFS.

Table 10.16: Test classification accuracy of naive Bayes

		Colon	Leukemia	CNS	DLBCL	Prostate	Lung	Ovarian	Breast	Average
CFS	CF	**90.00**	94.12	**70.00**	93.33	26.47	**100.00**	97.62	36.84	76.05
	DF	84.00	72.94	57.00	92.00	25.29	87.79	95.95	49.47	70.56
	DRF	85.00	88.24	60.00	93.33	20.59	98.66	**100.00**	36.84	72.83
	DRF0	**90.00**	88.24	60.00	93.33	20.59	98.66	**100.00**	36.84	73.46
Cons	CF	85.00	91.18	55.00	86.67	**32.35**	85.91	**100.00**	36.84	71.62
	DF	85.00	74.12	53.00	90.67	26.47	86.71	95.00	62.11	71.63
	DRF	**90.00**	94.12	55.00	73.33	26.47	94.63	97.62	36.84	71.00
	DRF0	85.00	94.12	65.00	73.33	26.47	94.63	97.62	36.84	71.63
INT	CF	85.00	94.12	65.00	93.33	26.47	**100.00**	**100.00**	36.84	75.10
	DF	84.00	79.41	57.00	90.67	26.47	88.05	93.33	52.63	71.45
	DRF	90.00	88.24	65.00	93.33	23.53	98.66	98.81	36.84	74.30
	DRF0	85.00	88.24	65.00	93.33	23.53	98.66	98.81	36.84	73.68
IG 10%	CF	85.00	**97.06**	60.00	93.33	26.47	98.66	92.86	36.84	73.78
	DF	81.00	73.53	59.00	**94.67**	24.71	90.47	92.14	42.11	69.70
	DRF	85.00	94.12	60.00	86.67	26.47	90.60	100.00	42.11	73.12
	DRF0	85.00	94.12	55.00	86.67	26.47	90.60	97.62	31.58	70.88
IG 25%	CF	80.00	**97.06**	50.00	93.33	26.47	98.66	92.86	36.84	71.90
	DF	83.00	72.35	61.00	89.33	26.47	92.62	91.19	38.95	69.36
	DRF	85.00	94.12	35.00	86.67	26.47	96.64	98.81	36.84	69.94
	DRF0	85.00	94.12	55.00	86.67	26.47	96.64	97.62	36.84	72.30
ReliefF 10%	CF	85.00	94.12	65.00	93.33	26.47	99.33	92.86	68.42	78.07
	DF	84.00	73.53	60.00	85.33	25.29	92.21	94.05	73.68	73.51
	DRF	85.00	82.35	**70.00**	80.00	23.53	98.66	**100.00**	**78.95**	77.31
	DRF0	85.00	82.35	60.00	80.00	23.53	98.66	**100.00**	**78.95**	76.06
ReliefF 25%	CF	85.00	94.12	55.00	93.33	26.47	99.33	92.86	63.16	76.16
	DF	79.00	77.65	65.00	84.00	26.47	88.72	91.67	65.26	72.22
	DRF	85.00	91.18	65.00	86.67	23.53	98.66	98.81	**78.95**	**78.47**
	DRF0	85.00	91.18	55.00	86.67	23.53	98.66	**100.00**	73.68	76.71

Finally, from Table 10.18 we can confirm that the SVM classifier is very suitable for microarray data (as mentioned in Chapter 4), since it achieves high classification accuracies in average for all datasets. Notice that this classifier can successfully handle

this kind of datasets with a much higher number of features than samples. Particularly, the best result in terms of average accuracy was obtained by DRF0 combined with Information Gain when retaining the top 25% of features in each partition. It is worth commenting the excellent result for Prostate dataset (obtained by CF with INTERACT and DRF with Information Gain 10%), which outperforms in more than 35% the highest accuracy reported by the other classifiers.

Table 10.17: Test classification accuracy of k-NN

		Colon	Leukemia	CNS	DLBCL	Prostate	Lung	Ovarian	Breast	Average
CFS	CF	80.00	85.29	65.00	86.67	32.35	**100.00**	**100.00**	63.16	76.56
	DF	79.00	78.24	61.00	88.00	31.18	91.68	99.05	57.89	73.25
	DRF	80.00	88.24	55.00	**93.33**	47.06	97.32	98.81	57.89	77.21
	DRF0	80.00	88.24	55.00	**93.33**	**61.76**	97.32	98.81	73.68	**81.02**
Cons	CF	85.00	91.18	65.00	73.33	26.47	81.88	**100.00**	47.37	71.28
	DF	79.00	76.47	59.00	82.67	32.94	88.72	97.14	58.95	71.86
	DRF	80.00	**94.12**	60.00	80.00	26.47	93.96	98.81	63.16	74.56
	DRF0	75.00	**94.12**	65.00	80.00	26.47	93.96	98.81	68.42	75.22
INT	CF	80.00	85.29	60.00	86.67	32.35	**100.00**	**100.00**	52.63	74.62
	DF	77.00	78.24	61.00	88.00	38.82	91.68	97.86	56.84	73.68
	DRF	70.00	88.24	60.00	**93.33**	32.35	97.32	**100.00**	73.68	76.87
	DRF0	85.00	88.24	60.00	**93.33**	41.18	97.32	**100.00**	68.42	79.19
IG 10%	CF	90.00	76.47	50.00	86.67	26.47	98.66	97.62	73.68	74.95
	DF	80.00	75.29	59.00	85.33	34.71	88.99	99.05	62.11	73.06
	DRF	70.00	88.24	70.00	86.67	38.24	90.60	**100.00**	**78.95**	77.84
	DRF0	80.00	88.24	65.00	86.67	52.94	90.60	**100.00**	68.42	78.98
IG 25%	CF	**95.00**	79.41	60.00	80.00	38.24	99.33	96.43	73.68	77.76
	DF	82.00	73.53	54.00	81.33	34.71	92.48	97.14	65.26	72.56
	DRF	80.00	91.18	45.00	86.67	26.47	97.32	97.62	63.16	73.43
	DRF0	70.00	91.18	50.00	86.67	26.47	97.32	98.81	47.37	70.98
ReliefF 10%	CF	90.00	76.47	50.00	86.67	26.47	98.66	97.62	73.68	74.95
	DF	83.00	80.00	56.00	90.67	37.06	94.23	98.10	54.74	74.22
	DRF	70.00	82.35	65.00	86.67	26.47	96.64	**100.00**	73.68	75.10
	DRF0	80.00	85.29	60.00	86.67	50.00	96.64	**100.00**	73.68	79.04
ReliefF 25%	CF	85.00	73.53	55.00	86.67	29.41	97.99	96.43	**78.95**	75.37
	DF	79.00	75.29	54.00	84.00	34.71	93.56	95.95	63.16	72.46
	DRF	80.00	91.18	**75.00**	**93.33**	23.53	97.99	**100.00**	63.16	78.02
	DRF0	55.00	91.18	**75.00**	**93.33**	29.41	97.99	**100.00**	52.63	74.32

Since for the CF, DRF and DRF0 approaches the experiments were run only once, it is not possible to conduct statistical tests to check if the differences among the methods

are statistically significant. Nevertheless, one can see easily from Tables 10.15 - 10.18 that the average results are very similar. In any case, the best average accuracy was obtained by a distributed approach for all the classifiers tested. Therefore we can affirm that, in terms of classification accuracy, our proposed distributed approaches at least maintain the performance compared with that of the standard centralized approach.

Table 10.18: Test classification accuracy of SVM

		Colon	Leukemia	CNS	DLBCL	Prostate	Lung	Ovarian	Breast	Average
CFS	CF	**85.00**	82.35	65.00	**93.33**	91.18	98.66	**100.00**	73.68	86.15
	DF	76.00	82.35	67.00	89.33	66.47	91.01	99.05	67.37	79.82
	DRF	80.00	88.24	50.00	86.67	76.47	95.30	98.81	73.68	81.15
	DRF0	80.00	88.24	70.00	86.67	85.29	95.30	98.81	73.68	84.75
Cons	CF	70.00	73.53	65.00	86.67	26.47	52.35	98.81	52.63	65.68
	DF	78.00	73.53	64.00	89.33	68.24	91.81	98.81	61.05	78.10
	DRF	75.00	67.65	**80.00**	80.00	35.29	91.28	98.81	68.42	74.56
	DRF0	75.00	85.29	70.00	80.00	26.47	91.28	98.81	63.16	73.75
INT	CF	80.00	82.35	55.00	86.67	**97.06**	98.66	**100.00**	73.68	84.18
	DF	76.00	78.82	67.00	90.67	78.82	91.14	98.57	67.37	81.05
	DRF	75.00	88.24	70.00	**93.33**	91.18	95.30	98.81	73.68	85.69
	DRF0	75.00	88.24	60.00	**93.33**	82.35	95.30	98.81	**84.21**	84.66
IG 10%	CF	80.00	**94.12**	60.00	**93.33**	82.35	**99.33**	98.81	73.68	85.20
	DF	77.00	78.82	59.00	88.00	57.65	86.04	98.33	68.42	76.66
	DRF	75.00	82.35	65.00	86.67	**97.06**	74.50	**100.00**	68.42	81.12
	DRF0	75.00	88.24	65.00	86.67	85.29	74.50	**100.00**	63.16	79.73
IG 25%	CF	80.00	**94.12**	65.00	**93.33**	73.53	**99.33**	98.81	57.89	82.75
	DF	74.00	78.24	65.00	84.00	60.00	94.23	99.05	60.00	76.81
	DRF	**85.00**	91.18	60.00	**93.33**	79.41	97.99	**100.00**	63.16	83.76
	DRF0	75.00	91.18	**80.00**	**93.33**	79.41	97.99	98.81	73.68	**86.18**
ReliefF 10%	CF	70.00	91.18	65.00	**93.33**	79.41	**99.33**	98.81	57.89	81.87
	DF	77.00	78.82	57.00	90.67	68.24	91.54	98.10	67.37	78.59
	DRF	80.00	79.41	60.00	80.00	91.18	95.97	**100.00**	52.63	79.90
	DRF0	80.00	82.35	65.00	80.00	70.59	95.97	**100.00**	63.16	79.63
ReliefF 25%	CF	75.00	88.24	65.00	**93.33**	76.47	**99.33**	98.81	63.16	82.42
	DF	75.00	78.24	63.00	90.67	69.41	93.69	98.57	70.53	79.89
	DRF	75.00	85.29	75.00	86.67	73.53	96.64	**100.00**	57.89	81.25
	DRF0	80.00	85.29	70.00	86.67	64.71	96.64	**100.00**	68.42	81.47

10.4.3.3 Runtime

Table 10.19 reports the runtime of the feature selection algorithms studied applied to the eight microarray datasets. In the distributed approaches (DF, DRF and DRF0), all the subsets can be processed at the same time, so the time displayed in the table is the maximum of the times required by the filter for all the subsets generated at the partitioning stage.

Table 10.19: Runtime (in seconds) for the feature selection methods tested.

		Colon	Leukemia	CNS	DLBCL	Prostate	Lung	Ovarian	Breast
CFS	CF	4.92	7.31	198.14	47.52	1225.87	13.73	202.62	13323.50
	DF	0.18	0.19	0.17	0.17	0.27	0.16	0.38	0.22
	DRF	0.20	0.25	0.35	0.18	0.32	0.28	0.42	0.28
	DRF0	0.19	0.19	0.18	0.18	0.31	0.18	0.42	0.28
Cons	CF	1.00	2.73	3.06	1.39	8.83	4.89	14.40	25.26
	DF	0.20	0.21	0.20	0.19	0.43	0.19	0.47	0.28
	DRF	0.22	0.20	0.20	0.19	0.40	0.20	0.49	0.29
	DRF0	0.23	0.20	0.21	0.19	0.34	0.19	0.50	0.30
INT	CF	1.64	15.47	12.75	3.69	123.10	38.23	224.72	285.21
	DF	0.24	0.24	0.24	0.23	0.31	0.23	0.48	0.28
	DRF	0.24	0.27	0.23	0.22	0.33	0.33	0.48	0.29
	DRF0	0.24	0.24	0.23	0.22	0.34	0.23	0.53	0.29
IG	CF	0.66	1.08	1.11	0.86	1.79	1.49	3.51	3.32
	DF	0.16	0.17	0.17	0.16	0.24	0.16	0.31	0.21
	DRF	0.16	0.17	0.16	0.16	0.23	0.28	0.31	0.25
	DRF0	0.16	0.16	0.16	0.15	0.24	0.15	0.32	0.22
ReliefF	CF	0.61	1.14	1.19	0.78	4.50	1.56	12.92	8.14
	DF	0.19	0.20	0.20	0.28	0.27	0.18	0.34	0.24
	DRF	0.20	0.19	0.19	0.18	0.28	0.24	0.34	0.24
	DRF0	0.18	0.19	0.18	0.18	0.28	0.18	0.34	0.24

As expected, when applying a distributed approach the time is reduced for all datasets. It is worth pointing out the case of Brain dataset combined with the CFS filter, in which the centralized approach took almost 4 hours whilst the time required by the distributed approaches was under 1 second. Having said this, it is necessary to remind that the distributed approaches have also a stage for combining the results from the different partitions. However, in this case, the time required to merging the

features is in the order of minutes, therefore our proposed method is able to shorten the execution time impressively compared to the standard version of the filter algorithm.

In light of the above, the most important advantage of our distributed methods is the large reduction in both execution time and number of features needed whilst maintaining the classification accuracy at reasonable levels, and in some cases even improving it. Two main different versions of the distributed algorithm have been proposed: the distributed ranking filter (DRF) and the distributed filter (DF), which performs a random partition of the data. DRF and its variant DRF0, using Information Gain in the first ranking stage, obtained the best results. However, as mentioned in the introduction of this chapter, the reason for which a user would like to apply distributed feature selection is two-fold: (i) data is sometimes distributed in multiple locations and often with multiple parties; and (ii) most existing feature selection algorithms do not scale well and their efficiency deteriorates significantly or even becomes inapplicable when dealing with large-scale data. In the first scenario, an ordered ranking of the features as a first step is not possible, since features are distributed in origin, therefore the distributed algorithm cannot take advantage of the ranking provided by Information Gain. On the other hand, they might appear cases in which the extremely huge amount of features prevent the use of Information Gain in order to obtain a previous ordered ranking of the features, so the random partition (DF) is the only option left. To sum up, we consider interesting to count on these two different options for distributed feature selection, although we recommend to use DRF or DRF0 if possible.

10.5 Incremental vertical partitioning

As mentioned in the introduction of this chapter, there are several ways in which a feature selection task could be distributed. An extreme case of a large data volume is streaming data that arrives in real time as effectively an infinite stream. To deal with the frequent arrival of new training data we need to use learning algorithms that are *incremental*, i.e. where an algorithm already constructed can be updated using new data without needing to re-process data already used (Bramer, 2007). To simulate this situation, the approach presented in this section consists of process parts of the data by different processors acting in parallel and then combining the individual results in an incremental way. In this manner, the proposed methodology will pave to way to its application to real incremental learning situations.

The idea of this approach is to deal with distributed learning problems through distributing vertically the data and performing a feature selection process which can be carried out at separate nodes. Since the computational complexity of most of feature selection methods is affected by the number of features, the complexity in each node will be reduced with respect to the centralized approach. Then, the selection procedure required for the data reduction will be integrated with the classifier learning. For the feature selection step, we choose the χ^2 metric (see Chapter 2), because of its simplicity and effectiveness. However, this filter requires data to be discrete, so a discretization stage has to be added to preprocess the data. Finally, a classifier is necessary, and the well-known naive Bayes (see Appendix I, Section I.5) was chosen. This decision has been made because after performing the three stages in each node (discretization, selection and classification), the learned models are combined in a incremental manner, and naive Bayes has some characteristics that makes it inherently incremental. With the proposed methodology, it is expected that the global learning process will be sped up and so become more computationally efficient.

10.5.1 The proposed method

As stated before, distributed feature selection on vertically partitioned data has not been deeply explored yet. Distributed methods usually consist of three stages:

1. Partition of the dataset (if the dataset is not distributed from origin).
2. Application of learning methods in each node. In the case of the method proposed herein, it consists of three steps:
   (a) Discretization.
   (b) Feature selection.
   (c) Classification.
3. Combination of the results.

The interest of this work relies on the independence of the methodology, that can be performed on all the nodes at the same time. Besides, the novelty in the combination stage is that it is done in an incremental manner. As explained before, the learning methodology to be applied to each node consists of three steps: discretization, feature selection and classification (see Figure 10.2).

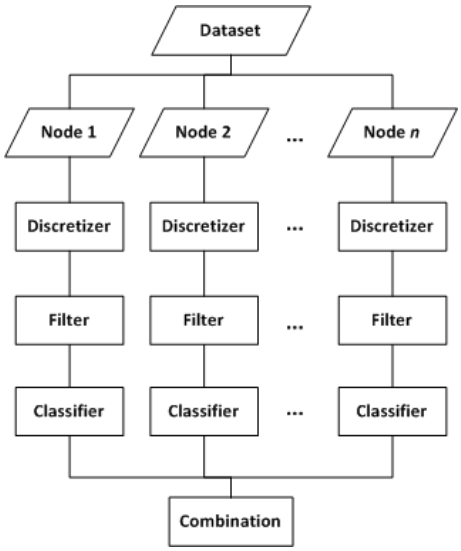


Figure 10.2: Flow chart of proposed methodology

10.5.1.1 Partition of the Dataset

In some cases, data can be originally distributed by features. In this manner, different features belonging to the same sample are recorded in different locations. Each node gathers the values for one or more features for a given instance and then, each node has a different "view" of the data. For instance, a sensor network usually records a single feature in each sensor. Another example may be a patient that performs several medical tests in different hospitals. In such these situations, a distributed learning approach can be much more efficient computationally than moving all distributed datasets into a centralized site for learning the global model. Moreover, even when data are stored in a single site, distributed learning can be also useful to speed up the process.

As most of the datasets publicly available are stored in a centralized manner, the first step consists of partitioning the dataset, i.e. dividing the original dataset into several disjoint subsets of approximately the same size that cover the full dataset. As mentioned in the introduction, in this research the partition will be done vertically. Notice that this step could be eliminated in a real distributed situation.

10.5.1.2 Learning Methods

In this methodology, the learning stage consists of three steps: discretizer, filter and classifier, which will be following described.

Discretizer

Many filter algorithms are shown to work effectively on discrete data (Liu & Setiono, 1997), so discretization is recommended as a previous step. The well-known k-means discretization algorithm (Tou & González, 1977; Ventura & Martinez, 1995) was chosen because of its simplicity and effectiveness. K-means moves the representative weights of each cluster along an unrestrained input space, where each feature is discretized independently, making it suitable for our purposes. This clustering algorithm operates on a set of data points and assumes that the number of clusters to be determined (k) is given. The partition is done based on certain objective function. The most frequently used criterion function in k-means is minimizing the squared error ϵ between the centroids μ_i of clusters $c_i, i = 1, \ldots, k$ and the samples in those clusters

$$\epsilon = \sum_{x \in c_i} |x - \mu_i|^2$$

Let C be the set of clusters and $|C|$ its cardinality. For each new sample x, the discretizer works as follows,

- *If* $|C| < k$ and $x \notin C$ *then* $C = \{x\} \cup C$, i.e. if the maximum number of clusters was not reached already and the new sample is not in C, then create a new cluster with its centroid in x.
- *else*
  1. Find the closest cluster to x.
  2. Update its centroid μ as the average of all values in that cluster.

The method assigns at most k clusters. Notice that the number of clusters is the minimum between the parameter k and the number of different values in the feature.

Filter

The χ^2 method (Liu & Setiono, 1995) evaluates features individually by measuring their chi-squared statistic with respect to the class labels. The χ^2 value of an attribute is defined as:

$$\chi^2 = \sum_{i=1}^{t} \sum_{j=1}^{l} \frac{(A_{ij} - E_{ij})^2}{E_{ij}} \tag{10.2}$$

where

$$E_{ij} = R_i * L_j / S \tag{10.3}$$

t being the number of intervals (number of different values in a feature), l the number of class labels, A_{ij} the number of samples in the i-th interval, j-th class, R_i the number of samples in the i-th interval, L_j the number of samples in the j-th class, S the total number of samples, and E_{ij} the expected frequency of A_{ij}. Note that the size of the matrices is related to the number of intervals. In this manner, a very large k in the discretizer will lead to a very large size of the matrices A and E. A very large matrix is computationally expensive to update and this should be taken into account for real-time applications.

After calculating the χ^2 value of all considered features in each node, these values can be sorted with the largest one at the first position, as the larger the χ^2 value, the more important the feature is. This will provide an ordered ranking of features, and a threshold needs to be established. In these experiments, the choice is to estimate the threshold from the effect on the training set, specifically using 10% of the training dataset available at each node so as to speed up the process. The selection of this threshold must take into account two different criteria: the training error, e, and the percentage of features retained, m. Both values must be minimized to the maximum possible extent. The fitness function is showed in equation (10.4), in which the function $f(v)$ is calculated using those features for which the χ^2 value is above v.

$$f(v) = \alpha e(v) + (1 - \alpha) m(v) \tag{10.4}$$

α being a value in the interval [0,1] that measures the relative relevance of both values. Following the recommendations given by de Haro García (2011), a value of $\alpha = 0.75$ was chosen, since in general the error minimization is more important than storage reduction. For the possible values of the threshold v, three options were considered for the experimental part:

- $v = mean(\chi^2)$
- $v = mean(\chi^2) + std(\chi^2)$
- $v = mean(\chi^2) + 2std(\chi^2)$

Classifier

Among the broad range of classifiers available in the literature, the naive Bayes method (see Appendix I, Section I.5) was chosen for the classification step. This classifier is simple, efficient and robust to noise and irrelevant attributes. Besides, it requires a small amount of input data to estimate the necessary parameters for classification.

Given a set of l mutually exclusive and exhaustive classes $c_1, c_2, \ldots, c_l$, which have prior probabilities $P(c_1), P(c_2), \ldots, P(c_l)$, respectively, and n attributes $a_1, a_2, \ldots, a_n$ which for a given instance have values $v_1, v_2, \ldots, v_n$ respectively, the posterior probability of class c_i occurring for the specified instance can be shown to be proportional to

$$P(c_i) \times P(a_1 = v_1 \text{ and } a_2 = v_2 \ldots \text{ and } a_n = v_n | c_i) \tag{10.5}$$

Making the assumption that the attributes are independent, the value of this expression can be calculated using the product

$$P(c_i) \times P(a_1 = v_1 | c_i) \times P(a_2 = v_2 | c_i) \times \cdots \times P(a_n = v_n | c_i) \tag{10.6}$$

This product is calculated for each value of i from 1 to l and the class which has the largest value is chosen. Notice that this method is suitable for a dynamic space of input features.

10.5.1.3 Combination of the results

After performing the previous stages, the methodology will return as many trained classifiers as nodes we have. These classifiers are trained using only the features selected in each node. The final step consists of combining all the trained classifiers in an incremental manner, in order to have a unique classifier trained on the subset of features as a result of the union of the features selected in every node. This combination is possible because the naive Bayes classifier is inherently incremental. In this algorithm, each feature makes an independent contribution toward the prediction of a class, as stated in the previous section.

Notice that the main contribution of this methodology relies in this merging step. The formulation of the naive Bayes classifier allows to build an exact solution, i.e. the same as would be obtained in batch learning. For this reason, the solution achieved is more reliable than other schemes, such as voting. Moreover, this methodology is flexible, since it can work independently of the number of nodes, the number of features selected and so on.

10.5.2 Experimental setup

In this case, in addition to the traditional approach of evaluating the performance of an algorithm in terms of test accuracy, our proposed distributed algorithm will be also evaluated in terms of speed-up (Bramer, 2007). Speed-up experiments evaluate the performance of the system with respect to the number of nodes for a given dataset. We measure the training time as the number of nodes is increased. This shows how much a distributed algorithm is faster than the serial (one processor) version, as the dataset is distributed to more and more nodes. We can define two performance metrics associated with speep-up:

- The *speedup factor* S_n is defined by $S_n = \frac{R_1}{R_n}$, where R_1 and R_n are the training times of the algorithm on a single and n nodes, respectively. This factor measures how much the training time is faster using n nodes instead of just one. The ideal case is that $S_n = n$, but the usual situation is that $S_n < n$ because of communication or other overheads. Occasionally, it can be a value greater than n, in the case of what is known as *superlinear* speedup.

- The *efficiency* E_n of using n nodes instead of one is defined by $E_n = \frac{S_n}{n}$, i.e. the speedup factor divided by the number of nodes.

Five binary classic datasets were selected for these experiments, the main characteristics of them can be consulted in Appendix I, Section I.2.2: Madelon, Mushrooms, Ozone and Spambase. A holdout validation was performed, using the common partition 2/3 for training and 1/3 for test (see Appendix I, Section I.6). Then, the training data have been scattered across either 2, 4, or 8 nodes; 1 node was also considered to perform a comparative study with the standard centralized approach. Experiments were run 10 times with random partitions of the datasets in order to ensure reliable results. We use the methodology proposed by Demšar (2006) to perform a statistical comparison of the algorithms over the multiple data sets. First a Friedman test (M. Friedman, 1940) is done and then, if significant differences are found, the Bonferroni-Dunn test (Dunn, 1961) is considered.

10.5.3 Experimental results

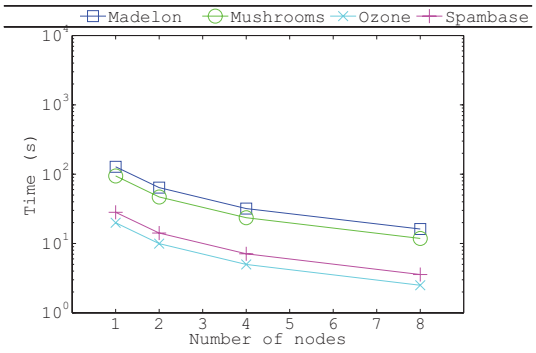
Table 10.20 shows the training time of the algorithm for the different datasets and number of nodes. As can be seen, the training time is dramatically reduced as the number of nodes is increased. Statistical tests demonstrate that doubling the number of nodes obtains significantly better results in terms of time. Table 10.21 shows the test accuracy on the different datasets for the different number of nodes. In general terms, the accuracy is maintained as the number of nodes in increased. Statistical tests prove this fact.

Table 10.20: Training time (s).

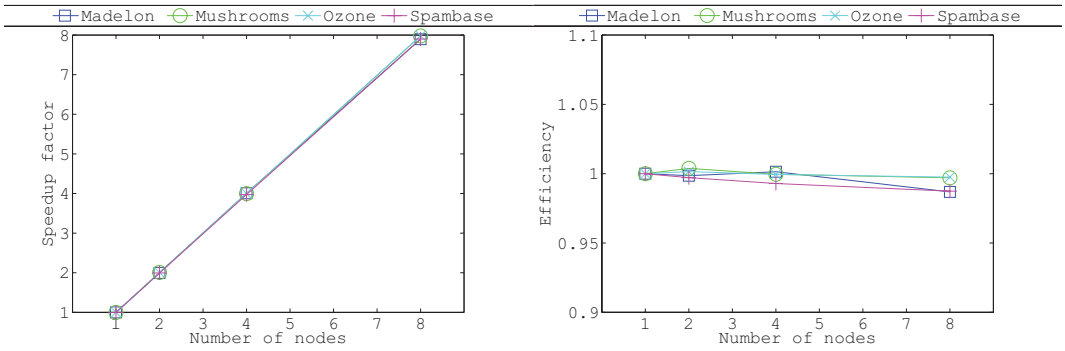
	Number of nodes			
	1	2	4	8
Madelon	127.74	63.96	31.89	16.18
Mushrooms	94.28	46.96	23.58	11.82
Ozone	19.95	9.96	4.99	2.50
Spambase	28.20	14.14	7.10	3.57

Table 10.21: Test accuracy (%).

	Number of nodes			
	1	2	4	8
Madelon	71.38	71.27	71.05	70.82
Mushrooms	93.25	92.04	91.81	91.78
Ozone	86.15	85.93	85.76	85.73
Spambase	88.78	88.72	88.76	88.81



(a) Nodes vs time



(b) Speed up

(c) Efficiency

Figure 10.3: Plots regarding time performance of the algorithm.

Finally, Figure 10.3 shows three graphs representing the different measures related with the time performance of the algorithm. Figure 10.3a plots the training time versus the number of nodes. Figure 10.3b shows a graph of speedup factor against the number of nodes. This form of display is often preferred to the more obvious plot of training time versus the number of nodes, as it makes straightforward to see the impact on the training time of increasing the number of nodes. As can be deduced from Figure 10.3c, the efficiency of the proposed method is close to 1, i.e. increasing the number

of nodes by n divides the training time by the same n. Notice the implications of these results when dealing with high dimensional datasets. The training time may be notably reduced without compromising the classification accuracy. In this manner, the proposed methodology allows to deal with problems which were intractable with classical approaches. However, this is a preliminary attempt and further experiment should be done on large-scale data, for example on microarray DNA classification.

10.6 Summary

Feature selection is usually applied in a centralized manner. However, if the data are distributed, feature selection may take advantage of processing multiple subsets in sequence or concurrently. The need to use distributed feature selection can be two-fold. On the one hand, with the advent of network technologies, the data are sometimes distributed in multiple locations and often with multiple parties. On the other hand, most existing feature selection algorithms do no scale well and their efficiency significantly deteriorates when dealing with large-scale data.

In this chapter several methodologies for distributing the process of feature selection have been proposed. First, we tackled the most common distribution in the literature: the horizontal partition. Then, we applied the same algorithm to data partitioned by features, showing satisfactory results.

A case of study was also presented for dealing with microarray data, in which we have opted for a vertical partition. In light of the experimental results, the most important advantage of our distributed method is the large reduction in execution time whilst maintaining the accuracy at reasonable levels, and sometimes even improving it. Moreover, our method has the additional advantage of allowing an easy parallel implementation. As could be seen in this chapter, the application of the filter algorithm to each subset of features is independent of all the remaining subsets, so all the subsets can be processed at the same time. It is worth mentioning that there is little communication among the nodes of parallel execution, only in the combination step to compute the final subset of selected features.

Finally, our last proposal consisted of performing a vertical partition of the data in which feature selection and classification are executed in parallel and, at the end, the learned models obtained from each node are combined in an incremental manner.

The proposed methodology has been tested considering different number of nodes to distribute the data. The experimental results showed that, in most of the datasets, increasing the number of nodes did not lead to a significant degradation in classification accuracy. As an additional advantage, the larger the number of nodes, the shorter the time required for the computation. Notice that this was a preliminary study, tested up to 8 nodes, and further experimentation is necessary to ensure these conclusions.

CHAPTER 11

Conclusions and future work

Continual advances in computer-based technologies have enabled researchers and engineers to collect data at an increasingly fast pace. To address this challenge, feature selection becomes an imperative preprocessing step which needs to be adapted and improved to handle high-dimensional data. This thesis is devoted to studying feature selection methods and their adequacy to be applied to large-scale data. The tendency nowadays is two-fold: on the one hand, to improve and to extend the existing methods to address the new challenges associated to high-dimensionality. And, on the other hand, to develop novel techniques to directly solving the arising challenges.

The first part of this thesis has dealt with the critical analysis of existing feature selection methods, by checking their appropriateness toward different challenges and aiming at being able to provide recommendations. The following issues have been addressed:

- **Critical review of the most popular feature selection methods in the literature.** A total of eleven feature selection methods were applied over eleven synthetic datasets, covering phenomena such as presence of irrelevant and redundant features, noise in the data or interaction between attributes. To test the effectiveness of the studied methods, an evaluation measure was introduced trying to reward the selection of the relevant features and to penalize the inclusion of the irrelevant ones. Some cases of study were also presented in order to decide among methods that showed similar behaviors and helping to find their adequacy in different situations. In light of the results obtained from the experiments, we suggest the use of filters (particularly, ReliefF), since they are independent of the induction algorithm and are faster than embedded and wrapper methods, as well as having a good generalization ability.

- **Analysis of the behavior of feature selection in DNA microarray classification.** The advent of DNA microarray data has posed a serious challenge for machine learning researchers, because of the large input dimensionality and small sample size. Thus, feature selection became an imperative step, in order to reduce the number of features (genes). We have analyzed the recent literature in order to describe in broad brushstrokes the trends in the development of feature selection methods for microarray data. In order to provide a complete picture of the topic, we have also mentioned the most common validation techniques. Since there is no consensus in the literature about this issue, we have provided some guidelines. Finally, we have proposed a practical evaluation for feature selection methods using microarray datasets in which we analyze the results obtained. This experimental study tries to show in practice the problematics that we have explained in theory. To this end, a suite of 9 widely-used binary datasets was chosen to apply over them 7 classical feature selection methods. In order to obtain the final classification accuracy, 3 well-known classifiers were used. This large set of experiments also aims at facilitating future comparative studies when a researcher proposes a new method.

- **Application of classic feature selection methods to real problems.** Feature selection plays a crucial role in many real applications. To illustrate this fact, two real problems were presented, in which feature selection has demonstrated to be useful to improve performance. First, we tackled the problem of tear film lipid layer classification. The time required by existing approaches dealing with this issue prevented their clinical use because they could not work in real time. A methodology for improving this classification problem was proposed, which included the application of feature selection methods. In clinical terms, the manual process done by experts could be automated with the benefits of being faster and unaffected by subjective factors, with maximum accuracy over 97% and processing time under 1 second. The second real scenario was the K-complex classification, which is a key aspect in sleep studies. Several feature selection methods were applied in combination with different machine learning algorithms, trying to achieve a low false positive rate whereas maintaining the accuracy. With the inclusion of this feature selection stage, the results improved significantly for all the classifiers tested.

- **Scalability in feature selection.** With the proliferation of large-scale data, researchers must focus not only on the accuracy but also on the scalability of the existing methods. First, the effectiveness of feature selection on the scalability of training algorithms for artificial neural networks (ANNs) was evaluated, both in

classification and regression tasks. The experimental results showed that feature selection as a preprocessing step is beneficial for the scalability of ANNs, even allowing certain algorithms to be able to train on some datasets in cases in which it was impossible due to the spacial complexity. Then, an analysis of the scalability of feature selection methods was presented, an issue which has not received much consideration in the literature. A total of eleven feature selection methods were evaluated, belonging to the three main groups of feature selection methods: filters, wrappers and embedded. The experimental study was performed with artificial datasets, so as to be able to assess the degree of closeness to the optimal solution in a confident way. In order to determine the scalability of the methods, new measures were proposed, based not only on accuracy but also on execution time and stability. Considering the experimental results, filters seemed to be the most scalable feature selection methods. Specifically, FCBF obtained the best performance in terms of scalability. As for the ranker methods, ReliefF is a good choice when having a small number of features, at the expense of a long training time. For this reason, when dealing with extremely-high datasets, Information Gain demonstrated better scalability properties. As future work, we plan to extend this research to real datasets in order to check if the conclusions drawn inhere can be extrapolated.

The second part of this thesis has been dedicated to proposing novel techniques for large-scale feature selection. Although the benefits of feature selection have been extensively proved, new feature selection methods are constantly emerging using different strategies. In fact, the current tendency in feature selection is not toward developing new algorithmic measures, but toward favoring the combination or modification of existing algorithms. Different strategies to deal with the new problematics derived from the big data explosion have been proposed:

- **A combination of discretization and feature selection methods.** Most feature selection algorithms only work on discrete data, so a common practice is to discretize the data before conducting feature selection. However, many studies entrust this task to default discretizers, as the case of the Weka tool. In an attempt to shed light on this issue, a framework was proposed which consists of combining discretization, filtering and classification methods to be applied to different scenarios. The adequacy of the combination method in terms of improvement in classification accuracy has been demonstrated on intrusion detection systems, microarray DNA data and a large suite of multiclass datasets.

- **An ensemble of filters and classifiers.** The goal of the proposed ensemble is to reduce the variability of the results obtained by different feature selection methods, based on the assumption that combining the output of multiple experts is better than the output of any single expert. Two general approaches were presented, based on the role the classifier plays. The first approach classifies as many times as filters there are, whilst the second one classifies only once with the result of joining the different subsets selected by the filters. A total of five implementations of the two approaches were proposed, tested on synthetic datasets, UCI classical datasets and microarray data. The appropriateness of using an ensemble instead of a single filter remained demonstrated, considering that for all scenarios tested, the ensemble was always the more successful solution.

- **A framework for cost-based feature selection.** There are some situations in which a user is not only interested in selecting the more accurate subset of features, but also in reducing the costs that may be associated to it. For example, for medical diagnosis, symptoms observed with the naked eye are costless, but each diagnostic value extracted by a clinical test is associated with its own cost and risk. Bearing this in mind, a new framework for cost-based feature selection was proposed. The objective was to solve problems in which it is interesting not only to minimize the classification error, but also to reduce the associated costs. The proposed framework consists of adding a new term to the evaluation function of any filter in order to reach a trade-off between a filter metric (e.g. correlation) and the cost associated to the input features. To test the adequacy of the proposed framework, three representative filters were chosen, applied to a broad suite of different datasets including a real-life problem. The obtained results demonstrated that the approach is sound and allows the user to reduce the cost without compromising the classification error significantly. We have left open the question of applying this framework together with other feature selection methods, such as embedded or wrappers, which might help to improve the classification accuracy. It would be also interesting to combine the cost-based feature selection with distributed learning.

- **Distributed and parallel feature selection.** When dealing with large-scale data, a possible strategy is to distribute the learning task over a number of processors. The main goal was to distribute the feature selection process, expecting that the execution time would be considerably shortened and the accuracy would not degrade to poor values. Several proposals have been presented in this chapter, in an attempt to deal with both horizontal and vertical partitioning of the data. The experimental results showed that our proposals have been able to reduce

the running time significantly with respect to the standard (centralized) feature selection algorithms. Actually, in terms of execution time, the behavior is excellent, being this fact the most important advantage of our method. Furthermore, with regard to the classification accuracy, our distributed approaches were able to match and in some cases even improve the standard algorithms applied to the non-partitioned datasets. As future work, it would be interesting to distribute other feature selection techniques, such as wrapper or embedded methods, as well as trying to improve the already proposed methods. On the one hand, the methodology for horizontal partitioning could be improved if the class distribution would be considered to distribute the samples across the different nodes. On the other hand, the incremental vertical partitioning leaves as future work to try another distributed approach in which all nodes share their results after each step. In this sense, the difficulty of this future line of research lies on the fact that all the methods have to be adapted to work in an incremental fashion. Finally, we plan to perform parallel feature selection using a cluster computing framework called Spark. This distributed programming model has been proposed to handle large-scale data problems. However, most existing feature selection techniques are designed to run in a centralized computing environment and their implementations have to be adapted to this new technology. By using Spark, the final user would be released of the decision of how to distribute the data.

As can be seen, this thesis covers a broad suite of problems arisen from the advent of high dimensionality. The proposed approaches have demonstrated to be sound, and it is expected that their contribution will be important in the next years, since feature selection for large-scale data is likely to continue to be a trending topic in the near future.

In addition to the future works that have been mentioned below, which are already in progress, there are some other lines of research that we would like to tackle. First, it would be interesting to study the relationship between data complexity and the accuracy of the feature selection process in classification tasks. There are several factors which can affect accuracy, such as the shape of the decision class boundary, the amount of overlap among the classes, the proximity of two classes and the number of informative samples available for training (Basu & Ho, 2006). Although some of these aspects have been commented in Chapter 4, it is necessary an analysis in depth. In a similar manner, a future line of research might be to develop distributed feature selection methods which address the possible data heterogeneity in the different partitions. There might

even exist situations in which a certain class would not be represented in some of the partitions, so this kind of problems need to be tackled.

Finally, when having a distributed feature selection method which provides an ordered ranking of features for each partition, it is not easy to combine the different rankings. For this reason, we plan to devise feature selection methods which learn from preference judgments. Thus, several rankings obtained from different feature selection methods can be combined in such a manner that the new method will take advantage of the coherence in the ranker given by the different methods.

APPENDIX

Materials and methods

This appendix describes the materials and methods used in this thesis, as well as the evaluation techniques and performance metrics employed.

I.1 Software tools

The experiments performed in this thesis were executed using the software tools Matlab and Weka, which will be following described.

- Matlab (MATLAB, 2013) is a numerical computing environment, well known and widely used by scientific researchers. It was developed by MathWorks in 1984 and its name comes from *Matrix Laboratory*. Matlab allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.

- Weka (Waikato Environment for Knowledge Analysis) (M. Hall et al., 2009) is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

I.2 Datasets

The feature selection methods used or proposed in this thesis are tested over a diverse suite of datasets. Some of them are synthetic (i.e. the relevant features are known *a priori*), others are classical datasets extracted from the widely used UCI repository (Asuncion & Newman, 2007) and others belong to the popular category of DNA microarray datasets. The following subsections will present the main characteristics of the datasets employed in this thesis.

I.2.1 Synthetic datasets

Several authors choose to use artificial data since the desired output is known, therefore a feature selection algorithm can be evaluated with independence of the classifier used. Although the final goal of a feature selection method is to test its effectiveness over a real dataset, the first step should be on synthetic data. The reason for this is two-fold (Belanche & González, 2011):

1. Controlled experiments can be developed by systematically varying chosen experimental conditions, like adding more irrelevant features or noise in the input. This fact facilitates to draw more useful conclusions and to test the strengths and weaknesses of the existing algorithms.

2. The main advantage of artificial scenarios is the knowledge of the set of optimal features that must be selected, thus the degree of closeness to any of these solutions can be assessed in a confident way.

The synthetic datasets used in this thesis try to cover different problems: increasing number of irrelevant features, redundancy, noise in the output, alteration of the inputs, non-linearity of the data, etc. These factors complicate the task of the feature selection methods, which are very affected by them as it will be shown afterwards. Besides, some of the datasets have a significantly higher number of features than samples, which implies an added difficulty for the correct selection of the relevant features. Table I.1 shows a summary of the main problems covered by them, as well as the number of features and samples and the relevant attributes which should be selected by the

feature selection methods. Notice that it is emphasized the main characteristic of each dataset, but it can have other characteristics too.

Table I.1: Summary of the synthetic datasets used. "Corr." stands for "Correlation".

Dataset	No. of features	No. of samples	Relevant features	Corr.	Noise	Non linear	No. feat >> No. samples
Corral	6	32	1-4	✓			
Corral-100	99	32	1-4	✓			✓
Led-25	24	50	1-7		✓		
Led-100	99	50	1-7		✓		✓
Madelon	500	2400	1-5		✓	✓	
Monk1	6	122	1,2,5			✓	
Monk2	6	122	1-6			✓	
Monk3	6	122	2,4,5		✓		
Parity3+3	12	64	1-3			✓	
SD1*	4020	75	G_1, G_2				✓
SD2*	4040	75	$G_1 - G_4$				✓
SD3*	4060	75	$G_1 - G_6$				✓
XOR-100	99	50	1,2			✓	✓

* G_i means that the feature selection method must select only one feature within the i-th group of features.

I.2.1.1 CorrAL

The CorrAL dataset (John, Kohavi, & Pfleger, 1994) has six binary features (i.e. $f_1, f_2, f_3, f_4, f_5, f_6$), and its class value is $(f_1 \wedge f_2) \vee (f_3 \wedge f_4)$. Feature f_5 is irrelevant and f_6 is correlated to the class label by 75%.

CorrAL-100 (Kim et al., 2010) was constructed by adding 93 irrelevant binary features to the previous CorrAL dataset. The data for the added features were generated randomly. Both datasets (CorrAL and CorrAL-100) have 32 samples that are formed by considering all possible values of the four relevant features and the correlated one (2^5). The correct behavior for a given feature selection method is to select the four relevant features and to discard the irrelevant and correlated ones. The correlated feature is redundant if the four relevant features are selected and, besides, it is correlated to the class label by 75%, so if one applies a classifier after the feature selection process, a 25% of error will be obtained.

I.2.1.2 XOR-100

XOR-100 (Kim et al., 2010) has 2 relevant binary features and 97 irrelevant binary features (randomly generated). The class attribute takes binary values and the dataset consists of 50 samples. Features f_1 and f_2 are correlated with the class value with XOR operation (i.e., class equals $f_1 \oplus f_2$). This is a hard dataset for the sake of feature selection because of the small ratio between number of samples and number of features and due to its non-linearity (unlike CorrAL dataset, which is a multi-variate dataset).

I.2.1.3 Parity3+3

The parity problem is a classic problem where the output is $f(x_1, \ldots, x_n) = 1$ if the number of $x_i = 1$ is odd and $f(x_1, \ldots, x_n) = 0$ otherwise. The Parity3+3 dataset is a modified version of the original parity dataset. The target concept is the parity of three bits. It contains 12 features among which 3 are relevant, another 3 are redundant (repeated) and other 6 are irrelevant (randomly generated).

I.2.1.4 The Led problem

The LED problem (Breiman, 1993) is a simple classification task that consists of, given the active leds on a seven segments display, identifying the digit that the display is representing. Thus, the classification task to be solved is described by seven binary attributes (see Figure I.1) and ten possible classes available ($C = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$). A 1 in a attribute indicates that the led is active, and a 0 indicates that it is not active.

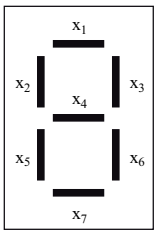


Figure I.1: LED Scheme

Two versions of the Led problem will be used: the first one, Led25, adding 17 irrelevant attributes (with random binary values) and the second one, Led100, adding 92

irrelevant attributes. Both versions contain 50 samples. The small number of samples was chosen because we are interested in dealing with datasets with a high number of features and a small sample size. Besides, different levels of noise (altered inputs) have been added to the attributes of these two versions of the Led dataset: 2%, 6%, 10%, 15% and 20%. In this manner, the tolerance to different levels of noise of the feature selection methods tested will be checked. Note that, as the attributes take binary values, adding noise means assigning to the relevant features an incorrect value.

I.2.1.5 The Monk problems

The MONK's problems (Thrun et al., 1991) rely on an artificial robot domain, in which robots are described by six different discrete attributes $(x_1, \ldots, x_6)$. The learning task is a binary classification task. The logical description of the class of the Monk problems are the following:

- Monk1: $(x_1 = x_2) \vee (x_5 = 1)$
- Monk2: $(x_n = 1)$ exactly two $n \in 1, 2, 3, 4, 5, 6$
- Monk3: $(x_5 = 3 \wedge x_4 = 1) \vee (x_5 \neq 4 \wedge x_2 \neq 3)$

In the case of Monk3, among the 122 samples, 5% are misclassifications, i.e. noise in the target.

I.2.1.6 SD1, SD2 and SD3

These three synthetic datasets (SD1, SD2 and SD3) (Zhu et al., 2010) are challenging problems because of their high number of features (around 4000) and the small number of samples (75), besides of a high number of irrelevant attributes.

SD1, SD2 and SD3 are three-class datasets with 75 samples (each class containing 25 samples) generated based on the approach described by Díaz-Uriarte and De Andres (2006). Each synthetic dataset consists of both relevant and irrelevant features. The relevant features in each dataset are generated from a multivariate normal distribution using mean and covariance matrixes (Zhu et al., 2010). Besides, 4000 irrelevant features are added to each dataset, where 2000 are drawn from a normal distribution of N(0,1) and the other 2000 are sampled with a uniform distribution U[-1,1]. It is necessary to introduce some new definitions of multiclass relevancy features: full class relevant (FCR) and partial class relevant (PCR) features. Specifically, FCR denotes genes (features) that serve as candidate biomarkers for discriminating all cancer types. However, PCR are genes (features) that distinguish subsets of cancer types.

SD1 is designed to contain only 20 FCR and 4000 irrelevant features. Two groups of relevant genes are generated from a multivariate normal distribution, with 10 genes in each group. Genes in the same group are redundant with each other and the optimal gene subset for distinguishing the three classes consists of any two relevant genes from different groups.

SD2 is designed to contain 10 FCR, 30 PCR, and 4000 irrelevant features. Four groups of relevant, i.e., FCR and PCR, genes are generated from a multivariate normal distribution, with 10 genes in each group. Genes in each group are redundant to each other and in this dataset, only genes in the first group are FCR genes while genes in the three last groups are PCR genes. The optimal gene subset to distinguish all the three classes consists of four genes, one FCR gene from the first group and three PCR genes each from one of the three remaining groups.

SD3 has been designed to contain only 60 PCR and 4000 irrelevant features. Six groups of relevant genes are generated from a multivariate normal distribution, with 10 genes in each group. Genes in the same group are designed to be redundant to each other and the optimal gene subset to distinguish all the three classes thus consists of six genes with one from each group.

It has to be noted that the easiest dataset in order to detect relevant features is SD1, since it contains only FCR features and the hardest one is SD3, due to the fact that it contains only PCR genes, which are more difficult to detect.

I.2.1.7 Madelon

The Madelon dataset (Guyon, 2006) is a 2 class problem originally proposed in the NIPS'2003 feature selection challenge. The relevant features are situated on the vertices of a five dimensional hypercube. Five redundant features were added, obtained by multiplying the useful features by a random matrix. Some of the previously defined features were repeated to create 10 more features. The other 480 features are drawn from a Gaussian distribution and labeled randomly. This dataset presents high dimensionality both in number of features and in number of samples and the data were distorted by adding noise, flipping labels, shifting and rescaling. For all these reasons, it conforms a hard dataset for the sake of feature selection.

I.2.2 Classical datasets

Several classical datasets are used in this thesis as a benchmark for testing feature selection methods. Most of them can be downloaded from the UCI Machine Learning Repository (Asuncion & Newman, 2007), which is a collection of databases, domain theories, and data generators that are used by the machine learning community for the empirical analysis of machine learning algorithms.

Table I.2: Dataset description for binary classic datasets

Dataset	Attributes	Samples	Distribution
Hepatitis	19	155	55% - 45%
Liver	6	345	42% - 58%
Madelon	500	2400	50% - 50%
Magic04	10	19020	35% - 65%
Mushrooms	112	8124	48% - 52%
Ozone	72	2536	97% - 3%
Pima	8	768	65% - 35%
Spambase	57	4601	61% - 39%
Splice	60	1000	48% - 52%

Table I.2 depicts the main characteristics of the binary classic datasets with only training test employed in this thesis, such as the number of attributes, samples and the class distribution. Table I.3 shows the description of the binary datasets that were

originally divided to train and test datasets. Finally, Table I.4 reports the number of classes, samples and features for the multiclass datasets considered in this research.

Table I.3: Dataset description for binary classic datasets with train and test sets

Dataset	Attributes	Train samples	Test samples
Forest	54	100000	50620
KDD Cup 99	42	494021	311029
MNIST	748	60000	10000

Table I.4: Dataset description for multiclass classic datasets

Dataset	Classes	Samples	Features
Connect 4	3	67557	42
Dermatology	6	366	34
Glass	6	214	10
Iris	3	150	4
KDD SC	6	600	60
Landsat	6	4435	36
Letter	26	20000	16
MFeat-Factor	10	2000	216
MFeat-Fourier	10	2000	76
MFeat-Karhounen	10	2000	64
MFeat-Pixel	10	2000	240
MFeat-Zernike	10	2000	47
MLL-Leukemia	3	57	12582
Optdigits	10	3823	64
Pendigits	10	7494	16
Sat	6	4435	36
Segmentation	7	2310	19
Splice	3	3190	61
Thyroid	3	3772	21
Vehicle	4	846	18
Vowel	11	990	13
Waveform	3	5000	21
Wine	3	178	13
Yeast	10	1033	8

Table I.5: Dataset description for multiclass classic datasets with train and test sets

Dataset	Classes	Features	Train samples	Test samples
Connect-4	3	42	60000	7557
Isolet	26	617	6238	1236

I.2.3 Datasets for regression

Although most of this thesis is devoted to improve classification tasks, several datasets for regression are also used, whose main characteristics are depicted in Table I.6. Notice that some of them (Covertype and MNIST) can be used both as classification and regression datasets. Covertype dataset can be downloaded from the UCI Repository (Asuncion & Newman, 2007) whilst MNIST is available on `http://yann.lecun.com/exdb/mnist/`. Friedman and Lorenz are artificial datasets. Friedman is defined by the equation $y = 10sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \sigma(0,1)$ where the input attributes $x_1, \ldots, x_{10}$ are generated independently, each of which is uniformly distributed over interval $[0, 1]$. Variables $x_6 - x_{10}$ are randomly generated. On the other hand, Lorenz is defined by the simultaneous solution of three equations $\frac{dX}{dt} = \delta Y - \delta X$, $\frac{dY}{dt} = -XZ + rX - Y$, $\frac{dZ}{dt} = XY - bZ$, where the systems exhibits chaotic behavior for $\delta = 10$, $r = 28$ and $b = \frac{8}{3}$.

Table I.6: Dataset description for datasets used in regression tasks

Dataset	Features	Train samples	Test samples
Forest	54	100000	50620
Friedman	10	1000000	100000
Lorenz	8	1000000	100000
MNIST	748	60000	10000

I.2.4 DNA microarray datasets

DNA microarray data classification (see Chapter 4) is a serious challenge for machine learning researchers due of its high dimensionality and small sample size. Typical values are around 10 000 gene expressions and a hundred or less tissue samples. For this reason, this type of datasets are usually employed to test the efficiency of a feature selection method. Some datasets were originally divided to training and test set whilst others have only a training set. On the one hand, Table I.7 shows the main characteristics of the datasets employed in this thesis having a unique training set (for example, to be applied a 10-fold cross validation). On the other hand, Tables I.8 and I.9 visualize the characteristics of the datasets used with separated training and test sets. For the sake of comparison, datasets with only training set were randomly divided using the common rule 2/3 for training and 1/3 for testing. This division introduces a more challenging scenario, since in some datasets, the distribution of the classes in the training set differs from the one in the test set. Notice that throughout this thesis, both versions of the datasets are used in different chapters. All the datasets described in this section are available for download in (Broad Institute, n.d.) and (Kent Ridge, n.d.).

Table I.7: Dataset description for binary microarray datasets

Dataset	Attributes	Samples	Distribution
Brain	12625	21	33% - 67%
Breast	24481	97	47% - 53%
CNS	7129	60	35% - 65%
Colon	2000	62	35% - 65%
DLBCL	4026	47	49% - 51%
GLI	22283	85	31% - 69%
Leukemia	7129	72	34% - 66%
Lung	12533	181	17% - 83%
Myeloma	12625	173	21% - 79%
Ovarian	15154	253	36% - 64%
Prostate	12600	136	43% - 57%
SMK	19993	187	48% - 52%

Table I.8: Dataset description for binary microarray datasets with train and test sets

Dataset	Attributes	Samples		Train	Test
		Train	Test	distribution	distribution
Brain	12625	14	7	36% - 64%	44% - 56%
Breast	24481	78	19	44% - 56%	37% - 63%
CNS	7129	40	20	65% - 35%	65% - 35%
Colon	2000	42	20	67% - 33%	60% - 40%
DLBCL	4026	32	15	50% - 50%	53% - 47%
GLI	22283	57	28	28% - 72%	36% - 64%
Leukemia	7129	38	34	71% - 29%	59% - 41%
Lung	12 533	32	149	50% - 50%	90% - 10%
Myeloma	12625	116	57	22% - 78%	19% - 81%
Ovarian	15 154	169	84	35% - 65%	38% - 62%
Prostate	12 600	102	34	49% - 51%	26% - 74%
SMK	19993	125	62	50% - 50%	44% - 56%

Table I.9: Dataset description for multiple class datasets.

Dataset	Attrib.	Samples		No. of
		Train	Test	classes
GCM	16063	144	46	14
Lymphoma	4026	64	32	9

I.3 Validation techniques

To evaluate the goodness of the selected set of features, it is necessary to have an independent test set with data which have not been seen by the feature selection method. In some cases, the data come originally distributed into training and test sets, so the training set is usually employed to perform the feature selection process and the test set is used to evaluate the appropriateness of the selection. However, not all the datasets come originally partitioned. For overcoming this issue, there exist several validation techniques, and we following describe those used in this thesis.

I.3.1 *k*-fold cross validation

This is one of the most famous validation techniques (Bramer, 2007). The data (D) is partitioned into k non-overlapping subsets $D_1, \ldots, D_k$ of roughly equal size. The learner is trained on $k-1$ of these subsets combined together and then applied to the remaining subset to obtain an estimate of the prediction error. This process is repeated in turn for each of the k subsets, and the cross-validation error is given by the average of the k estimates of the prediction error thus obtained. In the case of feature selection, notice that with this method there will be k subsets of selected features. Common practices are to merge the k different subsets (either by union or by intersection) or to keep the subset obtained in the fold with the best classification result.

I.3.2 Leave-one-out cross validation

This is a variant of k-fold cross validation where k is the number of samples (Bramer, 2007). A single observation is left out each time.

I.3.3 Bootstrap

This is a general resampling strategy (Efron, 1979). A *bootstrap sample* consists of n (being n the number of samples) equally-likely draws with replacement from the original data. Therefore, some of the samples will appear multiple times, whereas others will not appear at all. The learner is designed on the bootstrap sample and tested on the left-out data points. The error is approximated by a sample mean based on independent replicates (usually between 25 and 200). There exist some famous variants of the method such as *balanced bootstrap* or *0.632 bootstrap* (Efron & Tibshirani, 1993). As in the previous methods, there will be as many subsets of features as repetitions of the method.

I.3.4 Holdout validation

This technique consists of randomly splitting the available data into a disjoint pair training-test (Bramer, 2007). A common partition is to use 2/3 for training and 1/3

for testing. The learner is designed on the training data and the estimated error rate is the proportion of errors observed on the test data. This approach is usually employed when some of the datasets in a study come originally divided into training and test sets whilst others do not. In contrast to other validation techniques, a unique set of selected features is obtained.

I.4 Statistical tests

When performing several executions of a method, different results are obtained (e.g. after applying a k-fold cross validation). In this situation, statistical tests may be performed to check if there are significant differences among the medians for each method. In this thesis, the most used statistical methods were Kruskal-Wallis (Wolfe & Hollander, 1973) and a multiple comparison procedure (Tukey's) (Hsu, 1996). The experimental procedure is following detailed:

1. A Kruskal-Wallis test is applied to check if there are significant differences among the medians for each model. In this thesis, we have opted for a level of significance $\alpha = 0.05$.
2. If the non-parametric Kruskal-Wallis test is significant, it means that there exists at least a model that is better than the others. In this case, it is necessary to perform a multiple comparison procedure to figure out which model is the best.
3. Finally, the set of models with performance not significantly worse than the best is obtained. Among the models in this set, it should be selected the simplest one.

I.5 Classification algorithms

If we are dealing with real datasets, the relevant features are not known a priori. Therefore, it is necessary to use a classification algorithm to evaluate the performance of the feature selection, focusing on the classification accuracy. Unfortunately, the class prediction depends also on the classification algorithm used, so when testing a feature selection method, a common practice is to use several classifiers to obtain results as classifier-independent as possible. This section describes the most common

classification algorithms which are used along this thesis. Notice that some of them only can work with categorical features, whereas others require numerical attributes. In the first case, the problem is often solved by discretizing the numerical features. In the second case, it is common to use a conversion method which assigns numerical values to the categorical features.

I.5.1 Support Vector Machine, SVM

A Support Vector Machine (Vapnik, 1998) is a learning algorithm typically used for classification problems (text categorization, handwritten character recognition, image classification, etc.). More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. In its basic implementation, it can only work with numerical data and binary classes.

I.5.2 Proximal Support Vector Machine, PSVM

This method classifies points assigning them to the closest of two parallel planes (in input or feature space) that are pushed as far apart as possible (Fung & Mangasarian, 2001). The difference with a Support Vector Machine (SVM) is that PSVM classifies points by assigning them to one of two disjoint half-spaces. The PSVM leads to an extremely fast and simple algorithm by generating a linear or nonlinear classifier that merely requires the solution of a single system of linear equations.

I.5.3 C4.5

C4.5 is a classifier developed by Quinlan (1993), as an extension of the ID3 algorithm (Iterative Dicotomiser 3). Both algorithms are based in decision trees. A decision tree classifies a pattern doing a descending filtering of it until finding a leaf, that points to the corresponding classification. One of the improvements of C4.5 with respect to ID3 is that C4.5 can deal with both numerical and symbolic data. In order to handle

continuous attributes, C4.5 creates a threshold and depending on the value that takes the attribute, the set of instances is divided.

I.5.4 naive Bayes, NB

A naive Bayes classifier (Rish, 2001) is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. This classifier assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. A naive Bayes classifier considers each of the features to contribute independently to the probability that a sample belongs to a given class, regardless of the presence or absence of the other features. Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In fact, naive Bayes classifiers are simple, efficient and robust to noise and irrelevant attributes. However, they can only deal with symbolic data, although discretization techniques can be used to preprocess the data.

I.5.5 k-nearest neighbors, k-NN

K-Nearest neighbor (Aha, Kibler, & Albert, 1991) is a classification strategy that is an example of a "lazy learner". An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (where k is some user specified constant). If $k = 1$ (as it is the case in this thesis), then the object is simply assigned to the class of that single nearest neighbor. This method is more adequate for numerical data, although it can also deal with discrete values.

I.5.6 Multi-Layer Perceptron, MLP

A multi-layer perceptron (Rumelhart, Hinton, & Williams, 1985; Hornik, Stinchcombe, & White, 1989) is a feedforward artificial neural network model that maps sets of numerical input data onto a set of appropriate outputs. A MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a

nonlinear activation function. MLP utilizes a supervised learning technique called back-propagation for training the network. MLP is a modification of the standard linear perceptron and can distinguish data that are not linearly separable.

I.5.7 One-layer Feedfoward Neural Network, One-layer NN

This algorithm consists of training a single-layer feedfoward neural network using a new supervised learning method proposed by Castillo, Fontenla-Romero, Guijarro-Berdiñas, and Alonso-Betanzos (2002). The method is based on the use of an alternative cost function that measures the errors before the nonlinear activation functions instead of after them, as is normally the case. As an important consequence, the solution can be obtained easily and rapidly because the new cost function is convex. Therefore, the absence of local minima is assured in this situation.

I.5.8 AdaBoost, AB

AdaBoost ("Adaptive Boosting") (Freund & Schapire, 1995), is a meta-algorithm which can be used in conjunction with many other learning algorithms to improve their performance. AdaBoost is adaptive in the sense that subsequent classifiers built are tweaked in favor of those instances misclassified by previous classifiers. It generates and calls a new weak classifier in each of a series of rounds. For each call, a distribution of weights is updated that indicates the importance of examples in the data set for the classification. On each round, the weights of each incorrectly classified example are increased, and the weights of each correctly classified example are decreased, so the new classifier focuses on the examples which have so far eluded correct classification. AdaBoost is sensitive to noisy data and outliers.

I.6 Evaluation measures

In order to evaluate the behavior of the feature selection methods after applying a classifier, several evaluation measures need to be defined.

- *True positive (TP)*: percentage of positive examples correctly classified as so.

- *False positive (FP)*: percentage of negative examples incorrectly classified as positive.
- *True negative (TN)*: percentage of negative examples correctly classified as so.
- *False negative (FN)*: percentage of positive examples incorrectly classified as negative.
- $Sensitivity = \frac{TP}{TP+FN}$
- $Specificity = \frac{TN}{TN+FP}$
- $Accuracy = \frac{TN+TP}{TN+TP+FN+FP}$
- $Error = \frac{FN+FP}{TN+TP+FN+FP}$

I.6.1 Multiple-criteria decision-making

Multiple-criteria decision-making (MCDM) (Zeleny & Cochrane, 1982) is focused on evaluating classifiers from different aspects and produce rankings of them. A multi-criteria problem is formulated using a set of alternatives and criteria. Among many MCDM methods that have been developed up to now, *technique for order of preference by similarity to ideal solution* (TOPSIS) (Hwang & Yoon, 1981) is a well-known method that will be used. TOPSIS finds the best algorithms by minimizing the distance to the ideal solution whilst maximising the distance to the anti-ideal one. The extension of TOPSIS proposed by Olson (2004) is used in this research.

APPENDIX II

Author's key publications and mentions

The contents of the present research have been published in the following specialized journals and forums.

JCR Journals

- Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. *Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset.* Expert Systems with Applications (vol. 38, pp. 5947-5957, 2011)

- Porto-Díaz, I., Bolón-Canedo, V., Alonso-Betanzos, A. and Fontenla-Romero, O. *A study of performance on microarray data sets for a classifier based on information theoretic learning.* Neural Networks (vol. 24, no. 8, pp. 888-896, 2011)

- Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. *An ensemble of filters and classifiers for microarray data classification.* Pattern Recognition (vol. 45, no. 1, pp. 531-539, 2012)

- Peteiro-Barral, D., Bolón-Canedo, V., Alonso-Betanzos, A., Guijarro-Berdiñas, B. and Sánchez-Maroño, N. *Toward the scalability of neural networks through feature selection.* Expert Systems with Applications (vol. 40, no. 8, pp. 2807-2816, 2013)

- Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. *A review of feature selection methods on synthetic data.* Knowledge and Information Systems (vol. 34, no. 3, pp. 483-519, 2013)

- Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. *Data classification using an ensemble of filters.* Neurocomputing (2014). Web reference: `http://dx.doi.org/10.1016/j.neucom.2013.03.067`

- Remeseiro, B., Bolón-Canedo, V., Peteiro-Barral, D., Alonso-Betanzos, A., Guijarro-Berdiñas, B., Mosquera, A., Penedo, M.G., and Sánchez-Maroño, N. *A methodology for improving tear film lipid layer classification.* IEEE Journal of Biomedical and Health Informatics (2014). Web reference: `http://dx.doi.org/10.1109/JBHI.2013.2294732`

- Bolón-Canedo, V., Porto-Díaz, I., Sánchez-Maroño, N. and Alonso-Betanzos, A. *A framework for cost-based feature selection.* Pattern Recognition (2014). Web reference: `http://dx.doi.org/10.1016/j.patcog.2014.01.008`

JCR Journals (Under review process)

- Hernández-Pereira, E., Bolón-Canedo, V., Sánchez-Maroño, N., Álvarez-Estévez, D., Moret-Bonillo, V. and Alonso-Betanzos, A. *A comparison of performance of K-complex classification methods using feature selection.* Information Sciences (2014)

- Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. *A distributed filter approach for microarray data classification.* Applied Soft Computing (2013)

- Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A., Benítez, J.M. and Herrera, F. *An insight into microarray datasets and feature selection methods: a framework for ongoing studies.* Information Sciences (2013)

- Bolón-Canedo, V., Rego-Fernández, D., Peteiro-Barral, D., Alonso-Betanzos, A., Guijarro-Berdiñas, B. and Sánchez-Maroño, N. *On the Scalability of Filter Techniques for Feature Selection on Big Data.* IEEE Computational Intelligence Magazine (2013)

- Peteiro-Barral, D., Bolón-Canedo, V., Fernández-Francos, D., Alonso-Betanzos, A., Guijarro-Berdiñas, B. and Sánchez-Maroño, N. *A Proposal for Online Feature Selection and Classification Based on Chi-Square Filter and One-Layer Artificial Neural Networks.* Knowledge-Based Systems (2014)

Conferences

- Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. *A combination of discretization and filter methods for improving classification performance in KDD Cup 99 dataset.* In Proc. International Joint Conference on Artificial Neural Networks (IJCNN'09), pp. 359-366, Atlanta (GA), USA, June 2009.

- Bolón-Canedo, V., Sánchez-Maroño, N., Alonso-Betanzos, A. and Hernández-Pereira, E. *Feature selection and conversion methods in KDD Cup 99 dataset: A comparison of performance.* In Proc. IASTED International Conference on Artificial Intelligence and Applications (AIA'10), pp. 58-66, Innsbruck, Austria, February 2010.

- Sánchez-Maroño, N., Alonso-Betanzos, A., García-González, P. and Bolón-Canedo, V. *Multiple classifiers vs multiple binary classifiers using filters for feature selection.* In Proc. International Joint Conference on Artificial Neural Networks (IJCNN'10), pp. 2836-2843, Barcelona, Spain, June 2010.

- Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. *On the effectiveness of discretization on gene selection of microarray data.* In Proc. International Joint Conference on Artificial Neural Networks (IJCNN'10), pp. 3167-3174, Barcelona, Spain, June 2010.

- Bolón-Canedo, V., Seth, S., Sánchez-Maroño, N., Alonso-Betanzos, A. and Principe, J.C. *Statistical dependence measure for feature selection in microarray datasets.* In Proc. 19th European Symposium on Artificial Neural Networks (ESANN'11), pp. 23-28, Bruges, Belgium, April 2011.

- Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. *On the behavior of feature selection methods dealing with noise and relevance over synthetic scenarios.* In Proc. International Joint Conference on Artificial Neural Networks (IJCNN'11), pp. 1530-1537, San Jose (CA), USA, August 2011.

- Bolón-Canedo, V., Peteiro-Barral, D., Alonso-Betanzos, A., Guijarro-Berdiñas, B. and Sánchez-Maroño, N. *Scalability analysis of ANN training algorithms with feature selection.* In Lecture Notes in Artificial Intelligence (CAEPIA'11), vol. 7023, pp. 84-93, San Cristobal de la Laguna, Spain, November 2011.

- Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. *Toward an ensemble of filters for classification.* In Proc. 11th International Conference on Intelligent Systems Design and Applications (ISDA'11), pp. 331-336, Cordoba, Spain, November 2011.

- Peteiro-Barral, D., Bolón-Canedo, V., Alonso-Betanzos, A., Guijarro-Berdiñas, B. and Sánchez-Maroño, N. *Scalability analysis of filter-based methods for feature selection.* In Advances in Smart Systems Research (KES'12), vol. 2, no. 1, pp. 21-26, San Sebastian, Spain, September 2012.

- Bolón-Canedo, V., Peteiro-Barral, D., Remeseiro, B., Alonso-Betanzos, A., Guijarro-Berdiñas, B., Mosquera, A., Penedo, M.G. and Sánchez-Maroño, N. *Interferential tear film lipid layer classification: An automatic dry eye test.* In Proc. 24th International Conference on Tools with Artificial Intelligence (ICTAI'12), pp. 359-366, Athens, Greece, November 2012.

- Bolón-Canedo, V., Sánchez-Maroño, N. and Alonso-Betanzos, A. *A distributed wrapper approach for feature selection.* In Proc. 21st European Symposium on Artificial Neural Networks (ESANN'13), pp. 173-178, Bruges, Belgium, April 2013.

- Bolón-Canedo, V., Sánchez-Maroño, N. and Cerviño-Rabuñal, J. *Scaling up feature selection: a distributed filter approach.* In Lecture Notes in Artificial Intelligence (CAEPIA'13), vol. 8109, pp. 121-130, Madrid, Spain, September 2013.

- Bolón-Canedo, V., Remeseiro, B., Sánchez-Maroño, N. and Alonso-Betanzos, A. *mC-ReliefF: An extension of ReliefF for cost-based feature selection.* In Proc. of International Conference of Agents and Artificial Intelligence (ICAART'14), pp. 42-51, Angers, France, March 2014.

- Bolón-Canedo, V., Peteiro-Barral, D., Alonso-Betanzos, A., Guijarro-Berdiñas, B. and Sánchez-Maroño, N. *Learning on Vertically Partitioned Data Based on Chi-Square Feature Selection and Naive Bayes Classification.* In Proc. of International Conference of Agents and Artificial Intelligence (ICAART'14), pp. 350-357, Angers, France, March 2014.

- Rego-Fernández, D., Bolón-Canedo, V. and Alonso-Betanzos, A. *Scalability analysis of mRMR for microarray data.* In Proc. of International Conference of Agents and Artificial Intelligence (ICAART'14), pp. 380-386, Angers, France, March 2014.

- Bolón-Canedo, V., Sánchez-Maroño, N. and Cerviño-Rabuñal, J. *Toward parallel feature selection from vertically partitioned data.* In Proc. 22nd European Symposium on Artificial Neural Networks (ESANN'14), (in press), Bruges, Belgium, April 2014.

Book chapters

- Alonso-Betanzos, A., Bolón-Canedo, V., Fernández-Francos, D., Porto-Díaz, I. and Sánchez-Maroño, N. *Up-to-date feature selection methods for scalable and efficient machine learning* in *Efficiency and Scalability Methods for Computational Intellect*, B. Igelnik and J. M. Zurada, Eds: IGI Global, 2013, pp. 1-26.

Mentions

- GENIL (Granada Excellence Network of Innovation Laboratories) award to the best doctoral proposal for the article *Novel Feature Selection Methods Applied to High Dimensionality Datasets*. Conference of the Spanish Association of Artificial Intelligence (CAEPIA), 2011.

APPENDIX III

Resumen del trabajo

La aparición de conjuntos de datos de alta dimensión ha supuesto un reto sin precedentes para los investigadores de aprendizaje máquina, haciendo que las tareas de aprendizaje se vuelvan más difíciles y costosas computacionalmente. El término *alta dimensión* se aplica a una base de datos que presenta una de las siguientes características: (a) el número de muestras es muy elevado; (b) el número de características es muy elevado; o (c) tanto el número de muestras como el de características son muy elevados. Se considera que un conjunto de datos es de alta dimensión cuando tiene más de 10 000 datos (considerando datos como características por muestras).

Cuando se trata con bases de datos de alta dimensión, el rendimiento de los algoritmos de aprendizaje puede verse degradado debido al sobreajuste, los modelos aprendidos disminuyen su interpretabilidad cuando son más complejos y, además, la velocidad y eficiencia de los algoritmos decae en concordancia con el tamaño. El apendizaje máquina puede beneficiarse de los métodos de selección de características, ya que éstos son capaces de reducir la dimensión de un problema dado. La *selección de características* es el proceso de detectar las características relevantes y eliminar las redundantes e irrelevantes, tratando de obtener un subconjunto de características lo más pequeño posible que resuelva el problema dado con una degradación mínima en el rendimiento. La selección de características, ya que es una actividad importante en el preprocesado de los datos, ha sido un área activa de investigación en la última década, obteniendo resultados exitosos en diferentes aplicaciones reales, especialmente aquellas relacionadas con problemas de clasificación.

Esta tesis está dedicada a la investigación en métodos de selección de características y su aplicación a datos de alta dimensión. En primer lugar, se realiza un análisis crítico de los métodos de selección de características existentes, para comprobar su idoneidad frente a diferentes problemas y para poder dar recomendaciones a los usuarios sobre qué método usar. Teniendo presente este análisis, se aplican las técnicas más adecuadas a

varios problemas reales, obteniendo una notable mejora en el rendimiento. Además de la eficiencia, otro aspecto crítico para las aplicaciones de gran escala es la escalabilidad. La eficiencia de los métodos de selección de características puede verse significativamente degradada, si no totalmente inaplicable, cuando el volumen de datos se incrementa constantemente. Por este motivo, se analiza la escalabilidad de los métodos de selección de características.

Además, también se proponen nuevas técnicas para selección de características en datos de gran escala. En primer lugar, ya que la mayoría de métodos existentes necesitan que los datos sean discretos, se propone una nueva metodología que consiste en la combinación de un discretizador, un filtro y un clasificador simple, obteniendo resultados prometedores. Otra de las propuestas trata de usar un conjunto de filtros en lugar de uno sólo, liberando al usuario de tomar la decisión de qué técnica es la más apropiada para un problema dado. Otro tema interesante a tener en cuenta es considerar el coste asociado de las diferentes características (económico, o relacionado con los requisitos temporales o computacionales), por lo que se porpone una metodología para selección de características basada en coste, demostrando su idioneidad en un problema real. Por último, es bien sabido que una forma de tratar con datos de gran escala es transformar el problema de gran escala en varios subproblemas de pequeña escala, distribuyendo los datos. Con este objetivo, se proponen varias alternativas para abordar la selección de características de forma distribuida o paralela. A continuación se resumen las aportaciones principales de cada uno de los bloques de la tesis.

Análisis de los métodos de selección de características

Los métodos de selección de características suelen estar divididos en tres grandes grupos: filtros, envolventes y embebidos. Los filtros se basan en las características generales de los datos de entrenamiento y llevan a cabo el proceso de selección como un paso de preprocesado independiente del algoritmo de inducción. Por el contrario, los métodos envolventes involucran la optimización de un predictor como parte del proceso de selección. Entre estos dos modelos se encuentran los métodos embebidos, que realizan la selección de características en el proceso de entrenamiento y normalmente son específicos para un algoritmo de aprendizaje en particular.

En la literatura especializada existe un gran número de métodos de selección de características, incluyendo filtros basados en distintas métricas (p.e. entropía, distribu-

ciones de probabilidad o teoría de la información) y métodos envolventes y embebidos que usan distintos algoritmos de inducción. Sin embargo, la proliferación de algoritmos de selección de características no ha traído consigo una metodología general que permita una selección inteligente de entre los algoritmos existentes. Para poder tomar una decisión acertada, un usuario no sólo debe conocer bien el dominio, sinó que también debe entender los detalles técnicos de los algoritmos disponibles. Además de esto, la mayoría de los algoritmos fueron desarrollados cuando las bases de datos eran mucho más pequeñas, pero hoy en día se requieren compromisos distintos para los problemas de aprendizaje de grande y pequeña escala. Los problemas de pequeña escala están sujetos al compromiso habitual de aproximación-estimación. En el caso de los problemas de gran escala, la solución de compromiso es más compleja porque involucra no sólo a la precisión de la selección sinó también otros aspectos como la estabilidad (i.e. la sensibilidad de los resultados ante variaciones en el conjunto de entrenamiento) o la escalabilidad.

La primera parte de esta tesis está dedicada a analizar los métodos que forman el estado del arte de la selección de características y demostrar su eficacia en aplicaciones reales. Las principales contribuciones son las siguientes:

- Revisión crítica de los métodos de selección de características más populares en la bibliografía estudiando su comportamiento en un escenario artificial controlado. De esta manera, se evalúa la habilidad de los algoritmos para seleccionar las características relevantes y descartar las irrelevantes sin permitir que el ruido o la redundancia obstruya este proceso. Además, se presentan varios casos de estudio para ayudar a decidir entre métodos que muestran un comportamiento similar. A la vista de los resultados obtenidos, se recomienda usar filtros (en particular, ReliefF), ya que son independientes del algoritmo de clasificación y son más rápidos que embebidos o envolventes, además de presentar una buena capacidad de generalización.

- Análisis del comportamiento de la selección de características en un campo exigente: la clasificación de microarrays de ADN. Este tipo de datos supone un reto para los investigadores en aprendizaje máquina debido a su elevado número de características (sobre 10 000) frente a un número de muestras muy pequeño (típicamente cien o menos). Para esto, es necesario revisar los algoritmos más recientes desarrollados a medida para este tipo de datos, así como también estudiar sus particularidades. Se propone una evaluación práctica de los métodos de selección de características sobre conjuntos microarray para estudiar los resul-

tados. Se cuenta con un amplio abanico de 9 conjuntos de datos ampliamente usados en la literatura, 7 métodos de selección de características clásicos y 3 algoritmos de clasificación. Este amplio conjunto de experimentos facilitan futuras comparaciones cuando un investigador propone un nuevo método.

- Aplicación de métodos de selección de características clásicos a problemas reales para comprobar su idoneidad. Concretamente, se prueba la eficacia de la selección de características en dos problemas del dominio médico. En el primero de ellos, se aborda la clasificación de la capa lipídica de la película lagrimal. El tiempo que requerían los métodos existentes para tratar este problema era prohibitivo para su uso clínico, ya que no era posible trabajar en tiempo real. En esta tesis se propone una metodología para resolver este problema, que incluye la aplicación de métodos de selección de características. En términos clínicos, se puede así automatizar el proceso manual realizado por expertos, con las ventajas de hacerlo más rápido e independiente de factores subjetivos, obteniendo una precisión máxima por encima del 97% con un tiempo de procesado menor que un segundo. El segundo caso de estudio es la clasificación de complejos K en apnea del sueño, que es un punto clave en los estudios relacionados con el sueño. Se aplican varios métodos de selección de características en combinación con varios algoritmos de aprendizaje máquina, intentando obtener una tasa baja de falsos positivos al mismo tiempo que se mantiene la precisión. Con la inclusión del paso de selección de características, los resultados mejoran de forma significativa para todos los clasificadores considerados.

- Escalabilidad de los métodos de selección de características. Con la proliferación de los conjuntos de datos de gran escala, los investigadores deben centrarse no sólo en la precisión sinó también en la escalabilidad de los métodos existentes. En primer lugar, se evalúa la eficacia de la selección de características en la escalabilidad de los algoritmos de entrenamiento para redes de neuronas artificiales (RNAs), tanto en tareas de clasificación como de regresión. Los resultados experimentales muestran que la selección de características como paso de preprocesado es beneficiosa para la escalabilidad de RNAs, incluso haciendo que ciertos algoritmos sean capaces de entrenar en algunos conjuntos de datos en los que era imposible debido a la complejidad espacial. Además, se analiza también la escalabilidad de los métodos de selección de características, que es un tema que no ha recibido suficiente consideración en la bibliografía. Se evalúa un total de 11 métodos de selección pertenecientes a las tres categorías existentes: filtros, envolventes y embebidos. El estudio experimental se realiza sobre conjuntos de datos artificiales, para ser capaz de evaluar el grado de proximidad a la solución

óptima de forma segura. Para poder evaluar la escalabilidad de los métodos, es necesario proponer nuevas medidas que estén basadas no sólo en la precisión sino también en el tiempo de ejecución y la estabilidad. Considerando los resultados experimentales, los filtros parecen ser la opción más escalable. En particular, FCBF presenta el mejor comportamiento en cuando a escalabilidad. De entre los métodos que devuelven un orden de las características, ReliefF es una buena opción cuando se cuenta con un número no muy elevado de características, a costa de un tiempo de entrenamiento grande. Por este motivo, el filtro basado en la ganancia de información demuestra ser más escalable cuando se abordan problemas de alta dimensión.

Nuevos métodos de selección de características

La segunda parte de esta tesis trata del desarrollo de nuevos métodos de selección de características que sean aplicables a conjuntos de alta dimensión. Aunque los beneficios de la selección de características están ampliamente demostrados, la mayoría de los investigadores coinciden en que no existe el "mejor" método de selección de características y sus esfuerzos se centran en encontrar un método que sea bueno para un problema específico. Por este motivo, costantemente aparecen nuevos métodos de selección de características que usan estrategias diferentes. De hecho, la tendencia actual en la selección de características no está enfocada hacia el desarollo de nuevas medidas algorítmicas, sinó hacia la combinación o modificación de algoritmos existentes. Por lo tanto, el objetivo de esta parte de la tesis se centra en explorar distintas estrategias para tratar con las nuevas problemáticas que han surgido con la aparición de los datos de gran escala.

La primera aproximación está relacionada con las técnicas de preprocesado, por tanto se introduce una etapa de discretización previa a la etapa de selección de características para tratar de mejorar el rendimiento de los algoritmos de inducción. Otra línea de investigación muy interesante y popular en los últimos tiempos es el aprendizaje de conjuntos (*ensemble*), que se basa en la suposición de que un conjunto de expertos es mejor que un único experto, por lo que se propone un método que consiste en usar un conjunto de filtros y clasificadores. También es interesante considerar casos en los que las características tienen su propio coste o riesgo asociado, ya que este factor debe tenerse en cuenta además de la precisión. Por este motivo, se proponen métodos de selección de características que tengan en cuenta el coste. Por último, recientemente

ha surgido un nuevo tema de interés que consiste en distribuir el proceso de selección de características con el objetivo de mejorar la precisión al mismo tiempo que se reduce el tiempo de entrenamiento.

Las principales contribuciones de la segunda parte de esta tesis son las siguientes:

- Combinación de métodos de discretización y selección de características. La mayoría de los algoritmos de selección de características sólo pueden trabajar con datos discretos, por lo que una práctica ampliamente utilizada es discretizar los datos antes de realizar el proceso de selección. Sin embargo, muchos trabajos confían esta tarea a discretizadores "por defecto", como es el caso de la famosa herramienta Weka. Para arrojar luz sobre este tema, se propone una metodología que consiste en la combinación de distintos métdos de discretización, selección de características y clasificación y que se aplica a escenarios muy diferentes. La idoneidad de la metodología propuesta se demuestra sobre conjuntos de datos de un sistema de detección de intrusos, datos de microarrays de ADN y un amplio conjunto de datos multiclase.

- Aprendizaje por conjunto de filtros y clasificadores. El objetivo del método propuesto es reducir la variabilidad de los resultados obtenidos por métodos de discretización y selección de características diferentes, basándose en la suposición de que combinar la salida de varios expertos es mejor que la salida de un único experto. Se presentan dos algoritmos genéricos, en función del rol del clasificador. La primera propuesta clasifica los datos tantas veces como filtros se estén usando, mientras que la segunda clasifica una única vez con el resultado de unir los distintos subconjuntos de características seleccionados por los distintos filtros. Se proponen un total de cinco implementaciones de las dos propuestas, que se prueban tanto en conjuntos artificiales, como conjuntos de datos clásicos y de microarray de ADN. Se demuestra así lo apropiado de usar un conjunto de filtros en lugar de uno solo, ya que para todos los escenarios considerados, el conjunto de filtros siempre aparece como la mejor opción.

- Metodología para selección de características basada en coste. En algunas situaciones, un usuario no está interesado sólo en seleccionar el subconjunto de características más preciso, sino también en reducir los costes asociados a los datos. Por ejemplo, para un diagnóstico médico, los síntomas que se pueden observar a ojo no tienen coste, pero cada dato diagnóstico extraído de una prueba clínica está asociado con su propio coste y riesgo. Teniendo esto en cuenta, se propone

una nueva metodología que permite realizar selección de características basada en coste. Esta metodología consiste en añadir un nuevo término a la función de evaluación de cualquier filtro para tratar de obtener una solución intermedia entre la métrica de un filtro (p.e. correlación) y el coste asociado con las características. Para probar la efectividad del método se seleccionan tres filtros representativos y se aplica a un amplio conjunto de bases de datos, incluyendo un problema real. Los resultados experimentales demuestran que la propuesta es sólida y permite al usuario reducir el coste sin comprometer el error de clasificación de forma significativa.

- Selección de características distribuida y paralela. Cuando se trata con datos de gran escala, una posible estrategia consiste en distribuir la tarea de aprendizaje en varios procesadores. El objetivo principal de esta estrategia es distribuir el proceso de selección de características, suponiendo que se obtendrá una reducción considerable en el tiempo de ejecución y la precisión no se verá afectada en exceso. Se presentan varias propuestas para tratar tanto con una distribución horizontal como vertical de los datos. Los resultados experimentales muestran que las propuestas permiten reducir el tiempo de ejecución significativamente con respecto a la aproximación estándar (centralizada). De hecho, en cuanto a tiempo de ejecución, el comportamiento de las propuestas distribuidas es excelente, siendo ésta la ventaja más importante del método. Además, en cuanto a la precisión de clasificación, estas propuestas distribuidas son capaces de igualar –y en algún caso de mejorar– los resultados obtenidos por los algoritmos estándar aplicados a datos no distribuidos.

Organización de la tesis

En este resumen se han introducido los temas principales que se tratan en esta tesis. La primera parte (Análisis de los métodos de selección de características) se trata en los capítulos 2 - 6. El capítulo 2 presenta los fundamentos de la selección de características, además de una descripción de los métodos que se emplean en esta tesis. A continuación, el capítulo 3 hace una revisión de los métodos más populares de la literatura y prueba su comportamiento en un escenario artificial controlado, proponiendo varias pautas para su uso en distintos dominios. El capítulo 4 analiza las contribuciones más recientes de la investigación de selección de características aplicada al campo de la clasificación de microarrays de ADN, mientras que el capítulo 5 se dedica a demostrar los beneficios

de la selección de características en otras aplicaciones reales como la clasificación de la capa lipídica de la película lagrimal y los complejos K para detectar apena del sueño. El capítulo 6 cierra la primera parte de la tesis estudiando la escalabilidad de los métodos de selección de características existentes.

La segunda parte de la tesis (Nuevos métodos de selección de características) está formada por los capítulos 7 - 10. El capítulo 7 presenta un método que está compuesto por la combinación de discretizadores, filtros y clasificadores. El método propuesto se aplica a un conjunto de datos de detección de intrusos, ampliamente utilizado como banco de prueba, así como a otros escenarios exigentes como los microarrays de ADN. El capítulo 8 presenta un método basado en un conjunto de filtros para poder ser aplicado a escenarios diversos. La idea se centra en la suposición de que un conjunto de filtros es mejor que un método simple, ya que de este modo es posible aprovechar sus fortalezas individuales y superar sus puntos débiles al mismo tiempo. El capítulo 9 propone una nueva metodología para selección de características basada en coste. El objetivo es resolver problemas en los cuales es interesante no sólo minimizar el error de clasificación sinó también reducir los posibles costes asociados a las características. El capítulo 10 presenta varias aproximaciones para tratar la selección de características distribuida o paralela, mediante un particionado de los datos vertical y horizontal.

Bibliography

Abeel, T., Helleputte, T., Van de Peer, Y., Dupont, P., & Saeys, Y. (2010). Robust biomarker identification for cancer diagnosis with ensemble feature selection methods. *Bioinformatics*, *26*(3), 392–398.

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

Aliferis, C. F., Statnikov, A., Tsamardinos, I., Mani, S., & Koutsoukos, X. D. (2010). Local causal and markov blanket induction for causal discovery and feature selection for classification part i: Algorithms and empirical evaluation. *The Journal of Machine Learning Research*, *11*, 171–234.

Alizadeh, A., Eisen, M., Davis, R., Ma, C., Lossos, I., Rosenwald, A., ... Yu, X. (2000). Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, *403*(6769), 503–511.

Allwein, E. L., Schapire, R. E., & Singer, Y. (2001). Reducing multiclass to binary: A unifying approach for margin classifiers. *The Journal of Machine Learning Research*, *1*, 113–141.

Alonso-Betanzos, A., Bolón-Canedo, V., Fernández-Francos, D., Porto-Díaz, I., & Sánchez-Maroño, N. (2013). Efficiency and scalability methods for computational intellect. In (pp. 1–26). IGI Global.

Alonso-Betanzos, A., Sánchez-Maroño, N., Carballal-Fortes, F. M., Suárez-Romero, J. A., & Pérez-Sánchez, B. (2007). Classification of computer intrusions using functional networks. a comparative study. In *15th european symposium on artificial neural networks-esann* (Vol. 7, pp. 25–27).

Alonso-González, C. J., Moro, Q. I., Prieto, O. J., & Simón, M. A. (2010). Selecting few genes for microarray gene expression classification. In *Current topics in artificial intelligence* (pp. 111–120). Springer.

Aly, M., & Atiya, A. (2006). Novel methods for the feature subset ensembles approach. *International Journal of Artificial Intelligence and Machine Learning*, *6*(4).

Ambroise, C., & McLachlan, G. (2002). Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the National Academy of Sciences*, *99*(10), 6562–6566.

Anaissi, A., Kennedy, P. J., & Goyal, M. (2011). Feature selection of imbalanced gene expression microarray data. In *Software engineering, artificial intelligence, networking and parallel/distributed computing (snpd), 2011 12th acis international conference on* (pp. 73–78).

Ananthanarayana, V., Subramanian, D., & Murty, M. (2000). Scalable, distributed and dynamic mining of association rules. *High Performance ComputingHiPC 2000*, 559–566.

Asuncion, A., & Newman, D. (2007). *UCI machine learning repository.* Retrieved November 2013, from `http://www.ics.uci.edu/$\sim$mlearn/{MLR}epository.html`

Axelsson, S. (1999). The base-rate fallacy and its implications for the difficulty of intrusion detection. In *Proceedings of the 6th acm conference on computer and communications security* (pp. 1–7).

Bahamonde, A., Bayón, G., Díez, J., Quevedo, J., Luaces, O., Del Coz, J., ... Goyache, F. (2004). Feature subset selection for learning preferences: a case study. In *Proceedings of the twenty-first international conference on machine learning* (p. 49-56).

Bankman, I., Sigillito, V., Wise, R., & Smith, P. (1992). Feature-Based Detection of the K-Complex Wave in the Human Electroencephalogram Using Neural Networks. *IEEE Transactions on Biomedical Engineering*, *39*(12), 1305-1310.

Barnett, V., & Lewis, T. (1994). *Outliers in statistical data* (Vol. 3). Wiley New York.

Basu, M., & Ho, T. K. (2006). *Data complexity in pattern recognition.* Springer.

Belanche, L., & González, F. (2011). Review and evaluation of feature selection algorithms in synthetic problems. *arXiv preprint arXiv:1101.2320*.

Beretta, L., & Santaniello, A. (2011). Implementing relieff filters to extract meaningful features from genetic lifetime datasets. *Journal of Biomedical Informatics*, *44*(2), 361–369.

Bishop, C. (2006). *Pattern recognition and machine learning* (Vol. 4).

Blagus, R., & Lusa, L. (2012). Evaluation of smote for high-dimensional class-imbalanced microarray data. In *Machine learning and applications (icmla), 2012 11th international conference on* (Vol. 2, pp. 89–94).

Blanco, R., Larrañaga, P., Inza, I., & Sierra, B. (2004). Gene selection for cancer classification using wrapper approaches. *International Journal of Pattern Recognition and Artificial Intelligence*, *18*(08), 1373–1390.

Bolón-Canedo, V., Sánchez-Maroño, N., & Alonso-Betanzos, A. (2009). A combination of discretization and filter methods for improving classification performance in kdd cup 99 dataset. In *Neural networks, 2009. ijcnn 2009. international joint conference on* (pp. 359–366).

Bolón-Canedo, V., Sánchez-Maroño, N., & Alonso-Betanzos, A. (2010). On the effectiveness of discretization on gene selection of microarray data. In *Neural networks (ijcnn), the 2010 international joint conference on* (pp. 18–23).

Bolón-Canedo, V., Sánchez-Maroño, N., & Alonso-Betanzos, A. (2011). Feature selection and classification in multiple class datasets: An application to kdd cup 99 dataset. *Expert Systems with Applications*, *38*(5), 5947–5957.

Bolón-Canedo, V., Sánchez-Maroño, N., & Alonso-Betanzos, A. (2012). An ensemble of filters and classifiers for microarray data classification. *Pattern Recognition*, *45*(1), 531–539.

Bolón-Canedo, V., Sánchez-Maroño, N., & Alonso-Betanzos, A. (2013a). Data classification using an ensemble of filters. *Neurocomputing*. (In Press)

Bolón-Canedo, V., Sánchez-Maroño, N., & Alonso-Betanzos, A. (2013b). A distributed wrapper approach for feature selection. In *Proceedings of the 21st european symposium on artificial neural networks* (pp. 173–178).

Bolón-Canedo, V., Seth, S., Sánchez-Maroño, N., Alonso-Betanzos, A., & Principe, J. (2011). Statistical dependence measure for feature selection in microarray datasets. In *19th european symposium on artificial neural networks-esann* (pp. 23–28).

Bontempi, G., & Meyer, P. E. (2010). Causal filter selection in microarray data. In *Proceedings of the 27th international conference on machine learning (icml-10)* (pp. 95–102).

Bosin, A., Dessì, N., & Pes, B. (2007). Capturing heuristics and intelligent methods for improving micro-array data classification. In *Intelligent data engineering and automated learning-ideal 2007* (pp. 790–799). Springer.

Braga-Neto, U. (2007). Fads and fallacies in the name of small-sample microarray classification-a highlight of misunderstanding and erroneous usage in the applications of genomic signal processing. *Signal Processing Magazine, IEEE*, *24*(1), 91–99.

Braga-Neto, U., & Dougherty, E. (2004). Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, *20*(3), 374–380.

Bramer, M. (2007). *Principles of data mining*. Springer.

Breiman, L. (1993). *Classification and regression trees*. CRC press.

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123–140.

Broad Institute. Cancer Program Data Sets. (n.d.). Retrieved November 2013, from `http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi`

Brown, G., Pocock, A., Zhao, M.-J., & Luján, M. (2012). Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. *The Journal of Machine Learning Research*, *13*, 27–66.

Brown, M. P., Grundy, W. N., Lin, D., Cristianini, N., Sugnet, C. W., Furey, T. S., ... Haussler, D. (2000). Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proceedings of the National Academy of Sciences*, *97*(1), 262–267.

Bruzzone, L., & Serpico, S. (2000). A technique for feature selection in multiclass problems. *International Journal of Remote Sensing*, *21*(3), 549–563.

Byeon, B., & Rasheed, K. (2008). Simultaneously removing noise and selecting relevant features for high dimensional noisy data. In *Machine learning and applications, 2008. icmla'08. seventh international conference on* (pp. 147–152).

Canul-Reich, J., Hall, L., Goldgof, D., Korecki, J., & Eschrich, S. (2012). Iterative feature perturbation as a gene selector for microarray data. *International Journal of Pattern Recognition and Artificial Intelligence*, *26*(05), 1260003.

Castillo, E., Fontenla-Romero, O., Guijarro-Berdiñas, B., & Alonso-Betanzos, A. (2002). A global optimum approach for one-layer neural networks. *Neural Computation*, *14*(6), 1429–1449.

Catlett, J. (1991). *Megainduction: machine learning on very large databases*. Unpublished doctoral dissertation, University of Sydney Australia.

Çesmeli, E., & Wang, D. (2001). Texture segmentation using gaussian-markov random fields and neural oscillator networks. *Neural Networks, IEEE Transactions on*, *12*(2), 394–404.

Chan, P., & Stolfo, S. (1993). Toward parallel and distributed learning by meta-learning. In *Aaai workshop in knowledge discovery in databases* (pp. 227–240).

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Chidlovskii, B., & Lecerf, L. (2008). Scalable feature selection for multi-class problems. In *Machine learning and knowledge discovery in databases* (pp. 227–240). Springer.

Chuang, L., Yang, C., Wu, K., & Yang, C. (2011). A hybrid feature selection method for dna microarray data. *Computers in biology and medicine*, *41*(4), 228–237.

Clausi, D. A., & Jernigan, M. E. (1998). A Fast Method to Determine Co-occurrence Texture Features. *IEEE Transactions on Geoscience and Remote Sensing*, *36*(1), 298–300.

Collobert, R., & Bengio, S. (2001). Support vector machines for large-scale regression problems. *The Journal of Machine Learning Research*, *1*, 160.

Dash, M., & Liu, H. (2003). Consistency-based search in feature selection. *Artificial intelligence*, *151*(1), 155–176.

de Haro García, A. (2011). *Scaling data mining algorithms. application to instance and feature selection*. Unpublished doctoral dissertation, Universidad de Granada.

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

Díaz-Uriarte, R., & De Andres, S. A. (2006). Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, *7*(1), 3.

Ding, C., & Peng, H. (2005). Minimum redundancy feature selection from microarray gene expression data. *Journal of Bioinformatics and Computational Biology*, *3*(02), 185–205.

Doak, J. (1992). *An evaluation of feature selection methods and their application to computer security.* University of California, Computer Science.

Dougherty, E. (2001). Small sample issues for microarray-based classification. *Comparative and Functional Genomics*, *2*(1), 28–34.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. In *Icml* (pp. 194–202).

Dudoit, S., Fridlyand, J., & Speed, T. (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, *97*(457), 77–87.

Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association*, *56*(293), 52–64.

Dy, J. G., Brodley, C. E., Kak, A., Broderick, L. S., & Aisen, A. M. (2003). Unsupervised feature selection applied to content-based retrieval of lung images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *25*(3), 373–378.

Efron, B. (1979). Bootstrap methods: another look at the jackknife. *The annals of Statistics*, 1–26.

Efron, B., & Tibshirani, R. (1993). *An introduction to the bootstrap* (Vol. 57). Chapman & Hall/CRC.

Egozi, O., Gabrilovich, E., & Markovitch, S. (2008). Concept-based feature generation and selection for information retrieval. In *Aaai* (pp. 1132–1137).

El Akadi, A., Amine, A., El Ouardighi, A., & Aboutajdine, D. (2011). A two-stage gene selection scheme utilizing mrmr filter and ga wrapper. *Knowledge and Information Systems*, *26*(3), 487–500.

Elkan, C. (2000). Results of the kdd'99 classifier learning. *ACM SIGKDD Explorations Newsletter*, *1*(2), 63–64.

Fahad, A., Tari, Z., Khalil, I., Habib, I., & Alnuweiri, H. (2013). Toward an efficient and scalable feature selection approach for internet traffic classification. *Computer Networks*, *57*, 2040–2057.

Fayyad, U., & Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *13th international joint conference on artificial intelligence (ijcai)* (pp. 1022–1029).

Feature Selection at Arizona State University. (n.d.). Retrieved November 2013, from `http://featureselection.asu.edu/datasets.php`

Feddema, J., Lee, C., & Mitchell, O. (1991). Weighted selection of image features for resolved rate visual feedback control. *Robotics and Automation, IEEE Transactions on*, *7*(1), 31–47.

Ferreira, A., & Figueiredo, M. (2012). An unsupervised approach to feature discretization and selection. *Pattern Recognition*, *45*(9), 3048–3060.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *The Journal of machine learning research*, *3*, 1289–1305.

Freund, Y., & Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory* (pp. 23–37).

Friedman, J. H. (1989). Regularized discriminant analysis. *Journal of the American statistical association*, *84*(405), 165–175.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, *32*(200), 675–701.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, *11*(1), 86–92.

Fugate, M., & Gattiker, J. R. (2003). Computer intrusion detection with classification and anomaly detection, using svms. *International Journal of Pattern Recognition and Artificial Intelligence*, *17*(03), 441–458.

Fung, G., & Mangasarian, O. L. (2001). Proximal support vector machine classifiers. In *Proceedings of the seventh acm sigkdd international conference on knowledge discovery and data mining* (pp. 77–86).

Furey, T., Cristianini, N., Duffy, N., Bednarski, D., Schummer, M., & Haussler, D. (2000). Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, *16*(10), 906–914.

Gabor, D. (1946). Theory of communication. part 1: The analysis of information. *Electrical Engineers-Part III: Radio and Communication Engineering, Journal of the Institution of*, *93*(26), 429–441.

Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2012). A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, *42*(4), 463–484.

Galar, M., Fernández, A., Barrenechea, E., & Herrera, F. (2013). Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*.

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

García-Resúa, C., Fernández, M. J. G., Penedo, M. F. G., Calvo, D., Penas, M., & Yebra-Pimentel, E. (2013). New software application for clarifying tear film lipid layer patterns. *Cornea*, *32*(4), 538–546.

Gevaert, O., Smet, F., Timmerman, D., Moreau, Y., & Moor, B. (2006). Predicting the prognosis of breast cancer by integrating clinical and microarray data with bayesian networks. *Bioinformatics*, *22*(14), 184–190.

Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., ... Caligiuri, M. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, *286*(5439), 531–537.

Gomez, J. C., Boiy, E., & Moens, M.-F. (2012). Highly discriminative statistical features for email classification. *Knowledge and information systems*, *31*(1), 23–53.

Gonzalez, R., & Woods, R. (2008). *Digital image processing*. Pearson/Prentice Hall.

Gonzalez-Navarro, F. (2011). *Feature selection in cancer research: Microarray gene expression and in vivo 1h-mrs domains*. Unpublished doctoral dissertation, Technical University of Catalonia.

González Navarro, F., & Belanche Muñoz, L. (2009). Gene subset selection in microarray data using entropic filtering for cancer classification. *Expert Systems*, *26*(1), 113–124.

Gordon, G., Jensen, R., Hsiao, L., Gullans, S., Blumenstock, J., Ramaswamy, S., ... Bueno, R. (2002). Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer research*, *62*(17), 4963–4967.

Grkabczewski, K., & Jankowski, N. (2003). Transformations of symbolic data for continuous data oriented models. In *Artificial neural networks and neural information processingicann/iconip 2003* (pp. 359–366). Springer.

Guillon, J.-P. (1998). Non-invasive tearscope plus routine for contact lens fitting. *Contact Lens and Anterior Eye*, *21*, S31–S40.

Gulgezen, G., Cataltepe, Z., & Yu, L. (2009). Stable and accurate feature selection. In *Machine learning and knowledge discovery in databases* (pp. 455–468). Springer.

Guyon, I. (2006). *Feature extraction: foundations and applications* (Vol. 207). Springer.

Guyon, I., Bitter, H.-M., Ahmed, Z., Brown, M., & Heller, J. (2005). Multivariate nonlinear feature selection with kernel methods. In *Soft computing for information processing and analysis* (pp. 313–326). Springer.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection.

The Journal of Machine Learning Research, *3*, 1157–1182.

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine learning*, *46*(1-3), 389–422.

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

Hall, M., & Smith, L. (1998). Practical feature subset selection for machine learning. *Computer Science*, *98*, 181–191.

Hall, M. A. (1999). *Correlation-based feature selection for machine learning.* Unpublished doctoral dissertation, The University of Waikato.

Haralick, R. M., Shanmugam, K., & Dinstein, I. H. (1973). Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*(6), 610–621.

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, *21*(9), 1263–1284.

Hecht-Nielsen, R. (1990). *Neurocomputing.* Addison-Wesley, Menlo Park, California.

Ho, T. (1998). The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *20*(8), 832–844.

Ho, T. K., & Basu, M. (2002). Complexity measures of supervised classification problems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *24*(3), 289–300.

Hoi, S. C., Wang, J., Zhao, P., & Jin, R. (2012). Online feature selection for mining big data. In *Proceedings of the 1st international workshop on big data, streams and heterogeneous source mining: Algorithms, systems, programming models and applications* (pp. 93–100).

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359–366.

Hsu, J. C. (1996). *Multiple comparisons:: Theory and methods.* CRC Press.

Hua, J., Tembe, W. D., & Dougherty, E. R. (2009). Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recognition*, *42*(3), 409–424.

Huang, C., & Wang, C. (2006). A ga-based feature selection and parameters optimizationfor support vector machines. *Expert Systems with applications*, *31*(2), 231–240.

Hwang, C.-L., & Yoon, K. (1981). *Multiple attribute decision making.* Springer.

Iber, C., Ancoli-Israel, S., Chesson, A., & Quan, S. (2007). *The aasm manual for scoring of sleep and associated events: Rules, terminology and technical specifications.* Westchster, Illinois: American Academy Of Sleep Medicine.

Inza, I., Larrañaga, P., Blanco, R., & Cerrolaza, A. (2004). Filter versus wrapper gene selection approaches in dna microarray domains. *Artificial intelligence in Medicine*, *31*(2), 91–103.

Inza, I., Sierra, B., Blanco, R., & Larrañaga, P. (2002). Gene selection by sequential search wrapper approaches in microarray cancer class prediction. *Journal of Intelligent and Fuzzy Systems*, *12*(1), 25–33.

Jain, A., & Zongker, D. (1997). Feature selection: Evaluation, application, and small sample performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *19*(2), 153–158.

Janssens, D., Brijs, T., Vanhoof, K., & Wets, G. (2006). Evaluating the performance of cost-based discretization versus entropy-and error-based discretization. *Computers & operations research*, *33*(11), 3107–3123.

Jiawei, H., & Kamber, M. (2001). *Data mining: concepts and techniques*. San Francisco, CA, itd: Morgan Kaufmann.

Jirapech-Umpai, T., & Aitken, S. (2005). Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes. *BMC bioinformatics*, *6*(1), 148.

John, G. H., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Icml* (Vol. 94, pp. 121–129).

Kadota, K., Tominaga, D., Akiyama, Y., & Takahashi, K. (2003). Detecting outlying samples in microarray data: A critical assessment of the effect of outliers on sample classification. *Chem-Bio Informatics*, *3*(1), 30–45.

Kalousis, A., Prados, J., & Hilario, M. (2007). Stability of feature selection algorithms: a study on high-dimensional spaces. *Knowledge and information systems*, *12*(1), 95–116.

KDD Cup 99 Dataset. (n.d.). Retrieved November 2013, from `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, *30*(1/2), 81–93.

Kent Ridge Bio-Medical Dataset. (n.d.). Retrieved November 2013, from `http://datam.i2r.a-star.edu.sg/datasets/krbd`

Khoshgoftaar, T. M., Gao, K., & Ibrahim, N. H. (2005). Evaluating indirect and direct classification techniques for network intrusion detection. *Intelligent Data Analysis*, *9*(3), 309–326.

Kim, G., Kim, Y., Lim, H., & Kim, H. (2010). An MLP-based feature subset selection for hiv-1 protease cleavage site analysis. *Artificial intelligence in medicine*, *48*(2), 83–89.

Kira, K., & Rendell, L. A. (1992). A practical approach to feature selection. In

Proceedings of the ninth international workshop on machine learning (pp. 249–256).

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, *97*(1), 273–324.

Koller, D., & Sahami, M. (1995). Toward optimal feature selection. In *In 13th international conference on machine learning* (pp. 284–292).

Kononenko, I. (1994). Estimating attributes: analysis and extensions of relief. In *Machine learning: Ecml-94* (pp. 171–182).

Kryger, M., Roth, T., & Dement, W. (2005). *Principles and practice of sleep medicine*. Elsevier/Saunders. Retrieved from `http://books.google.es/books?id=YRdvQgAACAAJ`

Kudo, M., & Sklansky, J. (1997). A comparative evaluation of medium and large–scale feature selectors for pattern classifiers. In *Proc. of the 1st int. workshop on statistical techniques in pattern recognition* (pp. 91–96).

Kumar, R., & Vassilvitskii, S. (2010). Generalized distances between rankings. In *Proceedings of the 19th international conference on world wide web* (pp. 571–580).

Kuncheva, L. (2004). *Combining pattern classifiers: Methods and algorithms*. Wiley-Interscience.

Lan, L., & Vucetic, S. (2011). Improving accuracy of microarray classification by a simple multi-task feature selection filter. *International journal of data mining and bioinformatics*, *5*(2), 189–208.

Langley, P., & Iba, W. (1993). Average-case analysis of a nearest neighbor algorithm. In *International joint conference on artificial intelligence* (Vol. 13, pp. 889–889).

Lee, C., & Leu, Y. (2011). A novel hybrid feature selection method for microarray data analysis. *Applied Soft Computing*, *11*(1), 208–213.

Lee, J., Lee, J., Park, M., & Song, S. (2005). An extensive comparison of recent classification tools applied to microarray data. *Computational Statistics & Data Analysis*, *48*(4), 869–885.

Lee, W., Stolfo, S. J., & Mok, K. W. (2000). Adaptive intrusion detection: A data mining approach. *Artificial Intelligence Review*, *14*(6), 533–567.

Leung, Y., & Hung, Y. (2010). A multiple-filter-multiple-wrapper approach to gene selection and microarray data classification. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, *7*(1), 108–117.

Li, T., Zhang, C., & Ogihara, M. (2004). A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, *20*(15), 2429–2437.

Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., ...

Cunningham, R. K. (2000). Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *Darpa information survivability conference and exposition, 2000. discex'00. proceedings* (Vol. 2, pp. 12–26).

Liu, H., Hussain, F., Tan, C. L., & Dash, M. (2002). Discretization: An enabling technique. *Data mining and knowledge discovery*, *6*(4), 393–423.

Liu, H., Liu, L., & Zhang, H. (2008). Feature selection using mutual information: An experimental study. In *Pricai 2008: Trends in artificial intelligence* (pp. 235–246). Springer.

Liu, H., & Setiono, R. (1995). Chi2: Feature selection and discretization of numeric attributes. In *Tools with artificial intelligence, 1995. proceedings., seventh international conference on* (pp. 388–391).

Liu, H., & Setiono, R. (1997). Feature selection via discretization. *Knowledge and Data Engineering, IEEE Transactions on*, *9*(4), 642–645.

Liu, H., & Setiono, R. (1998). Scalable feature selection for large sized databases. In *Proceedings of the 4th world conference on machine learning* (pp. 101–106).

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on*, *17*(4), 491–502.

López, V., Fernández, A., García, S., Palade, V., & Herrera, F. (2013). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, *250*(0), 113 - 141.

Lorena, A. C., Costa, I. G., Spolaôr, N., & de Souto, M. C. (2012). Analysis of complexity indices for classification problems: cancer gene expression data. *Neurocomputing*, *75*(1), 33–42.

Loscalzo, S., Yu, L., & Ding, C. (2009). Consensus group stable feature selection. In *Proceedings of the 15th acm sigkdd international conference on knowledge discovery and data mining* (pp. 567–576).

Lovato, P., Bicego, M., Cristani, M., Jojic, N., & Perina, A. (2012). Feature selection using counting grids: application to microarray data. In *Structural, syntactic, and statistical pattern recognition* (pp. 629–637). Springer.

Luo, D., Wang, F., Sun, J., Markatou, M., Hu, J., & Ebadollahi, S. (2012). Sor: Scalable orthogonal regression for non-redundant feature selection and its healthcare applications. In *Siam data mining conference* (pp. 576–587).

Maldonado, S., Weber, R., & Basak, J. (2011). Simultaneous feature selection and classification using kernel-penalized support vector machines. *Information Sciences*, *181*(1), 115–128.

Mallat, S. G. (1989). A theory for multiresolution signal decomposition: the wavelet representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions*

on, *11*(7), 674–693.

Mamitsuka, H. (2006). Query-learning-based iterative feature-subset selection for learning from high-dimensional data sets. *Knowledge and information systems*, *9*(1), 91–108.

Martínez-Rego, D., Fontenla-Romero, O., Porto-Díaz, I., & Alonso-Betanzos, A. (2009). A new supervised local modelling classifier based on information theory. In *Neural networks, 2009. ijcnn 2009. international joint conference on* (pp. 2014–2020).

MATLAB. (2013). *version 8.1.0.604 (r2013a)*. Natick, Massachusetts: The MathWorks Inc.

McConnell, S., & Skillicorn, D. (2004). Building predictors from vertically distributed data. In *Proceedings of the 2004 conference of the centre for advanced studies on collaborative research* (pp. 150–162).

Mejía-Lavalle, M., Sucar, E., & Arroyo, G. (2006). Feature selection with a perceptron neural net. In *Proceedings of the international workshop on feature selection for data mining* (pp. 131–135).

Meyer, P., Schretter, C., & Bontempi, G. (2008). Information-theoretic feature selection in microarray data using variable complementarity. *Selected Topics in Signal Processing, IEEE Journal of*, *2*(3), 261–274.

Michiels, S., Koscielny, S., & Hill, C. (2005). Prediction of cancer outcome with microarrays: a multiple random validation strategy. *The Lancet*, *365*(9458), 488–492.

Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., & Euler, T. (2006, August). Yale: Rapid prototyping for complex data mining tasks. In L. Ungar, M. Craven, D. Gunopulos, & T. Eliassi-Rad (Eds.), *Kdd '06: Proceedings of the 12th acm sigkdd international conference on knowledge discovery and data mining* (pp. 935–940). New York, NY, USA: ACM. Retrieved from `http://rapid-i.com/component/option,com_docman/task,doc_download/gid,25/Itemid,62/` doi: http://doi.acm.org/10.1145/1150402.1150531

Molina, L. C., Belanche, L., & Nebot, À. (2002). Feature selection algorithms: A survey and experimental evaluation. In *Data mining, 2002. icdm 2003. proceedings. 2002 ieee international conference on* (pp. 306–313).

Moller, M. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, *6*(4), 525–533.

More, J. (1978). The levenberg-marquardt algorithm: implementation and theory. *Numerical analysis*, 105–116.

Moreno-Torres, J. G., Raeder, T., Alaiz-Rodríguez, R., Chawla, N. V., & Herrera, F. (2012). A unifying view on dataset shift in classification. *Pattern Recognition*, *45*(1), 521–530.

Moreno-Torres, J. G., Sáez, J. A., & Herrera, F. (2012). Study on the impact of partition-induced dataset shift on k-fold cross-validation. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(8), 1304–1312.

Mukherjee, S., Tamayo, P., Slonim, D., Verri, A., Golub, T., Mesirov, J., & Poggio, T. (1999). Support vector machine classification of microarray data. *CBCL Paper*, *182*.

Mukkamala, S., & Sung, A. H. (2002). Feature ranking and selection for intrusion detection systems using support vector machines. In *Proceedings of the second digital forensic research workshop.*

Mukkamala, S., Sung, A. H., & Abraham, A. (2005). Intrusion detection using an ensemble of intelligent paradigms. *Journal of network and computer applications*, *28*(2), 167–182.

Mundra, P., & Rajapakse, J. (2010). SVM-RFE with mRMR filter for gene selection. *NanoBioscience, IEEE Transactions on*, *9*(1), 31–37.

Nemenyi, P. (1963). *Distribution-free multiple comparisons.* Unpublished doctoral dissertation, Princeton University.

Nie, F., Huang, H., Cai, X., & Ding, C. (2010). Efficient and robust feature selection via joint l2, 1-norms minimization. *Advances in Neural Information Processing Systems*, *23*, 1813–1821.

Okun, O., & Priisalu, H. (2009). Dataset complexity in gene expression based cancer classification using ensembles of¡ i¿ k¡/i¿-nearest neighbors. *Artificial intelligence in medicine*, *45*(2), 151–162.

Olson, D. L. (2004). Comparison of weights in topsis models. *Mathematical and Computer Modelling*, *40*(7), 721–727.

Opitz, D. (1999). Feature selection for ensembles. In *Proceedings of the national conference on artificial intelligence* (pp. 379–384).

Orriols-Puig, A., & Bernadó-Mansilla, E. (2009). Evolutionary rule-based systems for imbalanced data sets. *Soft Computing*, *13*(3), 213–225.

Peng, H., Long, F., & Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *27*(8), 1226–1238.

Peng, Y., Wu, Z., & Jiang, J. (2010). A novel feature selection approach for biomedical data classification. *Journal of Biomedical Informatics*, *43*(1), 15–23.

Perner, P., & Apte, C. (2000). Empirical evaluation of feature subset selection based on a real-world data set. In *Principles of data mining and knowledge discovery* (Vol. 1910, p. 575-580). Springer Berlin Heidelberg.

Peteiro-Barral, D., Bolón-Canedo, V., Alonso-Betanzos, A., Guijarro-Berdiñas, B., & Sánchez-Maroño, N. (2012). Scalability analysis of filter-based methods for fea-

ture selection. *Advances in Smart Systems Research*, *2*(1), 21–26.

Peteiro-Barral, D., Guijarro-Berdiñas, B., Pérez-Sánchez, B., & Fontenla-Romero, O. (2013). A comparative study of the scalability of a sensitivity-based learning algorithm for artificial neural networks. *Expert Systems with Applications*, *40*(10), 3900-3905.

Piatetsky-Shapiro, G., & Tamayo, P. (2003). Microarray data mining: facing the challenges. *ACM SIGKDD Explorations Newsletter*, *5*(2), 1–5.

Pradhananga, N. (2007). *Effective linear-time feature selection.* Unpublished doctoral dissertation, Citeseer.

Provost, F. (2000). Distributed data mining: Scaling up and beyond. *Advances in distributed and parallel knowledge discovery*, 3–27.

Provost, F., & Kolluri, V. (1999). A survey of methods for scaling up inductive algorithms. *Data mining and knowledge discovery*, *3*(2), 131–169.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81–106.

Quinlan, J. R. (1993). *C4. 5: programs for machine learning* (Vol. 1). Morgan kaufmann.

Rakotomamonjy, A. (2003). Variable selection using svm based criteria. *The Journal of Machine Learning Research*, *3*, 1357–1370.

Ramos, L., Penas, M., Remeseiro, B., Mosquera, A., Barreira, N., & Yebra-Pimentel, E. (2011). Texture and color analysis for the automatic classification of the eye lipid layer. In *Advances in computational intelligence* (pp. 66–73). Springer.

Remeseiro, B., Ramos, L., Penas, M., Martinez, E., Penedo, M. G., & Mosquera, A. (2011). Colour texture analysis for classifying the tear film lipid layer: a comparative study. In *Digital image computing techniques and applications (dicta), 2011 international conference on* (pp. 268–273).

Rich, E., & Knight, K. (1991). *Artificial intelligence.* NY: McGraw-Hill.

Rish, I. (2001). An empirical study of the naive bayes classifier. In *Ijcai 2001 workshop on empirical methods in artificial intelligence* (Vol. 3, pp. 41–46).

Rokach, L. (2009). Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational Statistics & Data Analysis*, *53*(12), 4046–4072.

Rokach, L., Schclar, A., & Itach, E. (2013). Ensemble methods for multi-label classification. *arXiv preprint arXiv:1307.1769*.

Ruiz, R., Riquelme, J., & Aguilar-Ruiz, J. (2006). Incremental wrapper-based gene selection from microarray data for cancer classification. *Pattern Recognition*, *39*(12), 2383–2392.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation* (Tech. Rep.). DTIC Document.

Saari, P., Eerola, T., & Lartillot, O. (2011). Generalizability and simplicity as criteria in feature selection: application to mood classification in music. *Audio, Speech, and Language Processing, IEEE Transactions on*, *19*(6), 1802–1812.

Saeys, Y., Abeel, T., & Van de Peer, Y. (2008). Robust feature selection using ensemble feature selection techniques. In *Machine learning and knowledge discovery in databases* (pp. 313–325). Springer.

Saeys, Y., Inza, I., & Larrañaga, P. (2007). A review of feature selection techniques in bioinformatics. *Bioinformatics*, *23*(19), 2507–2517.

Saez, J. A., Luengo, J., & Herrera, F. (2013). Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. *Pattern Recognition*, *46*, 355–364.

Sánchez-Maroño, N., Alonso-Betanzos, A., García-González, P., & Bolón-Canedo, V. (2010). Multiclass classifiers vs multiple binary classifiers using filters for feature selection. In *Neural networks (ijcnn), the 2010 international joint conference on* (pp. 18–23).

Sánchez-Maroño, N., Alonso-Betanzos, A., & Tombilla-Sanromán, M. (2007). Filter methods for feature selection–a comparative study. In *Intelligent data engineering and automated learning-ideal 2007* (pp. 178–187). Springer.

Schapire, R. (1990). The strength of weak learnability. *Machine learning*, *5*(2), 197–227.

Seth, S., & Principe, J. C. (2010). Variable selection: A statistical dependence perspective. In *Machine learning and applications (icmla), 2010 ninth international conference on* (pp. 931–936).

Shah, M., Marchand, M., & Corbeil, J. (2012). Feature selection with conjunctions of decision stumps and learning from microarray data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *34*(1), 174–186.

Sharma, A., Imoto, S., & Miyano, S. (2012). A top-r feature selection algorithm for microarray gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, *9*(3), 754–764.

Shreem, S., Abdullah, S., Nazri, M., & Alzaqebah, M. (2012). Hybridizing ReliefF, MRMR filters and GA wrapper approaches for gene selection. *Journal of Theoretical and Applied Information Technology*, *46*(2), 1034-1039.

Singh, D., Febbo, P., Ross, K., Jackson, D., Manola, J., Ladd, C., ... Richie, J. (2002). Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell*, *1*(2), 203–209.

Sivagaminathan, R., & Ramakrishnan, S. (2007). A hybrid approach for feature subset selection using neural networks and ant colony optimization. *Expert systems with applications*, *33*(1), 49–60.

Skillicorn, D., & McConnell, S. (2008). Distributed prediction from vertically partitioned data. *Journal of Parallel and Distributed computing*, *68*(1), 16–36.

Song, L., Smola, A., Gretton, A., Bedo, J., & Borgwardt, K. (2012). Feature selection via dependence maximization. *The Journal of Machine Learning Research*, *98888*, 1393–1434.

Song, Q., Ni, J., & Wang, G. (2013). A fast clustering-based feature subset selection algorithm for high-dimensional data. *Knowledge and Data Engineering, IEEE Transactions on*, *25*(1), 1-14.

Sonnenburg, S., Franc, V., Yom-Tov, E., & Sebag, M. (2008). Pascal large scale learning challenge. In *25th international conference on machine learning (icml2008) workshop. j. mach. learn. res* (Vol. 10, pp. 1937–1953).

Spearman, C. (1904). The proof and measurement of association between two things. *The American journal of psychology*, *15*(1), 72–101.

Statnikov, A., Aliferis, C., & Tsamardinos, I. (n.d.). *GEMS: Gene Expression Model Selector.* Retrieved November 2013, from `http://www.gems-system.org`

Stolfo, S. J., Fan, W., Lee, W., Prodromidis, A., & Chan, P. K. (2000). Cost-based modeling for fraud and intrusion detection: Results from the jam project. In *Darpa information survivability conference and exposition, 2000. discex'00. proceedings* (Vol. 2, pp. 130–144).

Student, S., & Fujarewicz, K. (2012). Stable feature selection and classification algorithms for multiclass microarray data. *Biology Direct*, *7*(1), 33.

Sun, Y., Babbs, C., & Delp, E. (2006). A comparison of feature selection methods for the detection of breast cancers in mammograms: adaptive sequential floating search vs. genetic algorithm. In *Engineering in medicine and biology society, 2005. ieee-embs 2005. 27th annual international conference of the* (pp. 6532–6535).

Sun, Y., & Li, J. (2006). Iterative relief for feature weighting. In *Proceedings of the 23rd international conference on machine learning* (pp. 913–920).

Sun, Y., Todorovic, S., & Goodison, S. (2008a). A feature selection algorithm capable of handling extremely large data dimensionality. In *Sdm* (pp. 530–540).

Sun, Y., Todorovic, S., & Goodison, S. (2008b). A feature selection algorithm capable of handling extremely large data dimensionality. In *In proceedings of the 2008 siam internactional conference in data mining* (pp. 530–540).

Sun, Y., Wong, A. K., & Kamel, M. S. (2009). Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, *23*(04), 687–719.

Teich, J. (2001). Pareto-front exploration with uncertain objectives. In *Evolutionary multi-criterion optimization* (pp. 314–328).

Thrun, S. B., Bala, J., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., ... Fisher,

D. (1991). *The monk's problems a performance comparison of different learning algorithms* (Tech. Rep. No. CMU-CS-91-197). Pittsburgh, PA: Carnegie Mellon University, Computer Science Department.

Tou, J., & González, R. (1977). *Pattern recognition principles.* Addison-Wesley.

Tsoumakas, G., Katakis, I., & Vlahavas, I. (2010). Mining multi-label data. In *Data mining and knowledge discovery handbook* (pp. 667–685). Springer.

Tsoumakas, G., & Vlahavas, I. (2002). Distributed data mining of large classifier ensembles. In *Proceedings companion volume of the second hellenic conference on artificial intelligence* (pp. 249–256).

Tsymbal, A., Pechenizkiy, M., & Cunningham, P. (2005). Diversity in search strategies for ensemble feature selection. *Information fusion*, *6*(1), 83–98.

Tuv, E., Borisov, A., Runger, G., & Torkkola, K. (2009). Feature selection with ensembles, artificial variables, and redundancy elimination. *The Journal of Machine Learning Research*, *10*, 1341–1366.

Vainer, I., Kraus, S., Kaminka, G. A., & Slovin, H. (2011). Obtaining scalable and accurate classification in large-scale spatio-temporal domains. *Knowledge and information systems*, *29*(3), 527–564.

Vapnik, V. N. (1998). *Statistical learning theory.* Wiley.

Ventura, D., & Martinez, T. (1995). An empirical comparison of discretization methods. In *Proceedings of the tenth international symposium on computer and information sciences* (pp. 443–450).

Victo Sudha, G., & Raj, V. C. (2011). Review on feature selection techniques and the impact of svm for cancer classification using gene expression profile. *International Journal of Computer Science & Engineering Survey*, *2*(3), 16–27.

VOPTICAL_I1, VARPA optical dataset annotated by optometrists from the Faculty of Optics and Optometry, University of Santiago de Compostela (Spain). (n.d.). Retrieved November 2013, from `http://www.varpa.es/voptical_I1.html`

VOPTICAL_Is, VARPA optical dataset annotated by optometrists from the Faculty of Optics and Optometry, University of Santiago de Compostela (Spain). (n.d.). Retrieved November 2013, from `http://www.varpa.es/voptical_Is.html`

Wanderley, M., Gardeux, V., Natowicz, R., & Braga, A. (2013). Ga-kde-bayes: An evolutionary wrapper method based on non-parametric density estimation applied to bioinformatics problems. In *21st european symposium on artificial neural networks-esann* (pp. 155–160).

Wang, G., Song, Q., Xu, B., & Zhou, Y. (2013). Selecting feature subset for high dimensional data via the propositional foil rules. *Pattern Recognition*, *46*(1), 199 - 214.

Wang, I., Y.and Tetko, Hall, M., Frank, E., Facius, A., Mayer, K., & Mewes, H. (2005).

Gene selection from microarray data for cancer classification: a machine learning approach. *Computational Biology and Chemistry*, *29*(1), 37–46.

Wang, J., Wu, L., Kong, J., Li, Y., & Zhang, B. (2013). Maximum weight and minimum redundancy: A novel framework for feature subset selection. *Pattern Recognition*, *46*(6), 1616 - 1627.

Weiss, S., & Kulikowski, C. (1991). *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. Morgan Kaufmann, San Francisco.

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., & Vapnik, V. (2000). Feature selection for svms. In *Nips* (Vol. 12, pp. 668–674).

Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Wolfe, D. A., & Hollander, M. (1973). Nonparametric statistical methods. *Nonparametric statistical methods*.

Woods, J. (1972). Two-dimensional discrete markovian fields. *Information Theory, IEEE Transactions on*, *18*(2), 232–240.

Wright, J., Yang, A. Y., Ganesh, A., Sastry, S. S., & Ma, Y. (2009). Robust face recognition via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *31*(2), 210–227.

Yang, F., & Mao, K. (2011). Robust feature selection for microarray data based on multicriterion fusion. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, *8*(4), 1080–1092.

Yang, J., & Honavar, V. (1998). Feature subset selection using a genetic algorithm. *Intelligent Systems and Their Applications, IEEE*, *13*(2), 44–49.

Yang, S.-H., & Hu, B.-G. (2008). Efficient feature selection in the presence of outliers and noises. In *Information retrieval technology* (pp. 184–191). Springer.

Yang, Y., & Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Icml* (Vol. 97, pp. 412–420).

Yang, Y., & Webb, G. I. (2001). Proportional k-interval discretization for naive-bayes classifiers. In *Machine learning: Ecml 2001* (pp. 564–575). Springer.

Yang, Y., & Webb, G. I. (2002). A comparative study of discretization methods for naive-bayes classifiers. In *Proceedings of the pacific rim knowledge acquisition workshop (pkaw)* (pp. 159–173).

Ye, Y., Wu, Q., Zhexue Huang, J., Ng, M., & Li, X. (2013). Stratified sampling for feature subspace selection in random forests for high dimensional data. *Pattern Recognition*, *46*(3), 769 - 787.

Yeung, K., & Bumgarner, R. (2003). Multiclass classification of microarray data with repeated measurements: application to cancer. *Genome Biology*, *4*(12), 83–83.

You, D., Hamsici, O. C., & Martinez, A. M. (2011). Kernel optimization in discriminant analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *33*(3), 631–638.

Yu, L., & Liu, H. (2003). Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Machine learning-international workshop then conference-* (Vol. 20, p. 856).

Yu, L., & Liu, H. (2004a). Efficient feature selection via analysis of relevance and redundancy. *The Journal of Machine Learning Research*, *5*, 1205–1224.

Yu, L., & Liu, H. (2004b). Redundancy based feature selection for microarray data. In *Proceedings of the tenth acm sigkdd international conference on knowledge discovery and data mining* (pp. 737–742).

Zeleny, M., & Cochrane, J. L. (1982). *Multiple criteria decision making* (Vol. 25). McGraw-Hill New York.

Zhang, M.-L., Peña, J. M., & Robles, V. (2009). Feature selection for multi-label naive bayes classification. *Information Sciences*, *179*(19), 3218–3229.

Zhang, Y., Ding, C., & Li, T. (2008). Gene selection algorithm by combining relieff and mrmr. *BMC genomics*, *9*(Supl 2), S27.

Zhao, Z., & Liu, H. (2007). Searching for interacting features. In *Ijcai* (Vol. 7, pp. 1156–1161).

Zhao, Z., Zhang, R., Cox, J., Duling, D., & Sarle, W. (2013). Massively parallel feature selection: an approach based on variance preservation. *Machine Learning*, *92*, 195-220.

Zhao, Z. A., & Liu, H. (2011). *Spectral feature selection for data mining.* Chapman & Hall/CRC.

Zheng, Z., & Webb, G. (1998). Stochastic attribute selection committees. *Advanced Topics in Artificial Intelligence*, 321–332.

Zhu, Z., Ong, Y.-S., & Zurada, J. M. (2010). Identification of full and partial class relevant genes. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, *7*(2), 263–277.