FREDERICO GUALBERTO FERREIRA COELHO

SEMI-SUPERVISED FEATURE SELECTION

SEMI-SUPERVISED FEATURE SELECTION

FREDERICO GUALBERTO FERREIRA COELHO

Thesis presented to the Electrical Engineering Post-Graduate Program of UFMG and to the "Commission Doctoral de Domaine des Sciences de l'Ingénieur" of UCL, in an Agreement for International joint supervision of PhD thesis, as partial fulfillment of the requirements for the degree of Philosophy Doctor of the "Universidade Federal de Minas Gerais" and the degree of Doctor in "Sciences de l'ingénieur" of "Université catholique de Louvain"

Jury:
Prof. **A. Braga**, Supervisor (UFMG - School of Engineering, Department of Electronics Engineering, LITC)
Prof. **M. Verleysen**, Supervisor (UCL - ICTEAM Institute, Machine Learning Group)
Prof. **F. Rossi**, (Université Paris I Panthéon Sorbonne)
Prof. **M. Vellasco**, (PUC-RJ - Department of Electrical Engineering)
Prof. **H. Yehia**, (UFMG - School of Engineering, Department of Electronics Engineering)
Prof. **J. Lee**, (UCL - ICTEAM Institute, Machine Learning Group)

Scholarship from CNPq during stay in Brazil - Proc. n°41818/2010-7
Scholarship from CAPES during stay in Belgium - Proc. n°1456/10-5

Universidade Federal de Minas Gerais - Université catholique de Louvain

March 2013

Rapadura é doce...
mas não é mole não.

(Ditado popular de autoria desconhecida).

Dedicado à minha família.

(Dedicated to my family).

ABSTRACT

As data acquisition has become relatively easy and inexpensive, data sets are becoming extremely large, both in relation to the number of variables, and on the number of instances. However, the same is not true for "labeled" instances . Usually, the cost to obtain these labels is very high, and for this reason, unlabeled data represent the majority of instances, especially when compared with the amount of labeled data. Using such data requires special care, since several problems arise with the dimensionality increase and the lack of labels. Reducing the size of the data is thus a primordial need. In the midst of its outstanding features, usually we found irrelevant and redundant variables, which can and should be eliminated. In attempt to identify these variables, to despise the unlabeled data, implementing only supervised strategies, is a loss of any structural information that can be useful. Likewise, ignoring the labeled data by implementing only unsupervised methods is also a loss of information. In this context, the application of a semi-supervised approach is very suitable, where one can try to take advantage of the best benefits that each type of data has to offer. We are working on the problem of semi-supervised feature selection by two different approaches, but it may eventually complement each other later. The problem can be addressed in the context of feature clustering, grouping similar variables and discarding the irrelevant ones. On the other hand, we address the problem through a multi-objective approach, since we have arguments that clearly establish its multi-objective nature.

In the first approach, a similarity measure capable to take into account both the labeled and unlabeled data, based on mutual information, is developed as well, a criterion based on this measure for clustering and discarding variables. Also the principle of homogeneity between labels and data clusters is exploited and two semi-supervised feature selection methods are developed. Finally a mutual information estimator for a mixed set of discrete and continuous variables is developed as a secondary contribution. In the multi-objective approach, the proposal is try to solve both the problem of feature selection and function approximation, at the same time. The proposed method includes considering different weight vector norms for each layer of a Multi Layer Perceptron (MLP) neural networks, the independent training of each layer and the definition of objective functions, that are able to eliminate irrelevant features.

RÉSUMÉ

Que l'acquisition de données est devenue relativement simple et peu coûteux, les ensembles de données sont de plus extrêmement importante, tant en ce qui concerne le nombre de variables, et sur le nombre d'instances. Cependant, ce n'est pas le cas pour les instances étiquetées. Habituellement, le coût d'obtention de ces labels est très élevé, et pour cette raison, les données non étiquetées représentent la majorité des cas, surtout en comparaison avec la quantité de données étiquetées. En utilisant ces données nécessite un soin particulier, car plusieurs problèmes se posent à l'augmentation de la dimensionnalité et de l'absence d'étiquettes. La réduction de la taille des données est donc une nécessité primordiale. Au milieu de ses caractéristiques exceptionnelles, le plus souvent nous avons trouvé des variables non pertinentes et redondantes, qui peuvent et doivent être éliminés. Pour tenter d'identifier ces variables, à mépriser les données non étiquetées, la mise en œuvre des stratégies supervisées seulement, est une perte d'information structurelle qui peut être utile. De même, en ignorant les données étiquetées en mettant en œuvre uniquement des méthodes non supervisées est aussi une perte d'information. Dans ce contexte, l'application d'une approche semi-supervisé est très convenable, où l'on peut essayer de profiter des meilleurs avantages que chaque type de données a à offrir. Nous travaillons sur le problème de la sélection de caractéristiques semi-supervisé par deux approches différentes, mais il peut éventuellement se compléter plus tard. Le problème peut être résolu dans le cadre du regroupement option, le regroupement des variables similaires et en supprimant celles non pertinentes. D'autre part, nous abordons le problème par une approche multi-objectifs, puisque nous avons des arguments qui établissent clairement son multi-objective de la nature.

Dans la première approche, une mesure de similarité capable de prendre en compte à la fois les données marquées et non marquées, basée sur l'information mutuelle, se développe aussi, un critère basé sur cette mesure pour le clustering et les variables rejets. De plus, le principe d'homogénéité entre les étiquettes et les clusters de données est exploitée et deux semi-supervisé méthodes de sélection de caractéristiques sont développés. Finalement un estimateur informaton mutuelle pour un ensemble mixte de variables discrètes et continues est conçu comme une contribution secondaire. Dans l'approche multi-objectifs, la proposition est d'essayer de résoudre à la fois le problème de la sélection de caractéristiques et d'approximation de fonction, dans le même temps. La méthode proposée tient compte des normes différentes pour chaque couche d'un réseau MLP, l'entraînement in-

dépendante de chaque couche et la définition des fonctions objectives, qui sont capables d'éliminer des variables non pertinents.

RESUMO

Como a aquisição de dados tem se tornado relativamente mais fácil e barata, o conjunto de dados tem adquirido dimensões extremamente grandes, tanto em relação ao número de variáveis, bem como em relação ao número de instâncias. Contudo, o mesmo não ocorre com os "rótulos" de cada instância. O custo para se obter estes rótulos é, via de regra, muito alto, e por causa disto, dados não rotulados são a grande maioria, principalmente quando comparados com a quantidade de dados rotulados. A utilização destes dados requer cuidados especiais uma vez que vários problemas surgem com o aumento da dimensionalidade e com a escassez de rótulos. Reduzir a dimensão dos dados é então uma necessidade primordial. Em meio às suas características mais relevantes, usualmente encontramos variáveis redundantes e mesmo irrelevantes, que podem e devem ser eliminadas. Na procura destas variáveis, ao desprezar os dados não rotulados, implementando-se apenas estratégias supervisionadas, abrimos mão de informações estruturais que podem ser úteis. Da mesma forma, desprezar os dados rotulados implementando-se apenas métodos não supervisionados é igualmente disperdício de informação. Neste contexto, a aplicação de uma abordagem semi-supervisionada é bastante apropriada, onde pode-se tentar aproveitar o que cada tipo de dado tem de melhor a oferecer. Estamos trabalhando no problema de seleção de características semi-supervisionada através de duas abordagens distintas, mas que podem, eventualmente se complementarem mais à frente. O problema pode ser abordado num contexto de agrupamento de características, agrupando variáveis semelhantes e desprezando as irrelevantes. Por outro lado, podemos abordar o problema através de uma metodologia multi-objetiva, uma vez que temos argumentos estabelecendo claramente esta sua natureza multi-objetiva. Na primeira abordagem, uma medida de semelhança capaz de levar em consideração tanto os dados rotulados como os não rotulados, baseado na informação mútua, está sendo desenvolvida, bem como, um critério, baseado nesta medida, para agrupamento e eliminação de variáveis. Também o princípio da homogeneidade entre os rótulos e os clusters de dados é explorado e dois métodos semissupervisionados de seleção de características são desenvolvidos. Finalmente um estimador de informaçã mútua para um conjunto misto de variáveis discretas e contínuas é desenvolvido e constitue uma contribuição secundária do trabalho. Na segunda abordagem, a proposta é tentar resolver o problema de seleção de características e de aproximação de funções ao mesmo tempo. O método proposto inclue a consideração de normas

diferentes para cada camada de uma rede MLP, pelo treinamento independente de cada camada e pela definição de funções objetivo que sejam capazes de maximizar algum índice de relevância das variáveis.

First, I thank my God through Jesus Christ for you all... (Rom-1,8a)

—

ACKNOWLEDGMENTS

I thank God first! Sure, it could not be in another way. For me, it's simply impossible to deny that he leads my steps and that he always gives me the victory. After all that I have lived and witnessed from God in my life I can not deny their existence and his great love for me. It's very nice to walk with Someone who is able to solve any problem and always give me the best. I also include here a special thanks to my mother Mary, who always accompanies me, pleading for me to Jesus!

I could not found in Portuguese, English or even French, words that are able to express my gratitude to the people who made this possible. Without their loyal and unconditional support I would not have gotten this far. In particular, what can I say to my beloved wife and my beloved children. Actually, I think I can only express my gratitude in the language-of love, with many hugs and kisses to you, who have stood by my side, often giving up to your comfort and safety already established in our lives. I thank you for understanding my absences and fatigue, for the moral support in times of discouragement, and for all the happy times, and difficulties that we faced together. You were true gladiators, putting yourselves beside me in this experience of living in another culture outside of our beloved country. You my children, faced a change of country, culture, friends, life and ultimately come out of this fight victorious, because you overcame all difficulties, to learn the new language, to make new friends and to defend yourselves when was necessary. Bravo! And you Cris, who dedicated yourself with all your body and soul to give full support to our family, unconditionally, fighting against everything and everyone, and, at the final, winning always at our side. How to thank all of this? Moreover, how to explain in words all that we lived together? But one thing I know, my eternal gratitude and admiration for you all. Love you!

Likewise, I want to thank all our family, for understanding and bearing our absence. By giving us support and taking care of our things for us. Thanks Cassio, Vera, TiÃÂ£o, Beth, Herminio, Juninho, Herberth and everyone for the support and affection. We love you all.

And last but not least, I wish to thank all the teachers, colleagues and friends for their support, hospitality and help. Special thanks to you Michel, for the welcome, the dedication and commitment with me and my work, for your availability as well as for all support that was given to me. For me and my family was very important to know that we could count on you always. And I'm sure that I can continue to count on you in the continuity of the work and in future works

too. Merci beaucoup. Professor Braga, I don't know how to thank you! Once again I find myself without words to describe you were, and still is important to us. Thanks for the support, encouragement, commitment, availability, loyalty and friendship. Actually you are one of the pillars of my safe journey, and has been very good to walk with you and all colleagues and friends of LITC. And I'm glad that this is still just the beginning, and certainly, we still have a long way to walk together. Thank you.

I must thank all the friends and brothers of MMDP, especially to Elaine and her family, who, with all our friends, has struggled to provide support to our MM's family and community . And I wanto to give a special thanks to all friends and colleagues of LITC and DICE labs, for their support and welcome.

Well, in order to not forget anyone, I thank everybody. I really need to write the equivalent of another thesis, regarding the number of pages, just to try to thank more properly all of you.

Thank you very much! Merci Beaucoup!

AGRADECIMENTOS

Agradeço a Deus primeiramente! Claro, não podia deixar de ser. É inegável a ação de Deus em minha vida. Para mim, é simplesmente impossível negar que Ele conduz todos os meus passos e que Ele sempre me dá a vitória. Depois de tudo o que já vivi e presenciei de Deus em minha vida eu não tenho como negar sua existência, seu grande amor para comigo. E nem quero, pois é muito bom poder caminhar contando com Alguém capaz de resolver qualquer problema e que sempre me dará aquilo o que é melhor para mim. Também incluo aí um agradecimento especial à minha mãe Maria que sempre me acompanha, intercedendo por mim a Jesus!

Não encontrei na língua portuguesa, inglesa ou mesmo na francesa, palavras que sejam capazes de expressar minha gratidão às pessoas que tornaram tudo isto possível. Sem seu leal e incondicional apoio eu não teria chegado até aqui. Em particular, o que posso dizer à minha amada esposa e aos meus amados filhos. Na verdade, acho que só será possível expressar minha gratidão na linguagem do amor, com muitos abraços e beijos, à vocês que se colocaram ao meu lado, muitas vezes abrindo mão do seu conforto e da sua segurança já estabelecidas em nossas vidas. Pela compreensão e aceitação, muitas vezes das minhas ausências e cansaço, pelo apoio moral nos momentos de desânimo, e por todos os momento alegres e difíceis que passamos e ainda passaremos juntos. Vocês foram verdadeiros gladiadores, se colocando ao meu lado nesta experiência de viver em outra cultura, fora do nosso amado país. Vocês meus filhos, enfrentaram a mudança de país, de cultura, de amigos, de vida e ao final, saem desta luta,

vitoriosos, pois superaram todas as dificuldades, para aprender a nova língua, para fazer novos amigos e para se defenderem do que fosse necessário. Bravo!!! E você Cris que se dedicou de corpo e alma a dar todo o suporte à nossa família, incondicionalmente, lutando contra tudo e contra todos, e ao final, vencendo sempre ao nosso lado. Como agradecer tudo isso? Aliás, como explicar em palavras tudo o que vivemos juntos? Mas de uma coisa eu sei, da minha eterna gratidão e admiração por vocês. Amo vocês!

Da mesma forma, quero agradecer a toda a nossa família, por compreenderem e suportarem nossa ausência. Por nos darem o suporte tão essencial cuidando de nossas coisas por nós, para que pudéssemos nos lançar neste desafio. Valeu Cássio, Vera, Tião, Beth, Hermínio, Juninho, Herberth e todo mundo, pelo suporte e carinho. Amamos todos vocês.

E por último e não menos importante, gostaria de agradecer a todos os professores, colegas e amigos por todo o suporte, acolhida e ajuda. Agradeço em especial ao Michel pela acolhida, pela dedicação e comprometimento destinados a mim e ao meu trabalho, bem como por todo o suporte que me deu. Professor Braga, nem sei como agradecer a você! Mais uma vez me encontro sem palavras para descrever como você foi e é importante para nós. Obrigado pelo apoio, incentivo, comprometimento, pela disponibilidade, por todo companheirismo, pela lealdade e pela amizade. Realmente você é um dos pilares seguros desta minha caminhada e tem sido muito bom caminhar com você e com todos os colegas e amigos do LITC. E fico feliz por saber que isso ainda é só o começo, e que com certeza, ainda temos um bom caminho pela frente para trilharmos juntos. Muito obrigado.

Não posso deixar de agradecer a todos os amigos e irmãos do MMDP, em especial à Elaine e à sua família, que junto com todos os amigos, tem se esforçado para dar o suporte à nossa família dos MMs. E a todos os amigos e colegas do LITC e do DICE pelo suporte e acolhida. Agradeço também aos meus amigos da ESCHER por toda a compreensão e suporte, muito obrigado.

Bom, para não esquecer de ninguém termino agradecendo a todos. Eu realmente precisaria de escrever o equivalente a outra teste em número de páginas só pra tentar agradecer de maneira mais próxima ao necessário a todos vocês.

Obrigado.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACRONYMS

MLP Multi Layer Perceptron

MI Mutual Information

FS Feature Selection

LASSO Least Absolute Shrinkage and Selection Operator

MOBJ Multi OBJective

LMFS LASSO Multi objective Feature Selection method

SSFC Semi-Supervised Feature selection based on Clustering

RWP Reprinted with permition from [14]

LIST OF SYMBOLS

F, X	original feature set
Y	label output set (output variable set)
Γ	selected feature subset
Γ^{ℓ}	selected feature subset considering only labeled data
$\Gamma^{\ell u}$	selected feature subset considering labeled and unlabeled data
n_f	number of features of set F
n_s	number of features of set Γ
n_n	number of irrelevant features
n_r	number of remaining features in set F after selecting some feature
n_c	number of clusters
n_o	number of outputs
n_ℓ	number of labeled instances
n_t	number of test instances
n_u	number of unlabeled instances
n	total number of instances (sample size)
p_l	percentual of labeled data
n_d	number of crossvalidation folds
n_p	number of permutations
p	number of neurons
F_i	feature i from set F
X_i	feature i from set X
x	sample from X (is a vector composed by all features of X)
x_i	i^{th} sample from X (is a vector composed by all features of X)
Y_i	i^{th} output variable from set Y
y_i	i^{th} label sample from Y (is a vector composed by all output variables of Y)

Y_{cl}	cluster labels vector
R	relevance vector
R_c	relevance vector considering cluster label information
X	input data set or a random discrete variable
$X^{(l)}$	set of labeled data
$X^{(u)}$	set of unlabeled data
ω	vector of all weigths of a multi layer perceptron
W	vector of weigths of the output layer
Z	vector of weigths of the hidden layer or a discrete random variable
Λ	eigenvalues
ϑ	eigenvector
H	entropy
h	differential entropy
$\mathcal{H}$	mixed entropy
MI	mutual information
S	similarity measure
S_p	permutation of S
A	"unsupervised" term of S
B	"supervised" term of S
λ	parameter to balance influence of terms A and B of S
G	sum of the absolute values of weights of a feature in the hidden layer

Part I

BACKGROUND KNOWLEDGE

1

INTRODUCTION

Recently, due to the technological development of sensors, measurement systems and information storage hardware, collecting data is becoming easier and cheaper. Medical researches, as well as financial analysis, data mining and imaging are some examples of application fields that deal with, or are able to generate very large data sets. These applications have strongly demanded the development of methods that are capable to deal with new situations that appear in higher dimensions. Some problems that may arise involve dealing with redundant or irrelevant information, "curse of dimensionality" or "concentration of the euclidean norm" [96].

The analysis of higher dimensional data sets is difficult, not only because they are large in terms of number of observations, but also because of the large number of variables (features) that can be generated with the new sampling engines. In fact, in most applications that we aim at, the number of features n_f is much higher than the number of observed instances ($n_f >> n$). This is particularly true when considering that many usual statistical methods are developed for $n >> n_f$ under the normality assumption, what is not realistic for the majority of real problems. Analyzing three-dimensional data is quite different from analyzing instances with thousands of features, as in gene analysis applications [21]. For such higher dimensional space there is no way of directly visualizing the data set distribution. Moreover, for these huge number of variables it would be necessary to have an exponential number of instances (n^{n_f}) to overcome the curse of dimensionality problem. These are some reasons that explain the increasing interest in feature selection problems.

Particularly in these application fields new samples are easily generated, nevertheless, labeling data can be costly and time consuming [45]. Thus, it is usual to find data sets with scarce labeled instances and a large amount of untagged input samples, like in web-based information retrieval applications, and in many other problems whose data sets share the same following characteristics: high dimension, few labeled instances and a large number of unlabeled ones.

In this framework, the general problem of Semi-Supervised Feature Selection (SSFS) can be characterized by the selection of a minimal and sufficient feature subset, using the labeled data set $X^{(\ell)} = \{x_i, y_i\}_{i=1}^{n_\ell}$ and considering also the structural information implicit in the unlabeled data $X^{(u)} = \{x_i\}_{i=n_\ell+1}^{n}$. In other words, the principal aim of SSFS is to solve the problem using as less variables as possible [44], considering both labeled and unlabeled sources. Undeniably, reducing

the data dimensionality could help to get better understanding about the problems, allowing, in some cases, even some kind of visualization [96]. Increase in performance, reduction of computational and measurement times, reduction of measurement costs and data storage needs are also some of the well known advantages of reducing the number of features.

In order to reduce the number of features of a data set, there are three main strategies in which, more or less, all algorithms could be classified. In the first strategy there are the so-called *filter* methods whose objective is to rank the features according to some supervised [58, 72] or unsupervised [106, 105, 62] criterion. These methods are fast and easy to implement. In the second strategy, the algorithms are called *Wrappers* and they access the accuracy of a given model in order to find the best feature subset. The third category of methods refers to the *embedded* algorithms. As Wrapper methods, embedded ones uses the model as part of the selection process, but perform feature selection during the training process [26, 85, 89].

In this thesis we approach the semi-supervised feature selection problem by two different paths. Our first approach is based on the idea of feature clustering, i.e., features that are similar should be grouped, reducing the dimension of the problem, so that redundant information is reduced. We start the development of this approach from this main idea, applying it to a simple hierarchical clustering method. We propose here a new similarity criterion that is able to consider both labeled and unlabeled data. Also we defined a stopping criterion for the feature clustering method, which is based on the statistical significance of the similarity index. A criterion to identify relevant features is also proposed taking into account the statistical significance of some relevance measure related to the target labels and the relevance of each feature in relation to the data distribution.

Still in this clustering approach we exploit the principle of homogeneity between labels and data clusters in the development of another semi-supervised feature selection method. This principle permits the use of cluster information to improve the estimation of feature relevance in order to increase selection performance. We use Mutual Information in a Forward-Backward search process in order to evaluate the relevance of each feature to the data distribution and the existent labels, in a context of few labeled and many unlabeled instances.

Our second approach is based on the principle that the feature selection problem, as well as the learning problem are inherently multi-objective. We argue that the semi-supervised feature selection problem can be jointly solved with the induction of the Wrapper model, which should be selected according to Muti-Objective learning principles. Based on this we develop a feature selection method whose

objective function is designed to eliminate features while solving the learning problem.

Real problems are composed of a set of discrete and continuous variables. During our research we seek in the literature for estimators of mutual information able to handle with a mixed set of variables without sucsses. Then we developed an estimator that can deal with such kind of data, and it is also presented in this thesis as another contribution.

This thesis is organized as follows. We begin by discussing about the learning and feature selection paradigms in Chapter 2 trying to set a common background of the main used strategies, and introducing the principles of the semi-supervised approach, where we base our work. Then we address the Feature Selection itself in Section 3, where we try to explain the important general topics and its difference to the feature extraction task. We make a general picture of the supervised and unsupervised feature selection methods (Sections 3.1 and 3.2) establishing their bases and then we describe the short state of art of semi-supervised feature selection in Section 3.3. Then we propose two semi-supervised feature selection methods developed during this PhD in the feature clustering approach in Section 4, and one feature selection method proposed in the multi-objective approach in Section 5. Finally, in Chapter 6 we present the mutual information estimator for a mixed set of variables.

The methods proposed within the multi-objective approach, as well as within the feature clustering one, start from very simple ideas, and for while, they are unrelated paths. However, we believe that, maybe, they could be merged at some point in the future. We aim to continue developing these ideas in a efficient way and, only adding more elaborated solutions when they are necessary.

2

LEARNING AND FEATURE SELECTION PARADIGMS

The main paradigms of machine learning are Supervised Learning (SL), Unsupervised Learning (UL) and more recently Semi-Supervised Learning (SSL). In general, when there is a labeled data set $X^{(\ell)} = \{x_i, y_i\}_i^{n_\ell}$, with n_ℓ observed instances of a given problem, composed by the input variables vector $\mathbf{x}$, and by their respective output variable vector $\mathbf{y}$ (or the target concept variables), the SL schema could be applied. The main objective of the Supervised schema is learn a mapping function $\hat{\mathbf{y}} = f(\mathbf{x}|\alpha)$, where α is the set of parameters of the chosen model, in order to correct classify or predict the target value for any new observation. If the data set is ideally sufficient to describe the data distribution, the general function that originated the data can be found. The general scheme to perform the Supervised Learning is shown in Figure 2.1.

supervised schema

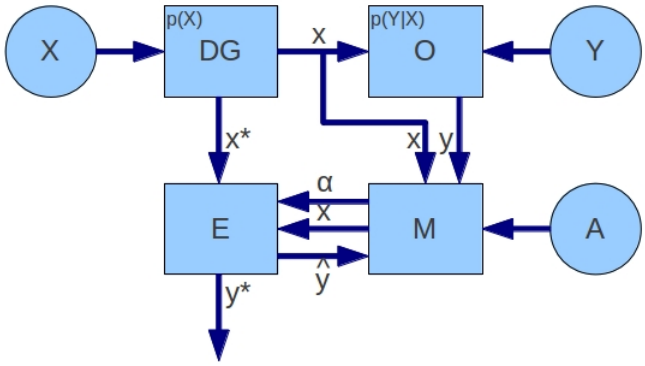


Figure 2.1: Supervised Learning Scheme. *RWP*

In Figure 2.1 there is a block, named DG, representing the Data Generator, that provides the data according to a generic marginal distribution $p(x)$. The data set $X^{(\ell)}$ is sampled from DG and delivered to the Oracle O which, for each observation in this data set, knows the target variable y. In other words, given x, the oracle, somehow, know the target y, so an approximation of the conditional distribution $p(y|x)$ is known. A Model M can be set with the data x and the target variable y, in order to be used by an Estimator E to classify or predict the target variable y^* for a new data x^*, given a set A of parameters α. Model M can be fed with the output estimation $\hat{y}$ in order to help in the fine tuning of α parameters.

unsupervised schema

If, for a given problem, there is no information about the target variable, and the only available information is a data set $X^{(u)} = \{x_i\}_{i=n_\ell+1}^{n}$ composed by n_u observations of the input variables, what can be done is try to uncover the data structure. In the Unsupervised Learning schema the main goal is, broadly speaking, to estimate the densities that are like to have generated the data set $X^{(u)}$. As shown in Figure 2.2, the conditional probability $p(y|x)$ isn't known, i.e., there is no labeling agent in the UL scheme. Considering this, the model M has to be built with sufficient ability to extract some structural information from $X^{(u)}$, in order to provide to the Estimator one parametric model to estimate the probability $p(x^*|\theta)$ of a new data x^* to belong or not to one specific class (given a parameter set Θ). Roughly speaking, we will look for clusters in this schema.

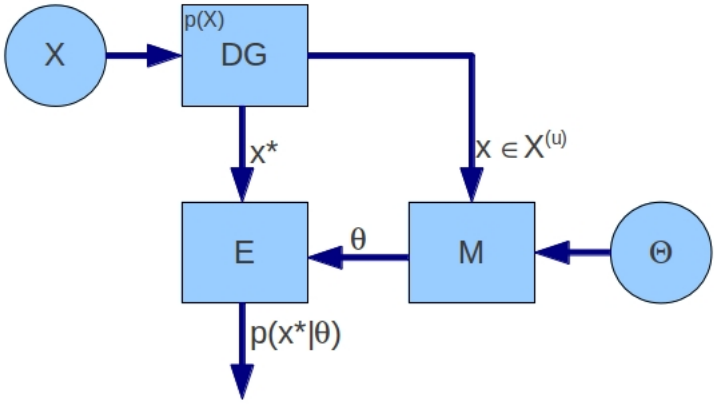


Figure 2.2: Unsupervised Learning Scheme. *RWP*

semi-supervised schema

The Semi-Supervised Learning, on the other hand, is considered to be halfway between Supervised and Unsupervised Learning. For this learning scheme, both $X^{(\ell)} = \{x_i, y_i\}_i^{n_\ell}$ and $X^{(u)} = \{x_i\}_{i=n_\ell+1}^{n}$ data sets are available, and the objective is try to use them, in some way, to build the model. The SSL can be performed within two different perspectives: in the Supervised Learning Perspective where the learning task is executed in a supervised scheme with additional structural information inferred from $X^{(u)} \cup X^{(\ell)}$; or in the Unsupervised Learning Perspective, where the learning task is performed in an unsupervised scheme with side constraints.

The general scheme for the SSL schema with the supervised perspective is shown in Figure 2.3. This scheme is basically the same for the SL one adding some information from the unsupervised data. The data $x \in X^{(\ell)}$, for which the target variables y are known by the labeling agent O, is drawn from DG and provided, together with the label information, to the model M_{ss}. This model M_{ss} differs from the one in the pure Supervised Learning because it is able to use some

probability density estimated by the estimator E_u. This estimator, in turn, was tuned by the unsupervised model M_u. By the way, the M_u model works together with the data $x \in X^{(u)}$, whose output variable values are not known by the Oracle, and, of course, with the data from $X^{(\ell)}$, both drawn by DG from the same distribution.

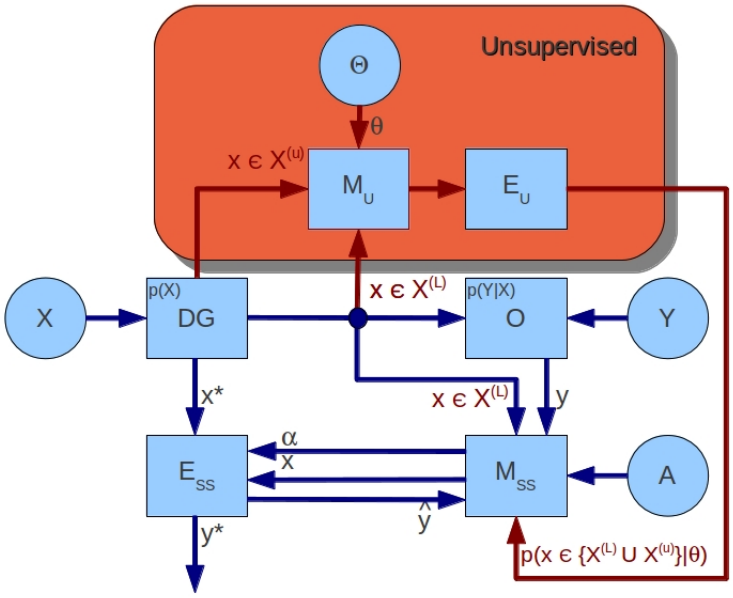


Figure 2.3: Semi-Supervised Learning Scheme within the Supervised Perspective.*RWP*

Figure 2.4 shows the scheme for the SSL under an Unsupervised Perspective. The approach for this perspective is based on the Unsupervised Learning scheme, adding the labeling agent to handle with the part of data whose target variables are known. Then, the model M has to provide some probability density inferred from the input data densities with support of the labeled data. In other words, the search for natural clusters in the data has to be consistent with the labeled data.

There are a lot of real problems where do not exist only labeled data. For example, just to mention a very simple one, every day a lot of web pages are created and stored in the internet and not properly classified or indexed. So, problems like data mining in the internet count with a huge unlabeled data set (not classified web pages). It is important to notice that these unlabeled data sets are usually huge when compared to the labeled data set, i.e. the number of unlabeled instances is much bigger than the number of instances of the labeled data set ($n_u >> n_\ell$).

This is, by the way, an important argument in favor of performing SSL. As n_ℓ is too small, the labeled data set $X^{(\ell)}$ have so few obser-

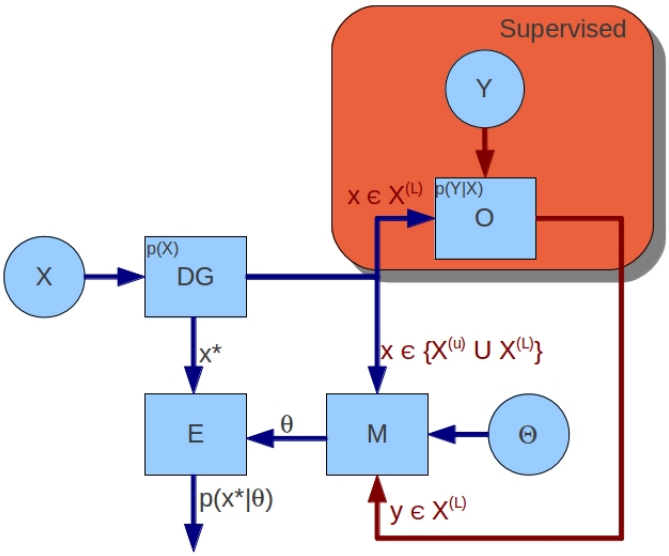


Figure 2.4: Semi-Supervised Learning Scheme within the Unsupervised Perspective. *RWP*

vations, that they are not sufficient to represent the class-condition densities (undersampling), favoring the occurrence of problems as biased sampling. This means that these few samples may have been obtained by a very specific condition, so, they are biased by this condition, for example, in the case of on-line polls. Individuals that have more strong convictions are more likely to fill the poll rather than persons with weak convictions about the subject. This fact could be even worse if the assumption of i.i.d. of the data is not meet, then, having more information should be interesting. In that sense, if there are a lot of data, even unlabeled, probably it is possible to get some useful structural information from data distribution, which could be used in order to increase the model accuracy. In other words, as the learning problem may not be completely specified by $X^{(\ell)}$, maybe, prior information (or assumptions) will be needed, and, the knowledge about $p(x|\theta)$ should improve the estimation of $p(y|x)$. That is why the Semi-supervised approach is used in the learning task.

high dimensional problems

Together with all these problems, frequently, the learning task is faced with another important detail: in many problems, not only the small number of labeled instances is the main problem, but also the number of variables. Nowadays, with the advance of technologies, like microarrays analyses in the bio-informatics field for example, the number of variables could reach thousands, or even tens of thousands. In theory, dealing with large number of features, especially when it is larger than the number of instances ($n_f >> n_\ell$), is possible, but, in practice it couldn't be feasible. Problems like the "curse of

dimensionality" [96], "concentration of measure phenomenon" [96] and overfitting have to be taken into account. Of course, as the required size of the labeled data set, in order to assure good representation of the data densities, is not met, two actions could be taken in order to avoid or minimize these problems:

- label new instances: but labeling data is, in general, very costly and time demanding. It would be nice, however, by definition, this is not a feasible solution;
- reduce the number of variables: if we succeed in finding redundant or irrelevant features we can eliminate them and reduce the dimension of the data set.

So, feature selection seems to be a good way to deal with these problems. In order to have a broader view of the problem that we want to address, we will formally start defining its scope. We are interested in dealing with databases characterized by a large number n_f of features, a small number n_ℓ of labeled instances and by a high number n_u of unlabeled samples. The data set $X = \left\{X^{(\ell)} \cup X^{(u)}\right\}$ is defined by the union of the labeled data set $X^{(\ell)} = \{x_i, y_i\}_i^{n_\ell}$ and the unlabeled data set $X^{(u)} = \{x_i\}_{i=n_\ell+1}^{n}$ where each instance is defined as a vector $x_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,n_f}\}$ where n_f is the number of features, and the labels or target variables are defined as a vector $y_i = \{y_{i,1}, y_{i,2}, \ldots, y_{i,n_o}\}$ being n_o the number of target variables. In our case, for simplicity we will consider only one output variable then y_i will be the class label or the target value for instance x_i in the labeled data set. Finally, lets define that the data set X has $n_u >> n_\ell$ and $n_f >> n_\ell + n_u$.

problem definition

Performing feature selection in a data set with a small number of labeled instances, whose number of features are huge, tends to fail if done in a Supervised scheme. For example, the relevance of a feature can be measured by its correlation with the target variable and, when evaluated in a data set too small, features that seems to be very relevant, could not be at all. On the other hand, unsupervised methods do not take into account the labeled information, which are few but important. As the problem may not be completely specified by $X^{(\ell)}$ or $X^{(u)}$ separately, maybe, considering the $X^{(\ell)}$ data set with some additional information from unlabeled data set $X^{(u)}$ could help the feature selection task.

Lets define the set $F = \left\{f_1, f_2, \ldots, f_j\right\}_{j=1}^{n_f}$ of all features, where each feature $f_j = \left\{f_{j,1}, f_{j,2}, \ldots, f_{j,i}\right\}_{i=1}^{n_\ell+n_u}$ is a vector composed by the values of feature j of all instances. The selection of a minimal feature subset Γ, where $\Gamma \subseteq F$, which is necessary and sufficient to induce a model capable to explain the data generation and rightly map the input data to the target variable, using both the labeled and unlabeled data sets is called *Semi-Supervised Feature Selection*.

In the same way that was defined before for the Semi-Supervised Learning, in this *selection* scheme, both $X^{(\ell)} = \{x_i, y_i\}_i^{n_\ell}$ and $X^{(u)} = \{x_i\}_{i=n_\ell+1}^{n}$ data sets are available, and the objective is try to use them, in some way, to build the model. The *Semi-Supervised Feature Selection* schema (SSFS) also can be performed within the two different perspectives: within a *supervised perspective* or within an *unsupervised perspective*.

semi-supervised FS schema

The general scheme for the SSFS with the supervised perspective is shown in Figure 2.5. This scheme is basically the same shown for the SSL one, but here, the so-called M_{ss} model has also the capability to select features. Likewise in the SSL, the data $x \in X^{(\ell)}$, with marginal distribution $p(x)$, and whose target variables Y are known by the labeling agent O, is drawn from the Data Generator DG and provided, as well as the label information, to the model M_{ssfs}. This model is able to use some information Ψ inferred from X by the unsupervised model M_{ufs}, which in turn, was tuned by a given parameter set Θ. The M_{ufs} model works with $x \in X$ drawn by DG without the need of known the output variable values. A subset Γ is defined by M_{ssfs} and given to a specific learning model M in order to predict the target variable or classify the data, for any new instance x^*. The model M_{ssfs} may need to access the accuracy of the model M if it is a wrapper or a embedding method.

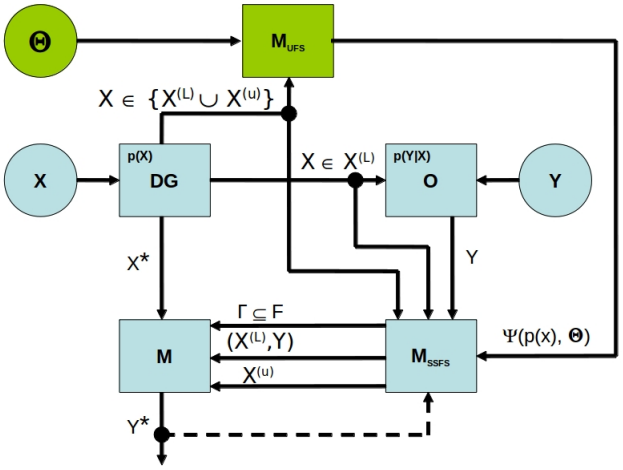


Figure 2.5: Semi-Supervised Feature Selection Scheme within the Supervised Perspective. *RWP*

Figure 2.6 shows the scheme for the SSFS under the *unsupervised perspective*. Basically, the model M_{ufs}, given a parameter set Θ tries to select features based on the data densities over all instances in X and using the label information given by the labeling agent O for the part of data whose target variables are known.

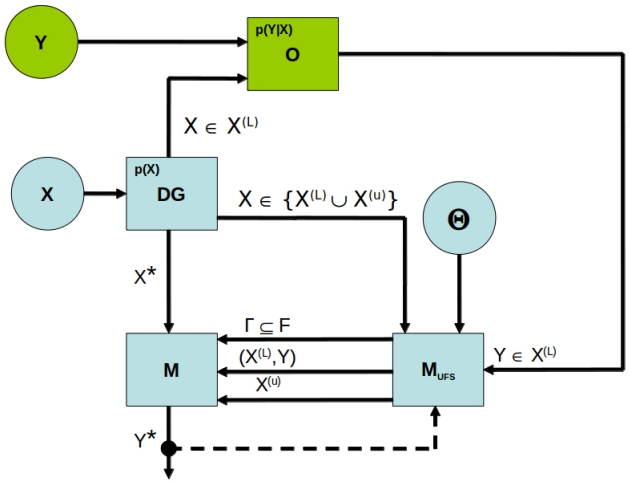


Figure 2.6: Semi-Supervised Feature Selection Scheme within the Unsupervised Perspective. *RWP*

To classify any semi-supervised method into one of these paradigms is not always a simple task. Nevertheless the semi-supervised methods proposed here in this thesis are more likely to be classified into the *semi-supervised feature selection scheme within the supervised perspective*.

3

STATE-OF-THE-ART

Feature Selection is a topic whose importance is growing enormously lately, especially in fields like medical research, financial analysis, data mining and imaging, just to cite a few ones. There is a wide range of applications in these and other fields, like EEG recordings and gene analysis which have, among their most important objectives, the attempt to better understand the pathologies and, obviously, find more adequate treatments. Analysis of financial time series by the stock market agents and the consumer behavior by credit card companies are also applications that deal or produce huge data sets and have been strongly demanding the utilization of tools to "select the features" that are more relevant for their analysis.

The principal aim of feature selection is to solve a problem using as few variables as possible [44], either for data mining, function approximation or identification.

dimension reduction

The fact that the means for collecting data are becoming more easy and cheap, coupled with the very need to understand the problems and find solutions, like in medical researches using gene expressions of cancer tumors for example, produces incredibly large databases. However, just collect as much data as possible does not means that it will be more easy to solve the problems. Intuitively we are led to think that more information is better than less information, but having a lot of data does not mean that we have the right information. Then, problems related to the increase of the space dimensionality, like redundant or irrelevant information, the "curse of dimensionality" or the "concentration of the Euclidean norm" [96] arise and have to be considered in order to be successful when dealing with such databases.

high dimensional issues

It is difficult to analyze these databases because they are large in terms of the number of observations, and because of the large number of variables. It is very different to analyze one hundred observations of three-dimensional data, whose data can be visualized in a three dimensional graph, from analyzing one hundred instances with thousands of features, as is the case in gene analyses applications. For such large number of variables it would be necessary near n^{n_f} instances to overcome the curse of dimensionality problem. Especially considering that many classical analysis tools are developed for $n >> n_f$ and under the assumption that data is normally distributed, which is not usually true in real problems. In these cases feature selection is a way to overcome all these difficulties.

Nevertheless, there is another specific characteristic of these large databases that has to be considered. It is not so easy to collect data that are associated to the output variables or to the target concepts, for a given problem, because labeling data is costly and time consuming [45]. Therefore it is usual to find data sets composed with more unlabeled data than labeled data. In the data mining domain, for example, when someone types some words in the search engine of his browser, he want to receive a list of "all" pages related to that subject. However, there are thousands or even millions of web pages on the internet, without any proper index or classification about their contents. Of course the search mechanism, in order to return to the user the list of web pages most related to a specific subject, can deal only with some well indexed web pages, but this would reduce drastically the search space. One good idea is try to get some information from the large number of non-indexed web pages. The feature selection performed in such way, using labeled and unlabeled data, is called Semi-Supervised Feature Selection (SSFS) and this concept will be better explained in Section 3.3. Like in this example, there are a lot of problems sharing the same characteristics: high dimension, few labeled instances and a lot of unlabeled ones. The challenge is to use some structural information from the labeled and unlabeled instances in order to improve the solutions.

scarce labeled data

To reduce the number of features of a data set, there are three general strategies where, more or less, algorithms could be classified. In the first strategy there are the so-called filter methods whose objective is to rank the features according to some relation (or correlation)[58, 72] between the input variables and the target concepts [106, 105, 62]. These methods are fast and easy to implement, but, filters that use a univariate index, in general, fail in considering the relevance of a given feature in the presence of other features [44]. In the second strategy, the algorithms are called wrappers and they access the accuracy of a given model in order to find the best feature subset. Technically they are the best approach, but they usually have to perfor an exhaustive search over all possible subsets of features which is costly and, even for problems with relatively small dimension, unfeasible. The third category of methods refers to the embedded algorithms. As Wrapper methods, embedded ones uses the model as part of the selection process, but perform feature selection during the training process [26, 85, 89].

feature selection strategies

The feature selection task can be also classified into three distinct categories according to the type of available data. This classification was explained in details at item 2 and here we limit ourselves to list them with a brief resume of their characteristics:

- **Supervised Feature Selection**: where there are only labeled data available, to perform the task in a supervised scheme, as detailed at item 2;

feature selection categories

- **Unsupervised Feature Selection**: the data has no labels or target variables, and the selection task has to be done in a unsupervised scheme;
- **Semi-Supervised Feature Selection**: there are both labeled and unlabeled data available to perform the task.

Many methods of each one of these categories can be implemented in, at least, two basic search strategies named *Forward Selection* or *Backward Elimination*. In the forward feature selection the algorithms begin with an empty set of features and go on adding relevant variables, given previously selected variables. This strategy has smaller capability to finding more complementary features, compared to backward feature selection, because it starts within a context of none selected feature. In the other hand even the smallest nested subset is predictive. This last characteristic is more interesting when dealing with the trade performance for number of features. The backward elimination begins with a set of all features and goes on eliminating irrelevant variables, given the remaining selected variables. This method is capable of finding complementary features, because it start its analysis with all features, however, its performance is degraded for smallest nested subsets [42].

FEATURE SELECTION VERSUS FEATURE EXTRACTION

The term *feature extraction* has a different meaning from *feature selection*. They are different in the sense that the second one selects a reduced subset from the set composed by all features, while the first one aims to construct a second smaller set of variables, using all features from the original set. In the literature there are a lot of methods that perform feature extraction and some of them are very popular as Principal Component Analysis (PCA) or ISOMAP.

feature extraction methods

The PCA [56] method transforms the data to a new coordinate system in which the greatest variance of projected data will be the first coordinate. The ISOMAP [88] method tries to discover the manifold structure generating a mapping that preserves the geodesic structure. Other methods that perform feature extraction also can be cited, as Locally Linear Embedding (LLE) [78], Laplacian Eigenmaps [9], a linear method called Locality Preserving Projections [50] (LPP) and a method that tries to find the low-dimensional representation of each data point, looking for a function that minimizes the error when reconstructing the high-dimensional data point representation [104].

Another well known method is the supervised Linear Discriminant Analysis one (LDA) [32], which calculates projection vectors in order to maximize the between-class-variance and minimize the within-class-variance of projected data points. In [2] the authors extend this method to the *semi-supervised* case, the SDA (Semi-supervised Dis-

semi-supervised feature extraction methods

criminant Analysis), whose objective is to find a projection, respecting the discriminant structure inferred from labeled data, and respecting the intrinsic geometrical structure inferred from both labeled and unlabeled data. The labeled data is used to maximize the separability between classes, and the unlabeled data is used to estimate the intrinsic geometrical structure of the data.

In [103] the authors proposed the so called SSDR algorithm (Semi-supervised Dimensionality Reduction). This method tries to preserve in the projected low-dimensional space, the structure of original data space (with high-dimension) and the pairwise constraints previously defined. It exploit the *must-links* and *cannot-links* constraints together with *unlabeled data*. Its motivation using unlabeled data is enhance performance and stability when constraints are few.

Hou et. al. in [54] extend the Semi-supervised Dimension Reduction method [103] cited before, in order to use multiple representation forms of data, employing the domain knowledge in the form of pairwise constraints.

Alternatively, in [101] the authors tries to use some prior information to improve the stability of solutions, withal, they show that in the case of LLE and Local Tangent Space Alignment methods the prior information only helps in limiting the freedom of translation and scaling, and, when applying to ISOMAP method, it does not result in any important improvement.

All these methods, somehow, tries to project the data into a new feature space with reduced dimension, nevertheless, in this thesis we are interested in deal with the *Feature Selection* task, instead of *Feature Extraction*.

In the literature there are a lot of methods that perform Feature Selection in a supervised and in a unsupervised way. In this section we intend to present some of them, trying to explain their main ideas and assumptions before introducing properly the state of art of the semi-supervised feature selection. It is important not only know the existing methods related to the semi-supervised feature selection task, but also to search for ideas in the purely supervised and unsupervised methods in order to get some good insights to our goal.

3.1 SUPERVISED FEATURE SELECTION

In the following lines we present some existing supervised methods as examples of each one of the approaches mentioned before: filter, wrapper and embedded methods, which could also be classified as univariate or multivariate methods, or even a mixture of concepts. All these methods are based on a supervised feature selection paradigm which can be derived from the general scheme to perform the supervised learning as shown in Figure 2.1 from Section 2.

Pearson based method

We start with one classical individual relevance index that is very simple and well known, the **Pearson correlation coefficient** [71]. This coefficient is shown by equation 3.1:

Pearson coefficient

$$C(j) = \frac{\left|\sum_{i=1}^{n}\left(x_{i,j} - \bar{x}_j\right)\left(y_i - \bar{y}\right)\right|}{\sqrt{\sum_{i=1}^{n}\left(x_{i,j} - \bar{x}_j\right)^2 \sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}}, \tag{3.1}$$

where n is the number of instances, X_j is the vector containing all instances of the j^{th} feature, $x_{i,j}$ is the i^{th} instance of X_j, y is the vector containing the target values and the bar notation means average over the i elements of X_j.

This very simple linear method is a univariate one, because it makes independence assumptions between features. The features are ranked according to their coefficients, with a low computational and statistical complexity. It has a linear computational cost with respect to the number of features and instances. Nevertheless, the fact of not considering the effect of the iterations among features is its drawback, because single features that do not have any relevance, can be extremely useful when considered in the presence of others! This constitutes the main reason that why we do not use this coefficient in the development of our methods.

Gram-Schmidt based methods

The Gram-Schmidt orthogonalization procedure [6] can be applied in a method to select features. It is a linear multivariate and selects variables by adding progressively variables, which correlate to the target, in the space orthogonal to the variables already selected.

In the work of Oukhellou et. al. [69], a version of Gram-Schmidt orthogonalization procedure is used to iteratively rank the features. In a first step the relevance of each feature is determined measuring the angle between itself and the model output. The feature with the lower angle is then better ranked. In a second step the method orthogonalizes the remaining features with respect to the first ranked one. In the sequence, it chooses the next feature with lower angle between itself and the model output, as done in the first step. And the procedure goes on recursively until all features are ranked. Once all features are ranked a threshold has to be fixed in order to determine the number of features in the final selected subset. The *Akaike Information Criterion* is used to set this threshold, however it is efficient when the number of observations is much greater than the number of features. If it is not the case other criterion is used. A random feature (descriptor) is created and added to the initial feature set. The method rank all features, including the random feature, using the orthogonalization

Gram-Schmidt

method as explained before. At the end, all features ranked below the random feature are considered irrelevant and discarded. However, it is necessary to generate a lot of random descriptors to make a good estimation of the cumulative distribution function of its rank.

This last criterion is applied in another work [85] in a slightly different way. The Gram-Schmidt method is also used to rank features, but in this case a new procedure to stop selection and to select the best ranked features was developed. The idea is not to rank all features and to select a threshold, but to stop the process at any point. The criterion is also based on the cumulative distribution function of the rank position of a random probe added to the initial feature set. It states that since the probe is a random variable, its rank position is also a random variable. At each step of Gram-Schmidt method the features are orthogonalized, and then the feature with smallest angle value related to projected output vector is selected. The cumulative distribution value is then evaluated and if this value is smaller than a defined risk this feature is definitely selected and the next step of Gram-Schmidt can be performed. Otherwise the process is stopped. This risk is defined as being the probability of the selected feature be less significant than a random one (probe).

Recursive Feature Elimination Support Vector Machine algorithm

The *Recursive Feature Elimination Support Vector Machine algorithm* (RFE-SVM) [43] is an example of a backward elimination wrapper method. The idea is very simple: a SVM is trained and then, the feature with smallest weight in absolute value is discarded. Then the model is trained again without this feature and the process continues recursively until the desired number of selected features is reached. Of course, this method has a expensive computational cost that could be a little less if more than one feature is discarded at each iteration, however, it can compromise the optimality of the result. This multivariate method can be extended to the non-linear case too.

RFE-SVM

RELIEF based methods

Feature Selection, by definition, is the problem of selecting one small feature subset which is "*ideally, sufficient and necessary*" [58] to represent the target variable (or target concept). Filter methods usually evaluate a feature subset by a criterion function and, of course, with an exhaustive search over all possible feature subsets, the minimum subset can be found, but this option, mostly, is not feasible. On the other hand, heuristics strategies cannot guarantee optimality. Instead of searching over all feature subsets combinations, RELIEF performs a direct search over the set of all features, assigning different weights to each feature and then, identifying those that are statistically relevant.

The supervised RELIEF method was first proposed by Kira et al. [58], and its main propose was to avoid the problem of an exhaustive search in a Feature Selection framework, or even prescind from choosing an heuristic strategy to this search. It is a non-linear feature ranking filter.

RELIEF is based on the maximal margin concept, so, the weight calculated for one feature will be as high as discriminative this feature could be. Roughly speaking, this method takes for each instance in the training set, the nearest neighbor from the same class (Near Hit) and the nearest neighbor from the other class (Near Miss). Therefore the distance[1] between each instance and its *near hit* and *near miss* neighbors, for each feature, is evaluated. The Relevance value assigned to each feature will be the sum of the difference between these two distances over all instances. The far from each other, along the axis defined by the considered feature, are these differences for each instance, the more discriminative is this feature and the larger is its weight. Finally the selected features will be those whose relevance values are higher than a given threshold.

RELIEF

It is important to highlight that RELIEF only works well if the relevance values of the relevant features are sufficiently larger than those of the irrelevant features, and, of course, if a threshold can be defined to split them. However, these are not the only problems for RELIEF. As the search is performed assigning weights to each feature instead of selecting subsets, these weights can be thought as a space mapping, therefore, the features in the original data space are being mapped into a new feature space that we will call *induced* feature space. The problem here is that two instances that are close in the original feature space, may not be, in the induced feature space. Another important problem is concerning the margin definition: the margin is calculated as an average margin and any outliers will have a high influence over its value. Then, the I-RELIEF [86] was developed based on RELIEF in the attempt to avoid these problems, being able to handle with large data sets.

I-RELIEF

The main idea of I-RELIEF, to overcome these issues, is to compute the margin in the "induced" feature space. But in this case, the weights are not known before learning, so the solution was developed one probabilistic model where the neighbors are treated as "latent variables". The margin defined in terms of this probabilistic model takes into account the probability of neighbor of an instance belonging to the same class of another one or not. See [86] for details.

This last method (I-RELIEF), was extended to the semi-supervised case in [62], considering the margin of the unlabeled data (large margin principle) in the objective function of the Logistic I-Relief method presented in [86]. This semi-supervised method is detailed in Section 3.3.

1 in this case, was used the euclidean distance

Optimal Brain Damage

The central idea of OBD method [26] (*Optimal Brain Damage*) is to determine, in a neural network, which are the connections (weights) to be pruned first using the *saliency* value. The saliency measures how important is one parameter in relation to the output of the network, or, in other words, which are the weights, that when deleted, less affects the training error. Basically, one has to train a network, evaluate the saliency values of each weight, and discard the one with the smallest saliency value. In this work the saliency of each network parameter is evaluated using one meaning justified formula, and not just considering the magnitude weights as a measure of the saliency. See [26] for more details. The procedure is recursively performed until a stopping criterion is reached. OBD is an example of backward elimination embedded method. It is non-linear and, as all variables are considered together in the calculations, it is a multivariate method.

OBD

Mutual Information based methods

In [61] the authors propose a very simple supervised method to select the important features using the mutual information as a similarity measure. Similar features are clustered in an hierarchical way and after the clustering step, the relevance of each cluster to the output is evaluated within a forward procedure. The Mutual Information (MI) evaluated between the features is used to evaluate the redundancy in the first step and the MI between the features and the output variable is used to evaluate their relevance in the next step. There is one slight detail in the way that they apply the hierarchical method: here only consecutive features could be clustered because the method was being applied to a frequency spectral data. The clustered features are then replaced by a *mean* feature. For further information please refer to [61].

MI based methods

Another method based on MI to select features was developed by François et al. [35]. It uses the permutation test to set a threshold for the mutual information between each variable to the output and therefore select those features whose MI value is greater than this limit. The permutation test is also used in [92, 28] to find statistically significant relevant and redundant features by means of filters using mutual information between regression and target variables. After that, good candidate feature subsets are searched by a wrapper, taking into account the regression model. This is an hybrid filter/wrapper features selection algorithm.

• • •

Here we presented only some examples of supervised methods. A complete review on this topic is beyond the scope of this work,

however, a very good review on these supervised methods and feature selection can be found in [44].

3.2 UNSUPERVISED FEATURE SELECTION

Given the nature of the problems that we want to address, and the fact that we have to deal with a lot of unlabeled data, we decided to give an special attention to the unsupervised feature selection methods. Maybe, we can get some good ideas in order to develop our method and achieve our goals. These methods have some interesting advantages and among then we can highlight that they are unbiased by any experimental expert, they perform well in the absence of any a priori knowledge and reduce the overfitting risk. Nevertheless, their main drawback is that they rely on some mathematical principles with no guarantee that it will work for any kind of problems. These feature selection methods are based on a unsupervised paradigm which can be derived from the general scheme to perform the unsupervised learning as explained in Section 2 schematized in Figure 2.2. We found some interesting unsupervised methods in the literature and the following paragraphs are dedicated to their presentation.

KNN based method

The first method to be presented is a KNN based one which performs feature selection using feature similarity. In [68] the authors propose a new metric called *maximum information compression index* (MICI) to be used in the algorithm. This new metric is defined as the smallest eigenvalue of the covariance matrix evaluated for any two random variables

KNN

The value of MICI is zero when features are linearly dependent indicating that these two features are very similar and can be grouped. Basically, the algorithm selects the most compact feature set, i.e. the cluster of neighbors of each variable which has the smallest distance to its k^{th} neighbor is selected. Then, all k nearest features of the selected cluster are discarded, and process is repeated until all features have been selected or discarded. This method is very sensitive to the parameter k and adjusting this value is not a simple task. Please refer to [68] for a detailed explanation.

SVD-entropy based method

One very interesting idea is shown in [95]. In this work the authors propose a new unsupervised criterion based on the SVD-entropy (Singular Value Decomposition). This SVD-based entropy is different from the one defined based on probabilities. Here the the SVD-entropy is defined based on the distribution of the eigenvalues.

SVD-entropy

This entropy varies between 0 and 1, meaning that the distribution of the eigenvalues varies from very non-uniform to uniform. The contribution CE_i of the i^{th} feature of input data set X, in the value of the SVD-entropy H_{SVD} is defined as

$$CE_i = H_{SVD}(X) - H_{SVD}\left(X_{\setminus i}\right) , \tag{3.2}$$

using the leave-one-out comparison method, where the i^{th} feature is removed from the second term of Equation 3.2.

Finally, features can be ranked by their CE_i values and selected by different strategies:

- features with CE larger than the mean value plus standard deviation has a high contribution and could be selected or;
- features with CE smaller than the mean value minus standard deviation has a low contribution and could be discarded.

For further information please refer to [95].

An unsupervised dimensionality-reduction technique

collinearity

In [64] a quite simple method is developed. The scalar product of the unit-norm vectors of the features is used in order to check collinearity. Features that are very collinear are discarded. The scalar product $P_{i,j}$ between the normalized vectors associated to features i and j is evaluated and assumes a value in a range between 0 and 1. The closer to unity $P_{i,j}$ is, the higher the collinearity between variables i and j is. The next step is to define a collinearity threshold to be used to determine which features are redundant and should be removed. This process is nonsupervised since there is no need to know about sample category a priori. In [64] this method is used as a first step in order to eliminate redundant features and a second feature selection method is applied to the remaining feature subset. For further details the reader is encouraged to read the refered work.

Iterative feature selection method

iterative method

There are some methods that aim to find the features that best explain the output or that are very correlated to it. In the absence of labels, in other words, in the lack of the target values this kind of correlation cannot be evaluated. However, in [102] the authors try to find the features that best explain the data cluster structure. The idea is very simple: one feature will be considered irrelevant to the data cluster if this cluster is indistinguishable when projected onto this feature.

A measure based on the variance of each cluster is defined as shown in Equation 3.3:

$$Score_l = \frac{1}{k}\sum_{j=1}^{k} Score_{l,j} = \frac{1}{k}\sum_{j=1}^{k}\left(1 - \frac{s_{l,j}^2}{s_l^2}\right), l = 1, ..., n_f \ , \quad (3.3)$$

where k is the number of clusters, $s_{l,j}^2$ is the variance of the cluster j over the dimension l, s_l^2 is the global variance considering all data over the dimension l and n_f is the number of features (dimension size). The $Score_{l,j}$ is the relevance measure of feature l with respect to the cluster j, then, the $Score_l$ will be an average of the relevance measure of each cluster with respect to the feature l. If $Score_l$ is near 1, all the cluster local variances, over feature l, are considerably small when compared with the global variance in this dimension. This method is sensitive to the considered number of clusters and, trying to mitigate it, the authors used an cluster method called *Rival Penalized EM Algorithm for the Gaussian Mixture Clustering* (see [102]) that automatically defines the final number of clusters. The ideas contained in this work inspired the development of the semi-supervised method proposed in Section 4.2.

Multi-cluster based method

preserve geometric structure

The main idea in [18] is to find a feature subset able to preserve the data intrinsic geometrical structure, found in the original space. In order to do that, the method developed here is based on spectral clustering techniques. Basically, the first step of the algorithm is to compute the graph Laplacian L, from a graph built using the data set X. With L the following generalized eigen-problem can be solved:

$$L\vartheta = \Lambda D\vartheta \ , \quad (3.4)$$

where D is a diagonal Matrix, the Λ is the eigenvalues and ϑ is the eigenvectors of the generalized eigen-problem. Each row of ϑ is the "flat" embedding for each data point, or, in other words, the "unfolded" data. Selecting the K eigenvectors with the largest eigenvalues, a relevant feature subset can be found solving the L1-regularized regression problem (LASSO) stated in equation 3.5:

$$\begin{aligned} &\min_{a_k} \| \vartheta_k - X^T a_k \|^2 \quad (3.5)\\ &\text{s.t.} \quad | a_k | \leqslant \gamma \ , \end{aligned}$$

where a_k have the combination coefficients for different features in approximating ϑ_k, and γ is a parameter set using the *Least Angel*

Regression (LAR)[31] that specifies the cardinality of a_k. In other words, LAR defines the number of non-zero entries, which is much more interesting for feature selection according to 3.5. Therefore features can be ranked by a *score* defined as

$$\text{Score}_j = \max_k | a_{k,j} | , \tag{3.6}$$

in order to select the top ones.

Clustering ensembles method

clustering ensembles

The process of selecting features without knowing the labels, can be viewed as a search for the minimal feature set that, when submitted to any clustering method, has the ability to reproduce, approximately, the same cluster structure, if this same clustering method is applied to the data set considering all features. This is the main idea in [53]. In this work the *clustering ensembles method* is used. The authors define how to compare clusters based on the instances shared and not by each cluster, and define a relevance measure to select the best candidate feature subset, based on the clusters similarity. The search for the optimal feature subset is performed using a genetic algorithm.

• • •

Just in order to cite a few more methods, there are many other unsupervised feature selection methods like in [97], [13] and [51]. In the first one a forward scheme is used to select the features based on the value of the Residual Mutual Information calculated by a fuzzy algorithm. The main idea in the second method[13] is to find a feature subset whose PCA projection is very close to the PCA projection considering all features. In [51] the Laplacian score is used in a filter approach and examines the intrinsic properties of the data to evaluate the features prior. A feature is *good* if data points that are close enough to each other in the original space, remain close in the reduced subspace.

3.3 SEMI-SUPERVISED FEATURE SELECTION

Despite the large number of methods related to the supervised and unsupervised feature selection subjects, there are too few works dealing with the challenge of using the unlabeled data, in addition to the labeled one, to help with the Feature Selection task. Some of them are wrapper-types, i.e. they access the model accuracy in order to evaluate the quality of the feature subset and others are filter-type, which in turn are more likely a pre-processing step, selecting features regardless

the model choice. Each one of the following methods methods can be classified in a semi-supervised feature selection scheme with the supervised or unsupervised perspective as described in Section 2 and schematized in Figures 2.3 and 2.4 respectively, however this is not evident so far.

Forward Semi-Supervised method

Following the wrapper strategy the Forward Semi-Supervised Feature Selection method [77] selects features accessing the accuracy of a given model. It fits the general scheme for semi-supervised feature selection in a supervised perspectivepointed in Section 2. Like in other approaches, this method is based on the assumption that both labeled and unlabeled data are drawn from the same distribution (they are i.i.d) and in the principle of margin maximization. In this work the idea is to add one feature per time in a *Supervised Sequential Forward Feature Selection* scheme. In this category of wrapper methods at each iteration, the not yet selected features are added, one by one, to the set of selected features Γ, and the accuracy of the model, trained considering only this new feature subset is then evaluated. The feature that better improves the model accuracy, is then permanently added to the set of selected features Γ. This process is repeated until a stop criterion is meet. *FSSFS*

However, at each iteration a model is trained using the labeled data considering only the selected features. Then part of the unlabeled data is randomly choosen, tagged by this model and added to the labeled data to perform the feature selection process. This process is repeated many times in order to avoid a biased distribution problem, or the bias on the sample selection. At the end of these trials, we count the frequency which each feature was selected in the previous process and add, permanently, the most frequent ones to Γ subset. For further information on this method please refer to [77].

The problem here is the computational cost of this method. Even for problems with a small number of features this exhaustive search will be unfeasible as well as the time for training the model several times to avoid the bias on the sample selection and access the model accuracy.

• • •

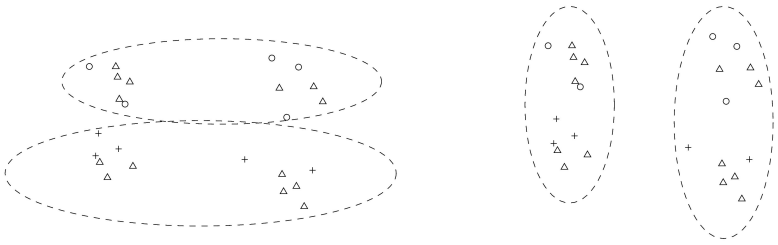
In the "filter type" approach the existent semi-supervised methods are based on the margin maximization principle and make the assumption that all labeled and unlabeled data are i.i.d.. The Semi-supervised Feature Selection via Spectral Analysis [106], the Locality sensitive semi-supervised feature selection method [105], the *FS-manifold* method [99] and the method in [107] apply, somehow, the

spectral graph theory, but they differ in how to apply it. There are also other filter methods, sharing the same basic assumptions, that were developed within the logistic I-RELIEF framework [62] and try to measure each feature discrimination power.

Spectral analysis method

spectral graph theory

Zhao et al. in [106] developed a semi-supervised feature selection algorithm based on spectral graph theory (spectral analysis), that fits the general scheme for semi-supervised feature selection under a unsupervised perspective like shown in Section 2. It assumes that if patterns are in the same cluster, they are considered to belong to the same class. The method searches for clusters, trying to maximize both cluster separability and the consistency with labeled information at the same time. This is done by means of cluster indicators. To better understand this cluster indicator idea, imagine that for two features we construct their two respective clusters indicators in different ways, providing cluster structures as shown in Figure 3.1. If we consider only the unlabeled data (represented by triangles in the figure), both clusters indicators schemes define good separable cluster structures, but, if we consider also the labeled information (represented by circles and crosses), the cluster indicator g_i that generated the cluster structure shown in Figure 3.1a will be more consistent, because all labeled data inside the clusters are from the same class. Considering it, probably, the feature f_i that corresponds to the cluster indicator g_i is more relevant to the output than the feature f_j related to the cluster indicator of Figure 3.1b.



(a) Scheme A (b) Scheme B

Figure 3.1: Result of Cluster indicator

cluster indicator

The proposition is to transform the features into the cluster indicators of the normalized min-cut method [82]. It is also shown in [82] that, a vector g, is a cluster indicator only if it is orthogonal to the elements dm of the degree matrix defined when applying the Graph Laplacian theory. So, if a feature vector f_j is orthogonal to dm, then it is a cluster indicator and this vector can be evaluated by the sep-

arability that it implies to the clusters, and by its consistency with the labeled data. However, as, generally, the feature vectors in the data set are not orthogonal to dm, then the method transforms them into orthogonal vectors to dm. This transformation is done using the FC-transformation equation. Refer to [106] for details.

Once feature f_i is transformed into a cluster indicator g_i with the labeled and unlabeled data set, we can evaluate how well the clusters formed by g_i are separated and whether or not they are consistent with the labels. This is done by means of an equation whose first term gives a measure of the separability of the clusters formed by g_i and the second term measures the consistency with class labels. At the end, the features will be ranked according to this degree of quality of the clusters and all features with this quantity larger than a threshold are selected to compose the final feature subset Γ. For further information see [106].

Locality Sensitive method

A similar approach, applying the spectral graph theory is the Locality Sensitive Semi-supervised Feature Selection method (LSDF) [105] . It can be classified as a filter method because it evaluates a relevance degree for each feature, despite the model that would be used for the learning task. It is an extension of the Locality Sensitive Discriminant Analysis method [17] to the semi-supervised case. This method follows the semi-supervised feature selection scheme as described in Section 2, however it is not evident in which perspective it is designed.

The principal aim is to discover the intrinsic structure of data, considering all features, and try to rank these features according to their contribution to keep the structure found unchanged in the low dimensional manifold. The labeled data is used to maximize the margin between the data from different classes and the unlabeled data is used to discover the geometrical structure of data space. The idea is quite simple and is based on the margin maximization principle, but it is a univariate method and it is one of its drawbacks. *LSDF*

Let us consider that two instances x_i and x_j from the data set X are close enough to each other, so we could expect that the values of the features for these two observations are also close to each other, i.e. $|f_{ki} - f_{kj}|$, with $1 < k < n_f$, is small, where f_{ki} and f_{kj} are respectively the values of feature k for instances x_i and x_j. If we are dealing with a supervised approach, by the margin maximization point of view, the feature k will be considered a "good" feature if $|f_{ki} - f_{kj}|$ is small for instances belonging to the same class and large when considering instances from different classes. Based on the Laplacian criterion proposed to score features [51] the LSDF method will consider the unlabeled data to infer about the geometrical structure and improve the feature ranking process made by supervised methods. Thus, a

feature will receive a high score if $|f_{ki} - f_{kj}|$ is small for instances belonging to the same class or that are close enough to each other, and if $|f_{ki} - f_{kj}|$ is large for instances belonging to different classes.

Shortly, LSDF generates two graphs: one is called *within-class graph* G_w and the other is the *between-class graph* G_b. The first one connects all observations that has the same label or that are close enough to each other. The second graph connects all observations that are close but doesn't have the same labels. The most important features are ranked according to their scores in such way that these two graphs structures can be best preserved in the reduced dimension, i.e., close data points on its original input space, remain near to each other on the final space, and, in the same way, far data points remain far from each other. The score of each feature r can be calculated as shown in equation 3.7,

$$L_r = \frac{\mathbf{f}_j^T L_b \mathbf{f}_j}{\mathbf{f}_j^T L_w \mathbf{f}_j} \tag{3.7}$$

where $\mathbf{f}_j$ is the feature vector as defined in Section 2, and L_b and L_w are respectively the Graph Laplacians of G_b and G_w. Please see [105] for more details.

FS-manifold method

FS-manifold

The so-called **FS-manifold** method [99] selects an optimal feature subset maximizing the classification margin between different classes, while exploiting the geometry of the probability distribution that generated data (labeled and unlabeled). This is done in a framework of manifold regularization, where a semi-supervised Support Vector Machine [10] can be obtained adding a manifold regularization term $\|R\|^2$ in a linear SVM formulation [93, 49] as shown in equation 3.8:

$$\begin{aligned} &\min_{w,b,\xi} \ \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n}\xi_i + \frac{\rho}{2}\|R\|^2 \\ &\text{s.t. } y_i\left(w^T x_i - b\right) \geqslant 1 - \xi, i = 1, ..., n, \\ &\quad \xi_i \geqslant 0, i = 1, ..., n\ , \end{aligned} \tag{3.8}$$

where ξ represent the slack variables, w is the weight vector, b is the bias, x_i is the i^{th} input data instance, n is the number of instances, C is a SVM parameter and ρ is used to balance the importance between the two regularization terms, restricted to $\rho \geqslant 0$. Once again, this method is based on the assumption that nearest instances share the same class labels and in the large margin principle.

When minimizing the regularization term $\|R\|^2$, by constitution, we are emphasizing that nearest instances have to share the same label,

i.e., this regularization smoothes the decision function with respect to the data distribution. This can be seen in the regularization term equation in 3.9:

$$\|\mathbf{R}\|^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\left(f(x_i) - f(x_j)\right)^2 W_{ij} = \mathbf{R}^T \mathscr{L} \mathbf{R}\,, \tag{3.9}$$

where W_{ij} are the weights for the edges between a pair of nodes (x_i, x_j) of the adjacency graph, $\mathbf{R} = [f(x_1), ..., f(x_n)]$ is the decision function values over all data examples, and $\mathscr{L}$ is the graph Laplacian. If one instance x_j belongs to the neighbors of an instance x_i, the weight value for the edge between these two instances will be large and, in order to minimize this regularization term, the algorithm has to find one solution which minimizes the difference $f(x_i) - f(x_j)$ between the function values for the considered data points.

Considering the decision function for a linear SVM, represented as $f(x_i) = w^T x_i - b$ the manifold regularization term is equal to $w^T X^T \mathscr{L} X w$, which does not depend on the bias b value.

To perform the feature selection, the regularization term is modified adding a diagonal Matrix $P = diag(p_1, ...p_{n_f})$, with $p_i \in \{0, 1\}$, $i = 1, ..., n_f$ where n_f is the number of features. This matrix will represent each selected feature. Then, the manifold regularization term will be $w^T P X^T \mathscr{L} X P w$.

However, solving the minimization problem, considering this manifold regularization term is very hard, so the constraint that p_i has to be binary is relaxed to the continuous case. At the end, the selected features will be those with p_i value higher than a given threshold. More details about the optimization process developed to solve this problem can be found in [99]. This method also follows the semi-supervised feature selection scheme as described in Section 2, however it is not possible to classify is it has a supervised or unsupervised perspective.

Semi-supervised I-RELIEF

I-RELIEF, from Section 3.1 is designed within a semi-supervised feature selection scheme under a unsupervised perspective as described in Section 2 (see Figure 2.4), and only works with labeled instances. If there is no label, the distances cannot be defined [86]. Meantime, the margin of an unlabeled sample will have the same absolute value regardless the class that it belongs to. Therefore, the use of an symmetric cost function to evaluate the margin of an unlabeled sample is the solution proposed in [62] in order to use the unlabeled data, leading to the optimization problem shown by Equation 3.10:

semi-supervised I-RELIEF

$$\min_{w} \|w\|_1 + \alpha \sum_{l=1}^{n\ell} \log\left(1 + \exp\left(-w^T \bar{z}_l\right)\right) + \beta \sum_{u=1}^{nu} \exp\left(-\frac{\left(w^T \bar{z}_u\right)^2}{\rho}\right), \tag{3.10}$$

subject to $w_j \geqslant 0$ and $1 \leqslant j \leqslant J$, where α and β controls the contribution of each labeled and unlabeled data sets, w is the weight vector, nl is the number of labeled instances, nu is the number of unlabeled instances, z is the difference between the distance among the instance (labeled or not) to the near and miss neighbors, and ρ is the kernel width controlling the shape of the cost function. As in [62], by making $w_j = v_j^2$ the equation 3.10 can be transformed into a unconstrained optimization problem as shown in equation 3.11 which can be solved using a gradient-descent based method:

$$\min_{v} \|v\|_2^2 + \alpha \sum_{l=1}^{n\ell} \log\left(1 + \exp\left(-\sum_j v_j^2 \bar{z}_l(j)\right)\right) + \beta \sum_{u=n\ell+1}^{n} \exp\left(-\frac{\left(\sum_j v_j^2 \bar{z}_u(j)\right)^2}{\rho}\right). \tag{3.11}$$

In other words, for the semi-supervised case, the sign of the margin is not relevant. The distance will be always the same, and with the probability of an unlabeled data belonging to the neighbors of x_i of the same class or not, it is able to evaluate this distance for the unlabeled data, and then, determine its margin.

Label propagation based method

Another interesting approach was developed by Zhong et. al. in [107], and the main idea is to iteratively "propagate" the labels from the labeled dataset to the unlabeled one and perform feature selection using the "new" labeled data. First they perform any Supervised Feature Selection on the labeled data $X^{(\ell)}$, selecting a initial feature subset Γ. After that a graph G is built using both labeled $X^{(\ell)}$ and unlabeled $X^{(u)}$ datasets, considering only the initial selected features, where each node corresponds to the instances of $X^{(\ell)} \cup X^{(u)}$, and the edges are weighted by

$$W_{ij} = \exp\left(-\frac{\| x_i - x_j \|^2}{\lambda^2}\right), \tag{3.12}$$

where λ is a bandwidth hyper-parameter. Then, the labels are propagated using the "Label Propagation" algorithm (see [108]).

label propagation

Then, based on a confidence measure, the top unlabeled data, that were tagged by the propagation method, can be selected, i.e., the tagged unlabeled data with largest confidence, given by expression 3.13, are selected to be added to the labeled dataset with their propagated labels, forming a new labeled dataset $X_{new}^{(\ell)}$. Over this new dataset, they perform again a supervised feature selection algorithm, construct a new graph and all the procedure is repeated for a certain number of iterations. The confidence value is given by

$$conf(y_i \mid x_i) = \frac{|\, f(i) - 0.5 \,|}{0.5}\,, \tag{3.13}$$

where f is an hamonic function (as defined in [108]) and can be viewed as the probability of hitting a labeled node with label 1, while $1 - f$ can be thought as the probability of hitting a node with label equal to 0.

At the end, the feature subset with highest average prediction confidence is then selected.

One interesting detail about the confidence in Equation 3.13 is that it has a direct relation with the normalized data margin, i.e., an increase of the confidence means an increase of the margin and vice-versa. This method is based on the general semi-supervised feature selection scheme with a supervised perspective as pointed in Section 2.

Mutual Information based method

The supervised method developed in [61] and listed in Section 3.1 use mutual information as a similarity measure. All steps in [61] were performed only considering labeled data, however, unlabeled data could be used in the clustering step, where labels are not needed at all. It was done in [73] where a semi-supervised feature selection method within the feature clustering approach was developed. Here the method cluster features using a semi-supervised metric based on the mutual information. It uses the Ward method [98] to cluster features according to the similarity measure between features X_i and X_j, described in equation 3.14:

mutual information based methods

$$\begin{aligned} S(X_i, X_j) &= \alpha_1 \left(H(X_i/X_j) + H(X_j/X_i) \right) \\ &+ \alpha_2 \left(MI(X_i, Y/X_j) + MI(X_j, Y/X_i) \right)\,, \end{aligned} \tag{3.14}$$

where H is the entropy, MI is the Mutual Information, X is the input data and Y is the output variable.

The first term in equation 3.14 is also known by Mantara's distance. After the end of the clustering process, the representative feature of

each cluster will be the one with the larger Mutual Information with the output Y. This method is designed in a semi-supervised feature selection scheme with a unsupervised perspective.

S^3VM

The Semi-Supervised version of the Support Vector Machines (S^3VM) [100], was used also to perform Feature Selection and was designed in a general semi-supervised scheme, however it is not possible to identify its perspective (refer to Section 2 for details). The idea is to use the S^3VM with 1-norm formulation which produces a weight matrix with a lot of zeros. The features whose weights goes to zero are not useful and could be discarded. One of the drawbacks of this method is that it doesn't work very well for bigger datasets, and there are parameters to be set. For further information please refer to [100].

SVM

• • •

Up to this moment the lack of material in the literature about the semi-supervised feature selection topic is an important constraint, but, at the same time, is a very big opportunity to develop new theories and methods. This is an not well explored field and offers a real challenge to all researchers who wish to launch themselves into this endeavor.

3.3.1 *Discussion about semi-supervised methods*

As far as we know, there is not any general algorithm capable of solving any Feature Selection problem for any database, whatever is the applied approach (unsupervised, semi-supervised or even supervised). Each method has advantages, drawbacks and specific technical characteristics that have to be taken into account when applying it to the problems. Depending on the own problem characteristics one method will be more interesting rather than others. For example, it will depend on the type of available data, the quantity of data (labeled and unlabeled), the proportions between each type of data, the nature of these data, the way how it is measured, the type of study, the objectives, and the available resources like time, hardware, etc.

Frequently, when developing a method, toy problems or real databases, whose desired solution is already known, are used in the tests, in order to see and prove their efficacy or not. It is easy to be led to think that one method has to be able to solve any problem and to be better than the others. However, there will be always some kind of data or problem that one method could not solve. So, when applying these methods to any real problem, for which the optimal solution is not known, there is no simple direct way to verify if the algorithm output is optimal or not. Actually, a good conduct is to apply different

methods to the problem and compare their results, always keeping in mind their main characteristics.

Two very important aspects when comparing any algorithm are the way how these methods are implemented and the computational cost associated to each one. Hereafter we do a comparison of the algorithms addressed in section 3.3, remarking some practical aspects concerning their implementation and computational complexity.

The LSDF method [105] is a univariate filtering method, meaning that features are ranked regardless of their behavior when taking into account the iteration with other features. It is very easy to implement it and ***not too much*** costly in terms of computational complexity. The cost to compute both the within and between-class graphs are about $O\left(n_f n^2\right)$, where n_f is the number of features and n is the number of instances (data points). So, the total cost is also linear with the number of features, and quadratic with the number of instances $O\left(n_f n^2\right)$. In the same way, the univariate filter *sSelect* method [106] has a similar computational complexity of $O\left(n_f n^2\right)$ and it is also very simple and fast to implement. The *Logistic Iterative Relief* [62] filter method, on the other hand, is not so easy to implement but its computational complexity is not so bigger when compared to LSDF or sSelect methods. Its computational complexity is $O\left(T n_f n^2\right)$, where T is the considered number of iterations to achieve the maximum margin. The *FS-manifold* method [99] by the way, is a multivariate filter method that is not so easy to be implemented and the computational complexity is the same of a Support Vector Machine [93] adding the cost of solving the minimization problem. The other filter method discussed in the previous section is the *IteraGraphFS* [107]. Despite its easiness of implementation, it has a high computational complexity: the initial supervised feature selection step consumes $O\left(n_f n^2\right)$, building the graph takes other $O\left(n_f n^2\right)$. The label propagation step is of the order of $O\left(n^3\right)$, the confidence evaluation is $O(n)$, the addition of the top confident tagged unlabeled data to the labeled data consumes near $O(n_f \log n_f)$, and, to select new features using the supervised schema costs $O\left(n_f n^2\right)$. Finally, the total computational complexity of this method is $O\left(T\left(n_f n^2 + n^3\right)\right)$.

computational costs

The filter method developed in [73] is quite easy to implement and not too costly when compared with the others: it is $O\left((n_f - 1)n^2\right)$ in addition to the cost of the entropy and the mutual information estimator considered, that could be high.

On the other hand, the wrapper method called *Forward Semi- Supervised Feature Selection* [77], is quite simple to implement but it has an extremely high computational cost. Even for datasets not too large this method could be unfeasible. However it can achieve, when applicable, the best results. Its computational complexity is of the order $O\left(T n_f n^2\right)$ plus the cost of the considered classifier (CL) at

each iteration multiplied by R times to get statistical precision, i.e. O(T O(CL) R).

3.4 MUTUAL INFORMATION AND ITS ESTIMATION

In the literature there is some feature selection methods were developed applying information theory. It uses the mutual information as a similarity measure to eliminate redundant features or as a relevance measure in order to eliminate irrelevant ones. A feature selection method is developed on this Thesis using a similarity measure also based on the mutual information as detailed in Chapter 4. Depending on the nature of data, mutual information is evaluated using estimators for the discrete or continuous case. Nevertheless real problems are composed by a mixed set of continuous and discrete variables, and no specific estimator is developed for this case in the literature as far as we know. Usually people do not care about this question which can lead to some inaccuracies that can be an issue depending on the problem. In this section we recall basic definitions of entropy and mutual information, that are important to the development of our feature selection method and in order to be able to develop an appropriate mutual information estimator for a set of mixed types of variables in Chapter 6.

3.4.1 *Mutual Information definition*

The first attempt to establish a measure of information is given in the 20s through the work of Nyquist, Hartley and Fisher, but it was only in 1948 that Shannon [81] established the most important concepts and fundamentals that led to the current theory information. Information theory has important contributions in many fields as in communication theory determining the maximum data compression (entropy) and the maximum transmission rate of communication (channel capacity) [24], as well as in computer science (Kolmogorov complexity) [63], in statistical physics (thermodynamics) [33], in statistical inference (Occam's Razor) [12], and in probability and statistics. Concerning data compression and transmission, quantities as entropy and mutual information are probability distribution functions that support the communication process.

discrete entropy

The entropy of a random variable is a lower bound on the average number of bits to represent this random variable [24]. As an example, consider one discrete random variable whose distribution is uniform over 64 different outcomes. Each outcome can be identified by a label,

so six bit strings are enough to represent them. The entropy of a *discrete* random variable X with probability mass function $p(x)$ is defined as

$$H(X) = -\sum_x p(x) \log_2 p(x). \tag{3.15}$$

The entropy defined using logarithms to base 2 measures the average uncertainty in the random variable in bits. In other words, it will be the number of bits, on average, to describe the random variable (to label each possible outcome). Note that entropy only depends on the probabilities and not on the actual values taken by the random variable , therefore it is a functional of the distribution of X. Unless otherwise specified, all logarithms on this thesis will be taken to base 2, so the indication 2 will be omitted in notation. Applying the entropy definition above to the last example we have

entropy is a functional of probabilities

$$H(X) = -\sum_i^{64} p(i) \log p(i) = -\sum_i^{64} \frac{1}{64} \log \frac{1}{64} = 6 \text{ bits}.$$

The entropy is a non-negative quantity ($H(X) \geqslant 0$) once probabilities are defined between 0 and 1.

Likewise the entropy for a discrete random variable was defined, the joint entropy of two discrete random variables X and Y with joint distribution $p(x, y)$ can be defined as

joint discrete entropy

$$H(X, Y) = -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(x, y).$$

It is also possible to define the conditional entropy that is the entropy of a random variable conditional on the knowledge of a second random variable as

conditional discrete entropy

$$\begin{aligned} H(X, Y) &= \sum_{x \in X} p(x) H(Y|X = x) \\ &= -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(y|x) \end{aligned}$$

The joint and conditional entropies relate by the *Chain rule* given by

$$H(X, Y) = H(X) + H(Y|X)$$

as can be seen in [24].

If there is some dependence between these two discrete random variables, the uncertainty of one variable will be reduced by the knowledge of the other. This reduction in uncertainty is called the *mutual*

discrete mutual information

information and it is the measure of the dependence among two random variables. The mutual information between two discrete random variables X and Y with joint distribution $p(x, y)$, and marginal probability mass functions $p(x)$ and $p(y)$, is given by

$$MI(X, Y) = H(X) - H(X|Y) = \sum_{x,y} p(x, y) \log \frac{p(x, y)}{p(x)\, p(y)}. \quad (3.16)$$

Mutual information has some interesting properties:

- it is symmetric in X and Y;
- it is non-negative;
- if the two variables are completely independent this quantity will be zero in theory.

differential entropy

The concept of entropy for the continuous case receives the name of differential entropy . Let now X be a continuous random variable with probability density function $f(x)$, therefore, the differential entropy h of X is defined as

$$h(X) = -\int_S f(x) \log f(x)\, dx, \quad (3.17)$$

where S is the support set of X where $f(x) > 0$. As in the discrete case, the differential entropy only depends on the probability density of the continuous random variable.

joint differential entropy

Extending the definition of differential entropy of a single continuous random variable to a set of two continuous random variables X and Y with joint density function $f(x, y)$, we have the definition of the joint differential entropy as

$$h(X, Y) = -\int f(x, y) \log f(x, y)\, dx\, dy.$$

conditional differential entropy

As X and Y have joint density function $f(x, y)$, the conditional differential entropy can be defined as

$$h(X|Y) = -\int f(x, y) \log f(x|y)\, dx\, dy,$$

differential mutual information

and, finally, the mutual information between these two continuous random variables is defined as

$$\begin{aligned} MI(X, Y) &= \int f(x, y) \log \frac{f(x, y)}{f(x)\, f(y)} dx\, dy \\ &= h(X) - h(X|Y) \end{aligned}$$

3.4.2 *Estimation*

In order to evaluate the mutual information, densities of dependent and independent variables should be estimated. In practice, evaluating mutual information is not straightforward, since it requires *a priori* knowledge of the corresponding densities. Usually, information about densities is not fully available and a proper density estimator is needed. There are, in the literature, specific estimators to be applied over datasets composed by discrete variables and over datasets composed by continuous variables.

Discrete setting

As the entropy and consequently the mutual information are clearly functions of the data distribution and joint distribution respectively, they can be estimated by estimating these distributions by its classic maximum likelihood estimator [67] [55], like histograms.

discrete estimators

Continuous setting

When the dataset is composed by continuous variables there are several approaches that could be applied. In the following we just list some of them giving an simple overview just in order to show their principles. For further details please refer to their respective citations.

continuous estimators

- The Shannon entropy can be estimated (and consequently the mutual information), for instance, by substituting the density of the random variable by an estimate evaluated from available independent realizations, by means of *kernel density estimation* [80]. However, the evaluation of the integral requires numerical integration and it is not a simple task when f is a kernel density estimator.
- Ahmad et. al. [4] proposed to estimate the entropy by

$$h(X) = -\frac{1}{n}\sum_{i=1}^{n} \ln f(X_i)$$

  where n is the number of elements of X and f is a kernel density estimate or even a histogram density estimator [48].
- Friedman et. al. proposed the projection pursuit density estimation method, where it extends the classical univariate density estimation methods to higher dimensions in such way that involves only univariate estimation [37].
- Another plug-in estimator is built splitting data estimate. The samples are decomposed into two sub-samples ($X_a = \{X_1, ...X_m\}$

and $X_b = \{X_{m+1}, ..., X_n\}$). X_a is used to construct a density estimate f_a and then, f_a and X_b are used to estimate the entropy. f_a can be evaluated by kernel density estimate [46] or even by histogram density estimate [47].

- Entropy/mutual information also can be estimated using the cross-validation estimate. This method is also known by leave-one-out density estimate, where f is estimate by kernel density estimator based on X leaving X_i out [7].
- Estimates of entropy based on sample-spacing take into account the spaces between values of X giving a rough idea of the probability density: the closer together the values are, the higher the probability density. This estimate has high variance and have issues when dimension is greater than one [87].
- Estimates also can be based on nearest neighbor distances. In fact the entropy is estimated from the distribution of the nearest-neighbor-distance of data points [59].
- Another way to estimate mutual information is based on a prior discretization of the random vectors by means of recursive partitioning algorithms [27].
- The one proposed by Kraskov et al. [60] is widely employed in the literature. Despite yielding high performances for many problems, even for scarce datasets, the Kraskov estimator is restricted to regression problems and continuous variables.

A mutual information estimator for classification problems was derived from the Kraskov estimator [60] and developed by Goméz et al. [39]. It addresses classification tasks by discretizing the output variable, and can also be applied to multi-class feature selection problems. Nevertheless, like the original Kraskov estimator, this approach is also restricted to continuous input variables. However, most real-domain applications contain not only continuous but also discrete variables, which are usually treated separately in current applications. When dealing with problems composed by a mixed set of variables usualy the continuous features are discretized. However there is a lack in the literature of an appropriated mutual information estimator able to handle both discrete and continuous variables togheter. In Chapter 6 we develop such estimator in order to be used in our feature selection methods.

3.4.3 *Feature selection using mutual information*

Mutual information [24] has been applied to a wide range of machine learning problems over the years [61, 73]. It is well established and

accepted among researchers, especially for estimating uni and multivariate non-linear relations. Roughly speaking, MI can be used to estimate the amount of information shared by two variables or groups of variables, i.e., it is a method to measure their dependencies. MI can also be viewed as the amount of reduction in the uncertainty of one variable due to the knowledge of another one [24], and can be applied to feature selection (Feature Selection (FS)) [8]. In filter methods, for instance, it can be used as a measure of the relevance of a feature or a group of features, avoiding the computational burden of wrapper methods that assess the outcomes of the classification or regression models in order to determine which features are relevant and which are not.

MI for feature selection

3.5 REMARKS

In the literature, there are too few material about semi-supervised feature selection. Some of the methods found, that deal with this kind of problem, are based on graph theory ([106],[105], [99] and [107]). One of them is based on Information theory ([73]). The method proposed in [73] is the only one that cluster features. The other methods cluster instances in order to select features. All methods try to combine somehow the labels and the structure information. We did not found a similarity measure able to take into account both labeled and unlabeled data and this is one of our objectives: to develop such similarity measure and to develop a feature selection method based on it, as proposed in Section 4.1.2. Also no semi-supervised multi-objective method to select features was found in the literature, and, the method developed and proposed in Section 5.4 is a first development to fill this gap.

Part II

PHD PRODUCTION

4

CLUSTERING APPROACH

A common approach to feature selection FS is to estimate the relevance and to rank each feature according to its relation (or correlation) with the output targets [58, 71]. This approach is intuitive and easy to implement but it usually fails to consider the relevance of a given feature in the presence of others[45], since most filter methods are univariate [71, 32]. In this context, mutual information [24] arises as a propper "relation" criterion, since it is a multivariate measure which is widely used to evaluate relations among sets of features and output labels.

Basically, labels and data are the available sources of information to perform FS. Many methods [58, 61] are able to deal only with labeled data while others only deal with unlabeled data [68, 64]. However, in many real situations, the amount of labeled data is not sufficient to well characterize the relations between input data and output classes. Since labeling by human experts can be costly, it is common in many kinds of problems to have large unlabeled data sets available and very few labeled data. Due to the availability of the large unlabeled data set, the question that arises in such a context is

> *why not to use information extracted from the unlabeled data in order to estimate feature relevance and to induce models?*

The joint use of labeled and unlabeled data to perform FS characterizes the semi-supervised feature selection paradigm.

Therefore feature selection can be performed through a feature clustering approach. Real problems could be composed by tens, hundreds or even thousands of features, and of course, many of these features could be redundant or even very similar to each other. Redundant features should be clustered, and one can do that using a clustering method. Whether the variables are relevant or not, if they are similar they could be grouped together, and one of those features can be chosen in order to represent the entire group. In order to identify and group pairs of "similar" features it is necessary to define a similarity measure. The main objective of this measure is to define if two features are similar and therefore discard one of them. Pairs of features with high values of similarity have to be "grouped" and those with low values might not be clustered. To evaluate the similarity, or distance, among features it is not necessary to know labels, once only the features themselves and a criterion are needed. However it would be very interesting to use the label information into the similarity criterion, when available.

In that sense, an index is proposed in this work in such way that, not only the "direct similarity" between the variables are evaluated, but also the similarity concerning the relation between each variable and the output labels. We define the term "direct similarity" as any kind of measure that aims to quantify how much information, for instance, one feature carries about others. Based on this measure a semi-supervised feature selection method is developed.

Also the principle of homogeneity between labels and data clusters is exploited in order to develop another semi-supervised feature selection method. This principle permits the use of data cluster information to improve the estimation of feature relevance in order to increase selection performance. Mutual Information is used in a Forward-Backward search process in order to evaluate the relevance of each feature to the data distribution and the existent labels, in a context of few labeled and many unlabeled instances. For both proposed methods tests are performed in order to evaluate their efficacy and results are discussed.

4.1 CLUSTERING AND SELECTING FEATURES

4.1.1 *Similarity criterion S*

The proposed index (S) is developed in order to consider both labeled and unlabeled data on its calculation (semi-supervised paradigm). Not only the "distance" between each feature needs to be small to consider both features as redundant, but also the individual influence of each one of these features over the output labels have to be similar. Therefore the general semi-supervised similarity criterion S between two random variables X_1 and X_2 can be defined as

$$S(X_1, X_2) = A(X_1, X_2) - \lambda B(X_1, X_2) , \tag{4.1}$$

$$A(X_1, X_2) = \Phi(X_1, X_2) \; \forall \; X^{(\ell)} \cup X^{(u)} , \tag{4.2}$$

$$B(X_1, X_2) = abs(\Phi(X_1, Y) - \Phi(X_2, Y)) \; \forall \; X^{(\ell)} , \tag{4.3}$$

where $X^{(\ell)}$ is the labeled data set, $X^{(u)}$ is the unlabeled data set, Φ is any correlation or similarity measure, Y is the output label vector (for $X^{(\ell)}$ only), and λ is a parameter to balance the influence of each *supervised* and *unsupervised* terms on the index. As only labeled data is considered in the calculation of term B this is considered as the supervised term, likewise, as labels are not necessary in the calculation of term A this is considered as the unsupervised term.

In this work the *mutual information* (MI) is chosen as Φ measure function, as it is very well accepted in the academic fields and once it is able to identify non-linear dependency between random variables. Of course any other *similarity* function can be used like the Pearson correlation or even the Relief Index for instance.

S is not a distance in a formal point of view despite being symetric. As defined in 4.3 this index fails in at least one necessary condition to be considered as a distance: it is not positively defined, i.e., the condition $S \geqslant 0$ cannot be sustained. Nevertheless the S index can be used as a similarity measure, even with the difficulty to set the λ value, as discussed in section 4.1.1.1. Once we are interested in ranking the pairs of features concerning their similarity degrees, since the similarity value is significative (see section 4.1.1.2), even if it is a negative number, we can rank the pairs of features.

Analyzing qualitatively and quantitatively this similarity definition it is possible to show its relevance to the task of identifying redundant features, keeping in mind the final goal: to perform feature selection. The first term in S is the MI between pairs of features and gives an idea of how much information one feature keeps from the other. In the extreme case, where two features are identical, this quantity is maximum. In other words, if two features are very similar, A will have a high value, increasing the similarity index. On the other hand, if two features are completely random to each other then, in theory, the mutual information has to be zero, and there is no similarity at all.

The second term adds to the concept of similarity the relation of each feature (from the pair) to the output variable. The similarity will be decreased if each feature from the pair being analyzed has, individually, different influence over output labels. If two features have similar relation with the output variable, then the B term will be very low and S will not be too much decreased. In the same way, if one of the two variables has nothing to do with the labels and the other is very correlated, then the difference in the B term will be high and S will be decreased. Note that the main objective of B term is to decrease the similarity between two features if they have different relevance levels to the output variables,

The A term takes into account all labeled and unlabeled data once the labels are not necessary, while the B term could only use the labeled ones. However, in Section 4.2 a strategy to consider unlabeled data in this term is developed.

In cases where there are too many features with different levels of relevance, identifying pairs of variables whose similarity represents also their importance to the output is a very interesting tool, especially in a feature selection framework. In these problems, redundant features must be discarded, so, if two variables are too similar, one of them could be eliminated. Nevertheless, features that share a certain amount of information could not necessarily have the same influence over the output, and considering only the mutual information between features as a criterion of similarity, could lead to discard, by chance, a relevant feature. In this case, for example, if we consider the B term, the two features could not be considered as redundant any more.

4.1.1.1 *Setting* λ

The S index can be thought of as the mutual information between two features, penalized by the difference of their relations to the output. Depending on the value of λ the S index could be more or less affected by the "supervised" B term. Obviously, if $\lambda = 0$ then S will be only the mutual information between the pair of features, i.e., the similarity will take into account only the "unsupervised" term, and their similarity will not be decreased by the difference between their relation to the output. For the case of $\lambda = 0$ we have that $S > 0$, at least in theory[1]. The *ideal* λ, if it exists, has to be the one whose value provides the optimum balance between the supervised and unsupervised terms of S.

Parameter λ has large or small influence on S depending on the degree of relevance of each variable in the considered pair of features. Therefore three general situations can be listed:

1. If two features are unrelated to Y, in theory, each part of B, i.e. $MI(X_1, Y)$ and $MI(X_2, Y)$, will be zero and then $S = A$ for any value of λ;
2. If the two features have the same relevance to Y, each part of B is greater than zero but equal leading again to $B = 0$ and $S = A$ for any value of λ;
3. and, if these two features has different relevance values to Y, then $B > 0$ and $S = A - \lambda B$. So, λ has a greater influence in the level of similarity in this case.

For the two first situations listed before λ has no influence over S, however, in the third situation this parameter has an important role as will be discussed next. Let P_1 be a pair of features whose MI is given by A_1 and the difference between their MI with the output is given by B_1. For a given λ we will have the similarity of this pair as $S_1 = A_1 - \lambda B_1$. In the same way let P_2 be another pair of features where $S_2 = A_2 - \lambda B_2$. The value S_1 can be greater or smaller than S_2 according to these following situations:

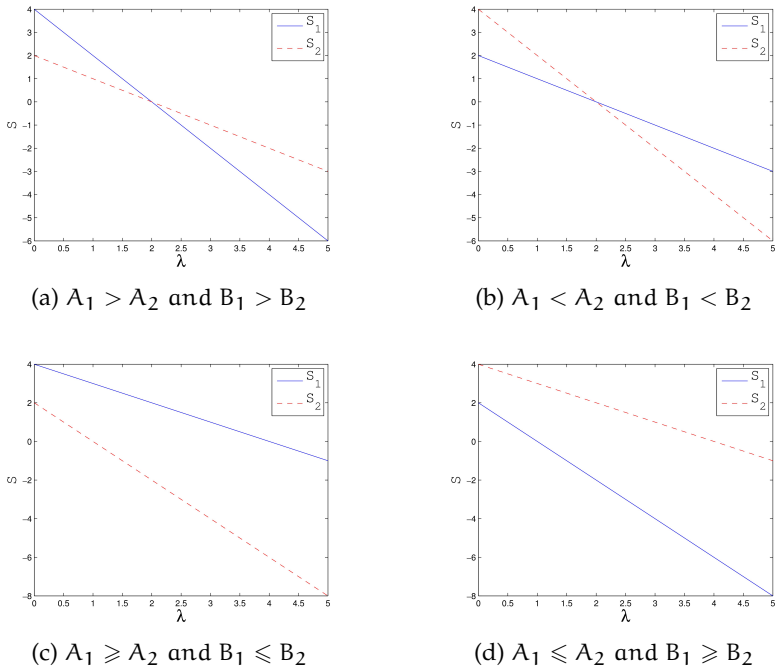
$$\begin{aligned} &\text{If } A_1 > A_2 \text{ and } B_1 > B_2 \quad \Rightarrow \quad \exists\, \lambda_k > 0 \mid \\ &\qquad S_1 > S_2 \;\forall\, \lambda < \lambda_k, \\ &\qquad S_1 < S_2 \;\forall\, \lambda > \lambda_k, \\ &\qquad S_1 = S_2 \text{ for } \lambda = \lambda_k. \end{aligned} \tag{4.4}$$

$$\text{If } A_1 > A_2 \text{ and } B_1 < B_2 \quad \Rightarrow \quad S_1 > S_2 \;\forall\, \lambda > 0. \tag{4.5}$$

$$\text{If } A_1 > A_2 \text{ and } B_1 = B_2 \quad \Rightarrow \quad S_1 > S_2 \;\forall\, \lambda > 0. \tag{4.6}$$

1 As estimators are used to evaluate mutual information, it's common to achieve some negatives values when there is no correlation between the variables

These relations can be graphically demonstrated as shown in Figure 4.1. For the cases described in equation 4.4, as can be seen in Figures 4.1a and 4.1b [2], there is one value of λ where $S_1 = S_2$ from which P_1 and P_2 change their relative positions in the similarity ranking. This happens because λ can assume only positive values. On the other hand, for the situations described by equations 4.5 and 4.6 there is no value of $\lambda > 0$ from which S_1 and S_2 can invert their relative positions in the magnitude scale, as shown by Figures 4.1c and 4.1d.



(a) $A_1 > A_2$ and $B_1 > B_2$

(b) $A_1 < A_2$ and $B_1 < B_2$

(c) $A_1 \geqslant A_2$ and $B_1 \leqslant B_2$

(d) $A_1 \leqslant A_2$ and $B_1 \geqslant B_2$

Figure 4.1: Importance of λ to S index.

One extreme situation occurs in the case of a pair of two identical features with the same relevance to the output. The A term will be maximum (and equal to the entropy of the variable) and B will be equal to zero. In this case S will be *maximum* and equal to A . Here λ makes no difference into the calculation of S. The other extreme is given when two features are completely independent from each other, but one of them is exactly equal to Y and the other is independent to the output. In this case A is equal to zero (in theory) and B will be maximum resulting in a minimum S index whose value is negative, however, variations in λ will change the point of minimum similarity. The question at this moment is how to set lambda in order to have a semi-supervised similarity measure well balanced.

2 Note that the situation drawn in Figure 4.1b is exactly the same situation shown in Figure 4.1a where there is one pair of features with A and B terms smaller than their respective terms in the other pair.

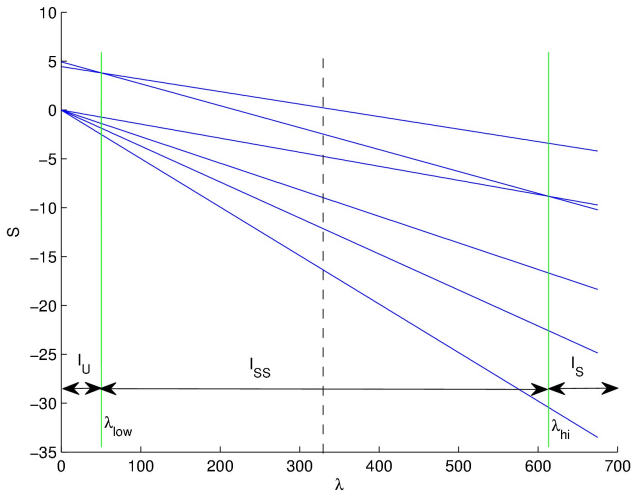


Figure 4.2: λ ranges example. Each curve shows the behavior of the S index in function of λ for each pair of features. I_U is the preponderantly unsupervised interval, while I_S is the mainly supervised interval. I_{SS} is the mainly semi-supervised interval. The first inversion of the S ranking occurs at λ_{low} and the last change occurs at λ_{hi}. The dashed line shows the chosen λ for a more balanced semi-supervised criterion.

Adopted strategy

Based on the behavior of S as a function of λ it is possible to define three main ranges of values that this parameter can assume, as shown in Figure 4.2. There are two ranges were, despite changes in S value, when increasing λ, the similarity ranking of the feature pairs doesn't changes. For the range $0 \leqslant \lambda < \lambda_{low}$ the S value is more related to the unsupervised similarity measure (A term). λ_{low} is the value where occurs the first change in the ranking order of similarities. On the other hand, in the range $\lambda_{hi} < \lambda < +\infty$ there is no more changes in the similarity ranking, and the index is then more affected by the supervised B term. λ_{hi} is the point where the last change in the similarity ranking occurs.

It is clear that for all values of $\lambda > 0$, somehow, both labeled and unlabeled data is taken into account in the similarity measure. Nevertheless, intuitively, a more balanced semi-supervised index can be achieved inside the range $\lambda_{low} < \lambda < \lambda_{hi}$. The idea is to set λ in such way that the S index is neither dominated by the A nor by the B terms.

Thinking S as a similarity measure given by the mutual information between the features, penalized by their relevance to the output variable, and, keeping in mind its objective, which is to perform feature selection in a semi-supervised framework, it is interesting to have a λ that provides a more balanced semi-supervised metrics.

All feature pairs whose B term is greater than zero will have their similarities decreased while λ increases. At certain value λ_{low} some of these similarities will change their relative positions in the magnitude scale and after another point λ_{hi} the similarities will not change positions anymore. In order to have a S index more balanced on its semi-supervised character, the λ must lay somewhere between λ_{low} and λ_{hi}. So, it can be defined as the half range value between these two limits, shown by the dashed curve in Figure 4.2.

The semi-supervised interval I_{ss} can be determined finding the minimum and maximum values of λ where occurs the first and last changes in the S index ranking, as shown in Figure 4.2. The possible intersection for each feature pair combination (P_k) is evaluated and after that the mean value of λ in the interval between the first and the last crossing points is considered as the chosen λ. The formulation is shown in equations 4.7, 4.8 and 4.9.

$$I_{ss} = \{\min(\lambda_{ij}); \max(\lambda_{ij})\} \tag{4.7}$$

where $i = 1, 2, \ldots p$, $j = 1, 2, \ldots p$, $i \neq j$, p is the number of possible pairs P of features, and λ_{ij} is the value where the similarity index for pairs i and j is the same. So,

$$\lambda_{ij} = \frac{A_j - A_i}{B_j - B_i} \tag{4.8}$$

and,

$$\lambda_{mean} = \frac{\min(\lambda_{ij}) + \max(\lambda_{ij})}{2} = \frac{\lambda_{low} + \lambda_{hi}}{2} \tag{4.9}$$

4.1.1.2 *Significance of S*

When evaluating the similarity between two features, using mutual information, estimators have to be used. Its estimates may well approach the correct value of MI or not. Anyway they are still approximations and, for that reason, if two features are completely unrelated their mutual information will not be zero (as in theory) but something that *means* zero. Based on this fact, it is imperative to know if a similarity value is really a meaningful measure or if it is only a number *meaning zero* to the estimator considered in the calculations. In the context of the similarity index proposed here, it is important to know the probability of an observed value of S not be only a "zero" value of the estimator, and this can be done with a hypothesis test. By definition, a hypothesis test evaluates the probability of a given hypothesis to be true, therefore the following hypothesis can be formulated:

> Given two random variables X_1 and X_2, let S be the similarity function between then. Let the null hypothesis be that the variables X_1 and X_2 have no similarity. In this case, we want to evaluate the distribution of S under the null hypothesis of no similarity.

If X_1 and X_2 do not have any information about each other, S is expected to be zero in theory, whatever are the values that these variables can assume. Nevertheless, as we are dealing with mutual information estimators in the calculation of S, this value will not be properly zero, but something near it and even a negative value. This means that the instances in X_1 are random concerning the instances in X_2 and, as they are random, any random changes in the instance values still keep the relation between X_1 and X_2 the same: *unrelated*. For different values of the instances in the variables, the result of the estimator will be different, following some unknown distribution, but still meaning zero mutual information, or, in the case of the similarity index, no similarity.

As both variables X_1 and X_2 are unrelated, permuting randomly the instances in one of them, keeping unchanged the instances on the other, will result in other unrelated pair of variables whose S should be approximately zero. Permuting one of these variables many times it is possible to get an approximation of the S index Cumulative Distribution Function (CDF) for the case where the two features are unrelated. Therefore, we can use this CDF curve to evaluate the probability of a similarity value not be only a zero estimation of a random *permutation* of the vectors instances by chance, but a *meaningful similarity value*.

Coming back to the hypothesis formulated before, one of the features in the pair is permuted n_p times and, for each permutation, the permuted similarity index S_p for features X_1 and X_2 is calculated and stored. With these n_p values a CDF of S_p is built and then a threshold can be set in order to determine if S is significant or not. This threshold can be set, for example, as the 95th percentile (S_p^{95th}) of the CDF of S_p: it means that 95% of the observed S_p values are smaller than this value in the cumulative distribution curve [57]. Therefore, if $S > S_p^{95th}$ then S can be considered as a meaningful similarity value, i.e. the null hypothesis is true. Otherwise, the null hypothesis is not verified and S means no similarity. This kind of test is called the permutation test [35].

4.1.2 *Feature Selection Method*

The semi-supervised feature selection method developed on this work, hereafter called Semi-Supervised Feature selection based on Clustering (SSFC) method, is very simple and can be split into two main steps:

- eliminating redundant features;

- eliminating irrelevant features.

In the first step the method tries to identify any redundant features. Many problems, especially those with a large number of features, could have similar features, regarding their own information. Taking into account the characteristics involved in the definition of S index, the proposed method will look for features sharing a certain amount of information between themselves and having similar *relevance* to the known output labels[3]. Therefore, in this step feature selection is done by eliminating redundant features, reducing the number of variables. Of course, the remaining features form the selected feature subset F_s.

The second step of the method is designed in order to find out and eliminate those features that are irrelevant to the problem. This is a very important step since there may exist redundant variables that mean nothing to the problem. Basically, in this step, the algorithm searches for those features whose mutual information value (R) with the output labels *means* zero. This is done in the same way explained in Section 4.1.1.2 for the S index: applying the permutation test[35] to R (Refer Section 4.1.2.2 for details).

4.1.2.1 *Eliminating redundancies*

Given a data set F, the algorithm will cluster similar features in an hierarchical procedure, like Ward's hierarchical clustering method [98], however, using the S index as a similarity criterion, which is based on mutual information. A hierarchical tree of similar features is built evaluating the similarity between each possible pair of features. The pair of features with the highest S value is "grouped" together forming one feature cluster. Of course there is no sense in averaging features in order to merge the clustered features. If a pair of features is considered similar then it is sufficient to discard one of them. The feature to be discarded is chosen based on the mutual information between them and the output variable Y, according to the following rule:

> *If* $MI(X_i, Y) \geqslant MI(X_j, Y)$ *then* X_j *is discarded, otherwise* X_i *is discarded*.

where X_i and X_j are features from the considered pair of features.

After one of these features is discarded, the process can continue clustering the next pair of features whose S is the biggest one among those remaining features in the data set. This process continues until a stopping criterion is met. At the end the remaining features constitute the initial selected feature subset F_s.

stopping criterion

The permutation test used to evaluate the significance of S index in Section 4.1.1.2 is the chosen stopping criterion to halt the process of eliminating redundant features. If the S value, for the pair of features

3 In Section 4.2 consider the data distribution (unlabeld data) in the calculation of feature relevance.

with largest similarity, is less than or equal to the statistical zero value S_p^{95th} defined by the permutation test, S has no significant meaning and process ends without any further feature elimination.

Summarizing:

> *Stopping criterion rule: If* $S \leqslant S_p^{95th}$ *process of elimination of redundant variables ends.*

The basic steps to eliminate redundant features are presented in the Algorithm 1.

Algorithm 1: Basic algorithm to eliminate redundant features

```
1  repeat
2      for feature i = 1 to n_f do
3          for feature j = 1 to n_f do
4              evaluate S(i, j) using equation 4.3 ;
5          end
6      end
7      select the pair of features (i, j) with the higher S ;
8      if S is significant (S > S_p^95th) then
9          discard the feature, from the selected pair, with smaller MI
           with relation to the output variable ;
10     end
11 until stopping criterion;
```

Note that Step 4 in Algorithm 1 is semi-supervised as it deals with labeled and unlabeled data, while Step 9, as defined so far, is supervised because it uses only labeled data. In Section 4.2 a way to consider unlabeled data in the calculation of the relevance of each feature is developed.

4.1.2.2 *Eliminating irrelevant features*

In this feature clustering approach the objective is to perform feature selection saving the relevant variables, i.e. discarding the redundant and irrelevant ones. A similarity measure is defined and used in previous sections in order to detect and eliminate redundancy, and now, it is necessary to define a relevance criterion in order to identify any irrelevant features among the remaining ones.

The Mutual Information plays, again, a decisive role here. Features sharing more information with the output vector will be saved, while those not related to the output will be discarded. For this criterion only the labeled data is considered so far (Refer Section 4.2 to see how unlabeled data could be considered on this calculation).

Basically, the mutual information between each remaining feature and the output is evaluated, and then the "statistical significance" of each result is investigated, in a similar process as done for the S index: permuting feature elements. Therefore the relevance index R of feature

relevance criterion

X_i can be defined as

$$R_i = MI(X_i, Y). \tag{4.10}$$

where Y is the label vector, $i = 1, 2, ..., \mathcal{N}$ and $\mathcal{N}$ is the number of features in the considered set. If R is being evaluated in the step of *redundancy elimination* $\mathcal{N}$ is equal to the number of features n_f of the original set F, and if R is being evaluated in the *irrelevance elimination* step $\mathcal{N}$ is the number of remaining features n_r in set F_s.

As the R index is evaluated using an estimator of mutual information, once more it is necessary to verify if these results have any meaning or if they are only a representation of zero. Each feature has its elements permuted keeping the output vector unchanged and then, the R index is evaluated again. This process is done n_p times allowing to build a Cumulative Distribution Function of the permuted relevance index (R_p) for each feature, and a threshold is set to determine if R is significant or not. This threshold is set as the 95^{th} percentile (R_p^{95th}) of the CDF of R_p: it means that 95% of the observed R_p values are smaller than this value in the cumulative distribution curve. Therefore, if $R > R_p^{95th}$ then R can be considered as a significant (nonzero) relevance value, and a rule to eliminate irrelevant features can be defined as:

significance of R

If $R_i < R_{ip}^{95th}$ *then* X_i *is discarded*.

The basic steps to eliminate irrelevant features are presented in the Algorithm 2.

Algorithm 2: Basic algorithm to eliminate irrelevant features

```
for feature i = 1 to n_r do
    evaluate R_i using equation 4.10 ;
    for j = 1 to n_p do
        permute elements of X_i ;
        evaluate R_ip ;
    end
    if R is irrelevant (R_i < R_ip^95th) then
        discard feature X_i ;
    end
end
```

Features with $R < R_p^{95th}$ will be discarded once their relevance index are equivalent to zero, meaning that they do not hold any information about Y.

4.1.3 *Experiments*

The method developed in Section 4.1.2 is applied to different real problems and the results are shown in this section. The tests are

conducted in order to evaluate the influence of the λ value in the selection of features and in the accuracy of different models trained with the selected features. The influence of the proportion of unlabeled data is also investigated.

The tests are performed over five different real classification problems, all of them from UCI Machine Learning Repository [1]:

- *SONAR*: the sonar data set is composed by instances of a sonar response. The task is to discriminate between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. This data set is composed of 208 instances with 60 features;
- *PEN*: the Pen-Based Handwritten Digits data set is composed by digit samples from 44 different writers. For this last problem we considered only instances of digits 6 and 9 in the experiments. It has 16 features and 2111 instances;
- *KDD*: data set used in The Third International Knowledge Discovery and Data Mining Tools Competition, in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining. This database contains a standard set of data, which includes a wide variety of intrusions simulated in a military network environment and normal connections. It has 600 patterns, 41 features each.
- *ILPD*: the Indian Liver Patient Dataset. This data set contains 416 liver patient records and 167 non liver patient records. It is composed of 10 features.
- *IONO*: the Johns Hopkins University Ionosphere database has 34 features with 351 patterns of radar returns from the ionosphere. Radar returns are classified into two classes: if there is a good return it shows the evidence of some type of structure in the ionosphere, if not their signals pass through the ionosphere.

For each problem the method is applied considering different values of λ varying from zero to some value sufficiently larger than λ_{hi} (see Section 4.1.1.1 for details),also passing through the intermediate values of λ as defined in 4.1.1.1 that better characterizes the semi-supervised approach. The feature selection process is applyied for different proportions of labeled and unlabeled data: 30%, 60%, 70% and 80% of labeled data. For each λ and for each proportion of labeled data the feature selection method is repeated 30 times. Each time the training and test sets are randomly chosen in order to avoid biased results. At the end of the feature selection task, a LDA model and a MLP network are trained considering only the selected features in a 10 fold cross-validation procedure in order to evaluate the accuracy achieved on each trial and the average results are saved. The MLP

is trained using the Levenberg-Marquardt[66] method. Algorithm 3 summarizes the basic script for tests.

Algorithm 3: Basic algorithm for the experiments of the feature selection method based on the proposed S index.

```
  // Feature Selection
1 for each proportion of p_l labeled data do
2     for each λ_i do
3         for j = 1 to n_d 10 cross validation folds do
4             Select training and test sets ;
              // Redundancy step
5             Cluster similar features as shown in Section 4.1.2.1 ;
              // Irrelevance step
6             Eliminate irrelevant features as shown in 4.1.2.2 ;
              // Accuracy
7             Train LDA model with selected subset Γ_(p_l,λ_i,j) ;
8             Evaluate LDA accuracy ;
9             Train MLP model with selected subset Γ_(p_l,λ_i,j) ;
10            Evaluate MLP accuracy ;
              // Partial results
11            save partial results ;
12        end
13    end
14 end
15 average results ;
```

Results

Table 4.1: Final number of slected features for KDD problem.

p_l	n_s	λ									
		0.1	1	10	50	100	500	1000	2250	5000	10000
30%	$\min n_s$	1	10	10	10	11	11	12	12	12	12
	$\max n_s$	8	12	12	13	13	13	13	13	13	13
	$\overline{n}_s$	6.5	11.6	11.1	11.5	12.2	12.1	12.7	12.8	12.7	12.8
60%	$\min n_s$	11	14	13	15	15	15	14	14	15	15
	$\max n_s$	11	16	16	17	17	17	17	17	16	17
	$\overline{n}_s$	11	15.8	14.3	15.7	16.2	16	15.7	15.7	15.1	15.6
70%	$\min n_s$	12	15	13	13	12	14	14	15	16	16
	$\max n_s$	12	17	17	18	17	17	18	17	17	18
	$\overline{n}_s$	12	16.8	15	16.3	16.3	15.2	15.8	16.4	16.7	16.7
80%	$\min n_s$	12	17	13	16	16	12	16	16	16	17
	$\max n_s$	12	17	16	17	18	18	18	17	17	17
	$\overline{n}_s$	12	17	15.2	16.4	17	15.2	16.9	16.6	16.6	17

Table 4.2: Accuracies achieved by classification models trained considering the final set of selected features for KDD problem.

λ	$Acc \pm \sigma$			
	LDA		MLP	
	p_l = 30%	p_l = 60%	p_l = 30%	p_l = 60%
0.1	0.919 ± 0.032	0.972 ± 0.019	0.946 ± 0.023	0.983 ± 0.016
1	0.998 ± 0.006	0.997 ± 0.007	0.997 ± 0.006	0.998 ± 0.005
10	0.996 ± 0.008	0.982 ± 0.011	0.994 ± 0.013	0.995 ± 0.011
50	0.993 ± 0.009	0.984 ± 0.010	0.997 ± 0.008	0.997 ± 0.006
100	0.995 ± 0.008	0.985 ± 0.010	0.997 ± 0.009	0.996 ± 0.01
500	0.993 ± 0.007	0.993 ± 0.008	0.995 ± 0.01	0.995 ± 0.012
1000	0.998 ± 0.006	0.997 ± 0.007	0.997 ± 0.007	0.992 ± 0.023
2250	0.995 ± 0.011	0.997 ± 0.007	0.996 ± 0.009	0.997 ± 0.007
5000	0.998 ± 0.005	0.997 ± 0.007	0.997 ± 0.007	0.998 ± 0.005
10000	0.998 ± 0.006	0.998 ± 0.006	0.995 ± 0.01	0.999 ± 0.003

Table 4.3: Accuracies achieved by classification models trained considering the final set of selected features for KDD problem.

λ	$Acc \pm \sigma$			
	LDA		MLP	
	p_l = 70%	p_l = 80%	p_l = 70%	p_l = 80%
0.1	0.983 ± 0.016	0.983 ± 0.016	0.981 ± 0.026	0.981 ± 0.014
1	0.998 ± 0.005	0.997 ± 0.007	0.994 ± 0.012	0.996 ± 0.009
10	0.974 ± 0.013	0.980 ± 0.013	0.992 ± 0.012	0.993 ± 0.009
50	0.993 ± 0.008	0.981 ± 0.012	0.994 ± 0.013	0.996 ± 0.008
100	0.991 ± 0.009	0.988 ± 0.010	0.995 ± 0.01	0.995 ± 0.01
500	0.988 ± 0.011	0.993 ± 0.010	0.995 ± 0.01	0.994 ± 0.008
1000	0.997 ± 0.007	0.995 ± 0.008	0.998 ± 0.006	0.998 ± 0.004
2250	0.995 ± 0.008	0.995 ± 0.008	0.999 ± 0.004	0.997 ± 0.008
5000	0.995 ± 0.008	0.995 ± 0.008	0.997 ± 0.006	0.998 ± 0.005
10000	0.997 ± 0.007	0.995 ± 0.009	0.997 ± 0.007	0.997 ± 0.007

For KDD problem our feature selection method was able to select from 11 to 18 features in average from the initial set composed by 41 features. In [3] the authors selected 12 features for the same classes that we considered (normal use and smurf intrusion). Also, as shown in Tables 4.2 and 4.3, all accuracy results are very high, showing that the selected features are relevant ones.

The more labeled information is available the more features are selected and the larger are the classification accuracy over the test set, for both LDA and MLP, for KDD problem. The increase of accuracy given the increase in the number of labeled data is expected, however, the relation between the number of labeled instances and the final number of selected features is not straightforward or clear in our framework. This will be data dependent. It is true that with a larger

Table 4.4: Final number of slected features for ILPD problem.

p_l	n_s	λ								
		0.10	40	90	135	200	270	500	1000	10000
30%	$min\ n_s$	0	1	1	1	2	1	2	1	1
	$max\ n_s$	0	2	2	2	2	2	2	2	2
	$\overline{n}_s$	0	1.9	1.9	1.9	2	1.6	2	1.8	1.4
60%	$min\ n_s$	0	2	3	3	3	3	3	3	3
	$max\ n_s$	2	3	4	4	4	4	4	4	4
	$\overline{n}_s$	0.3	2.8	3.5	3.2	3.6	3.1	3.7	3.5	3.7
70%	$min\ n_s$	0	1	3	3	2	2	3	2	3
	$max\ n_s$	1	4	4	4	4	4	4	3	4
	$\overline{n}_s$	0.4	2.1	3.7	3.3	2.6	3.3	3.7	2.6	3.4
80%	$min\ n_s$	0	0	2	3	2	2	2	2	2
	$max\ n_s$	1	3	3	3	3	3	3	3	3
	$\overline{n}_s$	0.6	2.1	2.5	3	2.9	2.9	2.7	2.8	2.1

Table 4.5: Accuracies achieved by classification models trained considering the final set of selected features for ILPD problem.

λ	$Acc \pm \sigma$			
	LDA		MLP	
	p_l = 30%	p_l = 60%	p_l = 30%	p_l = 60%
0.1	0	0.102 ± 0.011	0	0.143 ± 0.01
40	0.517 ± 0.068	0.532 ± 0.05	0.701 ± 0.025	0.703 ± 0.035
90	0.515 ± 0.068	0.534 ± 0.049	0.706 ± 0.024	0.701 ± 0.037
135	0.517 ± 0.068	0.534 ± 0.052	0.702 ± 0.03	0.702 ± 0.037
200	0.516 ± 0.068	0.534 ± 0.048	0.702 ± 0.027	0.699 ± 0.033
270	0.518 ± 0.068	0.534 ± 0.049	0.708 ± 0.021	0.7 ± 0.041
500	0.516 ± 0.068	0.533 ± 0.049	0.703 ± 0.028	0.701 ± 0.033
1000	0.517 ± 0.068	0.534 ± 0.049	0.705 ± 0.025	0.695 ± 0.041
10000	0.519 ± 0.068	0.533 ± 0.049	0.71 ± 0.017	0.697 ± 0.038

number of labeled data, the calculation of feature relevances will be more accurate. This will affect the two basic steps of the proposed method as follows:

- in the first step, where redundancies are eliminated, the B term will be affected changing the S index value of each pair of features. This will change the feature similarity rank and some feature clusters could have more or less features grouped together.
- in the second step irrelevant features are eliminated. To the extent that the number of labeled data is increasing, features that were irrelevant before may turn out to be relevant now and, therefore, they will not be discarded anymore.

Table 4.6: Accuracies achieved by classification models trained considering the final set of selected features for ILPD problem.

λ	$Acc \pm \sigma$			
	LDA		MLP	
	p_l = 70%	p_l = 80%	p_l = 70%	p_l = 80%
0.1	0.202 ± 0.022	0.3 ± 0.033	0.287 ± 0.007	0.428 ± 0.01
40	0.524 ± 0.054	0.477 ± 0.055	0.706 ± 0.026	0.636 ± 0.022
90	0.533 ± 0.048	0.534 ± 0.057	0.694 ± 0.04	0.7 ± 0.03
135	0.531 ± 0.051	0.537 ± 0.055	0.697 ± 0.038	0.699 ± 0.032
200	0.529 ± 0.057	0.536 ± 0.056	0.706 ± 0.027	0.701 ± 0.032
270	0.535 ± 0.05	0.536 ± 0.056	0.701 ± 0.038	0.695 ± 0.037
500	0.533 ± 0.048	0.534 ± 0.057	0.69 ± 0.037	0.703 ± 0.031
1000	0.536 ± 0.055	0.537 ± 0.055	0.703 ± 0.029	0.696 ± 0.033
10000	0.534 ± 0.049	0.535 ± 0.056	0.696 ± 0.035	0.703 ± 0.03

Table 4.7: Final number of slected features for SONAR problem.

p_l	n_s	λ						
		0.1	5000	12600	20000	40000	145000	400000
30%	$min\ n_s$	0	4	4	4	4	3	4
	$max\ n_s$	2	6	6	6	6	5	5
	$\overline{n}_s$	1	5.4	5.3	5	5	4.4	4.5
70%	$min\ n_s$	1	8	9	7	7	9	9
	$max\ n_s$	4	12	13	13	13	13	14
	$\overline{n}_s$	2.5	10.8	11.6	10.3	11	11.4	12.1

Table 4.8: Accuracies achieved by classification models trained considering the final set of selected features for SONAR problem.

λ	$Acc \pm \sigma$			
	LDA		MLP	
	p_l = 30%	p_l = 70%	p_l = 30%	p_l = 70%
0.1	0.612 ± 0.098	0.728 ± 0.1	0.604 ± 0.089	0.707 ± 0.101
5000	0.746 ± 0.111	0.769 ± 0.08	0.702 ± 0.107	0.724 ± 0.093
12600	0.737 ± 0.107	0.78 ± 0.071	0.719 ± 0.108	0.722 ± 0.091
20000	0.74 ± 0.108	0.764 ± 0.076	0.702 ± 0.114	0.711 ± 0.098
40000	0.735 ± 0.107	0.771 ± 0.072	0.693 ± 0.114	0.71 ± 0.109
145000	0.726 ± 0.0989	0.775 ± 0.100	0.716 ± 0.090	0.703 ± 0.090
400000	0.729 ± 0.0988	0.787 ± 0.103	0.713 ± 0.096	0.707 ± 0.097

For the KDD problem the final number of selected features and the accuracy of the final feature subsets on the test set are very similar for large values of λ. For example, the number of selected features for $\lambda = 500$ and for $\lambda = 5000$ is around 16 for a $p_l = 60\%$ (see Table 4.1). The same behavior is observed for different values of p_l and large values of λ. Actually, only for λ very small we have a smaller

Table 4.9: Final number of slected features for PEN problem.

p_l	n_s	λ					
		0.1	50	170	500	1000	3000
30%	$min\ n_s$	1	9	10	9	10	10
	$max\ n_s$	6	13	13	13	14	14
	$\overline{n}_s$	1.5	11.3	12.2	11.9	12.4	12.3
70%	$min\ n_s$	1	1	11	11	8	11
	$max\ n_s$	1	13	13	13	13	13
	$\overline{n}_s$	1	11.4	12.4	12.4	11.1	12.4

Table 4.10: Accuracies achieved by classification models trained considering the final set of selected features for PEN problem.

λ	$Acc \pm \sigma$			
	LDA		MLP	
	p_l = 30%	p_l = 70%	p_l = 30%	p_l = 70%
0.1	0.983 ± 0.008	0.981 ± 0.008	0.984 ± 0.008	0.983 ± 0.009
50	0.999 ± 0.002	0.998 ± 0.002	0.999 ± 0.002	0.998 ± 0.002
170	1 ± 0.001	1 ± 0.001	0.999 ± 0.002	0.999 ± 0.002
500	1 ± 0.001	0.999 ± 0.002	0.999 ± 0.002	0.999 ± 0.002
1000	1 ± 0.001	1 ± 0.001	0.999 ± 0.002	0.999 ± 0.002
3000	1 ± 0.001	1 ± 0.001	0.999 ± 0.002	0.999 ± 0.002

Table 4.11: Final number of slected features for IONO problem.

p_l	n_s	λ					
		0.1	500	1218	2000	5000	10000
30%	$min\ n_s$	1	2	4	2	2	4
	$max\ n_s$	1	23	22	25	30	17
	$\overline{n}_s$	1	10.4	12	9.4	9.1	9.6
70%	$min\ n_s$	1	2	4	4	2	2
	$max\ n_s$	2	6	12	8	10	15
	$\overline{n}_s$	1.1	4.6	6.9	5.6	5.4	5.5

number of selected features when compared with the results achieved for large λ. It is interesting to notice that except for the case with 30% of labeled data, the number of selected features is quite small for $\lambda = 0.1$ but the classification accuracy is not too much small. For example, for a p_l of 70%, for $\lambda = 0.1$ we have a feature subset with approximately 12 features with a classification accuracy of 98% in average for both LDA and MLP classifiers, while for $\lambda = 10000$ we have near 17 selected feature in average and 99% of accuracy (see Tables 4.2 and 4.3 for accuracy values). Therefore we have near 33% less features with only a reduction of 1% in the accuracy for this problem. In average, for $\lambda = 0.1$ we have 25% less features and only

Table 4.12: Accuracies achieved by classification models trained considering the final set of selected features for IONO problem.

λ	$Acc \pm \sigma$			
	LDA		MLP	
	p_l = 30%	p_l = 70%	p_l = 30%	p_l = 70%
0.1	0.572 ± 0.078	0.547 ± 0.084	0.8 ± 0.061	0.796 ± 0.054
500	0.848 ± 0.051	0.799 ± 0.05	0.862 ± 0.042	0.874 ± 0.05
1218	0.865 ± 0.045	0.807 ± 0.046	0.867 ± 0.048	0.879 ± 0.051
2000	0.849 ± 0.056	0.8 ± 0.048	0.856 ± 0.057	0.876 ± 0.05
5000	0.832 ± 0.058	0.807 ± 0.052	0.859 ± 0.05	0.873 ± 0.055
10000	0.852 ± 0.045	0.792 ± 0.055	0.861 ± 0.051	0.886 ± 0.046

Table 4.13: Crossing points.

Problem	p_l	λ_{low}	λ_{mean}	λ_{hi}
PEN	30	4e-06	265	531
	70	15-06	111	221
SONAR	30	7e-08	134214	268428
	70	8e-08	75015	150030
IONO	30	2e-06	17356	34712
	70	7e-07	4680	9360
ILPD	30	9e-05	39	78
	60	1e-05	21	42
	70	3e-06	16	31
	80	3e-05	16	33
KDD	30	16e-07	2257	4515
	60	2e-06	1746	3493
	70	2e-06	2647	5293
	80	1e-06	4102	8204

5% of reduction in the accuracy for the KDD problem, considering all proportions of labeled data and both classifiers. This shows that the feature subset selected for the smallest λ, in average is more efficient considering the reduction in the number of features compared to the reduction in the accuracy.

Our SSFC method selects in average 2 to 4 features from initial set of 10 features in the ILPD problem, as shown in Table 4.4. Tables 4.5 and 4.6 list the accuracy achieved by a LDA classifier and a MLP. Due to the non-linear nature of the problem MLP performed much better when trained with the selected feature subset. These results are not so good when compared to those obtained by Ramana in [75]. Ramana achieved 98% of accuracy with a backpropagation algorithm using 4 features. In [75] the best selected features are:

- total bilirubin;
- direct bilirubin;

- indirect bilirubin;
- albumin.

Our SSFC method selected when considering 70% of labeled data the following features:

- total bilirubin;
- direct bilirubin;
- sgot alamine aminotransferase;
- sgot aspartate aminotransferase.

The last two features selected by SSFC method are classified as 5^{th} and 6^{th} relevant features in [75], however, the *indirect bilirubin* feature is not present in the ILPD data set available which is used in our experiments. The data set used in [75] contains 12 features while ours contains only 10 features. Feature *albumin* was initially selected by the first step of our method, but was discarded by its lack of relevance to the output variable. This occurs because the relevance step of the method is univariate.

As can be seen in Table 4.4, in average, with 30% of labeled instances SSFC method selects 2 features and with 60% of labeled data the number of selected features increased to approximately 4. For larger proportions of labeled data there is no big difference in the final number of selected features. This is the same behavior observed for the KDD problem: for proportions of labeled data bigger than 60% the number of selected features does not change substantially. Nevertheless, the accuracy over the test set is not substantially affected by the increase of the number of labeled data, even because this problem have 583 instances with 10 dimensions with two unbalanced classes (see Tables 4.5 and 4.6 for accuracy values) . It is important to notice that in the experiments the labeled data are randomly chosen without concern in keeping the class proportions balanced, just in order to keep the problem as near to real situations as possible, once, in real world classes could be unbalanced.

In Table 4.13 we can see that the value of λ_{mean} as defined in Section 4.1.1.1, is around 40 depending on the number of labeled data. Table 4.4 shown that for $\lambda = 40$ the number of selected features is 2 in average for all proportions of labeled data. The selected features are the best two mentioned before and the accuracy achieved using this feature subset is the same than those achieve considering 4 selected features. Setting λ as defined in Section 4.1.1.1 yields a more efficient feature subset.

For the SONAR problem the SSFC method selects between 4 and 12 features in a set of 60 original features. This problem has very few data considering the total number of features. The curse of dimensionality

is a real issue here. In [41] the authors achieved results with accuracy of 81% in average, for different number of hidden neurons in a MLP network, considering all 60 features. Table 4.8 shows the accuracy achieved in a classification task using the feature subset selected by SSFC method for different proportions of labeled data and for different values of λ and Table 4.7 shows the final number of selected features in each situation. A LDA classifier reaches an accuracy of 78% using feature subsets composed of about 10 features, chosen when considering 70% of labeled data.

In the PEN problem the SSFC method is able to select, in average, 11 to 12 features from the original set containing 16 features and for different proportions of labeled data, as can be seen in Table 4.9. In all experiments and for all different values of λ either LDA or MLP classifiers achieve over 98% of accuracy in average (see Table 4.10). In [5] the authors achieved, depending on the network configuration, from 92% to 98% of accuracy on a test set using a MLP network and considering all 16 features. As we achieve at least 98% of accuracy in average, using about 10 features we can say, at least, that those 6 discarded features are irrelevant ones, and, of course, the SSFC method is able to select the important features. It is very interesting to notice that in the worst simulated case, where only 30% of data is labeled, and λ is equal to 0.1 (larger influence of unlabeled similarity term A of Equation 4.3) the SSFC method selects from 1 to 6 features. In average, in those conditions it selects 2 features and the accuracy achieved by the two classification models are very good (98%). Once more small value of λ produces a more efficient feature subset when comparing the reduction in the number of selected features and its respective reduction in the level of accuracy of the classifier models.

There are some works as [83] that achieved accuracies from 90% to 92% for a classification task using a MLP trained with all 34 features of IONO problem, in a supervised framework. The SSFC method selected from 4 to 12 features in average as can be seen in Table 4.11, and achieve a classification accuracy, considering these selected features, about 89% in average (see Table 4.12). Only for the case where $\lambda = 0.1$ the LDA classifier does not performed well with the selected subsets.

Making an analysis on the results we can observe that the solutions for the different settings of parameters used in simulations show, in general, a large overlap. Figures 4.3 to 4.9 show graphs with the average accuracy results and their ranges where the results can be found with 95% of confidence. This overlap of the result ranges indicates that the value of λ is not affecting too much the predective power of the selected feature subset as mentioned before. In general, for λ too small, in general, the set of selected features presented the the lowest accuracies. We can also observe that the decrease in the number of labeled data does not cause a reduction in the prediction power of the selected set of variables.

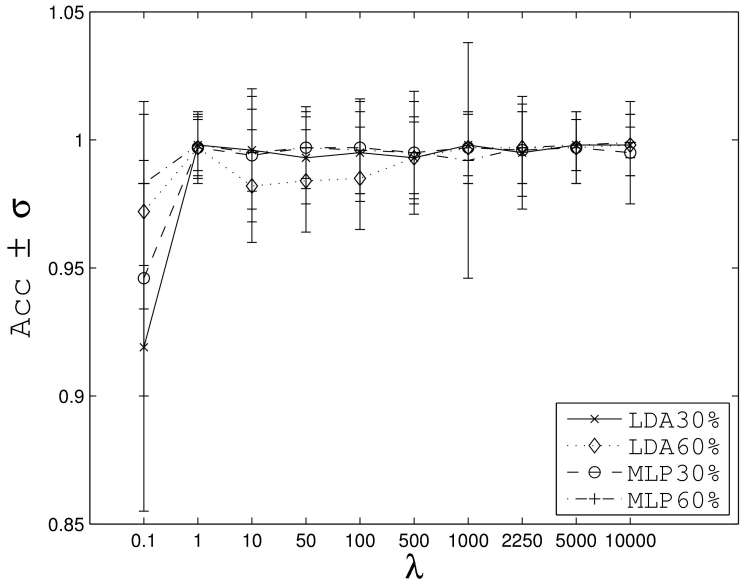


Figure 4.3: Analysis of prediction accuracy results for different parameter configurations for KDD problem.

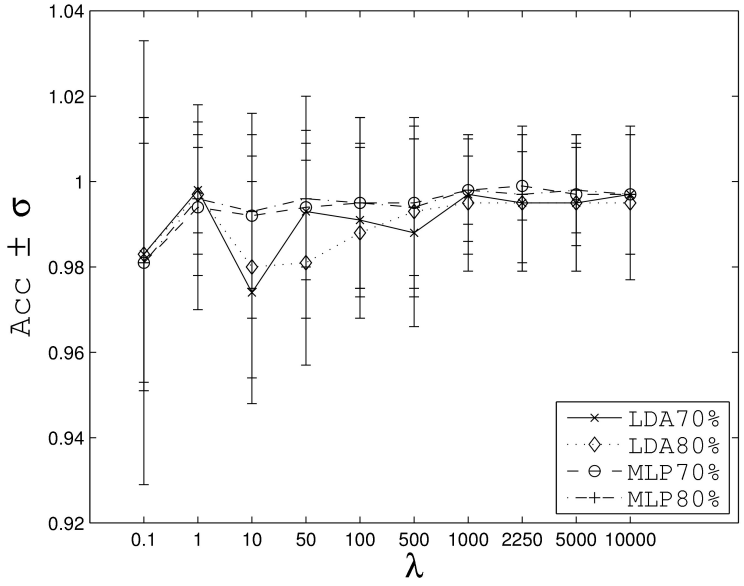


Figure 4.4: Analysis of prediction accuracy results for another different parameter configurations for KDD problem.

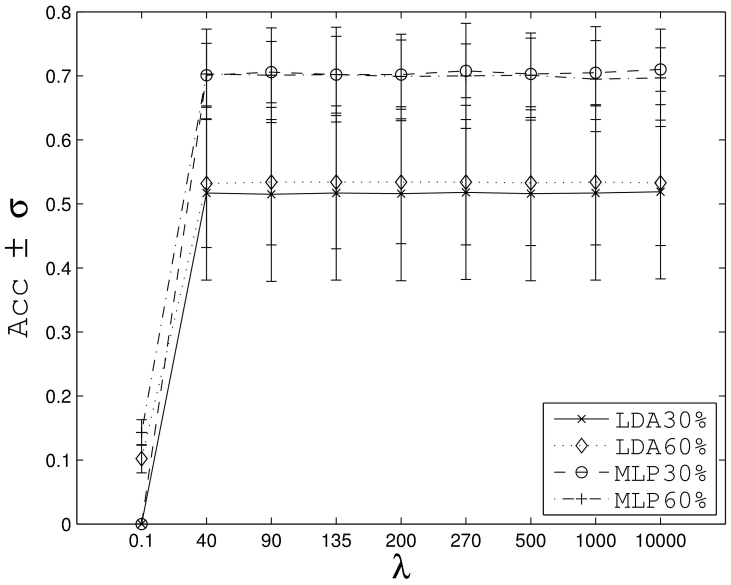


Figure 4.5: Analysis of prediction accuracy results for different parameter configurations for ILPD problem.

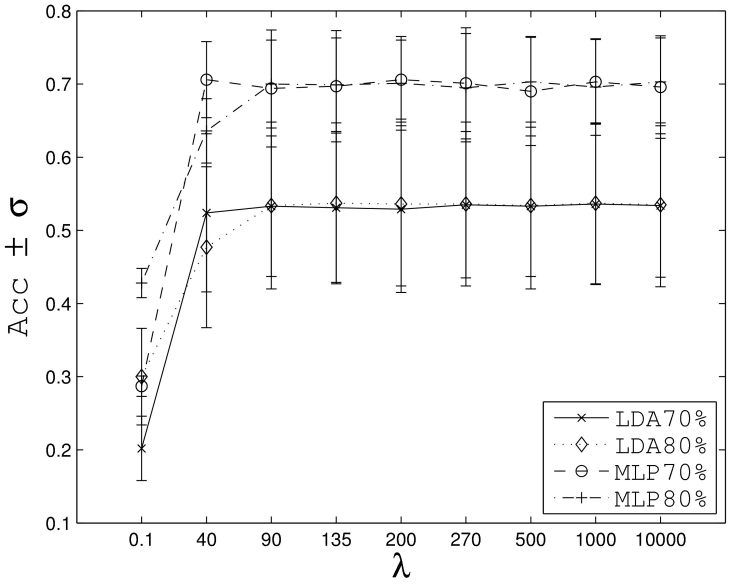


Figure 4.6: Analysis of prediction accuracy results for another different parameter configurations for ILPD problem.

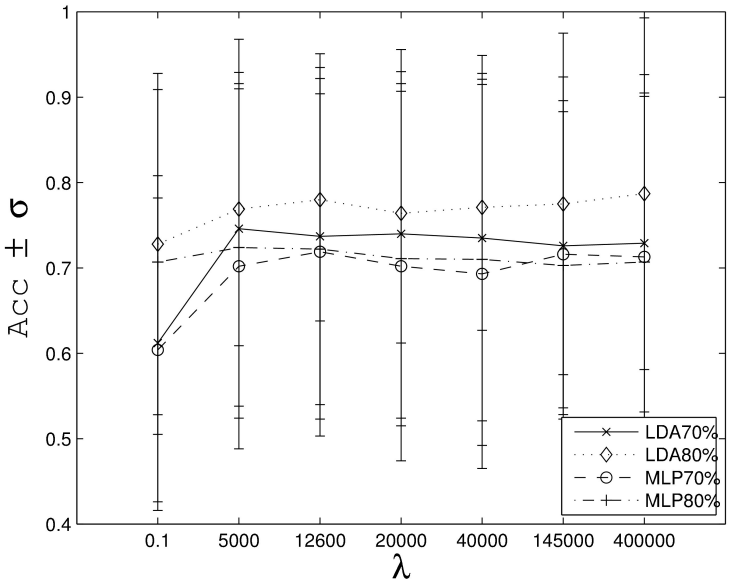


Figure 4.7: Analysis of prediction accuracy results for different parameter configurations for SONAR problem.

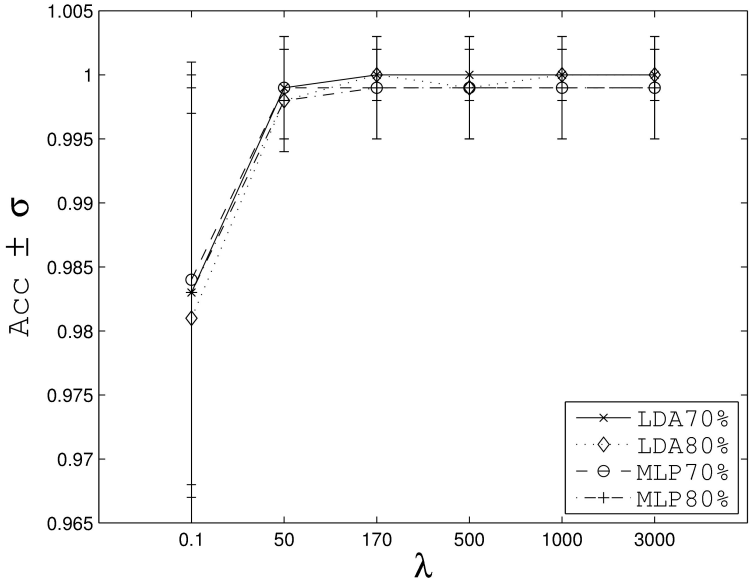


Figure 4.8: Analysis of prediction accuracy results for different parameter configurations for PEN problem.

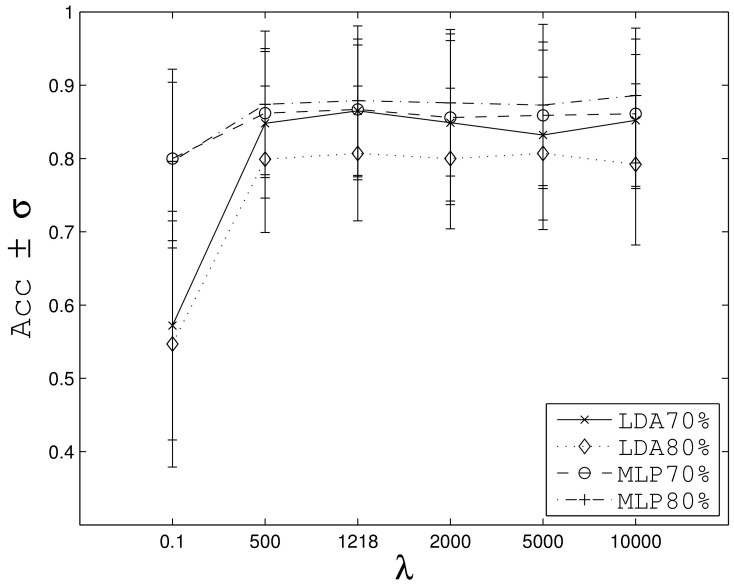


Figure 4.9: Analysis of prediction accuracy results for different parameter configurations for IONO problem.

Finally it is important to notice that this method worked well for the tested problems yielding good results, improving the results by the use of unlabeled data. Only for the ILPD problem it got the worse results when compared to the results from previous works, however, the database used in this test lacks the main feature for solving this problem. However the results are quite consitents for different parameter settings. This method has a high computational cost since we need to calculate the similarity index for all possible pairs of features and especially the calculation of the significance of this index. This involves the estimation of densities many times to get a reliable result but is a method that can be applied especially when you have a large number of unlabeled data.

4.2 FEATURE SELECTION METHOD BASED ON CLUSTER HOMOGENEITY

Some machine learning approaches include clustering methods in order to label instances. They are based on the assumption that the underlying distributions of the data, and their modes, can be estimated from the sampled data by clustering methods. One of the basic principles of structural data analysis is that labels are consistent with data distributions. Accordingly, the relevance of features to labels should also be reflected by the relevance of features to clusters.

In this section another semi-supervised feature selection strategy based on MI is introduced. The basic principle of the method is to replace, for unsupervised data, the label information by the cluster

information in order to estimate the relevance of each feature or feature subset. In Section 4.2.1 the chosen method to perform feature selection is presented and in Section 4.2.2 the unlabeled data is introduced in the method. The new method is applied to some problems and the results are shown in Section 4.2.3.

4.2.1 *Feature Selection method*

Feature selection is usually accomplished according to a relevance criterion and to a search strategy. The former aims to assess how relevant a single feature subset is, while the latter aims to guide the search towards the most relevant feature subset, since, in practice, testing all possible subsets (exhaustive search) can be unfeasible even for problems with few variables. Here a filter method is implemented using MI as a relevance criterion. Roughly speaking, MI measures the amount of information shared among two or more sets of variables [24] capturing even nonlinear relations among them. The multivariate properties of MI makes it an important approach to assess the relevance of subsets of features, since it may be affected by joint behavior of a feature in the presence of others. Equation 4.11 repeats bellow the relevance between the input data X_i and the output vector Y defined in Section 4.1.2.2, that is used in this method.

$$R_i = MI(X_i, Y). \tag{4.11}$$

The search technique chosen to be implemented in this method is the forward-backward (FB) procedure [34, 39, 45]. The forward strategy has smaller capability to finding more complementary features, compared to backward selection. On the other hand even the smallest nested subset is predictive. The backward strategy, in turn, is capable of finding complementary features, however, its performance is degraded for smallest nested subsets [42]. So, the forward-backward process tries to get the best of both approaches.

The Forward step

The method begins with an empty set of selected features $\Gamma = \{\emptyset\}$. In the very first iteration the method evaluates the individual relevance of each feature i from original feature set F. The feature i whose relevance to the output variable R_i is maximum ($R_i = max(R)$) is permanently added to Γ and removed from F. $R = \{R_1, R_2, ..., R_i, ..., R_{n_f}\}$ is the vector of all individual features relevance values and n_f denotes the total number of features in F.

After this first iteration, the method continues considering now the new selected feature subset $\Gamma = \{F_i\}$. The n_r remaining features ($n_r = n_f - 1$) are temporarily added to Γ, one at a time, to evaluate each

new subset $\{\Gamma \cup X_j\}$ relevance, with $j \neq i$. A new relevance vector $R_n = \{R_1, R_2, ..., R_j, ..., R_{n_r}\}$ is then calculated, where $R_j = MI(\Gamma \cup X_j, Y)$. If $MI(\Gamma \cup X_j, Y) > MI(\Gamma, Y)$ then, feature j will be permanently added to Γ and removed from F. Note that a direct comparison between $MI(\Gamma \cup X_j, Y)$ and $MI(\Gamma, Y)$ values is not a good idea once the selected feature subset at each iteration has one more dimension . Therefore a strategy can be afforded in order to allow this comparison: a random vector F_{jp} generated by permuting the elements of feature F_j can be added to Γ obtained from last iteration. To add a random feature to selected set Γ of last iteration will not change its MI with the output vector. Therefore, if

$$MI(\Gamma \cup F_j, Y) > MI(\Gamma \cup F_{jp}, Y) \, , \tag{4.12}$$

then feature j can be permanently added to Γ and eliminated from F. This process is repeated until condition defined by Equation 4.12 is not met anymore.

The Backward step

At this point, the Backward strategy is applied in order to verify if some of the selected features can be discarded. The idea is to remove one feature per time from the selected feature subset Γ and check if the mutual information with the output decrease or not. If $MI(\Gamma \setminus X_m, Y) \geqslant MI(\Gamma, Y)$, feature m can be permanently excluded from Γ. In other words, if mutual information of Γ leaving its m^{th} feature out does not changes or even increases feature m can be discarded. The issue in doing this comparison is the same for forward step: Γ and $\Gamma \setminus X_m$ have different dimensions and could not be directly compared. In order to compare similar numbers the same strategy from previous step is adopted but in a slightly different way. Actually, instead of removing feature m from Γ we just permute its elements in order to create a random vector X_{mp} without any relation with the output but keeping both sets in the same dimension. Therefore if

$$MI((\Gamma \setminus X_m) \cup X_{mp}, Y) \geqslant MI(\Gamma, Y) \, , \tag{4.13}$$

feature m can be eliminated from Γ. This step is repeated until condition defined by Equation 4.13 does not hold anymore.

After these two steps features remaining into Γ set constitute the final selected feature subset. A simple pseudo-code for this Forward-Backward process is shown by Algorithm 4.

Algorithm 4: Simple pseudo-code for the Forward-Backward process.

```
   // Forward Step
1  Γ = {∅}; // empty set of selected features
2  F = {F_1, F_2, ..., F_{n_f}}; // Set of original features
3  for i = 1 → n_f do
4      R_i = MI(F_i, Y); // individual feature subset relevance
5  Get feature i whose R_i = max(R), add it in Γ and remove from F;
6  while Forward stop criterion not match // Equation 4.12
7  do
8      for j = 1 → n_r do
9          R_j = MI(Γ ∪ F_j, Y); // feature subset relevance
10     Get feature j whose R_j = max(R), add it in Γ and remove from F if
       R_j > MI(Γ, Y)
   // Backward Step
11 while Backward stop Criterion not match // Equation 4.13
12 do
13     for i = 1 → n_s do
14         R_i = MI((Γ \ X_i) ∪ X_ip, Y); // feature subset relevance
15     Get feature j whose R_j = max(R) and remove it from Γ if R_j ⩾ MI(Γ, Y)
```

4.2.2 *Using unlabeled data*

Evaluating feature relevance using MI requires that data set contains some labeled data; however, small data sets may fail in well representing the general relation between input and output variables as shown in the illustrative example[4] of Figure 4.10a. In this example, the distribution of labels is not well represented if a small data set is sampled within the central circle as shown in Figure 4.10a. Since labeling can be costly, it is expected that unlabeled data could provide some information about the posterior probability of labels that could improve feature selection. The feature selection task could be performed by searching for those features that are important not only for labels, but also for clusters, which are expected to be consistent with labels. The use of both labeled and unlabeled data characterizes the semi-supervised paradigm.

Data distribution information can be useful even when there is a reasonable amount of labeled data. As an example, consider a forward feature selection procedure applied to a three dimensional problem, for which features X_1 and X_2 together fully explain the labels in Y and X_3 is completely random (Fig. 4.10a shows the relevant features). Individually none of the three features is able to explain the labels, so

4 This is an hypothetical example to illustrate the problem. In real problems labeled and unlabeled data are not expected to be concentrate in different space regions.

in the first iteration of the Forward-Backward algorithm defined in Section 4.2.1, by chance, feature X_3 could be ranked first, resulting in a poor initial subset selection. This can happen because the very first algorithm step is univariate and, in this hypothetic example the MI value for each feature will be low. In such a situation the distribution of the dataset may provide additional information about the relevance of X_1 and X_2.

Features X_1 and X_2, together, are able to discriminate the instances into four different clusters according to the distribution of the dataset, regardless of labels, as shown at Figure 4.10b. So, if we are able to estimate the cluster structure that best fits data generator functions, we can estimate the relevance of each feature subset according to the dataset distribution. Each pattern, especially the unlabeled ones, can be associated to a given cluster and receive a tag according to the cluster number (Figure 4.10b). These "cluster labels" assigned to each unlabeled data, generating the *cluster label vector* Y_{c1}. In addition, the number of clusters nc should be sufficiently large in order to guarantee label homogeneity within clusters.

In general, the MI between a feature set X and its vector of labels Y can be defined in terms of their joint and marginal probabilities as

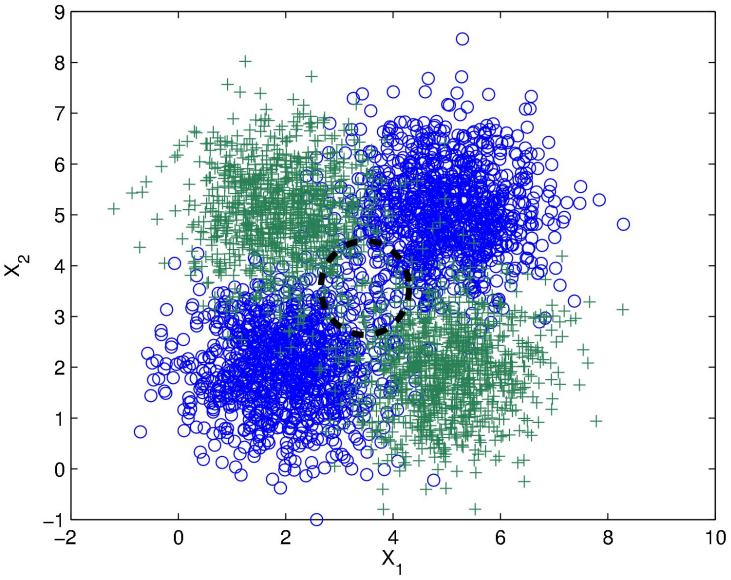
$$MI(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)}\,, \tag{4.14}$$

where x is a vector composed by the values of all features for a given pattern.

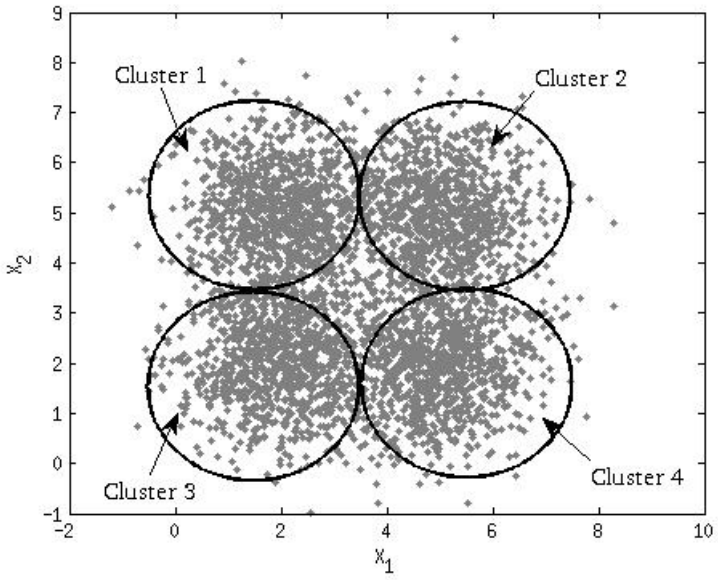
Equation 4.14 can be rewritten by splitting the data according to their classes as shown in Equation 4.15 for a binary case, where superscripts (1) and (-1) indicate respectively the data belonging to classes +1 and -1:

$$\begin{aligned} MI(X,Y) = & \sum_{x \in X^{(1)}} \sum_{y \in Y^{(1)}} p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)} + \\ & \sum_{x \in X^{(-1)}} \sum_{y \in Y^{(-1)}} p(x,y) \log \frac{p(x,y)}{p(x)\,p(y)}\,. \end{aligned} \tag{4.15}$$

Assuming that, after clustering procedures, clusters C_i, $i = 1, 2, ..., k$ are homogeneous and correspond to instances from the same class,



(a)



(b)

Figure 4.10: For the two class XOR problem, in 4.10a none of the features alone can explain the distribution of the classes, defined by circles and crosses, and in 4.10b, even without the labels, features 1 and 2 are still able to explain data distribution.

i.e., they were generated in such a way that $\{C_1, C_2, ..., C_i\} \subset Y^1$ and $\{C_{i+1}, ..., C_k\} \subset Y^{-1}$, the MI can be rewritten as

$$\begin{aligned} MI(X,Y) = & \sum_{x \in X^{(1)}} \sum_{c \in C_1} p(x,c) \log \frac{p(x,c)}{p(x)\,p(c)} + \ldots + \\ & \sum_{x \in X^{(1)}} \sum_{c \in C_i} p(x,c) \log \frac{p(x,c)}{p(x)\,p(c)} + \\ & \sum_{x \in X^{(-1)}} \sum_{c \in C_{i+1}} p(x,c) \log \frac{p(x,c)}{p(x)\,p(c)} + \ldots + \\ & \sum_{x \in X^{(-1)}} \sum_{c \in C_k} p(x,c) \log \frac{p(x,c)}{p(x)\,p(c)} \,. \end{aligned} \tag{4.16}$$

In such situation, $MI(X, Y_{cl}) = MI(X, Y)$. In practice, as we are dealing with unlabeled data, if the number of clusters is defined sufficiently large to allow that clusters encompass mostly instances from a single class, we have that $MI(X, Y_{cl}) \approx MI(X, Y)$. Equation 4.11 can now be rewritten as

$$R_c = MI\left(X^{(\ell)} \cup X^{(u)}, Y \cup Y_{cl}\right) \,, \tag{4.17}$$

where $X^{(\ell)}$ and $X^{(u)}$ are respectively the labeled and unlabeled data sets, Y is the label vector and Y_{cl} is the vector of cluster labels. Equation 4.17 can be directly used in our forward-backward feature selection filter method. In this way more information about the relevance of each feature subset is provided taking into account the cluster information. Therefore cluster information replaces the "label" information for unlabeled data in order to consider them in the evaluation of the MI.

4.2.3 *Experiments and results*

The experiments for this feature selection method based on cluster homogeneity consist in comparing the performances of feature subsets selected according to a pure supervised approach and the semi-supervised method proposed. A sequential forward-backward feature selection strategy [39, 45] was implemented and applied to some real and synthetic datasets, using a MI estimator tailored to classification problems. This estimator was developed by Goméz et al. [39] which has high performance even in a context of scarce data. Data was clustered with K-means algorithm [65]; the number of clusters n_c is shown in Table 4.14. The number of clusters was empirically chosen in such way to be sufficiently large in order to guarantee label homogeneity within clusters. This means that we vary the number of clusters and choose the best setting based on the results.

The final results aim at comparing the final feature subset obtained when using only labeled data Γ^{ℓ}, with the one obtained using both labeled and unlabeled data $\Gamma^{\ell u}$.The Linear Discriminant Analysis Method (LDA) was used in order to classify the test set in three different conditions: considering only Γ^{ℓ}, $\Gamma^{\ell u}$ or the set F of all features. The mean classification accuracy and standard deviation for 10 different trials are presented in Table 4.15. LDA was chosen to perform the classification tests due its simplicity and robustness.

Three data sets were used in the experiments. The first one (FBench) is a synthetic data set, originally developed for benchmark regression problems [36], whose output is a function of some of their random input variables. Its output was discretized into two classes (1 for $Y > 0$ and -1 for $Y < 0$) in order to transform it into a classification problem. Two other problems come from the UCI Machine Learning Repository [1]: the sonar data set, composed by instances of a sonar response from rocks and mines, and the Pen-Based Handwritten Digits data set, composed by digit samples from 44 different writers. For this last problem we considered only instances of digits 1 and 2 in the experiments.

On each trial a very small portion of data n_{ℓ} was chosen as labeled data because this method is designed for problems with few labeled data. Another n_t quantity was selected as a test set and the rest n_u instances was considered as unlabeled data, so their labels were not considered in the FS task. This values are shown in table 4.14. The number of clusters was varied from 2 to 150 for FBench problem, and from 2 to 60 for Sonar and Pen problems.

Table 4.14: Data and algorithm parameters: n is the total number of instances, n_f is the total number of features, n_{ℓ} is the number of labeled instances, n_u is the number of unlabeled instances, n_t is the number of instances in the test set, n_c is the number of clusters, Γ^{ℓ} is the final set of selected features considering only labeled data and $\Gamma^{\ell u}$ is the final set of features considering labeled and unlabeled data.

Problem	n_f	n	n_{ℓ}	n_u	n_t	n_c	Γ^{ℓ}	$\Gamma^{\ell u}$
FBench	10	10000	49	7952	1999	100	4 1 5	1-5-4-10-2-3
Sonar	60	208	11	147	41	40	46	46-36-20-27-30-16-43-24
Pen	16	2287	114	1717	456	30	4	4-15

In all experiments the obtained accuracy for the subset $\Gamma^{\ell u}$ is higher than those obtained using the features selected when using only the labeled data. It is possible to observe in Table 4.15 that, for Fbench and Sonar problems, there is no significant accuracy loss when using only the features in $\Gamma^{\ell u}$ instead of using all features. Only for the Pen data set there is a loss with respect to F. However, there is an improvement in accuracy with respect to using only the supervised

Table 4.15: Shows the results for each test, where n_s is the final number of features of each subset.

Problem	(n_s) Accuracy $\pm \sigma$		
	F	Γ^{ℓ}	$\Gamma^{\ell u}$
FBench	(10) 0.8502±0.0123	(3) 0.8199±0.0127	(6) 0.8504±0.0122
Sonar	(60) 0.7117±0.6667	(1) 0.6052±0.1343	(8) 0.6924±0.0803
Pen	(16) 0.9808±0.0075	(1) 0.8478±0.0251	(2) 0.8780±0.0264

set Γ^{ℓ}, as expected, since the objective here is to show that cluster information from unlabeled data, and consequently the proposed method, conveys information to improve feature selection.

Regarding the computational cost we have to make clear one point. In fact, what is being proposed and developed in this work is the idea of using structural information in the generation of cluster labels, that can be added to the label vector and used in any supervise feature selection method. This allows to transform a supervised method into a semi-supervised one. The idea is very simple and easy to implement which then leads us to conclude that the computational cost of applying it is associated with the computational cost of the clustering method and of the selection method chosen.

5

MULTI-OBJECTIVE APPROACH

One of the difficulties of dealing with feature selection problems is to assess the quality of the resulting feature subset. Since in most applications the subset is applied to a classification problem, a classifier is usually designed to verify the quality of the selected features for that particular problem. This is usually needed because the selection criterion methods are not able to fully represent the muti-variate nature of the non-linear input-output relations, so a non-linear inductive model is needed in order to assess the final result. Even MI adopted in Chapter 4, despite being a non-linear correlation measure, is not able to map the complex relations embodied in the data set. In other words, the quality of a set of features can only be assessed by using them to solve the target problem. Classification model induction and feature selection are usually accomplished separately especially due to the variability of parameters and methods. Nevertheless, Multi-Objective learning (MOBJ) of neural networks may provide a methology for jointly solving the two problems, as disscussed in this chapter.

It is well-known that supervised learning is inherently a multi-objective task [38, 94], since not only fitting to the data set should be accomplished, but also the complexity of the inductive model should be minimized. The optimization of the two conflicting objective functions imprint the multi-objective nature to the problem. The link that is established between MOBJ learning and feature selection results from the use of L1 norm (LASSO) as a measure of complexity. The MOBJ solutions yielded with LASSO tend to naturally select input features as will be shown in the sections that follow, since the magnitude of the weights tend to be discrepant for input variables. The features associated with small magnitudes are then discarded.

The development of the semi-supervised feature selection methods is done through the formalization of the problem, and it can be done by its characterization as an optimization problem with constraints, or even by a multi-objective optimization problem. The problem may be described as having multiple objective functions, such as the error of labeled data set, the final number of selected features and any function that could be quantified from the labeled and unlabeled data sets. The margin of separation in the region of lower density of samples is a natural choice of objective function capable to use unlabeled data, for instance. From this perspective, in order to solve the feature selection problem in this approach, one should solve the optimization problem considering appropriate objective functions.

However the main problem is exactly the definition of these functions, especially those concerning the set of unlabeled data.

In Section 5.1 we show that the problem of feature selection is naturally multi-objective and in Section 5.3.2 the well known LASSO algorithm is rewritten in such way to eliminate the irrelevant features during the training step. The LASSO is more appropriated to the task of selecting features because the weights are more likely to be zero as discussed in Section 5.3.1. Finaly a feature selection method is proposed in Section 5.4 and results of its aplication on syntethic and real problems are shown in Section 5.5.

5.1 MACHINE LEARNING MULTI-OBJECTIVE NATURE

The minimization of a loss function [94] would yield the approximate to the real generate function of the data. The most common loss function used in Machine Learning, especially in Supervised Learning, is the quadratic error. Equation 5.1 defines the quadratic error function as a loss function to evaluate the empirical risk [94]:

$$R_{emp} = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i, \omega))^2 \ , \tag{5.1}$$

where y_i is the target variable value of the i^{th} observed value of X (x_i), $f(x_i, \omega)$ is the function implemented by the model and n is the sample size.

In order to reduce the real risk [94] in the supervised learning approach, both the empirical risk R_{emp} and the model capacity have to be minimized, once these objectives are conflicting. Minimizing the R_{emp} implies rising the model complexity and vice-versa. Then, the solution for the supervised learning problem is characterized by the optimization of these two conflicting objectives [74, 15] that must be well balanced in the optimal solutions. The best solutions are restricted to the *Pareto Set*, where none of the objectives can be decreased without increasing the other one.

The set of all possible solutions for a given problem is defined or limited by the number of parameters of the model, i.e., these parameters determine the structure capacity of the model. Meanwhile, any restriction in the solution space defines the limits of the model capacity [15]. Figure 5.1 shows these limits. The external circle in this schematic graphic represents the limits of a set composed by all possible solutions that a given model, with a given number of parameters, is able to produce. The inner circle shows the subset of these solutions that meet any given set of restrictions in the solution space. This inner circle represents the effective capacity while the external circle represents the capacity of the model.

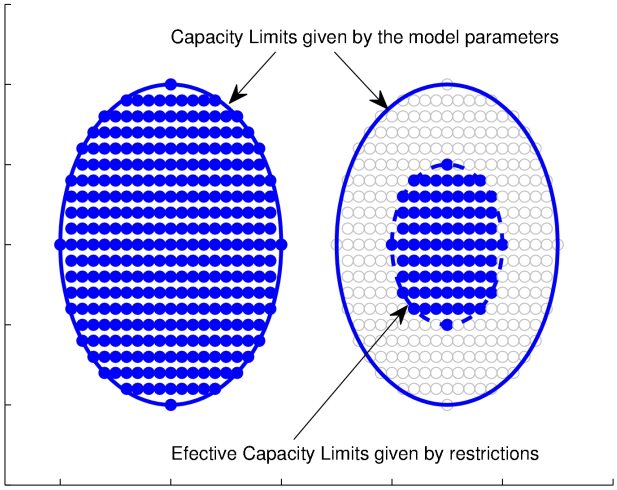


Figure 5.1: Structure capacity x Effective capacity.*RWP*

It can be shown that the effective capacity can be controlled by the square of the norm of the model parameter vector ($\| \omega \|^2$) [94]. A surface Ω of the $\| \omega \|^2$ for the case with only two parameters ω_1 and ω_2 is shown in Figure 5.2. If a determined value μ_1 of $\| \omega \|^2$ is fixed, then a set of possible values that the model parameters can assume, in order to have $\| \omega \|^2 = \mu_1$, is determined by the intersection between the surface of $\| \omega \|^2$ and the plane M_1 defined by μ_1. Saying in a different way, all the solutions with the norm value of the parameters vector equal to μ_1 are the ones in the intersection $\Omega \cap M$. Figure 5.3 shows the set of solutions with $\mu_1 = 2$ and $\mu_2 = 4$ projected on the plane $\omega_1 \times \omega_2$.

In order to solve the multi-objective problem, let us assume that we are dealing with two quadratic functions that have to be both minimized, and whose minimal points are non coincident, otherwise minimizing one of them is sufficient to find the solution. It is possible to see in the objective space defined by these two objective functions (Figure 5.4), that, the solutions corresponding to the curve points marked with triangles have no good candidate solutions, since, for all of them, both objectives or even, at least, one objective function can be yet minimized without increasing the other one. The solutions within the region defined by the curve points marked with circles has good candidate solutions, because there is no way to minimize one of the objectives without worsening the other one. This region is called the Pareto Front and the solutions lying on this front are optimal.

The Pareto Set is the frontier between all feasible and non feasible solutions and an example is shown in Figure 5.5. All solutions lying over the Pareto Front are non-optimal but feasible ones, while all solutions lying under the Pareto Front are unfeasible. The dashed

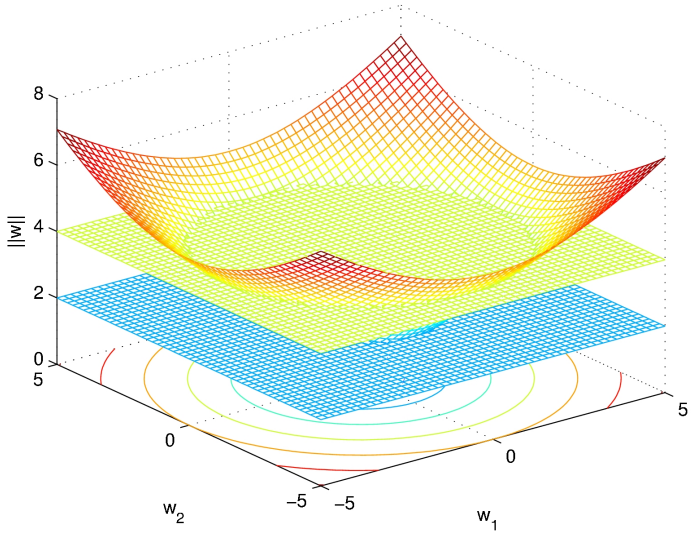


Figure 5.2: $\| \omega \|^2$ surface. Planes determine set of solutions with same value of $\| \omega \|^2$ in the intersection with its surface. *RWP*

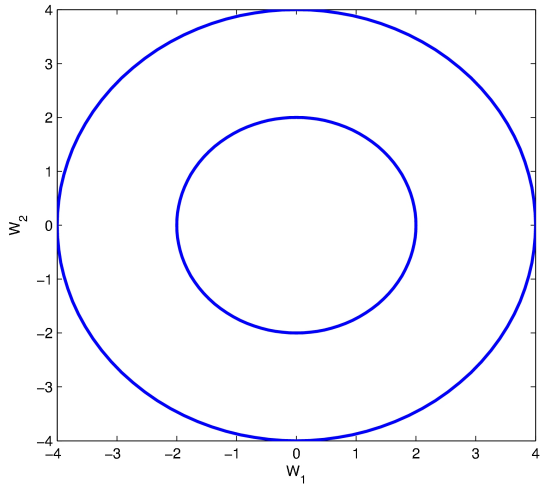


Figure 5.3: Solutions for $\mu_1 = 2$ and $\mu_2 = 4$.*RWP*

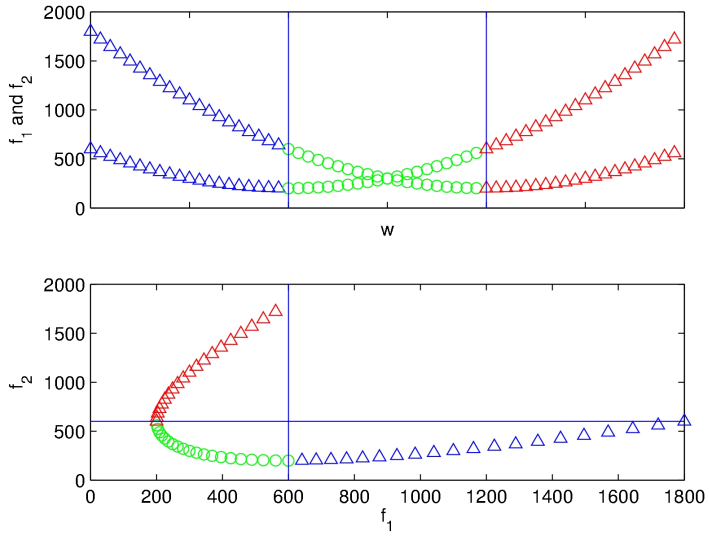


Figure 5.4: Candidate Solutions.*RWP*

line in Figure 5.5 represents the set of all solutions whose parameters vectors have the squared norm equal to 2, for example. Each circle on the line determined by $\mu_1 = 2$ is a model, whose vector of parameters ω has $\| \omega \|^2 = \mu_1 = 2$, but with different values for the second objective function (that in this case is the quadratic error function). It is interesting to note that, for a given model capacity μ_1 , all these solutions are feasible, but only one has the minimum value of the second objective function. This is the solution that lies in the Pareto Front (marked with a triangle in Figure 5.5). Moving from the solution with high error to the one with low error (in terms of the second objective function) for a given capacity, defines a trajectory in the objective space where the model parameters are adjusted by some method. Therefore, controlling this trajectory [23], fixing μ for different values, is a easily way to obtain the Pareto set. Summarizing, for each fixed value of μ, the second objective is minimized in order to find the Pareto solutions.

There will not exist a unique best solution and the algorithm will have to choose one among all the optimal solutions from the Pareto Set, taking into account the compromise between the objective functions that are being considered. There are a lot of strategies to choose the best solution in the Pareto Set in the literature.

Therefore, learning problems are inherently multi-objective. This statement can be easily extended to the feature selection problem, and constitutes the main reason to address this task with a Multi OBJective (MOBJ) approach.

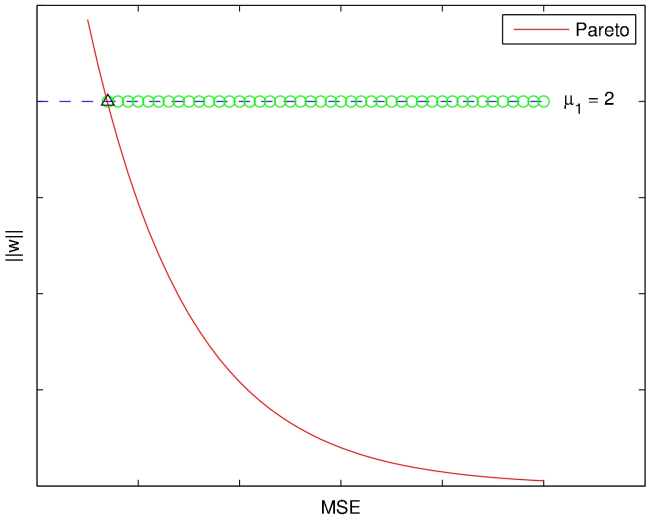


Figure 5.5: Finding Pareto solutions. Dashed curve indicates the Pareto solutions. *RWP*

5.2 INDEPENDENCE OF LAYERS IN NEURAL NETWORKS LEARNING

A Multi Layer Perceptron (MLP) is adopted in this thesis in order to develop the Multi-objective approach. In a MLP the problem of feature selection is solved in the hidden layer when a linearly separable mapping is determined. The output layer is encharged of finding the hyperplane that correctly classifies the mapped instances with best generalization. In order to have a trained network with good generalization we need any linearly separable mapping in the first layer and a maximal margin hyper plane in the output layer. If the MLP is able to classifies correctly all instances, the training process was able to find the relevant features in the input space using them to define a good mapping. When a MLP is trained to learn the data generation function, the feature selection problem is solved at same time because the mapping is done assigning larger weights to important features and lower weights to irrelevant features. We will use this characteristic in order to solve the feature selection task.

A classical approach to train a MLP neural network is to minimize an objective function based on Mean Square Error (MSE) between the desired output (labels) and the MLP output, given by

$$\text{MSE} = \frac{1}{2}\sum_{n}\left(Y - \hat{Y}\right)^2, \tag{5.2}$$

MLP classical training approach

where n is the number of instances, Y is the network output and $\hat{Y}$ is the desired output (the known label vector). This basic approach is also known as ordinary least squares estimates (OLS) [76]. However

one of the main drawbacks with OLS is the prediction accuracy: it may achieve low bias solution, but with large variance. MLP weights are updated in order to minimize MSE in the training step. The most known algorithm to train a MLP is the back-propagation [79] and most of learning methods are based on this algorithm. A review on those training methods can be found in any good neural network book as [16, 49, 30].

layers role

A neural network can be used to learn classification or regression functions of a given data set. Throughout this work a two layers Multi-Layer Perceptron neural network (MLP) within a classification framework will be considered. Each one of these two layers have distinct functions:

- the first layer, or hidden layer, is encharged to map patterns from input space to a higher dimensional feature space;
- the second layer, or the output layer, has the function of generating a separation hyperplane in order to classify those mapped patterns.

discussion about mapping

If the output layer is designed with one neuron with linear activation function, the hidden layer has to generate a linearly separable mapping in order to allow a correct classification. The methods proposed in this chapter are based on the following assumption:

> Any linearly separable mapping from input space to feature space in a two layers MLP neural network is sufficient in order to find a very good solution with good generalization.

Cover's theorem

This assumption is grounded in the theorem of separability of Cover [25] and in the *surface separation capacity* corollary resultant from this theorem.

Cover's theorem is formulated as:

> *A complex problem of pattern classification, designed nonlinearly into a high dimensional space, is more likely to be linearly separable than in a low dimensional space [25].*

Patterns that are nonlinearly separable in the input space of a MLP network are more likely to be linearly separable in the feature space if mapping is done with a nonlinear function to a dimension sufficiently greater. Cover also shows in [25] the following result from its theorem:

> *The expected maximum number of random patterns which are linearly separable in a space of dimension* d *is equal to* $2d$.

This result suggests that a natural definition for the separation capacity of a family of decision surfaces with d degrees of freedom

is 2d [25]. Therefore, if data can be correctly linearly separated in the feature space, it means that the first layer found a good mapping and many linear hyperplanes can be designed in order to classify correctly the instances solving the learning problem. However there is only one hyperplane equidistant to classes. This solution is known as maximal margin hyperplane and it provides the model with best generalization.

max margin hyperplane

A maximal margin solution has minimal weight norm as shown in [49]. So if we want a solution with maximal margin that correctly classifies the training patterns we do need to minimize the norm of the weights while minimizing the MSE. This claims for a Multi Objective approach (MOBJ), and there are also many algorithms to train this network as the one proposed by Costa in [22] and the Support Vector Machines [93]. However we do not have to train both layers at the same time or in the same way. Based on the assumption that any linearly separable mapping can lead to a good generalization solution, once such mapping is provided, a maximal margin solution can be easily found.

MOBJ solutions

Therefore, MLP layers can be trained individually, where the training of first layer has the objective to find a linearly separable mapping of input data, and the training of output layer has the objective of finding the maximal margin hyperplane.

Norm influence test

In a classical MOBJ training algorithm both error and weight vector norm have to be minimized and the following optimization problem can be defined:

$$\min_{\omega} \frac{1}{2} \sum_{n} \left(Y - \hat{Y}\right)^2 + \alpha \sum \|\omega\|_2 \ , \tag{5.3}$$

where α is a weight parameter.

The weight vector ω, originally, is composed by all weights from hidden and output layers, so its norm is minimized at same time in both layers [49, 93]. However, this norm do not need to be minimized always in that way. Controlling the weights of one layer does not influence the norm of the weights in the other layer and we want to use this "property" to design a feature selection method. Therefore, minimizing the norm of the weights only in the output layer, does not controls the norm of weights in the hidden layer and vice-versa. Some empirical tests are made in order to see what happens to the norm of the weights vector when only the output layer is controlled.

training layers with different objective functions

For the well known two moons toy problem, 50 paretos with 21 solutions each were generated by the classical approach, minimizing MSE and controlling the norm of all weights of entire network, i.e. $\omega = \{Z; W\}$, where W is the vector of weights of the output layer and Z is the vector of weights from hidden layer neurons.

controlling norm of entire MLP

Dominated solutions were eliminated and Figure 5.6 shows the relation between the norm of weight vectors from hidden and output layers.

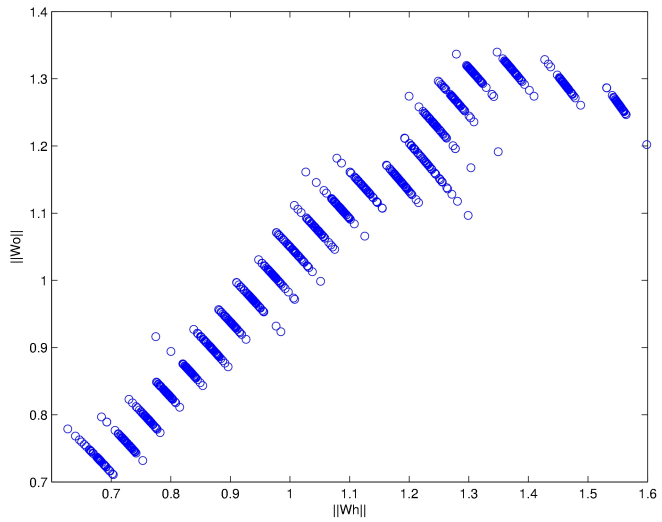


Figure 5.6: Relation between the weight vector norm from the hidden layer and from the output layer, when training a MLP controlling the norm of the weight vector from both layers.

Another 50 paretos with 21 solutions each were also generated, minimizing MSE and controlling only the norm W in the output layer and the relation between the norm of weight vectors from hidden and output layers is shown in Figure 5.7.

controlling only output layer

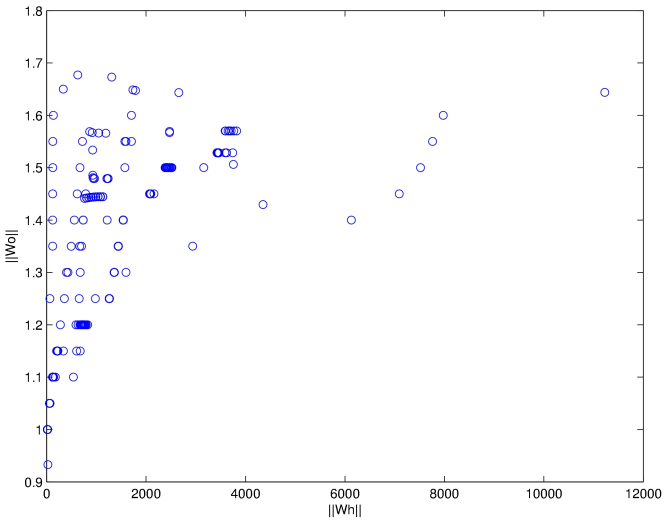


Figure 5.7: Relation between the norm of weight vector from the hidden layer and from the output layer, when training a MLP controlling only the norm of the weight vector from output layer.

As can be seen in Figure 5.7 the norm of weight vector of the hidden layer is not controlled when only the output weight vector

norm is controlled, which is a empirical confirmation that layers can be trained independently. This same principle is considered in [19] where the authors introduce a learning method based on sensitivity analysis, for a two-layer feedforward neural networks using a linear training algorithm for each one of the two layers. First the outputs from the hidden layer are set randomly plus a small error. Then it finds the weights solving individually a system of linear equations and evaluates the sum of the squared error of each layer individually.

same principle in the literature

5.3 LASSO

Minimizing the residual squared error is a way to obtain the ordinary least squares estimates (OLS), however prediction accuracy and interpretation are two main issues when dealing with OLS. OLS provides low bias and large variance, and, their estimates include all input features with different weight values, once it gives nonzero estimates to all coefficients[109], complicating feature selection task. Nevertheless if a problem has a lot of variables, it should be interesting to select a small subset of features whose relevance is strong.

improving accuracy

Prediction accuracy can be improved shrinking or even setting some weights to zero. This action will prejudice the bias, nevertheless, this will reduce the variance, improving the overall prediction accuracy. One way to control the amount of shrinkage that is applied to the estimates is to add a restriction to the minimization problem, limiting the sum of absolute values of weights, as shown in the following definition [90]:

$$\hat{\omega}\,(\mathrm{lasso}) = \arg\min_{\omega} \left\| y - \sum_{j=1}^{p} X_j \omega_j \right\|^2 + \lambda \sum_{j=1}^{p} |\omega_j| \,. \tag{5.4}$$

where ω is the vector of weights of the neural network, X is the input data, p is the number of neurons and λ is a parameter.

LASSO

If ω^* is the "full squares estimates" and $t^* = \sum |\omega^*|$ then setting $t < t^*$ will cause shrinkage of the solutions towards zero. Equation 5.4 is the definition of *least absolute shrinkage and selection operator* or LASSO.

As shown in [22], the LASSO operator applied to a multi layer perceptron neural networks may reduce or eliminate some weights and consequently some network inputs on the final solution. This characteristic may be very useful to feature selection. In a LASSO MOBJ approach the MLP network is trained minimizing the mean squared error, subject to the sum of the absolute weights being less than a constant t:

$$\min_{w} = \frac{1}{2}\sum_{n}\left(Y-\hat{Y}\right)^2 \qquad (5.5)$$
$$\text{subject to}: \sum_{i}|\omega_i| \leqslant t\ .$$

Nevertheless, LASSO has also its own drawbacks : *LASSO drawbacks*

- It has been shown that the L1 approach is able to discover the "right" sparse representation of the model, but only under certain conditions [29].
- The lasso shrinkage produces biased estimates for the large coefficients, and thus it could be suboptimal in terms of risk estimation.
- The optimal λ, in Equation 5.4 for prediction gives inconsistent variable selection results. In fact, many noise features are included in the predictive model[109].

5.3.1 *Better understanding LASSO - example*

Let us examine a synthetic classification problem composed by two Gaussian classes as shown in Figure 5.8.

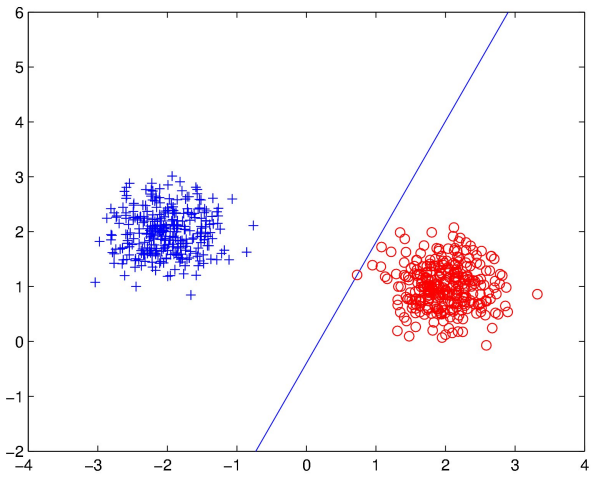


Figure 5.8: Two classes Gaussian problem

For this two-dimensional problem a hyperplane that correctly classifies the training patterns might be a line defined by

$$h = \omega^T X \qquad (5.6)$$

where ω is the weight vector of the neural network also shown in Figure 5.8.

This kind of problem can be easily solved by a single perceptron, i.e., by a single layer neural network. In this case, $\omega = \{\omega_1, \omega_2\}$ where each weight is applied to one input feature.

The MSE surface in function of weight values is shown in Figure 5.9, and Figure 5.10 shows the MSE contour in function of the weight values.

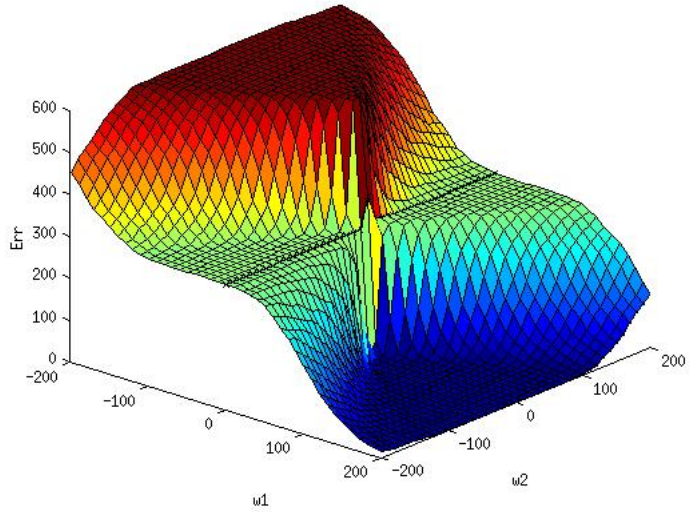


Figure 5.9: MSE surface for the Two Classes Gaussian problem

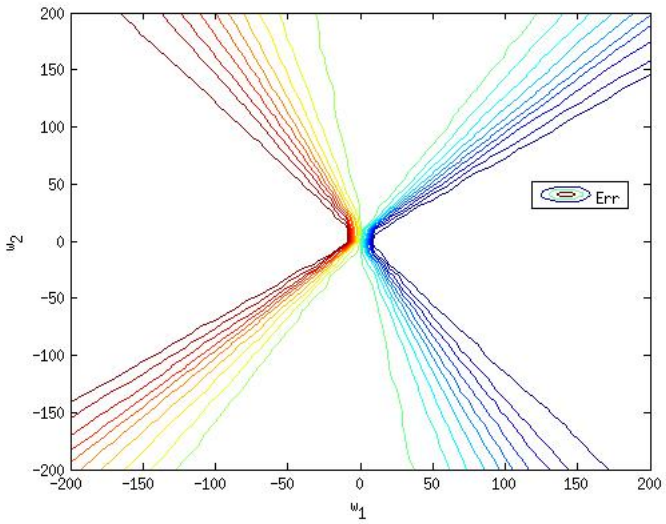


Figure 5.10: MSE contour for the Two Classes Gaussian problem

LASSO constraints

If we want to find a solution minimizing Equation 5.6, the constraints will define search areas in the weight space like shown in Figure 5.11.

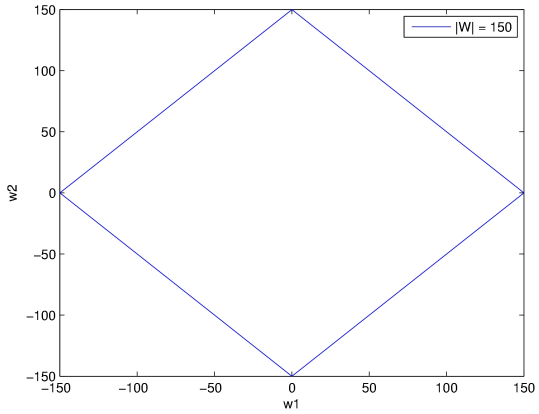


Figure 5.11: LASSO constraint contour for $|W| = 150$

Therefore, to minimize Equation 5.6 means to minimize MSE inside the region defined by the constraint $\|W\| \leqslant t$, whose t in this example is equal to 150. Figure 5.12 shows this region.

LASSO MOBJ approach

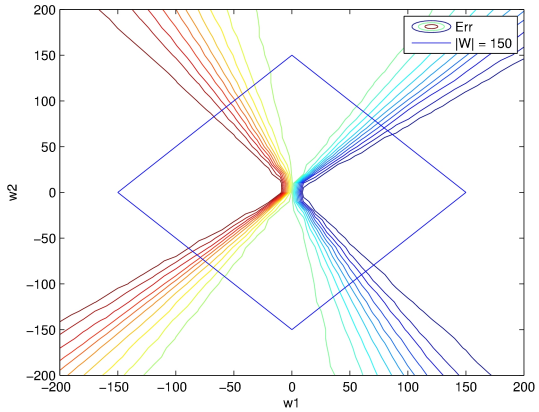


Figure 5.12: Error Surface with LASSO constraint contour

To find the solution with minimum MSE and with minimal weight norm an optimization method has to be applied, as the ellipsoidal method [11] for instance.

Inspired by the definition of LASSO a simple idea arose in the attempt to speed up the optimization step to select features. As can be seen in Figure 5.12, note that the constraint contour has corners. The solution at one corner has one of its weights equal to zero on this example. There will be as many other opportunities to have weights equal to zero as larger than two the number of weights is.

If we assume one of these weights equal to zero is possible to "walk" along these corners simplifying the optimization step. This strategy may have some utility when training, for example, the hidden layer of the MLP. Of course, the MSE on these corners may not be the smaller ones, however we just need a linearly separable mapping in the hidden

layer in order to achieve a maximal margin solution in the output layer, to solve the learning problem (see Section 5.2). If we are able to find a linearly separable mapping then, at same time, weights in the hidden layer select the best features.

Considering a n-dimensional problem, "walking" directly over the LASSO constraints corners is still possible. We only have to make all weights but one equal to zero, for a fixed norm, in order to find deterministically one corner. Doing that we also can determine directly the error at this point. Nevertheless there are, at least, two problems with this strategy:

- the problem will be treated as univariate. This is the same as evaluate any univariate relevance index between the features and the output;
- depending on the number of features and neurons we will have too many corners to evaluate, forcing the implementation of some heuristics.

Based on this idea of exploring corners of LASSO constraints to optimize MLP parameters we rewrite LASSO formulation, in Section 5.3.2, in order to better achieve the goals of solve learning and feature selection problems at same time.

5.3.2 *Re-Writing LASSO formulation*

Considering the objective of performing feature selection, there is another issue in the way how LASSO is defined in Equation 5.4:

> LASSO has more chances to produce null weights, *but not necessarily all in the same feature*.

new LASSO formulation

We are interested in selecting features while solving the learning problem. Therefore we prefer that all weights related to the irrelevant features go to zero in the hidden layer, where the selection problem is solved, and we need to minimize the norm of the weights in the output layer to find the maximal margin hyperplane. Actually we need to write LASSO formulation not in order to force any weight of the hidden layer go to zero, but in order to make all weights (from hidden layer) related to any irrelevant feature go to zero. However, instead of directly prune these weights as done in the *Optimal Cell Damage* method [20], we want minimize the objective funtcion subject to some constraints. Considering the independent training of layers we can rewrite the basic MOBJ optimization Equation 5.3 as

$$\min_{w,z} \frac{1}{2} \sum_{n} \left(Y - \hat{Y}\right)^2 + \alpha \sum \|W\|_2 + \beta \sum \|Z\|_1 \ , \tag{5.7}$$

where W is the vector of weights of output layer, Z is the vector of weights of the hidden layer and α and β are weight parameters.

The LASSO condition in the hidden layer is given adding the regularization term $\sum |Z|$. This regularization term can be rewritten as $\sum |Z| = \sum |Z_1| + \sum |Z_2| + \ldots + \sum |Z_{n_f}|$, where n_f is the number of features and $\sum |Z_i|$ is the sum of absolute values of all weights Z assigned to the feature i, in the hidden layer. The new optimization problem can be defined as:

$$\min \frac{1}{2} \sum_n \left(Y - \hat{Y}\right)^2 \tag{5.8}$$

$$\begin{aligned} \text{subjected to} : & \sum |Z_1| + \sum |Z_2| + \ldots + \sum |Z_{n_f}| \leqslant t, \\ & \sum \|W\|_2 \leqslant t_2 \end{aligned}$$

where t and t_2 are the desired norm bounds.

To perform feature selection while solving the learning problem, it is very interesting that weights assigned to irrelevant features vanish. In other words, if feature i is irrelevant we want that $\sum |Z_i| = 0$. Therefore the optimization problem can now be written as

$$\min_{w,z} \frac{1}{2} \sum_n \left(Y - \hat{Y}\right)^2 \tag{5.9}$$

$$\begin{aligned} \text{subjected to} : & \sum |Z| \leqslant t, \\ & \sum |Z_i| = 0 \; \forall \; i = 1, 2, ..., n_n \\ & \sum \|W\|_2 \leqslant t_2 \end{aligned}$$

where n_n is the number of irrelevant features and $\sum |Z_i| = |z_{i1}| + |z_{i2}| + ... + |z_{ip}|$, with p equal to the number of hidden neurons.

The way how the optimization problem is defined in 5.9 forces all weights from hidden layer that are assigned to irrelevant features to go to zero. Obviously we do not know a priori which are the irrelevant features and we cannot apply this equality constraint to all features, so the method has to decide which features apply it to.

applying equality constraints

During the training, features whose hidden layer weights are growing could be more important to the learning process than features whose weights values are smaller. As we want to select relevant features, the method can be set in order to apply the equality restrictions only on those features with small weights values, i.e., over the features whose sum of the absolute values of their hidden weights are small. To summarizing, during the optimization, those features whose sum of absolute values of their weights in the hidden layer (G) are larger,

will have their equality constraints released, while those with smaller sum of their weights will have their equality constraints held.

selecting features to apply equality constraints

One way to implement this idea and to decide on which features to apply the equality constraints, is to identify possible outliers in the range of G values, i.e, in the range of the sum of weight absolute values assigned to each feature. This can be done in two slightly different ways:

- emphasizing the elimination of irrelevant features or;
- emphasizing the selection of relevant features.

In the first case, those features whose G value is too low compared to the values of all features, will have their equality restriction applied. On the other hand, in the second case, those features whose G value is much larger than the others will have their equality constraints released. In order to define which are the outliers among the values of G, there are in the literature some tools as box plot, Dixon's test, Grubbs's test, z-score and others. A review on these methods can be found in [52].

Therefore, the steps considered in the proposed algorithm, which will be detailed in Section 5.4 are:

- input features are normalized with zero mean and unity variance;
- $G_i = \sum |z_i|$ is evaluated for each feature at each iteration;
- feature i whose G_i is an outlier will have the equality restriction activated according to the adopted emphasis, at each iteration.

5.4 MOBJ FEATURE SELECTION METHOD

Our LASSO multi-objective algorithm (LMFS) is designed to select the important features while solving the learning problem. It is implemented to train a multi-layer perceptron network composed by two layers where the selection problem is solved in the first layer and the learning problem is solved by both layers. The algorithm has four different objective functions:

- the mean squared error of the network;
- the L_1-norm of the weights in the hidden layer;
- the L_2-norm of the weights in the output layer;
- the sum of absolute values of weights assigned to the irrelevant features in the hidden layer.

The optimization problem is defined by Equation 5.9 in Section 5.3.2 and repeated here for convenience:

$$\min_{w,z} \frac{1}{2} \sum_{n} \left(Y - \hat{Y}\right)^2 \tag{5.10}$$

$$\begin{aligned} \text{subjected to} : & \sum \|W\|_2 \leqslant t_2 \\ & \sum |Z| \leqslant t, \\ & \sum |Z_i| = 0 \; \forall \; i = 1, 2, ..., n_n \end{aligned}$$

The *ellipsoidal algorithm* [11] is used to solve the optimization problem and the equality constraints are initially applied to all features and released only for those features whose G increases. The main algorithm for the proposed method is shown in Algorithm 5.

Algorithm 5: LMFS algorithm.

```
Γ = {∅}; // empty set of selected features
F = {F_1, F_2, ..., F_{n_f}}; // Set of original features
// Preparation
Normalize features;
// Ellipsoidal method
while Objective function minimum not found // Equation 5.10
do
    Run some initial iteration of optimization method without applying the
    equality constraints;
    for j = 1 to number of features remaining in F do
        Evaluate G_j = ∑ |Z_j|; // sum of absolute values of all weights
            in the hidden layer of feature j
    Identify outliers values of G;
    Apply equality constraints to all features and release these constraints for
    those features j whose G_j is an outlier with large value;
Select features whose weights in hidden layer are not zero (or that are not very
small) and assign it to Γ;
```

5.5 EXPERIMENTS

The multi-objective method developed in Section 5.4 is applied to different synthetic and real problems and the results are shown in this section. The tests are conducted in order to evaluate the efficiency of the objective functions defined in Equation 5.10, in selecting features while solving the learning problem.

The tests are performed on four real problems from UCI Machine Learning Repository [1] and two synthetic problems, as described in the following:

- *SONAR*: the sonar data set is composed by instances of a sonar response. The task is to discriminate between sonar signals

bounced off a metal cylinder and those bounced off a roughly cylindrical rock. This data set is composed by 208 instances with 60 features;

- *PEN*: the Pen-Based Handwritten Digits data set is composed by digit samples from 44 different writers. For this last problem we considered only instances of digits 6 and 9 in the experiments. It has 16 features and 2111 instances;
- *ILPD*: the Indian Liver Patient Dataset. This data set contains 416 liver patient records and 167 non liver patient records. It is composed by 10 features.
- *IONO*: the Johns Hopkins University Ionosphere database has 34 features with 351 patterns of radar returns from the ionosphere. Radar returns are classified into two classes: if there is a good return it shows the evidence of some type of structure in the ionosphere, if not their signals pass through the ionosphere.
- *XOR*: is a synthetic data set build with a exclusive-OR function. This problem has two important features (two dimensional problem), three random features and two more features that are equal to the first two plus a random noise. This data set is composed by 2.000 patterns.
- *PCIRC*: is a synthetic problem composed by two different classes with a distribution as shown in Figure 5.13. The first two features are the most relevant ones while the last two are a copy of these relevant features plus a random noise. There are also six more random features that are completely irrelevant. The problem has 2.277 patterns.

For each problem the feature selection method is repeated 10 times in a cross validation scheme. Each time the training and test sets are randomly chosen in order to avoid biased results. For each trial the selected features are assigned to the Γ feature subset that is used to train a MLP network. Features are selected according to their G values. The accuracy of this trained MLP is evaluated on the test set, and the

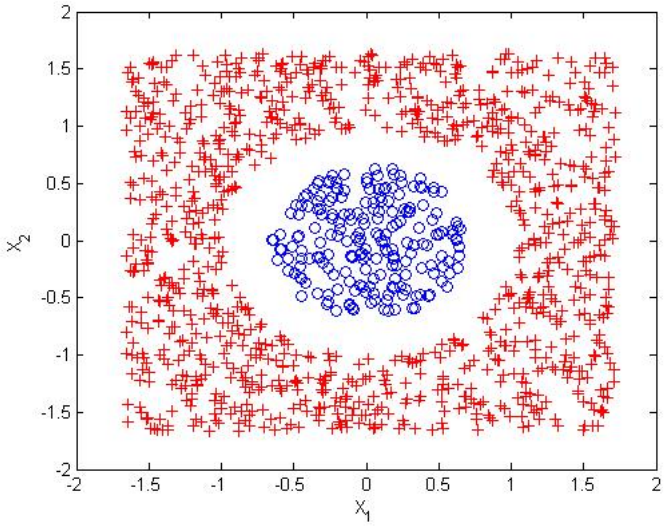


Figure 5.13: Two first features from PCIRC problem.

average results are shown in Section 5.5.1. Algorithm 6 shows the main steps of the experiments for each problem.

Algorithm 6: MOBJ experiments algorithm.

```
// Preparation
Normalize features;
Split data into 10 folds;
for each fold do
    Solve optimization problem 5.10; // Using ellipsoidal method
    Select features whose weights in hidden layer are not very small assigning
    them to Γ;
    Train a MLP using Γ subset;
    Evaluate network accuracy over the test set;
Summarize results;
```

5.5.1 *Results*

For the PCIRC problem, the parameter t of the optimization problem defined by Equation 5.10, is set to 3.5, and the chosen number of neuronis in the hidden layer is set to 5 according to previous tests. The tests to set these parameters are conducted with a classical MOBJ training method (see references in Section 5.2): a MLP is trained with several values of neurons and for several norm bound values. The parameters from the best trained network (considering the accuracy on a test set) are selected. This procedure is done for all other problems too.

Figure 5.14 shows the average sum of absolute values of weights G in the hidden layer assigned to each feature i of PCIRC problem. Note that features 1 and 2 as well as features 9 and 10 receive the larger

weights. It is natural that features 9 and 10 receive also large weights once feature 9 is equal to feature 1 plus a random noise, and feature 10 is equal to feature 2 plus a random noise. Analyzing the average values of final G values of each feature, features 1, 2, 9 and 10 are selected and assigned to the final selected feature subset Γ.

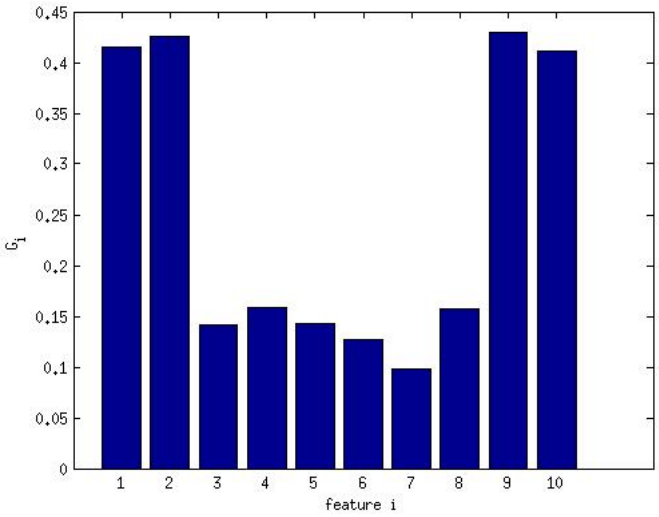


Figure 5.14: Average of the sum of absolute values of weights in the hidden layer of feature *i* for PCIRC problem. This is a synthetic problem where features 1 and 2 are the good ones and features 9 and 10 are redundant with the first two features.

The second synthetic problem evaluated is the XOR problem described before. The MLP network is set with 5 neurons in the hidden layer and with the optimization parameter $t = 4$. Figure 5.15 shows graphically the average value of G of each feature over 10 training trials. As expected, features 1,2, 6 and 7 received the larger G values in average. Features 6 and 7 are identical to features 1 and 2, and therefore they are selected together with features 1 and 2 after the analysis of results.

For the IONO problem, 23 features, in average, did not have the restriction of equality applied to its weights in the hidden layer, and are selected to compose the final Γ subset as can be seen in Figure 5.16. These 23 features are used to train the MLP network and a LDA model 10 times, using a cross validation framework, and their average accuracies on the test sets are shown in Table 5.1. The same test is performed considering only the 10 best features from Γ, i.e. the 10 features with larger G value. Results are also listed in Table 5.1

The LMFS method is applied to other three problems: SONAR, ILPD and PEN problems. For each one of them the neural network is set with 10 hidden neurons. The parameter t from the optimization problem is set as *1* for the PEN problem, as *3.5* for the ILPD problem, and as *1.5* for the SONAR problem. The average classification accuracy on

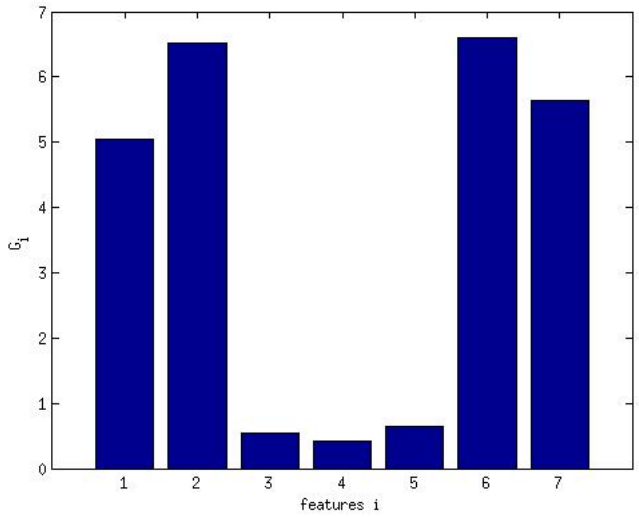


Figure 5.15: Average of the sum of absolute values of weights in the hidden layer of feature *i* for XOR problem. This is a synthetic problem where features 1 and 2 are the good ones and features 9 and 10 are redundant with the first two features.

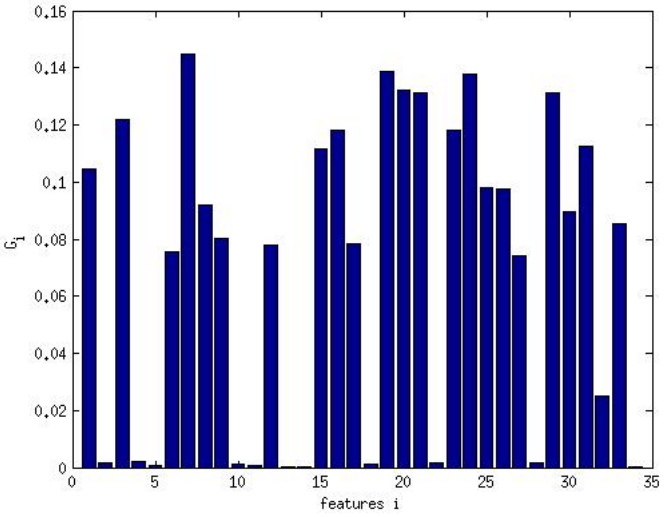


Figure 5.16: Average of the sum of absolute values of weights in the hidden layer of feature *i* for IONO problem.

Table 5.1: Average classification accuracies from models trained with subset Γ (with different number n_s of selected features) selected by LMFS method from IONO problem. n_f is the original number of features.

n_f	n_s	$Acc \pm \sigma$	
		LDA	MLP
34	23	0.8632 ± 0.0298	0.846 ± 0.0692
	10	0.8005 ± 0.0361	08630 ± 0.0425

the test set in a 10-fold cross validation test is shown in Table 5.2. This table also shows the number of selected features after analyzing the final G values.

Table 5.2: Average classification accuracies from models trained with subset Γ selected by LMFS method from problems SONAR, PEN and ILPD. n_f is the original number of features.

Problem	n_f	n_s	$Acc \pm \sigma$	
			LDA	MLP
SONAR	60	11	0.79 ± 0.08	0.77 ± 0.07
ILPD	10	5	0.61 ± 0.02	0.70 ± 0.03
PEN	16	11	0.9995 ± 0.001	0.9996 ± 0.001

It is interesting to emphasize that the LMFS method, developed in this thesis, selected the correct relevant features of all synthetic problems. Of course, as we know which are the relevant feature on these toy problems we do not have to check their accuracies on a test set. For the real problems used here Table 5.2 list the average results. Nevertheless the features selected by the LMFS method, for the SONAR problem, achieved 87% of correct classifications for a LDA model and 84% for a MLP model, many times, depending on the training set.

All results are also comparable with the results obtained by the SSFC method developed in Section 4.1.2 and with the results of previous works, where these data sets were used in a classification task. In [83] the authors achieved accuracies from 90% to 92% for a classification task using a MLP for the IONO problem, but using all 34 features. Our MOBJ method archives 86% of accuracy with a MLP model trained with the 10 first selected features, and 84% when considering the first 23 features. In [5] the authors achieved, depending on the network configuration, from 92% to 98% of accuracy over a test set using all 16 features for the PEN problem. We achieve better performances training a MLP and a LDA model using less features selected by the LMFS method: 99% for both LDA and MLP models in average. In [41] the authors achieved results with accuracy of 81% in average, for

different number of hidden neurons but considering all 60 features for the SONAR problem. Our MOBJ method selects a feature subset of 11 features whose classification accuracy reached 79% in average when training a LDA model.

In fact, which was developed in this thesis is an objective function to be minimized by an optimization method. The method chosen here was the ellipsoid one, and therefore the computational cost of this fetaure selection method is conditioned by the choice of the optimization method. The only additional cost added to the cost of the optimization method is the cost of deciding which features the equality restriction has to be applied. However this additional computational cost is much lower than the cost of the optimization method.

6

MIXED MI ESTIMATOR

Mutual information (MI) [24] has been applied to a wide range of machine learning problems [61, 73]. It is a well established approach, especially for estimating uni- and multi-variate non-linear relations, being also applied in the context of feature selection (FS) [8].

Estimating MI

In order to evaluate mutual information, densities of dependent and independent variables should be estimated. In practice, evaluating mutual information is not straightforward, since it requires *a priori* knowledge of the corresponding densities. Usually, information about densities is not fully available and a proper density estimator is needed.

Kraskov estimator

A mutual information estimator for classification problems, derived from the Kraskov estimator [60], was developed by Goméz et al. [39]. It addresses classification tasks by discretizing the output variable, and can also be applied to multi-class feature selection problems. Nevertheless, like the original Kraskov estimator, this approach is also restricted to continuous input variables. However, most real-domain applications contain not only continuous but also discrete variables, which are usually treated separately in current applications.

We propose a new estimator, based on the original Kraskov method, that is able to deal not only with continuous but also with discrete variables in order to perform Feature Selection.

As will be discussed later, the entropy of a n-bit quantization of a continuous random variable $\mathbf{X}$ is approximately the differential entropy of $\mathbf{X}$ plus n [24]. This indicates that the entropy of a discrete random variable is not comparable to the entropy of a continuous one and consequently the corresponding mutual information will not be the same too. It is noticeable the lack of a proper method to estimate MI for a set of continuous and discrete variables in the literature, and the use of an estimator that works only with one of them in problems composed by both types of variables could lead to inaccurate estimations and poor feature selection results as shown in Section 6.3.4.

entropies differences

We begin by discussing the statistical principles of this central question, which motivates the work, and by stating the differences between differential and discrete entropies. Next, a mutual information estimator for classification problems, that is able to deal with discrete and continuous variables together, is developed following the steps described in [39]. Next, it is shown with experiments that the new estimator method could improve feature selection results when used in problems composed by a mixed set of variables.

6.1 MIXED ENTROPY AND MUTUAL INFORMATION

Standard approaches for MI estimation include dynamic allocation of histograms' bins [8], recursive partitioning of the input domain [27] and kernel density estimators [84]. The Kraskov estimator [60] adopted in this work is also based on partitioning-and-counting in the input space. The density at a given point is obtained according to the distance from this point to the nearest neighbors, what makes the estimator robust to scarce datasets.

variable discretization

In order to deal with distortions that may appear in density estimations of mixed sets of variables, a strategy adopted in the literature is to discretize the continuous variables and then to apply the MI estimator [70]. However, this approach is likely to result in poor estimations due to discretization or due to continuous approximations of discrete variables. Therefore, in order to deal with this particular problem, which is frequent in most real applications, a method that is capable to handle both types of variables should be developed.

In order to emphasize the discrepancy in approximation highlighted in the previous section, let us consider a continuous random variable Z with a continuous probability density function $f(z)$. Consider now that the space of Z is discretized into fixed intervals Δ, where each interval i is defined as $[i\Delta, (i+1)\Delta[$. For each interval i, as a direct consequence of the *mean value theorem*, it is possible to find a value z_i for which

$$f(z_i)\Delta = \int_{i\Delta}^{(i+1)\Delta} f(z)dz \,. \tag{6.1}$$

A discrete random variable Z^Δ can be defined over a countable number of values z_i, being one per interval i of Z. In this case the probability p_i associated to z_i can be written on the basis of the pdf of Z as

$$p_i = f(z_i)\Delta \,. \tag{6.2}$$

discrete entropy

Cover [24] shows that the discrete entropy of the quantized variable Z^Δ is given by

$$\begin{aligned} H(Z^\Delta) &= -\sum_{-\infty}^{\infty} p_i \log p_i & (6.3)\\ &= -\sum \Delta f(z_i) \log f(z_i) - \log\Delta & (6.4) \end{aligned}$$

if $\sum f(z_i)\Delta = \int f(z) = 1$.

It can be shown [24] that the first term in Equation 6.4 tends to the integral of $-f(z)\log f(z)$ as $\Delta \to 0$, if $f(z)\log f(z)$ is Riemann integrable.

This implies that the entropy of the discrete random variable Z^Δ and the differential entropy of the continuous random variable Z relate as

$$H\left(Z^\Delta\right) + \log\Delta \to h(f) = h(Z) \text{ as } \Delta \to 0 \,; \tag{6.5}$$

See theorem 8.3.1 in [24].

It is easy to estimate entropies for discrete variables (by counts) and for continuous variables (for example, by using Kraskov estimator [60]). However, most real-world multivariate data include both continuous and discrete variables together. One way of estimating entropies on such multivariate data would be firstly to discretize the continuous variables and secondly to use a discrete estimator by counts. Nevertheless, Equation 6.5 shows that the entropies of the original continuous variables and their discretized versions do not match. A specific estimator for mixed variables is therefore necessary.

6.1.1 *Entropy of a mixed set of variables*

Given two discrete random variables X and Z , the joint entropy $H(X, Z)$ can be defined as in Equation 6.6 by the *Chain rule*: *joint entropy*

$$H(X, Z) = H(X) + H(Z \mid X) \,. \tag{6.6}$$

By definition [24], the conditional entropy $H(Z \mid X)$ is defined as

$$\begin{aligned} H(Z \mid X) &= \sum_{x \in X} p(x)\, H(Z \mid X = x) \\ &= -\sum_{x \in X} \sum_{z \in Z} p(x, z) \log p(z \mid x) \,, \end{aligned} \tag{6.7}$$

where $p(x)$ is the probability of x, $p(x, z)$ is the joint probability of x and z, and $p(z \mid x)$ is the probability of z once x is given.

If both variables are continuous then, by the *Chain rule* again, the differential joint entropy h becomes

$$h(X, Z) = h(X) + h(Z \mid X) \tag{6.8}$$

and, likewise, by definition, the equation above can be rewritten as *differential entropy*

$$\begin{aligned} h(Z \mid X) &= -\int\int f(x, z) \log f(z \mid x)\, dx dz \\ &= \int f(x)\, h(Z \mid X = x)\, dx \,, \end{aligned} \tag{6.9}$$

where $f(x)$ is the probability mass function of x, $f(x,z)$ is the joint probability mass function x and z and $f(z \mid x)$ is the probability mass function of z given x.

Considering X as discrete and Z as continuous, by analogy with Equations 6.6 and 6.8, the resulting mixed entropy $\mathcal{H}(Z,X)$ may be defined as

$$\mathcal{H}(Z,X) = H(X) + h(Z \mid X) \ . \tag{6.10}$$

It is interesting to notice that in Equation 6.10 the entropy is the sum of two different quantities: the differential entropy of a continuous variable and the discrete entropy of a discrete one. Since the random variable X is discrete the conditional differential entropy in Equation 6.10 is given by

$$h(Z \mid X) = \sum_{x \in X} p(X = x)\, h(Z \mid X = x) \ . \tag{6.11}$$

mixed entropy

Then, the mixed entropy of a discrete random variable X and a continuous one Z becomes

$$\begin{aligned} \mathcal{H}(Z,X) &= H(X) + \sum_{x \in X} p(X = x)\, h(Z \mid X = x) \\ &= H(X) - \sum_{x \in X} p(X = x) \int_S f(Z \mid X = x) \log f(Z \mid X = x)\, dz \end{aligned} \tag{6.12}$$

where S is the support set of the random variable Z.

6.1.2 *Mutual Information between a mixed set of variables and a discrete one*

Let us now consider V as a random variable set composed by a discrete random variable X and by a continuous random variable Z, such as $V = \{X \cup Z\}$. Consider also another discrete random variable Y.

mixed MI

The Mutual Information between V and Y can be defined, in terms of the mixed entropy, as:

$$\begin{aligned} MI(V,Y) &= \mathcal{H}(V) - \mathcal{H}(V \mid Y) \\ &= \mathcal{H}(Z,X) - \sum_{y \in Y} p(Y = y)\, \mathcal{H}(Z,X \mid Y = y) \ . \end{aligned} \tag{6.13}$$

Equation 6.13 can be rewritten as:

$$MI(V,Y) = H(X) + \sum_{x\in X} p(X=x)\, h(Z \mid X=x) - \sum_{y\in Y} p(Y=y) \left(H(X \mid Y=y) + \sum_{x\in X} p(X=x \mid Y=y)\, h(Z \mid X=x, Y=y) \right) \tag{6.14}$$

From Equation 6.14 the Mixed Mutual Information Estimator will be obtained in the next section.

6.2 MIXED MUTUAL INFORMATION ESTIMATOR

mixed MI estimator

An estimator of the Mixed Mutual Information can be developed by replacing the differential entropy quantities in Equation 6.14 by the Kozachenko-Leonenko entropy estimator

$$\hat{h}(Z) = -\psi(k) + \psi(N) + \log C_d + \frac{d}{N}\sum_{n=1}^{N} \log \varepsilon(n,k) \tag{6.15}$$

as presented in [39], where k is the number of nearest neighbors that should be set by the user, N is the number of patterns, d is the dimension of Z, C_d is the volume of the d-dimensional unitary sphere, $\psi(\cdot)$ is the digamma function and $\varepsilon(n,k)$ is twice the distance from z_n to its k^{th} neighbor.

Then, after some algebraic manipulations the following expression is obtained.

$$\begin{aligned} MI(V,Y) = &-\sum_{r} \frac{m_r}{N} \log\left(\frac{m_r}{N}\right) \\ &+ \sum_{r} \frac{m_r}{N}\psi(m_r) + \sum_{r} \frac{m_r}{N}\frac{d}{m_r}\sum_{z=1}^{m_r} \log \varepsilon(z,k) \\ &- \sum_{i} \frac{m_i}{N}\left(-\sum_{l=1}^{m_i} \frac{m_l}{m_i} \log\left(\frac{m_l}{m_i}\right)\right) \\ &- \sum_{i} \frac{m_i}{N}\sum_{l=1}^{m_i} \frac{m_l}{m_i}\psi(m_l) \\ &- \sum_{i} \frac{m_i}{N}\sum_{l=1}^{m_i} \frac{m_l}{m_i}\frac{d}{m_l}\sum_{j=1}^{m_l} \log \varepsilon(j,k) \end{aligned} \tag{6.16}$$

where m_i is the number of elements of X for which the elements in Y are equal to y_i, m_r is the number of elements of X that are equal to x_r, and m_l is the number of elements of X for which the elements in X are equal to x_l and the elements in Y are equal to y_i

Equation 6.16 was derived for the case of a one-dimensional variable X and a one-dimensional variable Z. This expression can be generalized for the case where $V = \{X, Z\}$, with $X = \{X_1, \ldots, X_n\}$ being a set of n discrete random variables, $Z = \{Z_1, \ldots, Z_t\}$ being a set of t continuous random variables and Y a discrete random variable:

$$\begin{aligned} MI(V,Y) = {} & H(X_1) + \sum_{g=2}^{n} p(X_1,\ldots,X_{g-1})\, H(X_g \mid X_{g-1},\ldots,X_1) \\ & - \sum_{y\in Y} p(Y=y)\,[H(X_1 \mid Y=y)] - \sum_{y\in Y} p(Y=y) \\ & \left[\sum_{g=2}^{n} p(X_1,\ldots,X_{g-1} \mid Y=y)\, H(X_g \mid X_{g-1},\ldots,X_1, Y=y)\right] \\ & + \sum_{x\in X} p(X=x)\, h(Z \mid X=x) - \sum_{y\in Y} p(Y=y) \\ & \sum_{x\in X|Y=y} p(X=x \mid Y=y)\, h(Z \mid X=x, Y=y)\ . \end{aligned} \tag{6.17}$$

The definition of Equation 6.17 as the Mutual Information Estimator for a set of continuous and discrete variables depends on the definition of the mixed entropy $\mathcal{H}$. This mixed entropy definition allows the computation of different quantities as discrete entropy and differential entropy in the same framework. This is the key point of this new MI Estimator: the ability to sum, in a proper way, discrete and differential entropies.

6.3 EXPERIMENTS

6.3.1 *Feature Selection methodology*

Selecting features by their individual relevance may lead to a suboptimal feature subset. As mentioned in [45] some features, considered irrelevant to the output when evaluated individually, could become relevant when evaluated in the presence of other features. However, in practice, testing all possible feature subsets by exhaustive search can be unfeasible depending on the dimension of the input space. Therefore, it is important to use a search strategy in order to overcome this issue: a forward-backward sequential supervised feature selection algorithm [39, 45] is implemented.

forward-backward procedure

The forward-backward procedure adopted here is the following:

- firstly the forward step is applied. During this step the selected feature subset Γ starts empty; at each iteration one feature is added. In the first iteration the MI between each individual feature from the full set F and the output labels Y is evaluated. The feature F_i from F with the highest MI with Y is added to

Γ and excluded from F. In the next iteration each one of the remaining features F_j in F is temporarily added to Γ (giving Γ_j); the MI between Γ_j and Y is evaluated. Feature F_j that, together with the previously selected subset, has the highest MI with the output labels is selected and permanently added to Γ (and excluded from F). If MI of Γ_j is greater than Γ, given the stopping criterion detailed next, then the procedure continues to the next iteration, otherwise the forward procedure is stopped.

- secondly the backward step is applied starting with the final selected feature subset Γ from the previous step. At each iteration each selected feature F_j is individually and temporarily excluded from Γ (giving Γ_j) and the MI between Γ_j and Y is evaluated. The set Γ_j with the highest MI value is selected and, if Γ_j is more relevant than Γ given a stopping criterion detailed below, then F_j is definitively excluded from Γ, otherwise the procedure is stopped.

Other forward-backward schemes could be adopted as well. For example, a backward step is sometimes performed at each iteration of the forward loop. Another possibility is to start from the full set and to apply backward steps; the advantage is that links between features are taken into account from the beginning of the procedure, the price to pay being the need for estimating high-dimensional entropies and mutual information.

6.3.2 *Stopping criterion*

Considering F_i one feature from the initial set F and Γ the actual selected feature subset, with $F_i \notin \Gamma$, the forward step used in the feature selection process tries to answer the following question: "is $\Gamma \cup F_i$ more relevant than Γ"? Since Γ has one dimension less than $\Gamma \cup F_i$, the $MI(\Gamma \cup F_i, Y)$ value can not be compared to the $MI(\Gamma, Y)$ value. In order to answer the question the permutation test idea [35, 40] is applied as follows: from the set $\Gamma \cup F_i$, the feature F_i has its elements randomly permuted forming another set $\Gamma \cup F_i^p$, where F_i^p is the permuted version of feature F_i. This permutation generates a random variable with the same distribution as F_i, but that does not have any relation with the output Y (the corresponding values of Y are not permuted). Actually, adding a random variable to set Γ, in theory, does not improve or prejudice the mutual information between Γ and Y, but it increases the dimension of the Γ set in order to make it comparable to $\Gamma \cup F_i$. Therefore, if $MI(\Gamma \cup F_i, Y) > MI(\Gamma \cup F_i^p, Y)$ then $\Gamma \cup F_i$ is more relevant than Γ. Therefore F_i can be added to Γ and the process continues. Otherwise the forward process is halted and no more features are added to Γ.

permutation test

The same principle is applied in the backward step, however in a slightly different way. As before, it is not possible to compare the result of $MI(\Gamma, Y)$ with the result of $MI(\Gamma \setminus F_i, Y)$ in order to verify if there is an increase of relevance when feature F_i is removed from Γ, because the sets have different dimensions. Permuting the feature F_i in Γ transforms this feature in a random variable with no relation with Y, thus, a set without the influence of F_i but with the same dimension of Γ is generated. Now it is possible to answer if $\Gamma \setminus F_i$ is more relevant than Γ. If $MI(\Gamma, Y) < MI((\Gamma \setminus F_i) \cup F_i^p)$, F_i can be definitively removed from Γ and the process continues, otherwise the backward process is halted.

6.3.3 *Problems*

The tests are applied to seven real datasets problems, and to a synthetic one, as described next:

- **DCbench** dataset which is specifically designed for testing the MMI estimator, since the relation between input features and the output variable is known and controlled. This dataset is composed by four discrete and six continuous features, that are sampled from different distributions. The DCbench dataset has 10.000 samples. The output results from a combination of three continuous features (X_1, X_2 and X_3) and two discrete ones (X_7 and X_8), in the following way:

$$Y = sign\left(tanh\left(X_1\right) + sin\left(X_2\right) + X_7 + X_8 + X_3\right) \ . \tag{6.18}$$

- **Boston Housing** dataset [1] is composed by 506 samples with 13 features (3 discrete and 10 continuous). Originally the output of this problem is the house prices, which is a continuous variable, however, here it is transformed into a classification problem by splitting the output into two classes: prices larger or smaller than a given threshold as in [91].

- **Page Blocks Classification** dataset [1], which is composed by 5473 samples with 10 features, being 6 discrete and 4 continuous.

- **Spambase** dataset (Spam) [1], a dataset of e-mails spams with 4601 samples composed by 55 continuous features and 2 discrete ones.

- **Multi-feature digit dataset** (Mfeat) [1] consists of features of handwritten numerals ("0" to "9") extracted from a collection of Dutch utility maps. It has 2000 samples (200 per class) with 190 continuous features and 459 discrete features.

- **Steel Plates Faults Data Set** (Steel) [1] is classified into 7 different types and has 1941 samples with 13 continuous features and 20 discrete ones.
- **Indian Liver Patient Dataset** (ILPD) [1] is composed by 579 liver patient records with 5 continuous features and 5 discrete ones.
- **KDD Cup 1999 Data** [1] from the *Third International Knowledge Discovery and Data Mining Tools Competition*. It has 15 continuous features and 26 discrete ones with 4.898.431 samples; 600 patterns were randomly sampled from the original dataset for the tests.

6.3.4 *Results*

Results are summarized in Table 6.1. Γ_{mix} is the feature subset selected using the mixed MI estimator considering all discrete and continuous features in the initial set, and Γ_{dd} is the set of selected features obtained when using a discrete MI estimator and considering all features as discrete (continuous features are discretized). Γ_d is the selected feature subset considering only discrete variables and a discrete MI estimator, and Γ_c is the set of selected features when considering only the continuous features and an exclusive continuous MI estimator. All these selected subsets are used to train a LDA classification model in a 10 fold cross-validation framework. $\overline{Acc}$ is the mean accuracy over and σ is its standard deviation. *results*

For the dcBench problem all relevant features are known. As expected, all these relevant features are correctly selected when using the MMI estimator

Table 6.1: Mean accuracy for LDA over final selected feature subset of each experiment.

Problem	LDA accuracy ($\overline{Acc} \pm \sigma$)		
	Γ_{mix}	Γ_{dd}	$\Gamma_d \cup \Gamma_c$
dcBench	0.9267±0.0082	0.8584±0.0111	0.9263±0.0080
Page Block	0.9011±0.0281	0.7955±0.0139	0.8730±0.0247
Boston	0.8440±0.0518	0.8459±0.0635	0.8341±0.0510
Spam	0.6740±0.0219	0.6727 ± 0.0206	0.6727±0.0206
MFeat	0.9661±0.0246	0.9445±0.0102	0.8072±0.0253
Steel	0.5365±0.0311	0.5322±0.0449	0.5776±0.0283
ILPD	0.5482±0.0478	0.5310±0.0348	0.5855±0.0443
KDD	0.9933±0.0111	0.9617±0.0249	0.9967 ± 0.0072

It can be observed from the results of Table 6.1 that:

- the average performance of the final set of selected features using the proposed estimator Γ_{mix} is, in half of the cases, significantly

better than the one achieved by the subset of selected variables discretizing all continuous variables and using estimator Γ_{dd} based on histograms (and in the worst case is not significantly different from the latter);

- considering continuous variables as continuous resulted in a gain of classifier performance for dcBench, Page Blocks, Mfeat, ILPD and KDD datasets;
- it can be observed that the performance achieved by Γ_{mix} is better when compared with $\Gamma_d \cup \Gamma_c$ for datasets like Page Block, MFeat and Boston;
- for dcBench, Spam and KDD the same average rates of accuracy is obtained while for Steel and ILPD datasets Γ_{mix} performed lower than $\Gamma_d \cup \Gamma_c$;
- it is also interesting to observe that the final feature subsets $\Gamma_d \cup \Gamma_c$ result, in most cases, to higher performance when compared to the performance of Γ_{dd} sets. This supports our arguments that it is worth avoiding the discretization of continuous variables. In addition, performances obtained with $\Gamma_d \cup \Gamma_c$ are comparable in average to performances obtained with Γ_{mix}; in the few cases where performances obtained with $\Gamma_d \cup \Gamma_c$ are higher (ex: Steel and ILPD), it comes from the fact that the $\Gamma_d \cup \Gamma_c$ procedure results in a higher number of selected features, because interactions between continuous and discrete variables are not taken into account in the $\Gamma_d \cup \Gamma_c$ case.

7

CONCLUSIONS

This work presented new methods for feature selection that are capable to consider sources of information from labeled as well as unlabeled data. In this semi-supervised feature selection framework we developed our research according to two distinct approaches. The first one is based on the idea of eliminating redundant features by clustering them and on the clustering homogeneity principle: near instances are more likely to be of the same class, and therefore share the same labels.

In the feature clustering approach, the SSFC method developed in Section 4.1.2 is based on a new similarity measure proposed in Section 4.1.1. This method is considered as a semi-supervised method because labeled and unlabeled data are taken into account in the similarity measure. According to the results in Section 4.1.3 the method achieves very good results, being comparable with results of past works over the same data sets. It performs well even for small number of labeled data.

The similarity measure has contributions of labeled and unlabeled data and a parameter balances these contributions. We performed the method for different values of this parameter in order to see its importance and to check if the way to set it, proposed in Section 4.1.1.1, is a good way. Actually the values of this balance parameter do not affect the simulations too much. In average the mean value of it, proposed as a more balanced choice to provide a more semi-supervised similarity measure, performs quite well as the other values on these problems. Therefore, as the proposed value for the balance parameter also allowed good performances of the selected feature subset, it is better to consider this rather than any other value, keeping a good balance between the "supervised" and "unsupervised" terms in the similarity measure proposed.

There are some points that deserve some discussion and further development in future works. In the feature selection method developed based on the proposed similarity measure (SSFC method), the relevance step needs to be improved. The way how it is implemented only perform a univariate filter, analyzing individually the relevance of each feature. It must be interesting to develop or to use some multivariate measure because features that are irrelevant alone, may be very relevant when considered together with other features. This seems to be the case in the ILPD problem analyzed in Section 4.1.3, and such change can improve the method performance.

The other point in the same method that deserves some attention is the stopping criterion used in the redundancy step. The implemented idea is to stop the clustering process only when the similarity measure has no more significance. The results show that this criterion performed very well, however we believe that it can be improved. The fact that the similarity value is not a "zero" of the estimator does not necessarily means that it is big enough in order to be consider two features as similar. Therefore a new way to set a threshold has to be developed.

Still in the feature clustering approach, in Section 4.2 another feature selection method based on the principle of homogeneity between labels and data clusters is proposed. According to this principle the label distribution is consistent and coherent with the distribution of data. In that sense, estimation of data clusters can provide some hints about the posterior label distribution. Therefore, features that are relevant to labels are also relevant to data distribution and, consequently to clusters. The results show that information retrieved from clusters can improve the estimation of feature relevance and of feature selection tasks, specially when the number of labeled data is too small and the unlabeled data is numerous.

This principle may be applied also to the SSFC method proposed in Section 4.1.2, in order to provide more "labeled" data to the *supervised* term of the proposed similarity measure.

In the multi-objective approach, in Section 5.4 the proposed LMFS method is based on the fact that layers in a MLP are independent. While the hidden layer maps the input space to a higher dimensional one in order to provide a higher chance to make classes linearly separable, the output layer provides the separation hyperplane. Based on this principle we proposed a new way to train MLPs, which is based in the minimization of the mean squared error, the euclidean norm of weights of the output layer and the L_1-norm of weights of hidden layer neurons. In order to obtain a linearly separable mapping to solve the learning problem, the weights are updated so that the magnitude of those weights assigned to relevant features are increased and the magnitude of those assigned to irrelevant features are decreased. The use of LASSO operator in a optimization problem provides a higher probability to have null weights. We rewrite the LASSO formulation in order to force the weights associated to irrelevant variables to be minimized. The results show that this method works very well, yielding results that are comparable to those found in the literature. It is worth noting that LASSO operator has some drawbacks, as listed in Section 5.3, and these issues can limit the use of this formulation in some problems.

The LMFS method is a supervised method and we are still studying a way to consider the unlabeled data on its framework. The principle of homogeneity between labels and data clusters discussed in Section

4.2 may be used here in order to provide more "labeled" data from unlabeled data set, and then, turn this method into a semi-supervised one.

For the MOBJ experiments we chose the ellipsoidal method to solve the optimization problem, and particularly we had some problems with the implementation of this algorithm. Other implementations of the ellipsoidal method or other methods can be used too in order to improve the selection results.

Finally the idea of the independence of layers in a MLP explored on Section 5.2 provides a new way to train these kind of neural network. The layers are trained independently with different objective functions. Training a neural network considering different norms for each layer makes it possible to work with 1-norm in the hidden layer, what is more consistent with the selection task.

It is worth noting that all these methods are designed to work with a small number of labeled instances (except the selection method based on multi-objective approach that is still just supervised). This detail makes comparison with results from other feature selection methods very difficult, even impossible to be done. The best solution to do this comparison would be to implement each one of the methods found in the literature and then perform simulations under the same test conditions applied the methods we have developed, but this could not be done.

Our final conclusion is that all three methods can, and should be used as feature selection methods. Among the three methods, the first one, based on the clustering approach, despite running well over the experiments, has the highest computational cost and can produce better results if some improvement are done. If the stopping criterion of the relevance elimination step is improved we are confident that the results will be even better. The same is true for the criterion used in second step which irrelevant features are removed. Up to this moment it is univariate.

From the three methods of feature selection developed here, The ideia behind the method based on cluster homogeneity is the most simplest and promising one. With a very low computational cost this idea can probably be implemented in most existent supervised methods of feature selection. In turn, the selection method developed based on the MOBJ approach is the one of the three which can be further developed so far. Its objective function still needs to be developed to address the data that are not labeled but it already shows great results as a supervised method.

CONTRIBUTIONS

We worked in two different approaches in order to develop and propose new semi-supervised features selection methods. We can highlight the following direct and indirect contributions:

- developement of a similarity measure that take into account even the labeled and unlabeled data;
- developement of a new semi-supervised feature select method based on the proposed similarity measure;
- developement of a second semi-supervised feature selection method based on the principle of homogeneity between labels and data clusters;
- developement of a new feature selection method in the MOBJ approach that is also a new way to train a MLP, where the different layers are trained independently with different objective functions;
- as an indirect contribution, developement of a Mutual Information estimator able to consider both discrete and continuous variables in the same data set.

FUTURE WORK

As a suggestion of future work we can highlight:

- the implementation of a better ellipsoidal algorithm in the LMFS method;
- the use of other optimization methods in the LMFS method;
- the elimination of the L_1-norm constraint from the training of the hidden layer, once for the feature selection task the equality L_1-norm constraint is the most important.
- introduce the principle of homogeneity between labels and data clusters in the LMFS method turning it into a semi-supervised method;
- search for other ways and objective functions in order to consider unlabeled data in the multi-objective approach;
- use the same principle of homogeneity between labels and data clusters in the SSFC method;

"Just keep swimming!!"

(Finding Nemo)

BIBLIOGRAPHY

[1] Uci machine learning repository. URL http://archive.ics.uci.edu/ml/.

[2] *Semi-supervised Discriminant Analysis*, 2007. doi: 10.1109/ICCV.2007.4408856. URL http://dx.doi.org/10.1109/ICCV.2007.4408856.

[3] Adeola S.Oladele. Adetunmbi A.Olusola. and Daramola O.Abosede. Analysis of kdd 99 intrusion detection dataset for selection of relevance features. volume 1, 2010.

[4] I. Ahmad and Pi-Erh Lin. A nonparametric estimation of the entropy for absolutely continuous distributions (corresp.). *IEEE Trans. Inf. Theor.*, 22(3):372–375, sep 2006. ISSN 0018-9448. doi: 10.1109/TIT.1976.1055550. URL http://dx.doi.org/10.1109/TIT.1976.1055550.

[5] Fevzi Alimoglu and Ethem Alpaydin. Methods of combining multiple classifiers based on different representations for pen-based handwritten digit recognition. In *Proceedings of the Fifth Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN 96*, 1996.

[6] George B. Arfken and Hans J. Weber. *Mathematical Methods for Physicists, Sixth Edition: A Comprehensive Guide*. Academic Press, 6 edition, jul 2005. ISBN 0120598760. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0120598760.

[7] M. N. Rozhkova A.V. Ivanov. Properties of the statistical estimate of the entropy of a random vector with a probability density. *Problems of Information Transmission*, 17:171–178, 1981.

[8] R. Battiti. Using mutual information for selecting features in supervised neural net learning. *Neural Networks, IEEE Transactions on*, 5(4):537 –550, jul 1994. ISSN 1045-9227. doi: 10.1109/72.298224.

[9] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, volume 14, pages 585–591, 2002. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.9400.

[10] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.*, 7:2399–2434, 2006. ISSN 1532-4435.

[11] Robert G. Bland, Donald Goldfarb, and Michael J. Todd. The ellipsoid method: A survey. Technical report, Ithaca, NY, USA, 1980.

[12] Alselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. *Inf. Process. Lett.*, 24(6):377–380, apr 1987. ISSN 0020-0190. doi: 10.1016/0020-0190(87)90114-1. URL http://dx.doi.org/10.1016/0020-0190(87)90114-1.

[13] Christos Boutsidis, Michael W. Mahoney, and Petros Drineas. Unsupervised feature selection for principal components analysis. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 61–69, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4. doi: http://doi.acm.org/10.1145/1401890.1401903. URL http://doi.acm.org/10.1145/1401890.1401903.

[14] A Braga. Class notes in multi-objective learning.

[15] AntÃ´nio Braga, Ricardo Takahashi, Marcelo Costa, and Roselito Teixeira. Multi-objective algorithms for neural networks learning. In Yaochu Jin, editor, *Multi-Objective Machine Learning*, volume 16 of *Studies in Computational Intelligence*, pages 151–171. Springer Berlin / Heidelberg, 2006.

[16] A.P. Braga. *Redes Neurais Artificiais. Teoria e aplicações.* LTC Editora, Rio de Janeiro, 2000.

[17] Deng Cai, Xiaofei He, Kun Zhou, Jiawei Han, and Hujun Bao. Locality sensitive discriminant analysis. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 708–713, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc. URL http://portal.acm.org/citation.cfm?id=1625275.1625389.

[18] Deng Cai, Chiyuan Zhang, and Xiaofei He. Unsupervised feature selection for multi-cluster data. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 333–342, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0055-1. doi: http://doi.acm.org/10.1145/1835804.1835848. URL http://doi.acm.org/10.1145/1835804.1835848.

[19] Enrique Castillo, Bertha Guijarro-Berdiñas, Oscar Fontenla-Romero, and Amparo Alonso-Betanzos. A very fast learning

method for neural networks based on sensitivity analysis. *J. Mach. Learn. Res.*, 7:1159–1182, dec 2006. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1248547.1248589.

[20] Tautvydas Cibas, Franoise Fogelman SouliÃ©, Patrick Gallinari, and Sarunas Raudys. Variable selection with optimal cell damage. In *In Proceedings of ICANN'94*, pages 727–730. Springer-Verlag, 1994.

[21] Frederico COELHO, AntÃ´nio de PÃ¡dua BRAGA, RenÃ© NATOWICZ, and Roman ROUZIER. Semi-supervised model applied to the prediction of the response to preoperative chemotherapy for breast cancer. In *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, july 2010. doi: 10.1007/s00500-010-0589-8.

[22] M.A. Costa and A.P. Braga. Optimization of neural networks with multi-objective lasso algorithm. In *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pages 3312 – 3318, july 2006.

[23] Marcelo Azevedo Costa, Antônio de Pádua Braga, and Benjamin Rodrigues de Menezes. Improving generalization of mlps with sliding mode control and the levenberg-marquardt algorithm. *Neurocomput.*, 70(7-9):1342–1347, 2007. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2006.09.003.

[24] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991. ISBN 0-471-06259-6.

[25] T.M. Cover. Geometrical and statistical properties of systems of linear inequalities with applications in patter recognition. *IEEE Transactions on Eletronic Computers, vol. EC-14,pp.326-334*, 1965.

[26] Yann Le Cun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems*, pages 598–605. Morgan Kaufmann, 1990.

[27] G.A. Darbellay and I. Vajda. Estimation of the information by an adaptive partitioning of the observation space. *Information Theory, IEEE Transactions on*, 45(4):1315 –1321, may 1999. ISSN 0018-9448. doi: 10.1109/18.761290.

[28] Gert Dijck and Marc M. Hulle. Speeding up feature subset selection through mutual information relevance filtering. In *Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, PKDD 2007, pages 277–287, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-74975-2. doi: 10.1007/978-3-540-74976-9_27. URL http://dx.doi.org/10.1007/978-3-540-74976-9_27.

[29] David L. Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via â1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, mar 2003. ISSN 1091-6490. doi: 10.1073/pnas.0437847100. URL http://dx.doi.org/10.1073/pnas.0437847100.

[30] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000. ISBN 0471056693.

[31] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[32] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals Eugen.*, 7:179–188, 1936.

[33] Alexandre F. Fonseca, Marcus V. Mesquita, Aurea R. Vasconcellos, and Roberto Luzzi. Informational–statistical thermodynamics of a complex system. *The Journal of Chemical Physics*, 112(9):3967–3979, 2000. doi: 10.1063/1.481000. URL http://link.aip.org/link/?JCP/112/3967/1.

[34] D. François, F. Rossi, V. Wertz, and M. Verleysen. Resampling methods for parameter-free and robust feature selection with mutual information. *Neurocomput.*, 70(7-9):1276–1288, mar 2007. ISSN 0925-2312. doi: 10.1016/j.neucom.2006.11.019. URL http://dx.doi.org/10.1016/j.neucom.2006.11.019.

[35] D. FranÃ§ois, V. Wertz, and M. Verleysen. The permutation test for feature selection by mutual information. In *in: ESANN 2006, European Symposium on Artificial Neural Networks*, pages 239–244, 2006.

[36] Jerome H. Friedman. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1–67, 1991. ISSN 00905364. doi: 10.2307/2241837. URL http://dx.doi.org/10.2307/2241837.

[37] Jerome H. Friedman, Werner Stuetzle, and Anne Schroeder. Projection pursuit density estimation. *Journal of the American Statistical Association*, 79(387):599–608, 1984. doi: 10.1080/01621459.1984.10478086.

[38] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Comput.*, 4(1):1–58, jan 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.1.1. URL http://dx.doi.org/10.1162/neco.1992.4.1.1.

[39] Vanessa Gómez-Verdejo, Michel Verleysen, and Jérôme Fleury. Information-theoretic feature selection for functional data classification. *Neurocomput.*, 72:3580–3589, October 2009. ISSN

0925-2312. doi: 10.1016/j.neucom.2008.12.035. URL http://portal.acm.org/citation.cfm?id=1609191.1609261.

[40] P Good. Permutation tests. *Technometrics*, 43(June):114–114, 2008. ISSN 00401706. doi: 10.1198/tech.2001.s575. URL http://pubs.amstat.org/doi/abs/10.1198/tech.2001.s575.

[41] Paul R. Gorman and Terrence J. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1(1):75–89, 1988.

[42] Isabelle Guyon and Creston Road. *Practical Feature Selection : from Correlation to Causality*, pages 1–17. IOS Press, 2008.

[43] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002. ISSN 0885-6125. URL http://dx.doi.org/10.1023/A:1012487302797. 10.1023/A:1012487302797.

[44] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lotfi A. Zadeh. *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 3540354875.

[45] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lotfi A. Zadeh. *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 3540354875. URL http://portal.acm.org/citation.cfm?id=1208773.

[46] Laszlo Gyorfi and Edward C. van der Meulen. Density-free convergence properties of various estimators of entropy. *Computational Statistics & Data Analysis*, 5(4):425–436, September 1987. URL http://ideas.repec.org/a/eee/csdana/v5y1987i4p425-436.html.

[47] Laszlo Gyorfi and Edward C. van der Meulen. An entropy estimate based on a kernel density estimation. *Limit Theorems in Probability and Statistics*, pages 229–240, 1989.

[48] Peter Hall and Sally Morton. On the estimation of entropy. *Annals of the Institute of Statistical Mathematics*, 45(1):69–88, March 1993. URL http://ideas.repec.org/a/spr/aistmt/v45y1993i1p69-88.html.

[49] S. Haykin. *Redes Neurais: PrincÃpios e PrÃ¡tica*. Bookman, 2001.

[50] Xiaofei He and Partha Niyogi. Locality preserving projections. Cambridge, MA, 2004. MIT Press. URL http://books.nips.cc/nips16.html.

[51] Xiaofei He, Deng Cai, and Partha Niyogi. Laplacian score for feature selection. In *In NIPS*, volume 17, 2005.

[52] Victoria Hodge and Jim Austin. A Survey of Outlier Detection Methodologies. *Artif. Intell. Rev.*, 22(2):85–126, oct 2004. ISSN 0269-2821. doi: 10.1023/B:AIRE.0000045502.10941.a9. URL http://dx.doi.org/10.1023/B:AIRE.0000045502.10941.a9.

[53] Yi Hong, Sam Kwong, Yuchou Chang, and Qingsheng Ren. Unsupervised feature selection using clustering ensembles and population based incremental learning algorithm. *Pattern Recogn.*, 41:2742–2756, September 2008. ISSN 0031-3203. doi: 10.1016/j.patcog.2008.03.007. URL http://portal.acm.org/citation.cfm?id=1379924.1380377.

[54] Chenping Hou, Changshui Zhang, Yi Wu, and Feiping Nie. Multiple view semi-supervised dimensionality reduction. *Pattern Recogn.*, 43(3):720–730, 2010. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2009.07.015.

[55] Marcus Hutter and Marco Zaffalon. Distribution of mutual information from complete and incomplete data. *Computational Statistics and Data Analysis*, 48:633–657, 2004.

[56] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2nd edition, oct 2002. ISBN 0387954422.

[57] S. M. Kay. *Intuitive Probability and Random Processes using MATLAB*. Springer, 2006.

[58] Kenji Kira and Larry A. Rendell. A practical approach to feature selection. In *ML92: Proceedings of the ninth international workshop on Machine learning*, pages 249–256, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc. ISBN 15586247X. URL http://portal.acm.org/citation.cfm?id=142034.

[59] L. F. Kozachenko and Leonenko. N. n. sample estimate of entropy of a random vector. *Problems of Information Transmission*, 23:95–101, 1987.

[60] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 69(6 Pt 2), June 2004. ISSN 1539-3755. URL http://view.ncbi.nlm.nih.gov/pubmed/15244698.

[61] C. Krier, D. François, F. Rossi, and M. Verleysen. Feature clustering and mutual information for the selection of variables in spectral data. *Neural Networks*, pages 25–27, 2007.

[62] Jian Li. Semi-supervised feature selection under logistic I-RELIEF framework. *2008 19th International Conference on Pattern Recognition*, pages 1–4, dec 2008. ISSN 1051-4651. doi: 10.1109/ICPR.2008.4761687. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4761687.

[63] Ming Li and Paul M.B. Vitnyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Publishing Company, Incorporated, 3 edition, 2008. ISBN 0387339981, 9780387339986.

[64] E. Llobet, O. Gualdron, J. Brezmes, X. Vilanova, and X. Correig. An unsupervised dimensionality-reduction technique. In *Sensors, 2005 IEEE*, 30 2005.

[65] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, mar 1982. ISSN 0018-9448. doi: 10.1109/TIT.1982.1056489. URL http://dx.doi.org/10.1109/TIT.1982.1056489.

[66] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[67] G. A. Miller. Note on the bias of information estimates, 1955.

[68] Pabitra Mitra, C. A. Murthy, and Sankar K. Pal. Unsupervised feature selection using feature similarity. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(3):301–312, 2002. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.990133.

[69] Latifa Oukhellou, Patrice Aknin, Hervé Stoppiglia, and Gérard Dreyfus. Application to the Classification of Non Destructive Testing Signatures. In *European SIgnal Processing COnference*, 1998.

[70] Hanchuan Peng, Fuhui Long, and C. Ding. 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, aug. 2005. ISSN 0162-8828. doi: 10.1109/TPAMI.2005.159.

[71] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1992. ISBN 0-521-43108-5.

[72] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1992. ISBN 0-521-43108-5.

[73] Ianisse Quinzan, Jose M. Sotoca, and Filiberto Pla. Clustering-based feature selection in semi-supervised problems. *Intelligent Systems Design and Applications, International Conference on*, 0:535–540, 2009. doi: http://doi.ieeecomputersociety.org/10.1109/ISDA.2009.211.

[74] R. H. C. Takahashi R. A. Teixeira, A. P. Braga and R. R. Saldanha. Improving generalization of mlps with multi-objective optimization. *Neurocomputing*, 35(1–4):189–194, 2000.

[75] Bendi Venkata Ramana, M.Surendra Prasad Babu, and N. B. Venkateswarlu. A critical study of selected classification algorithms for liver disease diagnosis. *International Journal of Database Management Systems*, 3:101–114, 2011.

[76] C.R. Rao. *Linear Statistical Inference and Its Applications*. 1965. URL http://books.google.com.br/books?id=UOajksS-ZI4C.

[77] Jiangtao Ren, Zhengyuan Qiu, Wei Fan, Hong Cheng, and Philip S. Yu. Forward semi-supervised feature selection. In *PAKDD'08: Proceedings of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining*, pages 970–976, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-68124-8, 978-3-540-68124-3.

[78] Sam T. Roweis and Lawrence K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000. doi: 10.1126/science.290.5500.2323. URL http://www.sciencemag.org/cgi/content/abstract/290/5500/2323.

[79] D.E.; Rumelhart and J.L. McClelland. *Parallel Distributed Processing, vol1: Foundations.* The MIT Press, 1986.

[80] D.W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley, 1992. ISBN 9780471547709. URL http://books.google.com.br/books?id=7crCUS_F2ocC.

[81] C. E. Shannon. A mathematical theory of communication. *Bell system technical journal*, 27, 1948.

[82] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000. URL http://citeseer.ist.psu.edu/shi97normalized.html.

[83] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, pages 262–266, 1989.

[84] R. Steuer, J. Kurths, C. O. Daub, J. Weise, and J. Selbig. The mutual information: Detecting and evaluating dependencies between variables. *Bioinformatics*, 18(suppl 2):S231–S240, October 2002. ISSN 1367-4803. doi: 10.1093/bioinformatics/18.suppl_2.S231. URL http://dx.doi.org/10.1093/bioinformatics/18.suppl_2.S231.

[85] Hervé Stoppiglia, Gérard Dreyfus, Rémi Dubois, and Yacine Oussar. Ranking a random feature for variable and feature selection. *Journal of Machine Learning Research*, 3:1399–1414, 2003. ISSN 1532-4435.

[86] Y. Sun, S. Todorovic, and S. Goodison. A feature selection algorithm capable of handling extremely large data dimensionality. In *Proceedings of the 8th SIAM International Conference on Data Mining*, pages 530–540, 2008. URL http://www.siam.org/proceedings/datamining/2008/dm08.php.

[87] F.P. Tarasenko. On the evaluation of an unknown probability density function, the direct estimation of the entropy from independent observations of a continuous random variable, and the distribution-free entropy test of goodness-of-fit. *Proceedings of the IEEE*, 56(11):2052 – 2053, nov. 1968. ISSN 0018-9219. doi: 10.1109/PROC.1968.6784.

[88] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000. doi: 10.1126/science.290.5500.2319. URL http://www.sciencemag.org/cgi/content/abstract/290/5500/2319.

[89] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.7574.

[90] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.

[91] Ferdinand van der Heijden, Robert Duin, Dick de Ridder, and David M. J. Tax. *Classification, Parameter Estimation and State Estimation: An Engineering Approach Using MATLAB*. Wiley, 1 edition, nov 2004. ISBN 0470090138.

[92] Gert Van Dijck and Marc M. Van Hulle. Speeding up the wrapper feature subset selection in regression by mutual information relevance and redundancy analysis. In *Proceedings of the 16th international conference on Artificial Neural Networks - Volume Part*

I, ICANN'06, pages 31–40, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-38625-4, 978-3-540-38625-4. doi: 10.1007/11840817_4. URL http://dx.doi.org/10.1007/11840817_4.

[93] C. Vapnik, V.N.; Cortes. Support vector networks. *Machine learning, vol.20, pp273-297.*, 1995.

[94] V.N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag: New York, 1995.

[95] Roy Varshavsky, Assaf Gottlieb, Michal Linial, and David Horn. Novel Unsupervised Feature Filtering of Biological Data. *Bioinformatics*, 22(14):e507–513, July 2006. ISSN 1460-2059. doi: 10.1093/bioinformatics/btl214. URL http://dx.doi.org/10.1093/bioinformatics/btl214.

[96] Michel Verleysen. *Learning high-dimensional data*, pages 141–162. IOS Press, Amsterdam, 2003.

[97] José R. Villar, María R. Suárez, Javier Sedano, and Felipe Mateos. Unsupervised feature selection in high dimensional spaces and uncertainty. In *Proceedings of the 4th International Conference on Hybrid Artificial Intelligence Systems*, HAIS '09, pages 565–572, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-02318-7. doi: http://dx.doi.org/10.1007/978-3-642-02319-4_68. URL http://dx.doi.org/10.1007/978-3-642-02319-4_68.

[98] J. H. Ward. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301): 236–244, 1963. ISSN 01621459. doi: 10.2307/2282967. URL http://dx.doi.org/10.2307/2282967.

[99] Zenglin Xu, Rong Jin, Michael R. Lyu, and Irwin King. Discriminative semi-supervised feature selection via manifold regularization. In *IJCAI'09: Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1303–1308, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

[100] Liming Yang and Laisheng Wang. Simultaneous feature selection and classification via semi-supervised models. *International Conference on Natural Computation*, 1:646–650, 2007. doi: http://doi.ieeecomputersociety.org/10.1109/ICNC.2007.666.

[101] Xin Yang, Haoying Fu, Hongyuan Zha, and Jesse Barlow. Semi-supervised nonlinear dimensionality reduction. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 1065–1072, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: http://doi.acm.org/10.1145/1143844.1143978.

[102] Hong Zeng and Yiu ming Cheung. Iterative feature selection in gaussian mixture clustering with automatic model selection.

In *in: Proceedings of the International Joint Conference on Neural Networks*, pages 2277–2282, 2007.

[103] Daoqiang Zhang, Zhi-hua Zhou, and Songcan Chen. Semi-Supervised Dimensionality Reduction. In *SIAM Conference on Data Mining (SDM)*, pages 629–634, 2007.

[104] Zhenyue Zhang and Hongyuan Zha. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *SIAM Journal of Scientific Computing*, 26:313–338, 2002.

[105] Jidong Zhao, Ke Lu, and Xiaofei He. Locality sensitive semi-supervised feature selection. *Neurocomput.*, 71(10-12):1842–1849, 2008. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2007.06.014.

[106] Z Zhao and H Liu. Semi-supervised Feature Selection via Spectral Analysis. In *SDM*, 2007.

[107] ErHeng Zhong, Sihong Xie, Wei Fan, Jiangtao Ren, Jing Peng, and Kun Zhang. Graph-based iterative hybrid feature selection. *Data Mining, IEEE International Conference on*, 0:1133–1138, 2008. ISSN 1550-4786. doi: http://doi.ieeecomputersociety.org/10.1109/ICDM.2008.63.

[108] Xiaojin Zhu. *Semi-supervised learning with graphs*. PhD thesis, Pittsburgh, PA, USA, 2005. AAI3179046.

[109] Hui Zou. The Adaptive Lasso and Its Oracle Properties. *Journal of the American Statistical Association*, 101(476):1418–1429, December 2006.

COLOPHON

This thesis was typeset with LaTeX 2ε using *Classic Thesis* available for LaTeX via CTAN as "classicthesis".

Final Version as of March 5, 2013 at 14:33.

DECLARATION

"I do not know everything about everything, I just know few things about almost anything..." *(Man in Black)*

Brazil - Belgium, March 2013

Frederico Gualberto Ferreira Coelho