POLITECNICO DI MILANO - Polo Territoriale di Como
Scuola di Ingegneria Industriale e dell'Informazione
Master of Science in Computer Engineering

Feature Selection for Microarray Classification Problems

Supervisor: Prof. Luigi Piroddi
Co-Supervisor: Dott.ssa Aida Brankovic

Candidate:
Marjan Hosseini, 833336

Academic Year 2017-2018

Contents

List of Figures

List of Tables

Abstract

DNA microarray datasets are characterized by a large number of features with very few samples, which is a typical cause of overfitting and poor generalization in the classification task. In this thesis we introduce a novel feature selection (FS) approach which employs the distance correlation (dCor) as a criterion for evaluating the dependence of the class on a given feature subset. The dCor index provides a reliable dependence measure among random vectors of arbitrary dimension, without any assumption on their distribution. Moreover, it is sensitive to the presence of redundant terms. The proposed FS method is based on a probabilistic representation of the feature subset model, which is progressively refined by a repeated process of model extraction and evaluation. A key element of the approach is a distributed optimization scheme based on a vertical partitioning of the dataset, which alleviates the negative effects of its unbalanced dimensions. The proposed method has been tested on several microarray datasets, resulting in quite compact and accurate models obtained at a reasonable computational cost.

Italian Abstract

I dataset di microarray di DNA sono caratterizzati da un grande numero di features e pochi campioni che sono la tipica causa di overfitting e povera generalizzazione nei processi di classificazione. In questa tesi introduciamo un nuovo metodo di features selection (FS) che utilizza l'indice di distance correlation (dCor) come criterio per valutare la dipendenza di una classe rispetto un gruppo di features dato. L'indice dCor fornisce una misura affidabile di dipendenza rispetto vettori casuali con dimensioni arbitrarie, senza nessuna assunzione sulla loro distribuzione. Inoltre, l'indice è sensibile alla presenza di termini ridondanti. Il metodo di FS proposto si basa sulla rappresentazione probabilistica del modello del sottoinsieme di features, che è progressivamente migliorato con un processo ripetitivo di estrazione del modello e valutazione. Un elemento chiave del metodo è uno schema di ottimizzazione distribuito basato sul partizionamento verticale del dataset, che riduce gli effetti negativi delle sue dimensioni sbilanciate. Il metodo proposto è stato testato su numerosi dataset di microarray, risultando in un modello compatto e accurato ottenuto con un costo computazionale ragionevole.

Chapter 1

Introduction

The high dimensional nature of bioinformatic data poses a severe challenge on machine learning methods. For example, microarrays allow to simultaneously measure the expression levels of a large number of genes, so that the resulting datasets are characterized by a large number of features (more than 50 thousand genes) and a very limited sample size [64]. Most of the genes provide little or no information useful for classification purposes, and it is particularly important to detect the smallest subset of features (referred to as *biomarkers*), that provide sufficient information to separate the classes represented in the dataset (which could distinguish cancerous and noncancerous samples, or identify different types of cancer [34]). This crucial task is referred to as feature selection (FS), which is a combinatorial optimization problem aiming at selecting from a set of available features only the relevant ones, in order to build a classifier with the required performance. FS reduces the computational cost of the classifier design and simplifies its structure, thus facilitating model interpretation and data understanding, and ultimately improving both accuracy and robustness of the designed classifier [16]. Indeed, the presence of redundant features may adversely affect the classification accuracy, as they can add more noise than useful information [42].

The highly unbalanced dimensions of microarray datasets greatly complicate the FS task, and unsatisfactory classification performances are often reported with standard methods [37], [1]. Indeed, large feature vectors significantly slow down the learning process, since the complexity of the FS problem grows exponentially with the number of features. At the same

time, the small number of samples may cause the classifier to overfit the training data, thus compromising model generalization [42]. In addition, microarray data are often affected by noise, which further aggravates the analysis. For all these reasons, specialized FS techniques must be developed to appropriately handle this type of datasets.

FS methods can be characterized as filter, wrapper or embedded methods. Filter methods select features based only on data-related properties, *i.e.* independently of the classifier design. Wrapper methods are more costly but potentially more accurate than filter-based ones, as they condition the FS process to the performance of the resulting classifier. Finally, embedded methods combine the benefits of both explained approaches: a feature screening is initially performed using a filter-based approach, followed by the application of a wrapper method to refine the final solution. In the following we focus on filter methods, which are the predominant choice in microarray problems. Indeed, the added cost of classifier design may be significant for large size problems. In addition, the classifier bias resulting from the relatively small number of samples can negatively affect the FS process [44].

Univariate filter methods are a common choice in view of their computational advantages. These methods are based on individual feature assessment, *i.e.* they rank the features based on their individual capabilities to discriminate among the classes. Once the features have been ranked, the top ones in the ranking are selected, according to some criterion. As interactions among features (in our case, the correlations among genes) are not taken into account, it is not infrequent that redundant terms might be selected in this way [45]. Furthermore, features that are individually not significant are discarded, although they may actually reveal strong discriminatory power in combination with others [42].

Multivariate filter methods overcome this problems by evaluating *subsets* of features according to some scoring function. Multivariate methods pose greater complexity than univariate ones in that, besides requiring a method to evaluate groups of features, they also involve a search mechanism in the space of all possible feature subsets. Regarding the first issue, many works employ correlation-oriented criteria based on the concept of mutual information (MI). Indeed, the MI between the features and the output reveals their discriminating capabilities, whereas the correlation among features indicates possible redundancy issues (see, *e.g.*, the Minimal Redundancy Maximal Relevance (MRMR) algorithm [18], and the Correlation Based Filtering (CFS) method [31]). It is important to note that the mentioned correlation-based

criteria operate on pairs of variables, so that their usage to assess subsets of features of arbitrary size requires some form of aggregation of the pairwise computed indices (*e.g.*, averaging), which does not necessarily capture the actual value of a given subset [75]. Ranking criteria natively designed for groups of variables of arbitrary size, as opposed to pairs, are highly desirable for the problem at hand.

The second crucial element in multivariate filter methods is the strategy for selecting feature subset candidates to be evaluated and ranked. Indeed, in view of the exponential complexity of the underlying combinatorial problem, the exhaustive approach is barely applicable with large feature sets. The space of feature subsets is typically explored with heuristic rules. A typical choice is the incremental strategy, due to its simplicity. For example, the sequential FS (SFS) approach incrementally builds the model, by adding at each step the feature that yields the maximum marginal improvement. This strategy has several drawbacks both conceptual and computational. First of all, the decision on which feature to add or remove at a given step of the selection process depends locally on the currently selected feature subset. In this respect, it can be easily seen that the marginal utility of a feature can greatly vary depending on the feature subset with respect to which it is evaluated. In other words, the relevance of a specific feature is not evaluated as a global property, but rather as a local one. This may stray the selection process from the optimal path. Also, what is optimized at every step is only the local improvement of the current feature subset with an elementary feature variation. In this way selection errors are propagated throughout the process. Finally, the incremental strategy depends critically on the threshold adopted as a stopping criterion. For all these reasons, methods based on greedy policies such as the SFS are subject to redundancy and overfitting issues, especially if applied to datasets with extremely unbalanced dimensions such as microarrays [42].

We here propose a novel multivariate filter-based FS method that can effectively tackle the two mentioned issues and is therefore suitable for classification problems with high data dimensionality and complex data distributions. The proposed method is based on the combination of the following three factors:

1. A selection criterion based on the distance correlation index (dCor);
2. A distributed combinatorial optimization approach;
3. A randomized FS procedure;

which are briefly explained below.

The dCor index [73, 74] provides an ideal criterion for the evaluation of feature subsets. Indeed, the dCor is a generalization of the correlation concept that provides a reliable dependence measure between random vectors of arbitrary dimension (not just pairs of random variables), without any assumption on their distribution. The higher the correlation between vectors, the higher the dependence measure. In the presented approach the dCor is employed to evaluate a feature subset by measuring the correlation of the latter with the target output. The dCor is inherently robust to redundancy and overfitting issues, and provides satisfactory performance even in the presence of nonlinear dependencies [86]. The dCor has been studied for variable selection in regression problems [86], where it was employed in combination with an incremental model building strategy. It has also been applied for feature screening purposes in ultrahigh-dimensional data [47], where it proved more effective than a classical screening procedure based on the classical Pearson's correlation coefficient.

The distributed combinatorial optimization scheme allows to efficiently tackle the prohibitive complexity of the combinatorial problem underlying the classification task on microarrays, as a result of the large number of features combined with the small number of samples. It is based on a *divide et impera* strategy that breaks the FS problem into smaller and more balanced subproblems, which are typically more tractable by classification methods. More in detail, the original set of features is partitioned into several smaller subsets (denoted feature bins) and an FS algorithm is run independently on each of them. Then, the features belonging to the best among the obtained local solutions are added to all feature bins, and the local FS processes are repeated. This sharing of the most promising features with all the local FS problems allows each of them to improve the local solution by combining the old features with the new ones. The algorithm stops when all local problems converge over the same solution. A noticeable benefit of the suggested distributed approach is the inherent parallelizability of the procedure.

The third contribution of the presented approach is to employ a randomized FS method to address the local FS problems, in which the utility of each feature is evaluated in a global fashion, as opposed to the local evaluation adopted in incremental methods. More in detail, the FS selection problem is reformulated in a probabilistic framework, where a probability distribution characterizes the likelihood that each feature belongs to the target model. The FS procedure alternates a generation phase, where different feature

subsets are extracted from the current distribution, to an assessment phase, where the distribution is updated based on an aggregate performance analysis carried out for each feature over all the extracted subsets. Features appearing more often in highly ranked feature subsets are re-enforced, and viceversa. Unlike incremental selection strategies the proposed method is occasionally capable of escaping local optima, and is also reported to be less prone to redundancy and overfitting issues [9].

The rest of the thesis is organized as follows. Chapter 2 provides some preliminary notions concerning the microarray datasets, feature selection and classification formulations, and briefly reviews the related literature. Then it introduces the dCor index, emphasizing the properties that make it particularly suited to the FS task. The proposed method is discussed in Chapter 3. Chapter 4 provides different experimental studies carried on well-known microarray datasets from the literature. Finally, Chapter 5 presents some concluding remarks.

Chapter 2

Preliminaries

In this chapter, we provide the reader with the basic theoretical frameworks and concepts used in this thesis. First, we introduce a fundamental terminology necessary to understand the microarray technology and the general characteristics of the microarray datasets used for diagnostics purposes. Then, it is discussed why FS is a prerequisite for obtaining good results in classification problems, especially in high-dimensional problems such as microarrays. In Section 2.3, a general introduction to classification in the context of supervised machine learning is provided, as well as the principles on which well-established classification algorithms are based. Finally, Section 2.3.8 and 2.3.9 provide different metrics used for model evaluation, and an introduction to the distance correlation (dCor) index as a dependence measure for random variables.

2.1 Microarrays

Since 2001, the year that the human genome sequence was completed, microarray technology has been largely used for new fields of research and experiments in biology, and so far abundant works regarding the detection, treatment and prevention of diseases have been carried out based on this technology. Microarrays datasets are actually the quantitative measurement of gene expression levels in humans and they are used for different purposes. In this document we are interested in their use to diagnose cancers and tumors. Short explanation of some basic concepts is provided in this section, before exploring microarray technology. Then, some of their

genaral characteristics, applications and challenges along with the datasets that have been analyzed in this thesis are discussed.

2.1.1 Genome, Chromosomes, Genes and DNA

According to Oxford Dictionary, the term *genome* has been created in 1930s to mix the words *gene* and *chromosome* [13]. In modern molecular biology and genetics terminology, a genome is all of a living thing's genetic material. It carries all the needed instructions for constructing, functioning and preserving an organism, and also information for passing life on to the next generation. Not only different species, but also every single individual on earth has its own distinctive genome, (with the exception of identical twins), although the variance among the genomes of two different humans is much less than the difference between a human genome and a chimpanzee's.

Figure 2.1: The human genome

Chromosomes are packages that organize and compact large amount of genetic information inside a cell. The number of these packages (chromosomes) depends on the size of the organism's genomes. For example, in bacteria the whole genome is packaged just into one chromosome, while for humans, where the genomes are a thousand or even a million times larger than bacteria, this number is 46. There are no specific rules on the number of chromosomes in genomes of living things, but in general, more complex organisms have more chromosomes and closely related species have a comparable number of chromosomes.

Each chromosome contains genes. Genes are similar to atoms in genetic terminology and are the fundamental unit of heredity and include enough DNS sequence to code for one protein. They characterize the features of each individual such as skin color. Like chromosomes, the number of genes that exist in a species is in direct relationship with the complexity of the organism. For instance, bacteria have only several hundred to several thousands of genes, while the number of genes in humans has been estimated to be from

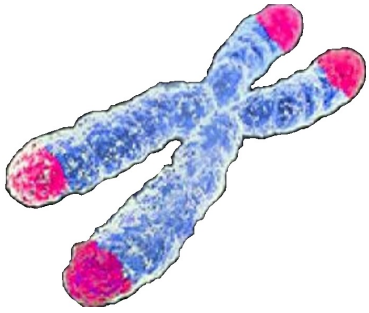

Figure 2.2: The human chromosome

25,000 to 30,000. In figure 2.3 a human gene containing some DNA sequences is displayed.

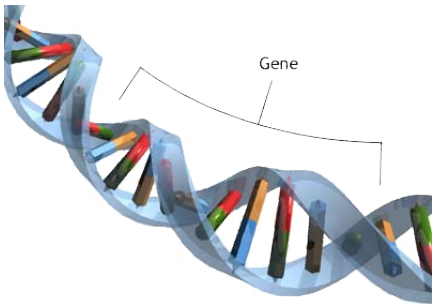

Figure 2.3: The human gene

Finally, DNA sequences are molecules that are actually responsible for building, maintaining and passing all the necessary genetic information in all living cells. In terms of shape, DNA is a very large molecule, consisting of a long row of nucleotide units and because of that the length of DNA is thousands of times longer than its width. More specifically, its structure is similar to a twisted spiral ladder. Moreover, each nucleotide in the DNA has two molecules and a structure. The molecules are sugar and phosphate, and the structure is known as nitrogenous base, referred to as *nucleotide* or *base* in this context. This structure is capable of carrying genetic information and can be of four types: adenine, cytosine, guanine, and thymine, abbreviated as A, C, G, and T, respectively.

In summary, the genome consists of chromosomes, chromosomes include genes, and genes contain DNA sequences. A visual illustration of these elements' hierarchy is reflected in 2.5.

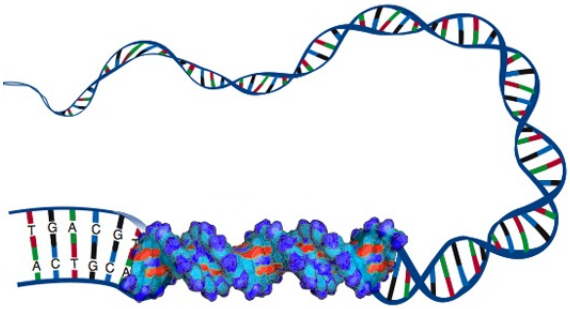

Figure 2.4: Human DNA Overview

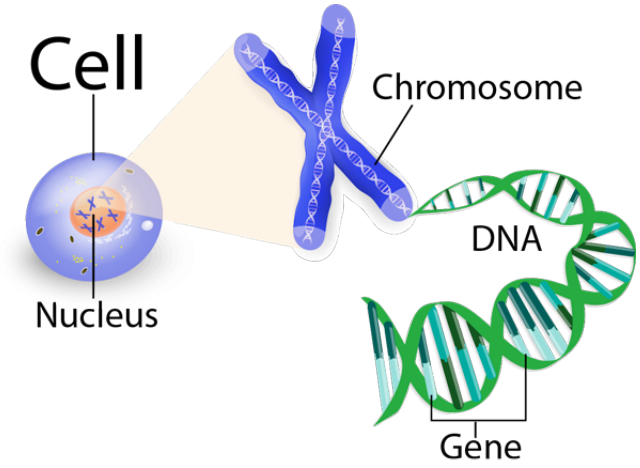

Figure 2.5: Hierarchy of genetic material

2.1.2 Microarray Technology

Up to 2000, research in Molecular Biology relied upon some well established technologies such as Northern blot and reverse transcriptase-polymerase chain reaction (RT-PCR), that were able to record only a small number of genes. However, with the technological improvement, these techniques are no longer adequate for recent researches, considering that even a minor diversity in individual characteristics could cause variations in their DNA sequence, and a comprehensive study of not preselected genes is required to capture all those characteristics. Microarray or *global expression profiling* technology has paved the way to investigate and address the issues that were not traceable before. Besides, gene microarray technology is a tool to capture a complete set of genes, orders of magnitude more than previous methods, by facilitating an approach to deposit distinct tens of thousands of human DNA sequences on a solid substrate, called *chip* which is usually a

glass slide or silicon thin-film cell and arrange them in a 2D array fashion in rows and columns. This is the reason this technology is called microarray. The position of fragments in the array is used to identify the identity of those fragments [28].

There are two main types of microarrays: Gene expression microarray and tissue microarray (TMA).

Both TMAs and gene expressions are arrays organized in rows and columns on the chip. Their difference is that TMAs offer the assessment of multiple patients in one experiment, in that different elements on the slide correspond to different patients' specific molecular markers, whereas gene expression arrays represent all the molecular markers for a single patient and all the elements belong to one sample.

The microarray technology principle involves mRNA and cDNA. mRNA is an intermediary molecule responsible for protein synthesis and carries all the needed genetic information from the cell nucleus to the cytoplasm. Several copies of mRNA corresponding to genes are generated in case of gene expression. This process in which mRNAs synthesize the corresponding protein is known as transcription. Consequently, genetic information is measurable by examining different mRNAs as substitute markers. Likewise, the processes involving genetic expressions can be understood using them. However, since they are degraded easily, it is necessary to convert them into more stable cDNA forms. Then cDNAs should be labeled through fluorochrome dyes Cy3 (green) and Cy5 (red) [28].

After that, by applying restriction endonucleases, the unknown DNA molecules are cut into fragments, to which fluorescent markers from the previous stage are able to attach, so that they can react with the probes of the DNA chip. Since complementary sequences will bind to each other, the target DNA fragments will stick to the DNA probes too. In the next step, the remaining DNA fragments are removed from the slide. Finally, the identification process of the target DNA fragments is done through the fluorescence they emit by passing a laser beam and recording the pattern of emission in a computer.

DNA chips used in this approach make this technique efficient, fast, sensitive and suitable for capturing multiple DNA pieces at the same time [28]. On the other hand, TMAs are generated by transferring cores of paraffin-embedded tissue to pre-cored holes in a recipient paraffin block. A single block has the capacity for over 500 cores applying this technique [28].

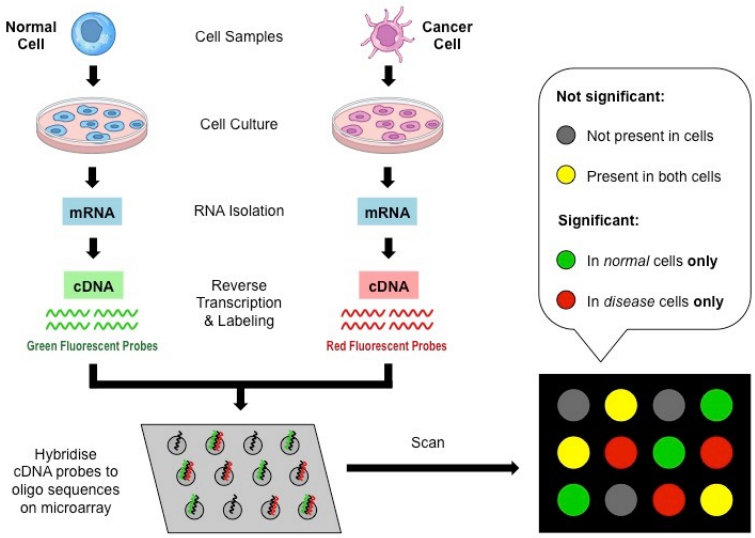

Figure 2.6: An overview of microarray technology

2.1.3 General Characteristics of Microarrays

The data containing microarray gene expression levels have some intrinsic characteristics which are listed below:

- *Large number of features:* The number of features provided by microarrays usually ranges from a few thousands to tens of thousands of gene expressions.

- *Very limited sample size:* Sample sizes in microarrays are typically very small compared to the number of features.

- *Class-imbalance:* Another property of microarrays is that the positive and negative classes, (*e.g.,* the number of healthy and cancerous tested samples) are not equal and typically there are many more samples of healthy patients.

- *Noise:* Expression levels registered for genes are often noisy, corrupted or with missing data.

- *Irrelevant features:* Most of the genes provide little or no useful information for classification purposes.

2.1.4 Applications of microarrays

DNA microarrays can be used as a tool in very large studies or for some clinical purposes and in areas such as classification of diseases, or molecular phenotyping, the study of gene function in relation to gene regulatory networks, or functional genomics, pharmacogenomics and developmental biology. It is also possible to establish a new molecular taxonomy of cancer, clustering cancers according to prognostic groups on the basis of gene expression profiles [28].

Cancer related applications:

Cancer is the simultaneous variation of lots of cells and genes and microarrays serve a suitable platform to investigate several genetic samples at the same time.

Scientists can identify the gene associated with a specific cancer by analyzing gene expression levels in a group of samples, consisting of patients having that particular cancer and healthy patients, and seek for the underlying patterns in regions of the DNA which control disease states. These patterns can be used to distinguish these patients from the healthy ones. Besides, it can help to investigate the process through which a non-invasive tumor can turn into an invasive tumor.

Microarrays can be also used to identify single-nucleotide polymorphisms (SNPs) and mutations, classify different types of tumors, recognize target genes of tumor suppressors, identify cancer biomarkers, find the genes related to chemoresistance, and develop drugs. The sensitivity and toxicity of some drugs on specific patients or in particular situations can also be assessed using microarrays in early clinical trials. Moreover, it is now possible to characterize gene categorization as well as recognize notable cellular and molecular events that might be involved in complex processes such as metastasis [28].

Antibiotic treatment

Antibiotics failure is as a result of the increase of resistant bacteria and superadded infections and affects the outcome of the disease process. DNA microarray analysis is useful, for instance, in the oral cavity where anaerobic bacteria might be the infective agent, and they usually are not easily culturable, while bacterial genomic DNA often persist more. Besides, there would be no need to a large number of bacteria DNA for culturing and a small amount of DNA is adequate to make diagnosis [28].

Early detection of oral precancerous lesions

The effectiveness of treatments of oral cancer is critically dependent on the early detection and management of it. Consequently, analyzing and identifying premalignant and early cancerous oral lesions could be another valuable use of microarrays. This is crucial for clinicians to distinguish between harmless white lesions and precancerous lesions or very early cancer, because the conditions might be reversible. However, discriminating among them and locating the lesions that lead to oral cancer is not possible using microscopic examination, while gene expression profiles or *genomic fingerprints* allows clinicians to differentiate these lesions [28].

2.1.5 Microarrays Challenges

In this work, we address the problem of cancer related applications, *i.e.* diagnosis or illness detection through classification in order to distinguish among healthy and cancerous patients based on their gene expression levels. Conventional learning algorithms are hardly applicable on this type of data. The reason is that all the microarray natural characteristics presented in 2.1.3 actually pose difficult challenges for machine learning methods in obtaining acceptable results. For example their high dimensionality combined with the makes it impossible to train a classifier without performing dimensionality reduction. Simultaneously, these properties make microarrays popular benchmarks for the evaluation of machine learning algorithms. Some important particularities of microarray datasets, concerning the classification problem we address in this work are listed below.

Curse of dimensionality

The term *"curse of dimensionality"* refers to the negative effect of high dimensional inputs on machine learning techniques. It occurs when the problem complexity grows exponentially with the problem size. Large dimensions (features) in the input would make many algorithms inapplicable, whereas they work well with low dimensional inputs. More broadly, correct generalization difficulty increases exponentially when the dimensionality grows [20]. The rationale behind many algorithms is directly or indirectly based on comparing similarity, a mechanism which breaks down in high dimensions. For example constructing a classifier when the dimension is two or three is easy, even just by visual inspection, but understanding the pattern

in high dimension is not intuitive. Besides, high dimensionality often results in overfitting. Although some might assume that having more features is always better because of the more information they provide, the drawbacks sometimes outweigh their advantages [20].

Microarray data intrinsically suffer from significantly high number of dimensions and, due to the abundance of human genes, they usually contain a large number of features (up to several hundreds of thousands of gene expressions). This is a major obstacle when analyzing microarray datasets.

Small sample size

The task of diagnosis, *i.e.* distinguishing among healthy and cancerous patients should be achieved using very small size of observations (usually less than 100) which are provided by microarray datasets. This amount is considered very small compared to the large amount of recorded gene expressions and as a result, serious issues affect the data analysis and the performance and reliability of the classification results. Two main difficulties are the definition of the classification rule and error estimation.

To construct the classification rule, many classifiers estimate the conditional probability of samples belonging to existing class labels, given the value of genes. However, these conditional probabilities are not deterministically known, and the classifier uses the values provided by observations to estimate that. The larger the training data, the more certain would be the final classification rule. Consequently, the reliability, consistency and accuracy of the classifier's decision boundary directly depend on the size of the data and also on how well distributed are the pairs of samples and labels, since the consistency and generalization capability of a learning algorithm increase as the number of samples grows. On the other hand, the performance of the classifiers which segment the feature space into disjoint areas is also in direct relationship with data size and distribution of sample and label pairs, since their decision is based on the ratio of the number of majority labels in a cell to all observations in it. For a distribution to be consistent, the cells should consider the local structure of the distribution and there should be enough labels in each cell so that the classifier's decision reflects the true conditional probability. Consequently, the cells and in turn, the classification rule depend on the number of samples [21].

There also should be enough test samples in order to evaluate the performance of the method. However, in many microarray datasets, no test part

is provided. Even when such section exists, the number of test samples is so small that each misclassified sample considerably reduces the accuracy. Apart from obtaining a high error rate, the result reflects only an approximation of the method's performance, since with few samples the error rate is quantized into a limited number of levels. The fewer the levels, the more deviated is the obtained performance from the real capabilities of the algorithm.

Class imbalance

In the classification of a dataset, the class distribution *i.e.* the proportion of samples belonging to each class, is one of the key points [24]. A dataset is said to be *imbalanced* if there are more instances of a certain class than the others. This imbalance or skewness in the class distribution is also referred to as small or rare class learning problem, and the *degree of imbalance* in the class distribution is the ratio of the minority to the majority class [70].

Since observations belonging to certain class labels are much fewer, classification rules aiming at predicting them are negatively weighted, and those rules are usually not discovered, ignored or considered as noise. Many well-known classifiers such as decision trees, support vector machines, neural networks and nearest neighbors are reported to be inadequate in the presence of class imbalance problems. They are usually biased towards the majority class, because more general rules are preferred. As a result, test observations from the minority class are misclassified more often than those of the majority class [7] [70].

This represents a major issue in some domains where the main interest is learning rare classes. An example is cancer detection in microarray datasets where predicting and preventing the disease plays a major role. Here, the number of normal patients (negative class) outnumbers the amount of sick ones [24]. The problem is even more severe when the imbalance appears in the test data more than in the training set, *i.e.* the test data do not follow the training set distribution. One example can be seen in the Lung test set in which the ratio between classes is 15:134, while this ratio in the training set is 16:16. This phenomenon is also called *dataset shift problem* [7]. Table 2.1 reports the degree of imbalance in the datasets that are being used in this work.

One of the factors that deteriorates the class imbalance problem is the small sample size. Due to the limited number of observations in microarrays, dis-

Table 2.1: Degree of imbalance in microarray datasets

Dataset	Degree of imbalance	
	Training Set	Test Set
Breast	34:44	7:12
CNS	21:39	n.a
Colon	22:40	n.a
DLBCL	19:58	n.a
Leukemia	13:25	10:24
Lung	16:16	15:134
Ovarian	91:162	n.a
Prostate	50:52	9:25

covering the patterns inherent in the rare class is difficult [70] and the size of the training data has a direct relation to the error rate caused by poorly distributed classes. Moreover, the absence of a highly discriminative pattern to distinguish among classes increases the complexity of the classification rules. Especially when the underlying patterns among classes overlap in some part of the feature space, inducing discriminative rules is hard. In fact, the experiment done by Prati *et al.* [59] concludes that the imbalance problem does not pose a problem by itself, but in the case of highly overlapping classes it can significantly decrease the number of correctly classified samples of the minority class. Also, in [36] it is claimed that linearly separable data are not sensitive to any level of imbalance [70]. On the other hand, the presence of noise in the data negatively impacts the performance of the classifier. Many real world datasets inherently involve various types of noise and the classifier usually treats the rare samples as background noise and fails to distinguish these two cases [70].

Noise

The noise in datasets is caused by errors in measurements or the features' natural variation and is called *the error in the variance of a measured variable* [34]. Microarray gene expression datasets are also believed to be extremely noisy because of many imperfections inherent in the sample preparation and the hybridization processes during the application of microarray technology [40] [77]. Noisy data pose a difficult challenge for machine learning methods since noise adds some unnecessary complexity to the inferred model and decreases the efficiency of the algorithm [69]. To avoid these problems, the method should be able to differentiate among informative

and noisy data.

There are generally two types of noise in this context. It is either *attribute noise*, *i.e.* there are wrongly measured variables or missing values in the data resulting in errors in the attribute values, or *class noise*, caused by mislabeled samples and/or labelled to belong to more than one class [88].

Irrelevant or redundant data

In the context of supervised learning, *irrelevant* features are those that carry no useful information for distinguishing among different classes. Due to the abundant number of gene expression levels that are provided as features in microarray datasets, there is a high probability that many of them are irrelevant with respect to the class label. These should be discarded before applying classification among cancerous and healthy patients, in order to avoid unnecessary complexity of the model and overfitting [17].

On the other hand, *redundant* features *carry* useful information, but the information they convey has been expressed by other features. Redundant features are typically highly correlated to each other. However, adding one of them in presence of another does not significantly improve the performance. The more features are correlated to each other, the less additional information is gained by adding them [17].

These two kinds of features should be detected and separated using feature selection algorithms, before employing any learning task, and considering that microarray datasets typically contain thousands of gene expression values, this poses a difficult challenge in analyzing them.

2.1.6 Microarray Databases

DNA Microarray data repositories are databases that measure gene expression coefficients which reflect the value of ribonucleic acid (mRNA) for a number of observations. They usually contain a large number of features, ranging from 2000 to hundreds of thousands while providing a few number of samples. In recent years the number of these datasets has been increasing for wide range of cancer types and these datasets are accessible to the public and the scientific community.

In this thesis, eight well-known biclass microarray cancer databases have been explored: Breast, CNS, Colon, DLBCL, Leukemia, Lung, Ovarian and

Prostate cancers. In the following, a short summary of their characteristcs along with some technological details about their construction is provided.

Breast:

Up to now several binary and multiple class Breast cancer microarray datasets have been introduced in the literature [78] [53] [33]. In this document the dataset first announced in 2002 in [78] has been used[1].

The construction of the dataset involved analysing the DNA microarray of primary breast tumours of all *sporadic* young patients under the age of 55 with negative lymph node. From each observation, the RNA was separated from snap-frozen tumor material in order to obtain the complementary RNA (cRNA). Then using the fluorescent dye reversal technique, hybridization was carried out on the microarray. Around 25,000 human genes have been synthesized, scanned and normalized.

The Breast cancer dataset is originally divided into training and test sets. The training data consist of 78 samples, 34 of which corresponding to patients who developed distant metastases within 5 years (relapse samples) and the rest (44 non-relapse patients) are the ones who remained disease-free 5 years after the primary diagnosis. Additional 19 samples have been added as the test set, among which 12 observations are relapsed and the remaining 7 are non-relapse samples. The total number of features (gene expression levels) measured in all these patients is 24481. Table 2.2 summarizes the Breast cancer dataset characteristics.

# features	24481		
classes	number	class labels	
	2	Relapse/non-Relapse	
Training set	Total	NP	NN
	78	34	44
Test set	Total	NP	NN
	19	12	7

Table 2.2: Breast cancer dataset characteristics

[1]Download link: http://datam.i2r.a-star.edu.sg/datasets/krbd

CNS:

Medulloblastoma is the most common malignant brain tumour of childhood. In this regard, a dataset consisting of a gene expression microarray for the Central Nervous System[2](CNS) embryonal tumour has been assembled by Pomeroy *et al.* [58] in 2002. The dataset comes in several versions, among which we selected version C. To prepare this version, a group of 60 young children with medulloblastoma were treated with chemotherapy. All the samples were obtained before starting the treatment, at the time of initial surgery, then they were frozen in liquid nitrogen and homogenized in guanidinium. Then the RNA was isolated by performing centrifugation, and a transcription reaction was carried out to generate and fragment the cRNA for 35 minutes. Then, they were hybridized. Arrays were washed and scanned on *Affymetrix* scanners so that the expression value for each gene could be calculated. Afterwards, a linear scaling method was performed to correct minor changes in microarray intensity.

The CNS/Embrional-T - Dataset C presented in [58] - includes the intensity level for 7129 gene expressions (features). It is a binary dataset and the samples are labeled as class0 and class1, class0 being the label for 39 medulloblastoma survivors and class1 representing 21 patients for whom the treatment had failed (non-survivors). Table 2.3 reports the general properties of this dataset.

# features	7129		
classes	number	class labels	
	2	class0/class1	
Training set	Total	NP	NN
	60	21	39
Test set	Total	NP	NN
	-	-	-

Table 2.3: CNS dataset characteristics

Colon:

The Colon dataset[3] has been first introduced in a study conducted by Alon *et al.* [3] in 1999. The result of their experiment is a binary dataset offering gene expression patterns of different cell types and composed of 40

[2]Download link: http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi
[3]Download link: http://www.molbio.princeton.eduycolon

tumor and 22 normal colon tissue observations. It was first analyzed with an Affymetrix oligonucleotide array complementary to more than 6,500 human genes and Expressed Sequence Tags (ESTs). Then the genes were normalized in terms of intensity to obtain a zero mean distribution across the tissues while having the magnitude equal to 1. This normalization is necessary to emphasize their relative intensity level as opposed to their absolute intensity. Then they applied a two-way clustering method to the gene expression dataset and produced a matrix in which groups of genes whose expression is correlated across tissue types are distinguishable. The final 2000 genes (features) presented in the dataset are the ones with the highest minimal intensity across the samples. Each gene is normalized such that the average intensity across the samples is 0 and the standard deviation is 1.

As illustrated in Table 2.4, this binary dataset is not originally divided into training and test set. As a whole, it provides gene expression levels for 62 patients, 22 of which were diagnosed cancer.

# features	2000		
classes	number	class labels	
	2	Tumor/Normal	
Training set	Total	NP	NN
	62	22	40
Test set	Total	NP	NN
	-	-	-

Table 2.4: Colon cancer dataset characteristics

DLBCL:

Lymphochip[4] is a specialized microarray, first designed and constructed in 2000 by Alizadeh *et al.* [2].

A biopsy sample was obtained from a group of *de novo* DLBCL patients that had not obviously arisen from pre-existing lowgrade malignancies such as follicular lymphoma, before starting their treatment. In the sampling, they preferentially expressed gene levels in lymphoid cells along with the known or suspected gene expressions having roles in processes which are notable in immunology or cancer and genes repressed or induced during B- and T- lymphocyte activation. In this experiment, they compared the relative abundance of genes in each observation with respect to a reference mRNA

[4]Download link: http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi

pool to reflect the fluorescence ratio for each gene. Due to the presence of a common pool, it was possible to measure the relative expression levels of genes in each experimental sample across all other samples [2].

Finally, in the resulting binary dataset, as reported in Table 2.5, the number of recorded features is 4026 for 77 patients, 58 of which were diagnosed as cancerous after the treatment (Fatal class label), while the remaining 19 patients were disease-free (Cured class label).

# features	4026		
classes	number	class labels	
	2	Fatal/Cured	
Training set	Total	NP	NN
	77	58	19
Test set	Total	NP	NN
	-	-	-

Table 2.5: DLBCL dataset characteristics

Leukemia:

The Leukemia DNA microarray[5] has been first introduced in 1999 by Golub *et al.* [26] for the purpose of tumor class discovery in order to distinguish among acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL), as the differentiation between ALL and AML is critical for successful treatment. To prepare the dataset, they obtained 38 bone marrow samples (27 ALL, 11 AML) from acute leukemia patients at the time of diagnosis. Using bone marrow mononuclear cells, the RNA was prepared and hybridized to high-density oligonucleotide microarrays, then it was generated by Affymetrix. The probes contained 6817 human genes, and a quantitative expression level was measured for each of them. Afterwards, an additional independent collection consisting of 34 Leukemia patients, which included a much broader range of samples, was added to the experiment. Among them, 24 observations were obtained from bone marrow and 10 from peripheral blood samples [26].

The final Leukemia binary dataset is composed of 38 observations in the training and 34 in the test set respectively, and presents 7129 gene expression levels. The characteristics of this dataset are summarized in Table 2.6.

[5]Download link: http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi

# features	7129		
classes	number	class labels	
	2	ALL/AML	
Training set	Total	NP	NN
	38	13	25
Test set	Total	NP	NN
	34	10	24

Table 2.6: Leukemia cancer dataset characteristics

Lung:

The Lung cancer dataset[6] has been originally produced by Gordon *et al.* [27] in 2002. To prepare the dataset, they collected 245 discarded Malignant Pleural Mesothelioma (MPM) and lung Adenocarcinoma (ADCA) surgical specimens from patients who underwent surgery. Then, the samples were snap-frozen and the RNA was prepared and processed. Afterwards, the cRNA was hybridized to human U95A oligonucleotide probe arrays (Affymetrix, Santa Clara, CA [27]). After inspection, 64 out of 245 samples were dropped due to the presence of some artifacts in the scanned samples. Among the rest, 31 observations were Mesothelioma (MPM) and 150 samples were Adenocarcinoma (ADCA). Then, gene expression measurements were scaled and a zero mean normalization was applied.

The Lung cancer dataset comes with separate training and test data. The training set contains 32 samples with equal distribution of MPM and ADCA patients, while the test set presents 149 observations, among which 15 are classified as Mesothelioma and 134 samples as ADCA. The total number of provided gene expression data is 12533. A summary of the descriptions is reported in Table 2.7.

# features	12533		
classes	number	class labels	
	2	Mesothelioma/ADCA	
Training set	Total	NP	NN
	32	16	16
Test set	Total	NP	NN
	149	15	134

Table 2.7: Lung cancer dataset characteristics

[6]Download link: http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi

Ovarian:

The Ovarian cancer dataset[7] was first published in 2002 by Petricoin *et al.* [57]. To construct this dataset, an initial set of spectra obtained from serum from 50 unaffected women and 50 patients with ovarian cancer were analyzed. Then a masked validation set, containing 116 observation has been added to the initial set, from which 50 samples belonged to cancerous patients, and the remaining 66 samples were unaffected women or those with non-malignant disorders. The serum samples were obtained prior to any examination, diagnosis, or treatment. Then, they were frozen in liquid nitrogen and manually analyzed on a C16 hydrophobic interaction protein chip [57].

As shown in the Table 2.8, the final binary dataset consists of a total of 253 observations, for which 15154 gene expression levels have been measured. Among these, 162 patients are diagnosed as cancerous while the rest (91) were unaffected.

# features	15154		
classes	number	class labels	
	2	Cancer/Normal	
Training set	Total	NP	NN
	253	162	91
Test set	Total	NP	NN
	-	-	-

Table 2.8: Ovarian cancer dataset characteristics

Prostate:

The Prostate dataset[8] construction experiment was undertaken originally by Singh *et al.* [67] in 2002. The samples were obtained from prostate tumors and normal tissues from a group of patients undergoing radical prostatectomy. Out of 235 observations, 65 patients were cancerous. After that the RNA were extracted, then the labeled cRNA were generated, fragmented and hybridized to Affymetrix arrays. The expression values were measured and normalized based on the median array intensity. Among all genes, the ones with variations greater than 5-fold between any two samples were kept.

[7]Download link: http://datam.i2r.a-star.edu.sg/datasets/krbd
[8]Download link: http://datam.i2r.a-star.edu.sg/datasets/krbd

Finally, high-quality expression profiles were derived using oligonucleotide microarrays which contained approximately 12,600 genes and ESTs.

The present dataset provides separate training and test datasets. Training data contain 102 observations, 52 of them corresponding to tumor (Relapse) and the remaining 50 normal (non-Relapse). On the other hand, the test data amounts to 34 samples, 25 of which are tumor while the rest (9) are normal. The general characteristics of the Prostate cancer dataset are reported in Table 2.9.

# features	12600	
classes	number	class labels
	2	Relapse/non-Relapse
Training set	Total	NP NN
	102	52 50
Test set	Total	NP NN
	34	25 9

Table 2.9: Prostate cancer dataset characteristics

2.2 The Feature Selection

Feature selection is one of the most crucial steps in the pre-processing of data, especially when the the number of features is large, such as with bioinformatic data, which pose severe challenges on machine learning methods, due their high dimensional nature. For example, microarrays allow to simultaneously measure the expression levels of a large number of genes, so that the resulting datasets are characterized by a large number of features (more than 50 thousand genes) and a very limited sample size [64]. Most of the genes provide little or no useful information for classification purposes, and it is particularly important to detect the smallest subset of features (referred to as *biomarkers*), that provide sufficient information to separate the classes represented in the dataset (which could distinguish cancerous and noncancerous samples, or identify different types of cancer [34]). This highly crucial task is referred to as feature selection (FS), which is a combinatorial optimization problem aiming at selecting from a set of available features only the relevant ones, in order to build a classifier with the required performance. FS reduces the computational cost of the classifier design and simplifies its structure, thus facilitating model interpretation and data understanding, and ultimately improves both accuracy and robustness

of the designed classifier [16]. Indeed, the presence of redundant features may adversely affect the classification accuracy, as they can add more noise than useful information [42].

The highly unbalanced dimensions of microarray datasets greatly complicate the FS task, and unsatisfactory classification performances are often reported with standard methods [37], [1]. Indeed, large feature vectors significantly slow down the learning process, since the complexity of the FS problem grows exponentially with the number of features. At the same time, the small number of samples may cause the classifier to overfit the training data, thus compromising model generalization [42]. Besides their unbalanced dimensions, microarray data are often affected by noise, which further aggravates the analysis. For all these reasons, specialized FS techniques must be developed to appropriately handle this type of datasets.

In the context of supervised learning, where the labels of samples are *a priori* known, as with microarray datasets, the goal of FS is to find all the relevant and non-redundant features among other irrelevant or noisy information in the training set, in order to eventually process the data faster and obtain more accurate and interpretable results [43].

Yu and Liu in [87] classified features into four categories: (i) irrelevant features, (ii) weakly relevant but redundant features, (iii) weakly relevant and non-redundant features, (iv) strongly relevant features. An optimal subset should contain features of type iii and iv. To fulfill this condition, a desirable feature selection algorithm should be capable of searching for a reduced subset of the feature space, such that if using only that subset to train a certain classifier, not only classification performance would not deteriorate, but it might also improve in some cases [23]. In this procedure the importance or performance of each feature or subset of features is measured using some specific criteria. In machine learning terminology, this problem is also referred to as *variable selection, attribute selection*, or *variable subset selection* [23] [43]. The FS problem is formally defined as follows:

Let the original set of features be $\mathcal{S}$ and $\mathcal{L}(.)$ be an evaluation criterion that should be maximized (optimized) and defined as $\mathcal{L} : \mathcal{S}' \subseteq \mathcal{S} \longrightarrow \mathcal{R}$. The optimal subset of features $\mathcal{S}^*$ is a subset of $\mathcal{S}$ such that $\mathcal{L}(\mathcal{S}^*) \geq \mathcal{L}(\mathcal{S}')$, meaning that the optimal subset maximizes the criterion. [9] Note that this definition does not imply the uniqueness of the optimal solution.

FS algorithms have four basic steps (see Figure 2.7): 1) subset generation,

[9] To avoid overfitting, model is evaluated on the validation part of the dataset.

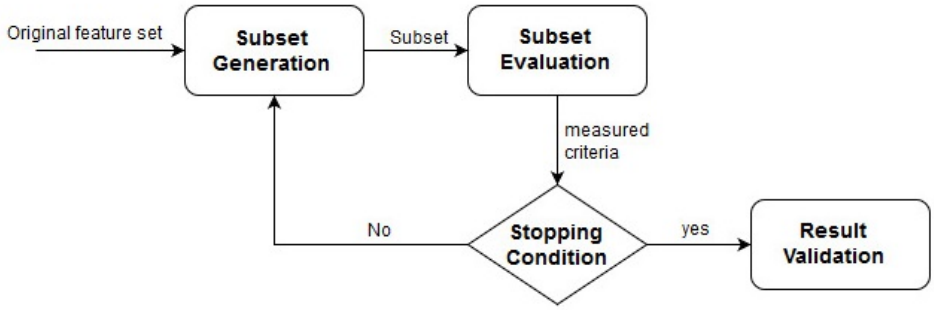

Figure 2.7: Main steps of the feature selection procedure

2) subset evaluation, 3) stopping criterion 4) validation. The first two steps are discussed in the following, the validation step is presented in 2.3.10 and the stopping condition(s) will be commented in section 3.1.

Subset Generation

Subset generation algorithms or search strategies are originally designed to achieve a trade-off between result optimality and computational efficiency [87], and they seek for a candidate subset [43]. For a N_f-dimensional dataset with a set of N_f features $\mathcal{R} = \{\varphi_1, \ldots, \varphi_{N_F}\}$, there exist $2^{N_F} - 1$ candidate subsets of features and as the number of features grows, the search space increases exponentially. Therefore, it is impractical for a FS algorithm to consider all the possible subsets and it needs to resort to some heuristic strategy. In general search mechanisms can be divided into sequential, exponential and randomized. In the following, the basic idea behind each strategy is discussed.

Sequential or Incremental Search are greedy procedures that progressively generate a candidate subset, either by adding or removing features starting from the subset obtained at the previous iteration. Some famous methods in this category include *Sequential Forward Selection* (SFS) and *Sequential Backward elimination/Selection* (SBS). In SFS, at the first iteration the subset is empty. Then, at each time, one uncollected feature is added to the subset, such that the classifier accuracy would be maximum compared to other uncollected features that could possibly be added. In SBS, starting from the full feature set, at each step the less important feature in the existing subset is selected to be removed. Different combination of these methods are also possible, for example the *bi-directional selection* or adding

or removing multiple features at a time instead of a single one [43] [81].

Sequential algorithms visit only a small portion of the search space, and consequently they do not guarantee optimality, since the optimal solutions might never be visited [19]. Moreover, the decision made at each step in these algorithms to remove or add features is dependent on the current subset, so their initialization significantly impacts the visited subsets and consequently the outcome [61]. On the other hand, in simple datasets with few features, they are easy to implement and relatively fast.

Exponential Search or *best subset selection* [35] searches for each possible combination of the features. In N_f-dimensional space it searches the whole $2^{N_F} - 1$ options. It has computational limitations and for $N_f \geq 40$, this approach is practically unfeasible. However, to reduce the search space, it may employ such methods as Branch and bound [54] and Beam search [19]. Nevertheless, employing exponential search algorithms might lead to overfitting, especially when the criterion to evaluate the subset is classifier performance, because selecting more features and more flexible models always results in better performance on the training set, so the algorithms are biased towards selecting large sized models.

Randomized Search methods choose the potential subsets by introducing some degree of randomness in the search strategy. As a consequence, they can avoid local minima and converge closer to global optimal solutions [19]. In some randomized search algorithms, a random subset is selected at the first step, then at the next iterations other completely random subsets are selected or extra features are added to the first selected subset until the optimal solution is obtained [43]. For example, the Las Vegas algorithm [50] at each step selects a random subset of features from the whole features, then evaluates it using an inconsistency criterion. If the size of the selected subset is smaller than the best subset identified so far and the inconsistency criterion holds, it will replace the obtained subset as the optimal solution. Also, if the size of the obtained solution is the same as the optimal solution and the inconsistency criterion is less than a threshold, it will keep the current answer as an equally good solution and this procedure continues until the algorithm reaches the maximum number of predefined rounds [60].

Applying random algorithms requires some level of knowledge about the underlying feature space. For example, randomization may not be effective enough, specifically in high dimensional spaces where only a very small subset of features is desirable or relevant, in which case the probability of converging to that particular subset is too small [60]. Also, randomized al-

gorithms are extremely sensitive to their tuning parameters [19]. Another issue is the stopping condition, since there is no *a priori* knowledge about the optimal solution, and the algorithm can stop when a performance threshold is met or a time or iteration counter is reached [60]. Some well-known subset generation algorithms are presented in Section 2.10.

Strategy	Algorithm name
Exponential	Exhaustive search
	Branch and bound
	Beam search
Sequential	Greedy forward selection or backward elimination
	Best-first
	Linear forward selection
	Floating forward or backward selection
	Beam search (and beam stack search)
	Race search
Randomized	Random generation
	Simulated annealing
	Evolutionary computation algorithms (e.g. genetic, ant colony optimization)
	Scatter search

Table 2.10: Algorithms in different search strategies

Subset Evaluation

After generating candidate subsets, their goodness should be measured according to some certain criterion. There are two main subset evaluation metrics, according to whether they are independent from or dependent on the FS algorithm.

Independent metrics are filters that measure the importance of a single or a subset of features, solely based on the essential characteristics of the training data, and do not involve the underlying FS algorithm [43]. Some of them include *distance* measures, *information or uncertainty* measures, *dependency* measures, *inter-class distance* measures, and *consistency* measures.

The distance measure computes the probabilistic distance among the class conditional probability densities. It is also known as divergence, discrimination and separability criterion. Information or uncertainty measures use the concept of entropy to calculate the information gain. In dependency or similarity measures, the features that are strongly correlated to the output

are preferred. Consistency metrics search for the smallest subset of features that separates the classes as consistently as the whole set. Inter-class distance evaluation methods measure the distance (usually Euclidean) between each classes' observations and then the distance between two classes is computed [43].

On the other hand, dependence criteria evaluate the goodness of a subset based on the performance of the classifier. Although their final outcome is better than other algorithms, they are computationally expensive since for each subset one has to calculate and evaluate the classifier.

FS algorithms are divided into three main categories: filter, wrapper and embedded methods. Typically, independent metrics are used in filter-based FS methods, and dependence measures are used in combination with wrappers. The three types of FS algorithms are discussed next.

2.2.1 Filter Methods

Filter or open-loop methods assess the importance of individual features (or subsets of features) by evaluating their intrinsic characteristics in terms of some criteria *i.e.*, dependency, information, distance, consistency and statistical measures [4] [38]. They are computationally very effective and fast, and can be easily scaled to large-sized datasets such as microarrays. Besides, since the process of measuring the criteria is done independently from any learning task, they can provide a general solution for every classifier. Filter methods are categorized into univariate and multivariate methods based on whether the goodness of a single feature is evaluated or a subset of features [4].

Univariate filter methods rank a single feature according to their degrees of relevance, without considering their mutual interaction [6]. Therefore, these algorithms remove only the irrelevant features, but not the redundant ones, because it is likely that similar features have similar rankings [87]. Besides, they ignore that a single feature may be irrelevant, but it can contain highly relevant information in combination with other features [4].

To address the first issue, several methods employ correlation-oriented criteria based on the concept of mutual information (MI) (see [52], [55], [56], [18], [68] and [75]). As it is the case in Redundancy Maximal Relevance (MRMR) algorithm [18] and in Correlation Based Filtering (CFS) method [31], the MI between the features and the output displays the features' discriminatory

powers, while the correlation among features reveals possible redundancy issues. It is essential to mention that correlation-based criteria operate on pairs of variables, so applying them for assessing subsets of features of arbitrary size requires some form of aggregation of the pairwise computed indices (e.g., averaging), which does not necessarily reflect the actual importance of a given subset [75]. Some methods using this type of approximated calculation of the group mutual information are presented in [31], [41], [5], [71] and [72]. It is apparent that the ranking criteria that are originally designed for the evaluation of groups of variables of arbitrary size are more desirable than those designed for pairs of features. This is the case in multivariate filter methods, in which, a subset of features is evaluated as opposed to a single feature, to account for the interaction among features and to capture the features that are valuable in the presence of other ones.

Unlike univariate methods, which provide a ranked list of features to select from, *multivariate filter methods* pose greater complexity, because in addition to the evaluation process, they also involve a search strategy for candidate subsets in the space of all possible feature subsets. Different search strategies proposed in the literature have been described in the previous section.

The most common strategies when using multivariate methods are exponential algorithms, sequential algorithms and randomized algorithms. Exponential algorithms search for candidate subsets exhaustively and as a result the number of subsets, as well as the computational complexity, increase exponentially when the feature space grows, so they are not practical solutions for large-sized datasets. On the other hand, heuristic or sequential algorithms are more practical in these situations. Two well-known approaches of this kind are Sequential Forward Selection (SFS) and Sequential Backward Selection (SBS) [18] [31] [65]. They add in or remove from the candidate subset one or more features sequentially such that the improvement in that stage would be maximized. Despite its simplicity, this strategy has several drawbacks both conceptual and computational. First of all, the decision regarding which feature should be added or removed at a given step depends locally on the currently selected feature subset. A feature with great discriminatory capabilities when combined with other features that are not present in the current subset would be neglected. In other words, the relevance of a specific feature is not evaluated as a global property, but rather as a local one. Thus, an unsuitable initialization would stray the selection process from the optimal path and lead the algorithm to a local minimum. Also, what is optimized at every step is only the local improvement of the

current feature subset with an elementary feature variation. In this way selection errors are propagated throughout the process. Finally, the incremental strategy depends critically on the threshold adopted as a stopping criterion. For all these reasons, methods based on greedy policies such as the SFS and SBS are subject to redundancy and overfitting issues, especially if applied to datasets with extremely unbalanced dimensions, such as microarrays. However, randomized algorithms overcome these issues and have the possibility of escaping from local minima, by selecting subsets of features randomly, such that the initialzation does not affect the future states since the selection of features at the next iteration does not completely rely on the initialization and they involve some degree of randomness [38]. Although if the desirable solution is a unique subset of features, the chance to find it is probably not high. After this stage, each obtained subset is assessed by a certain criterion and compared to other subsets according to the same criterion. This would prevent redundant features to appear in the same subset, as opposed to univariate methods, in which there is a high probability that redundant features have similar ranking measurements. However, despite the overall better performance of multivariate algorithms, they need more computational time to generate and compare subsets [6].

Due to their simplicity and speed, filter methods are widely used in analyzing microarray datasets. The following are some well-known filter methods proposed in the literature.

Correlation-based Feature Selection (CFS) [30] is a multivariate filter method that uses a correlation-based heuristic to measure the goodness of features, according to the hypothesis that a good feature subset includes the features highly correlated with the output (those with high predictive ability), but mutually uncorrelated, so that irrelevant and redundant features should be discarded. The search strategy in this method is an incremental one [30] [7].

Information Gain is another univariate filter method that makes an ordered ranking list of features, putting first those whose values in the observations provide more information to predict the class. The method works by approximating the conditional distribution $Pr(Y|X)$ where Y is the class and X is the feature vector [85] [34]. Like other ranking methods, a threshold can be used to select the best features among all, by evaluating each feature's discriminating ability, independently of other features [7].

Relief is a multivariate filter method for biclass problems inspired by instance learning. It randomly picks instances, then finds their nearest neigh-

bor from the same class and also from the other class. Then the values of features of these two samples are compared to the randomly selected sample, then used in updating the rank of each feature. In fact, it progressively picks observations and gives higher ranks to the features that have the same value in the same classes but different in the opposite class [7] [34]. So the features are not directly selected, but their weights are gradually updated [41]. **ReliefF** is an extension of Relief which adds the ability to deal with more than two classes [7]. These two filter methods are widely used in gene selection problems. However, in spite of their noise tolerant property, low bias and simplicity, they are often unable to find an optimal feature set size [39].

Minimal-Redundancy Maximal-Relevance (MRMR) [18] is another multivariate filter method that selects the features at maximum dissimilarity in terms of their mutual Euclidean distances or other correlation-based metrics to ensure minimum redundancy, and at the same time, highly correlated features to the output using relevance criteria such as mutual information to ensure maximum relevance. MRMR employs an incremental strategy to search for candidate subsets, thus it is not guaranteed to converge to the global optimum.

Markov Blanket Filtering (MBF) [41] is an iterative greedy multivariate filter method, that at each step eliminates weakly relevant redundant features, whose removal causes the least change in the distribution and also the least decrease in accuracy. It employs the concept of conditional independence between feature subsets and the class labels to determine the variables carrying the least amount of information to be removed at next stage. The set of other variables is called *Markov blanket* of that feature [41] [34] [38].MBF lacks a global evaluation of subsets, due to its incremental strategy for finding feature subset candidates.

2.2.2 Wrapper Methods

Wrapper methods are another major type of FS algorithms. In wrapper techniques, subsets are generated based on the search strategies similar to filter methods. However, the assessment and comparison of feature subsets is done in terms of the performance of the classifier. In contrast to filters, they can detect feature dependencies. [34].

Increasing the problem size increases the feature subset space, and since the process of evaluation should be repeated for each subset, it dramatically slows down the algorithm compared to filter methods. This is critical in

analyzing microarray datasets, which have thousands of features. Moreover, subset evaluation is done through a classifier, therefore the subsets that provide more accurate classification results employing that specific classifier are considered better and consequently their importance are biased towards the classifier. Also, due to the small sample size there is the risk of overfitting. Their performance is generally better than filter methods [7]. Two famous wrapper methods (SFS and GA-KDE-Bayes) are discussed below:

Sequential Feature Selection (SFS) is applied on microarrays by Sharma *et al.* [66] to overcome the existing problem in many FS algorithms, *i.e.* that some weakly ranked genes are not selected despite their informative nature in classification if they are combined with appropriate subsets of features. The algorithm first divides the features into small subsets. Then in each subset it selects informative smaller subsets of genes according to the classification accuracy and merges them together to update the optimal solution. The process of selection and merging is repeated until a single subset is obtained [7].

GA-KDE-Bayes [79] is an evolutionary wrapper algorithm, which uses a non-parametric density estimation method in combination with a *Bayesian* classifier, suggesting that in sparse and scarce data such as microarrays, non-parametric methods are favorable alternatives, since they do not make any assumption about the distribution and structure of the data, and all the information are extracted directly from the data itself. The experimental study of this method shows that local modeling selects small and relevant subsets of features and results are competitive in terms of classification accuracy [7].

2.2.3 Embedded Methods

Embedded methods are designed such that FS is embedded in the learning process. They are more efficient than wrapper methods in terms of time and computation, as they do not train the classifier for measuring the importance of each subset, but employ the classifier to establish a criterion for ranking the features. The probability of overfitting in embedded methods is less than with wrapper methods. However, they still suffer from computational complexity, especially when the size of the dataset grows and, similar to wrapper methods, they are dependent on the specific classifier they employ [4] [7]. Recently, most FS algorithms tend to combine the above mentioned algorithms to overcome their inherent drawbacks and combine their advantages.

Some of them are introduced in the following:

Support Vector Machines based on Recursive Feature Elimination (SVM-RFE) [29] is an iterative embedded method that, at each step trains an SVM classifier using the genes in the current subset, then computes the ranking criterion for all the features (or for all the subsets), and finally eliminates the least important one. This method is capable of removing redundant features and ranking the features based not only on their individual relevance, but also to their importance in the presence of other features [7].

Kernel Penalized SVM (KP-SVM) [51] is another SVM-based embedded method which embeds the FS task in the dual formulation of SVMs. In the dual problem formulation, a penalization function is added to optimize the shape of the Gaussian kernel and remove the less relevant features from the classifier [7].

FOIL Rule based Feature subset Selection algorithm (FRFS) [80] is an alternative embedded FS method, which uses FOIL rules (First Order Inductive Learner rules) for classification. First, it uses a modified propositional implementation of the FOIL algorithm in order to generate the FOIL classification rules. It then combines the features that appear in the antecedents of all rules to achieve a candidate feature subset that excludes redundant terms and maintains the interactive ones. Finally, the algorithm evaluates the relevance of each feature in the candidate subset using the CoverRatio [80] metric to remove the irrelevant features [7].

2.3 Classification

Classification is the process in which a learner or classifier aims to find the underlying relationship between the input and the output data in a given training set, and according to that makes a concise model of the distribution of the class labels in terms of the features, so that in the future it can predict the outcome of any other test set based only on the values of the features in the input data and accordingly assign a class label to that instance.

Let $\mathcal{D} = \{d^{(1)}, \ldots, d^{(N)}\}$ be a set of N available observations, each consisting of an input-output pair $d^{(k)} = (\boldsymbol{f}^{(k)}, c^{(k)})$, where $\boldsymbol{f} = [f_1, \ldots, f_{N_f}]$ denotes the vector of features, and $c \in \{1, \ldots, N_c\}$ the classes, with $k = 1, \ldots, N$. $\mathcal{D}$ is used to build a classifier, capable of predicting the class label of previously unseen samples of the features. The general form of the classifier is thus

given as:

$$\hat{c} = h(\boldsymbol{f}), \tag{2.1}$$

where $\hat{c}$ denotes the predicted class associated to the vector of features $\boldsymbol{f}$ and h is a suitable function of the feature values.

In this thesis, some well-known classifiers have been employed to evaluate the performance of the studied FS algorithms, which are listed below:

2.3.1 k-Nearest Neighbors

k-Nearest Neighbors or k-NN is a non-parametric lazy classifier. Being lazy means that it does not learn a particular model from the training dataset, but memorizes the data to use in the test stage in which it assigns the observation to a particular class label by estimating the conditional probability that the observation belongs to the class. It considers the k data points in the training set nearest to the observation and assigns the observation to the class label y_c, that is most common label amongst its k nearest neighbors [35]. Formally, if $\mathcal{N}_i$ is the neighborhood containing k samples whose distance to data point $d^{(i)} = f^{(i)}$ is the least, the probability of $d^{(i)}$ belonging to class label $c = c^{(j)}$ is given by:

$$Pr(c = c^{(j)} | d^{(i)} = f^{(i)}) = \frac{1}{k} \sum_{c^{(j)} \in \mathcal{N}_i} I(c = c^{(j)}) \tag{2.2}$$

where I is the identity matrix.

Since k-NN is non parametric, it does not make any prior assumption about the distribution of the data. This is quite helpful, especially in case of microarray datasets. However, since there is a minimal learning phase due to its laziness and absence of generalization, it tends to perform poorly in the presence of high dimensional data and irrelevant features. Moreover, the performance is significantly dependent on the appropriate choice of k, which balances the bias and variance tradeoff in the method. Increasing k reduces the bias and noise impact, but also the risk of ignoring small but important patterns increases as well, due to the fact that the decision boundary becomes less flexible. On the other hand, a small k introduces bias and an increase in the probability of overfitting [76]. The best choice of k is typically tuned using cross validation (see section 2.3.10).

2.3.2 Support Vector Machine

The support vector machine classifier (SVM) [35] is a generalization of the maximal margin classifier. In the binary classification setting, the SVM aims to classify the samples by solving an optimization problem in order to divide the N_f-dimensional space into two subspaces by a hyperplane that almost separates the classes, using a so-called soft margin. This hyperplane (maximal margin hyperplane) has the farthest minimum distance to the training observations. The optimization problem is formalized as follows:

$$\begin{aligned} &\underset{\beta_0,\beta_1,...,\beta_{N_f},\epsilon^{(1)},...,\epsilon^{(N)}}{\text{maximize}} M \\ &\text{subject to} \sum_{l=1}^{N_f} \beta_l^2 = 1 \\ &c^{(j)}(\beta_0 + \beta_1 f_{j1} + \beta_2 f_{j2} + ... + \beta_{N_f} f_{jN_f}) \quad \geq M(1 - \epsilon^{(i)}) \\ &\epsilon^{(i)} \geq 0, \ \sum_{i=1}^{N} \epsilon^{(i)} \leq \mathcal{C} \end{aligned} \tag{2.3}$$

where M defines the width of the margin. The goal is to keep it as large as possible, such that the second constraint holds.

In a N_f-dimensional space, a hyperplane is a flat affine subspace of dimension N_f, defined by:

$$\beta_0 + \beta_1 f_1 + ... + \beta_{N_f} f_{N_f} = 0 \tag{2.4}$$

Any N_f-dimensional point $d = (f_1, f_2, ..., f_{N_f})^T$ for which (2.5) holds is a point on the hyperplane. If point d does not satisfy the equation (*i.e.* does not reside on the hyperplane in equation 2.5), then the sign of $\beta_0 + \beta_1 f_1 + ... + \beta_{N_f} f_{N_f}$ tells us on which side of the hyperplane the point lies. In the binary setting, if we assume that all the observations fall into two classes $\{1, -1\}$ ($c^{(j)} \in \{1, -1\}$ for all class labels), if $c^{(j)} = 1$, then $\beta_0 + \beta_1 f_{j1} + \beta_2 f_{j2} + ... + \beta_{N_f} f_{jN_f} > 0$, and if $c^{(j)} = -1$, then $\beta_0 + \beta_1 f_{j1} + \beta_2 f_{j2} + ... + \beta_{N_f} f_{jN_f} < 0$. Evidently, this separating hyperplane has the property that

$$c^{(j)}(\beta_0 + \beta_1 f_{j1} + \beta_2 f_{j2} + ... + \beta_{N_f} f_{jN_f}) > 0 \tag{2.5}$$

However, $M(1 - \epsilon^{(i)})$ in the second constraint of equation (2.3) would let some training data reside on the wrong side of the hyperplane or along the

margin in the process of learning, depending on the $\epsilon^{(i)}$ value being greater than 1, or not. The $\epsilon^{(i)}$ s are called slack variables to make the soft margin possible.

$\mathcal{C}$ is a non-negative tuning parameter which bounds the sum of the $\epsilon^{(i)}$ s, to determine the number and severity of the violations. In fact $\mathcal{C}$ is the tolerance level that controls the bias-variance trade-off in the learning.

The data points residing along the margin or between the margin and the hyperplane are called support vectors, since they are N_f-dimensional vectors that support the maximal margin hyperplane. If these points were moved, the maximal margin hyperplane would move as well. The optimization problem depends directly only on the support vectors.

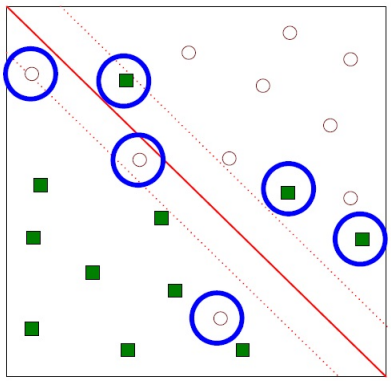

Figure 2.8: Support vector classifier

Interestingly, in equation (2.3) only the data points that violate the margin will affect the optimization solution. This implies that the decision rule is based only on a potentially small subset of the training observations (the support vectors), so it is quite robust to the behavior of observations that are far from the hyperplane.

2.3.3 Nonlinear SVM

In case a linear boundary between two classes does not exist, the performance of linear classifiers such as linear SVM would drop. In these conditions one should resort to nonlinear decision boundaries. This is typically addressed by enlarging the feature space by involving also functions of the features, *e.g.* adding quadratic or higher-order polynomial terms of the feature space. In nonlinear SVM, however, enlarging the feature space is

performed in a computationally efficient way, using *kernels*.

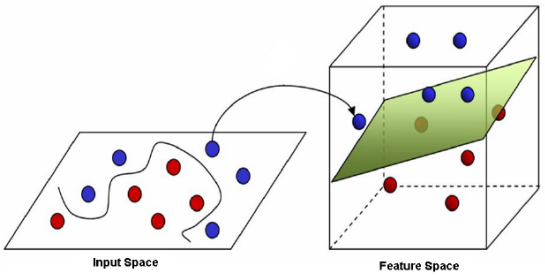

Figure 2.9: Enlarging feature space overview in nonlinear SVM

The inner product of two N_f-dimensional vectors $f^{(i)}$ and $f^{(i')}$ is defined as:

$$\langle f^{(i)}, f^{(i')} \rangle = \sum_{j=1}^{N_f} f_j^{(i)} f_j^{(i')} \tag{2.6}$$

It can be shown that the solution to the linear SVM illustrated in equation (2.3) involves only the inner products of the observations as opposed to the observations themselves and can be represented as:

$$g(x) = \beta_0 + \sum_{i=1}^{N} \alpha_i \langle f^{(i)}, f^{(i')} \rangle \tag{2.7}$$

where N is the number of observations, α_i s are the parameters that should be estimated using the inner product of each pair of samples and is non zero only for support vectors. To accommodate non-linearity, instead of the inner product $\langle f^{(i)}, f^{(i')} \rangle$, a general nonlinear function called kernel can be used.

$$g(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \mathcal{K}(f^{(i)}, f^{(i')}) \tag{2.8}$$

The kernel function quantifies the similarity of two observations. The inner product is the linear special case of the kernel.

Two well-known nonlinear kernels are the *polynomial* and the *radial* ones:

The polynomial kernel of degree d in N_f-dimensional space is as follows:

$$\mathcal{K}(f^{(i)}, f^{(i')}) = (1 + \sum_{j=1}^{N_f} f_j^{(i)}, f_j^{(i')})^d \tag{2.9}$$

and the radial kernel is constructed by:

$$\mathcal{K}(f^{(i)}, f^{(i')}) = exp(-\gamma \sum_{j=1}^{p} (f_j^{(i)} - f_j^{(i')})^2) \tag{2.10}$$

Using kernels instead of adding functions of original predictors for enlarging the feature space is computationally much more effective, since otherwise computations in the enlarged feature space become intractable [35]. In fact SVMs first map the training data into a higher dimensional feature space using some predefined nonlinear function, then construct a linear decision surface in that space to ensure maximum generalization [14].

2.3.4 Logistic Regression

Logistic regression is extensively used as a classifier to model the outcomes of a categorical dependent variable, in which linear regression cannot be used since its response values are not expressed in ratio scale, and the range of its predicted values can be any real number, while in the classification setting, the output is only a limited number of discrete values. Besides, in linear regression error terms are not normally distributed [15].

In linear regression the expected value of a dependent variable c is a linear combination of independent variables and parameters and the output is the probability of a class label (conventionally the default class, shown as $p(f^{(i)}) = Pr(c = 1|f^{(i)})$). For the logistic regression, this function is called the *logit transform*:

$$\log\Big(\frac{p(f^{(i)})}{1 - p(f^{(i)})}\Big) = \beta_0 + \sum_{j=1}^{N_f} \beta_j f_j^{(i)} \tag{2.11}$$

This ensures that the probability will fall between 0 and 1 for all values of N_f-dimensional samples $f^{(i)} = [f_1^{(i)}, f_2^{(i)}, ..., f_{N_f}^{(i)}]$, $i \in \{1, ..., N\}$ as inputs:

$$p(f^{(i)}) = \frac{e^{\beta_0 + \sum_{j=1}^{N_f} \beta_j f_j^{(i)}}}{1 + e^{\beta_0 + \sum_{j=1}^{N_f} \beta_j f_j^{(i)}}} \tag{2.12}$$

Function (2.12), also known as *logistic function*, forms an S-shaped curve in the desirable range.

The *maximum likelihood* method is used to estimate the parameters β_j such that the probability of the observed data is greatest [15].

The drawback of the logistic regression is that when the classes are well-separated, the parameter estimation process is surprisingly unstable. [35]. Also, if the number of samples are small, the logistic regression is less stable.

2.3.5 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) develops an approximation of a Bayes classifier by modeling the distribution of each of the predictors in $f^{(i)}$ in different classes, and then using Bayes' theorem to estimate $Pr(c = c^{(j)}|f^{(i)})$ [35].

Let N_c be the number of class labels ($N_c \geq 2$) and π_{c_j} be the *prior* probability that a randomly chosen observation $f^{(i)}$ belongs to the class c_j. We define $g_{c_j}(\boldsymbol{f}) \equiv Pr(f = f^{(i)}|c = c_j)$ as the *density function* of f (the probability that an observation in the c_jth class has $f \approx f^{(i)}$) and $p_{c_j}(\boldsymbol{f}) = Pr(c = c_j|\boldsymbol{f})$ is the *posterior* probability that an observation $\boldsymbol{f} = f^{(i)}$ comes from the c_jth class. Bayes theorem states that:

$$Pr(c = c_j|\boldsymbol{f} = f^{(i)}) = \frac{\pi_{c_j} g_{c_j}(\boldsymbol{f})}{\sum_{l=1}^{N_c} \pi_l g_l(\boldsymbol{f})} \tag{2.13}$$

π_{c_j} is estimated as the fraction of the training observations that belong to the c_jth class, $\frac{N_{c_j}}{N}$ where N_{c_j} is the number of observations belonging to class c_j and N is the sample size. Now we can plug in the estimates of π_{c_j} and $g_{c_j}(\boldsymbol{f})$ into equation (2.13).

To estimate $g_{c_j}(\boldsymbol{f})$, we assume that $\boldsymbol{f}$ has a multivariate Gaussian distribution ($\boldsymbol{f} \sim \mathcal{N}(\mu, \Sigma)$) , so in an N_f-dimensional space, $g_{c_j}(\boldsymbol{f})$ would be:

$$g_{c_j}(\boldsymbol{f}) = \frac{1}{2\pi^{N_f/2}\,|\Sigma|^{N_f/2}} exp\Big(-\frac{1}{2}(f^{(i)}-\mu)^T\Sigma^{-1}(f^{(i)}-\mu)\Big) \tag{2.14}$$

where $\mu = \mathbb{E}(\boldsymbol{f})$ is a N_f-dimensional vector, and $\Sigma = Cov(\boldsymbol{f})$ a $N_f \times N_f$ covariance matrix of $\boldsymbol{f}$ which is common to all N_c classes.

An observation $\boldsymbol{f} = f^{(i)}$ is assigned to the class for which (2.15) is the largest.

$$\delta_{c_j}(f^{(i)}) = f^{(i)T}\Sigma^{-1}\mu_{c_j} - \frac{1}{2}\mu_{c_j}^T\Sigma^{-1}\mu_{c_j} + log(\pi_{c_j}) \tag{2.15}$$

In fact, the LDA method approximates the Bayes classifier by plugging estimates for μ, Σ and π_k from the training data into $\delta_{c_j}(f^{(i)})$, which is called *discriminant function* and is a linear function with respect to $f^{(i)}$.

2.3.6 Naïve Bayes

Naïve Bayes is a simplae linear probabilistic classifier based on the Bayes probability theorem, which constructs simple yet efficient and well performing models in many real world problems, such as disease prediction. It applies the mutual independence assumption among features for given a class label. It is called naïve, because to compute the probability that each observation belonging to a class, it considers the contribution of each predictor unrelated to the contribution of others. However, unless this assumption is strongly violated or the decision boundaries among classes are inherently nonlinear, this classifier is accurate, easy to implement and fast, and works surprisingly well in many cases, especially for small sample size data, outperforming the more powerful classifiers. It is also robust to noise and irrelevant attributes [6] [62] [63].

The core of NB is the Bayes' theorem concept:

$$\text{posterior probability} = \frac{\text{conditional probability} \times \text{prior probability}}{\text{evidence}} \tag{2.16}$$

The *posterior probability* is the probability that a particular observation belongs to the class c_j given its observed feature values. It can be presented as $Pr(c = c_j|\boldsymbol{f} = f^{(i)})$ and the classifier assigns that observation to the most likely class according to the *Bayes decision boundary*, which in the binary

classification case is the line on which the probability of both classes are the same.

The posterior probability can be written as:

$$Pr(c = c_j|\boldsymbol{f} = f^{(i)}) = \frac{Pr(\boldsymbol{f} = f^{(i)}|c = c_j) \times Pr(c = c_j)}{Pr(\boldsymbol{f} = f^{(i)})} \tag{2.17}$$

The NB objective is to maximize the a posteriori probability (MAP) given the training data in order to formulate the decision rule.

$$\text{Predicted class label} \longleftarrow \underset{c=1,\ldots,N_j}{\text{argmax}}\, Pr(c = c_j|\boldsymbol{f} = f^{(i)}) \tag{2.18}$$

In equation (2.17) $Pr(\boldsymbol{f} = f^{(i)})$ is identical for all classes, and can thus be ignored in the solution. However, a direct estimation of $Pr(\boldsymbol{f} = f^{(i)}|c = c_j)$ from a given set of training observations is difficult when the feature space is high dimensional. Therefore, usually some approximations are applied, such as the mentioned assumption about the independence of the features [63].

The resulting discriminant functions indicating the decision boundaries among classes are:

$$g_{c_j}^{NB} = \prod_{j=1}^{n} Pr(\boldsymbol{f} = f^{(i)}|c = c_j)Pr(c = c_j) \tag{2.19}$$

2.3.7 Tree Based Classifiers

Tree based methods are simple and intuitive approaches and are suitable for regression or classification. In the classification setting, they employ a data structure called classification tree, which is usually described in graphical terms. Classification trees are usually constructed *upside down* from the root of the tree [11], using splitting rules which evaluate the quality of a particular split. Each parent node is split into two child nodes, and the child nodes will, in turn, be split, unless they are terminal nodes. The goal is to partition the N_f-dimensional feature space into a set of $\mathcal{R}$ distinct non-overlapping regions called *leaves* using the explanatory variables, such that each leave contains the most homogeneous labels possible [11] [35].

More in detail, at the top of the tree (*root*), all the observations belong to a single region, then the algorithm successively splits the predictor space into

two new branches based on some *splitting rules*.

Formally, the predictor f_j and the cutpoint s should be selected such that the resulting regions $\mathcal{R}_1 = \{f_j^{(i)} | f_j^{(i)} < s\}$, and $\mathcal{R}_2 = \{f_j^{(i)} | f_j^{(i)} \geq s\}$, $\forall i \in \{1, .., N\}$ and $\forall j \in \{1, .., N_f\}$, lead to the greatest possible reduction in impurity as measured through a criterion called *Gini index*. It roughly indicates the total variance across the N_c classes and is defined by:

$$G = \sum_{l=1}^{N_c} \hat{p}_{ml}(1 - \hat{p}_{ml}) \tag{2.20}$$

where $\hat{p}_{mk}$ is the proportion of training observations in the mth region that are from the lth class. The more $\hat{p}_{mk}$ s are closer to zero or one, the less is the Gini index, indicating that a specific region contains mostly the observations from a single class, which is desirable. Each of the resulting regions from the previous split are then split in turn based on the same criterion until the point where a further division is not possible or according to some user defined stopping condition (for example that no node should contain more than five observations). Then some subset of the original tree can be built by *pruning*, which prevents overfitting [11].

To predict the class of an observation, it is carried down the tree and when encountering the nodes, its next direction is decided based on its explanatory predictors' values, until it reaches a terminal node, namely region $\mathcal{R}_j$. The predicted class for the observation would be the most commonly occurring class for the existing observations in $\mathcal{R}_j$.

Another common splitting rule is the classification error rate which is employed when the final goal is to maximize the classification accuracy. It is the fraction of the training observations in that region that do not belong to the most common class. The problem of using the classification error rate as a splitting rule is that it is not sufficiently sensitive for tree-growing. Another natural alternative to purity is cross entropy.

Tree based algorithms are very easy to interpret and capable of being illustrated graphically. However, they typically have less accuracy rate compared to the best supervised learning approaches. Besides, they are not robust to variation in samples and a small change would structurally change the tree [35].

2.3.8 Performance Evaluation

Model evaluation criteria are categorized into two main groups: The first category includes model evaluation metrics which are based on model predictive capabilities such as the classification error rate (PE). These type of metrics are employed by wrapper methods. The second group are statistical based indices, such as mutual information (MI) and correlation which are typically used within the FS algorithm (filter methods) to assess the obtained model as presented in [18] and [31]. In both, the Pearson correlation coefficient has been used, which not only assumes the data have a normal distribution, but is also applicable only on two-dimensional feature vectors. These shortcomings led to resort to other statistical indices proposed in the literature, namely the distance correlation (dCor) statistic presented in [73].

In this work, some model evaluation criteria specifically designed for imbalanced data such as microarrays, along with various validation techniques are employed to measure the performance of the method. In the following, some of them are described:

Clasification error rate/Accuracy/PE(Percentage Error) measures the proportion of true results among all, and is defined by the ratio of misclassified samples over the total number of tested samples. Specifically in binary classification problems, the definition is as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{N}} \tag{2.21}$$

in which N is the number of tested samples and TP and TN are the number of true positive and true negative classified samples. N can be reformulated as the sum of misclassified and correctly classified samples, *i.e.* N = TP + TN + FP + FN, where FP and FN are the number of false positive and false negative samples, respectively.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.22}$$

Sensitivity/recall/True Positive Rate (TPR) of the classifier defines the extent to which true positives are not missed with the rationale that a highly sensitive test should not overlook a positive sample. This is important particularly in medical applications, since detecting a positive sample and avoiding false negatives is crucial in the treatment process. In a sense, it can be said that in the set of misclassified samples, false negatives weigh more than

false positive ones. It is defined by the ratio of the number of true positive classified samples to all true samples.

$$\text{Sensitivity/TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.23}$$

Specificity/True Negative Rate (TNR) reveals the extent to which a classifier precisely detects positive samples, *i.e.* how much positively classified samples are really positive. Therefore, a highly specific test rarely misclassifies a negative sample as a positive one. It is defined by the ratio between correctly classified negative samples over the total number of negative samples.

$$\text{Specificity/TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{2.24}$$

Precision/positive predictive value is the fraction of true positives (*i.e.* the number of samples correctly classified as positive) among the total number of samples labeled as belonging to the positive class.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.25}$$

F-score/F-measure combines precision and recall and is actually the harmonic mean of them.

$$\text{F-score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.26}$$

Gmean is the geometric mean of TPR and TNR as is defined as:

$$\text{Gmean} = \sqrt{\text{TPR} \times \text{TNR}} \tag{2.27}$$

One statistical based metric is the *Pearson product-moment correlation coefficient/Pearson correlation coefficient* denoted by r, that measures the strength of the possible linear relationship between two variables, by fitting the best model using data points and determining how well the points actually fit the resulted model. It ranges from -1 to +1. A value of 0 signifies that there is no linear association between two variables and as the value is getting closer to +1 or -1, it indicates that the linear relationship, either positive or negative, is stronger. The formula for computing r among two feature vectors f_1 and f_2 is provided by 2.28.

$$r(f_1, f_2) = \frac{n\Big(\sum_{i=1}^{n} f_{1i} f_{2i}\Big) - \Big(\sum_{i=1}^{n} f_{1i}\Big)\Big(\sum_{i=1}^{n} f_{2i}\Big)}{\sqrt{\Big[n\sum_{i=1}^{n} f_{1i}^2 - \Big(\sum_{i=1}^{n} f_{1i}\Big)^2\Big]\Big[n\sum_{i=1}^{n} f_{2i}^2 - \Big(\sum_{i=1}^{n} f_{2i}\Big)^2\Big]}} \tag{2.28}$$

In which n is the number of samples, and f_{1_i} and f_{2_i} are the values of features f_1 and f_2 for the ith sample.

For Pearson's correlation to be a true indication of the relationship among variables, the following properties should hold:

1) The two variables should be continuous.

2) The data should not be significantly noisy, because the fitted model is highly sensitive to outliers. Due to the presence of noise in microarrays, this factor may cause issues regarding the ranking of variables containing noisy data.

3) Variables should approximately follow a normal distribution. However, this assumption hardly holds in microarray datasets.

4) The index is computed only for pairs of variables, so in case of feature subset evaluation in multivariate methods, some pairwise comparisons should be employed.

Because of the discussed shortcomings of metrics such as Pearson's, another statistical based criterion called *distance correlation index (dCor)* has been employed throughout this thesis to test the dependence of random vectors. It is proposed in [74] and refined in [73]. Unlike Pearson's correlation, dCor provides a reliable correlation-based dependence measure, between random vectors of arbitrary dimensions. It is also applicable to both continuous and discrete random variables, and does not require any *a priori* assumption on data distribution. It ranges from zero (for zero dependency) to 1 (a linear dependency) among vectors. In the next section, a brief assessment of the dCor criterion based on [74] is provided.

2.3.9 Distance Correlation Index (dCor)

Various statistical tests have been developed in the literature to test the dependence of random vectors. In this thesis, we exploit the distance correlation index which is a recently proposed approach by Szekely *et al.* [74],

for measuring the dependency between two arbitrary dimensional random variables.

Unlike Pearson's correlation coefficient, dCor can be applied on both discrete and continuous random variables with finite moments, and does not require any priori assumption about the underlying data distribution. For the sake of completeness, we here briefly report the main results of [74].

The basic dCor index

Let $\boldsymbol{x} = [x_1, \ldots, x_p]^T$ and $\boldsymbol{y} = [y_1, \ldots, y_q]^T$ be two random vectors, such that $\mathbb{E}(\|\boldsymbol{x}\| + \|\boldsymbol{y}\|) < \infty$, where $\|\cdot\|$ denotes the Euclidean norm. Let also $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}$ be N i.i.d. realizations of $\boldsymbol{x}$, and $\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(N)}$ the corresponding i.i.d. realizations of $\boldsymbol{y}$. Now, the empirical distance covariance (briefly, dCov) is defined as

$$\nu_N^2(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{N^2} \sum_{k,\, l=1}^{N} A_{kl} B_{kl}, \tag{2.29}$$

where

$$\begin{aligned} A_{kl} &= a_{kl} - \bar{a}_{k\cdot} - \bar{a}_{\cdot l} + \bar{a}_{\cdot\cdot}, \\ B_{kl} &= b_{kl} - \bar{b}_{k\cdot} - \bar{b}_{\cdot l} + \bar{b}_{\cdot\cdot}, \end{aligned}$$

with

$$a_{kl} = \|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(l)}\|, \quad b_{kl} = \|\boldsymbol{y}^{(k)} - \boldsymbol{y}^{(l)}\|,$$

and

$$\begin{aligned} \bar{a}_{k\cdot} &= \frac{1}{N} \sum_{l=1}^{N} a_{kl}, \quad \bar{a}_{\cdot l} = \frac{1}{N} \sum_{k=1}^{N} a_{kl}, \quad \bar{a}_{\cdot\cdot} = \frac{1}{N^2} \sum_{k,\, l=1}^{N} a_{kl}, \\ \bar{b}_{k\cdot} &= \frac{1}{N} \sum_{l=1}^{N} b_{kl}, \quad \bar{b}_{\cdot l} = \frac{1}{N} \sum_{k=1}^{N} b_{kl}, \quad \bar{b}_{\cdot\cdot} = \frac{1}{N^2} \sum_{k,\, l=1}^{N} b_{kl}. \end{aligned}$$

Then, the empirical dCor is the square root of

$$\mathcal{R}_N^2(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} \frac{\nu_N^2(\boldsymbol{x}, \boldsymbol{y})}{\sqrt{\nu_N^2(\boldsymbol{x}) \nu_N^2(\boldsymbol{y})}}, & \nu_N^2(\boldsymbol{x}) \nu_N^2(\boldsymbol{y}) > 0 \\ 0, & \nu_N^2(\boldsymbol{x}) \nu_N^2(\boldsymbol{y}) = 0 \end{cases} \tag{2.30}$$

In the assumption that $\mathbb{E}(\|\boldsymbol{x}\|+\|\boldsymbol{y}\|) < \infty$, it holds that the sampled version of the dCor tends to the corresponding probabilistic quantity, denoted $\mathcal{R}$:

$$\lim_{N\to\infty} \mathcal{R}_N^2(\boldsymbol{x},\boldsymbol{y}) = \mathcal{R}^2(\boldsymbol{x},\boldsymbol{y}). \tag{2.31}$$

It also holds that $0 \leq \mathcal{R}(\boldsymbol{x},\boldsymbol{y}) \leq 1$, and $\mathcal{R}(\boldsymbol{x},\boldsymbol{y}) = 0$ iff $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent. Similarly, $0 \leq \mathcal{R}_N(\boldsymbol{x},\boldsymbol{y}) \leq 1$, and if $\mathcal{R}_N(\boldsymbol{x},\boldsymbol{y}) = 1$, then there exist a vector ζ, a nonzero real number τ and an orthogonal matrix C such that $\boldsymbol{Y} = \zeta + \tau\boldsymbol{x}C$.

In view of the last property, $\mathcal{R}_N(\boldsymbol{x},\boldsymbol{y})$ can be indeed used as a measure of the linear dependence between random vectors. It can be verified that the proposed index is also sensitive to nonlinear input-output relationships.

The Unbiased dCor Index

It is worth mentioning that the bias of the dCor index increases with the dimension of the random vectors. As discussed in [73], for fixed number of samples N the dCor tends to 1 as $p, q \to \infty$. Thus, it might be hard to interpret the obtained index in high dimensional cases. This problem is investigated in [73] where an unbiased version of the dCor index is introduced, which is amenable to high dimensional problems. Here, the following quantities A_{kl}^* and B_{kl}^* are used instead of A_{kl} and B_{kl}:

$$A_{kl}^* = \begin{cases} \frac{N}{N-1}(A_{kl} - \frac{a_{kl}}{N}), k \neq l \\ \frac{N}{N-1}(\bar{a}_{k\cdot} - \bar{a}_{\cdot\cdot}), k = l \end{cases} \tag{2.32}$$

$$B_{kl}^* = \begin{cases} \frac{N}{N-1}(B_{kl} - \frac{b_{kl}}{N}), k \neq l \\ \frac{N}{N-1}(\bar{b}_{k\cdot} - \bar{b}_{\cdot\cdot}), k = l \end{cases} \tag{2.33}$$

Let

$$\mathcal{U}_N^*(\boldsymbol{x},\boldsymbol{y}) = \sum_{k\neq l}^{N} A_{kl}^* B_{kl}^* - \frac{2}{N-2}\sum_{k=1}^{N} A_{kk}^* B_{kk}^*. \tag{2.34}$$

The modified dCov and dCor indices are given respectively by:

$$\nu_N^*(\boldsymbol{x},\boldsymbol{y}) = \frac{\mathcal{U}_N^*(\boldsymbol{x},\boldsymbol{y})}{N(N-3)}, \tag{2.35}$$

$$\mathcal{R}_N^*(\boldsymbol{x}, \boldsymbol{y}) = \frac{\nu_N^*(\boldsymbol{x}, \boldsymbol{y})}{\sqrt{\nu_N^*(\boldsymbol{x})\nu_N^*(\boldsymbol{y})}}. \tag{2.36}$$

For simplicity, in the rest of document, we will drop the asterisk symbol and use the notation R_N to denote the unbiased dCor index.

Sensitivity of the dCor to redundant terms

In this section we present some illustrative simulations that emphasize the robustness of the dCor index in the presence of redundant terms. Let $\boldsymbol{x} = [x_1, \ldots, x_6]^T$ be a random vector and $y = 3x_1$, and assume that N i.i.d. realizations of both $\boldsymbol{x}$ and y are available. All elements of the $\boldsymbol{x}$ vector are independently drawn from the same distribution. Table 2.11 reports the dCor value calculated for different subsets of inputs on average over 1000 Monte Carlo tests performed for data generated with different distributions (Normal, Poisson and Lognormal). The evaluated input subsets are $\{x_1, \ldots, x_{1+k}\}$, for $k = 0, \ldots, 5$, corresponding to the exact model and 5 redundant models with increasing number of redundant terms. While the dCor equals 1 for the true model (including only x_1), its value decreases as we introduce further terms, regardless of the distribution of the data.

Number of redundant terms	Data distribution		
	Normal	Poisson	Lognormal
0	1.0000	1.0000	1.0000
1	0.9873	0.9835	0.9765
2	0.9778	0.9729	0.9573
3	0.9697	0.9640	0.9406
4	0.9623	0.9560	0.9262
5	0.9555	0.9488	0.9130

Table 2.11: Average dCor measure over 1000 Monte Carlo tests for increasingly redundant models (true model: $y = 3x_1$).

A similar result holds even if the input-output relationship is nonlinear, *e.g.* $y = 3x_1^2$, although this time the dCor associated to the model containing only x_1 is less than 1: any further term added to the model decreases the dCor. The results are reported in Table 2.12.

Inspecting the results presented in Tables 2.11-2.12 leads to the conclusion that the dCor index is highly sensitive to the presence of redundant terms, and is maximal in the absence of redundant terms. This property proves to

Number of redundant features	Data distribution Normal	Poisson	Lognormal
0	0.5731	0.9682	0.9221
1	0.5447	0.9570	0.9106
2	0.5261	0.9490	0.8997
3	0.5120	0.9419	0.8897
4	0.5008	0.9352	0.8808
5	0.4916	0.9289	0.8716

Table 2.12: Average dCor measure over 1000 Monte Carlo tests for increasingly redundant models (true model: $y = 3x_1^2$).

be crucial for the detection of redundant terms in the FS task.

dCor dependence test

The dCor index can be employed in order to design a statistical test for measuring the dependence between two random vectors.
Let x and z be two random variables such that $\mathbb{E} = \|x\| + \|z\| < \infty$, where $\|\cdot\|$ denotes the absolute value. Let $H_0 : x$ *and* z *independent* be the null hypothesis.

Then, the statistical test proposed in [74] rejects H_0 if

$$\frac{N\,\nu_N^2(x,z)}{S} > \mathcal{N}^{-1}\Big(1-\frac{\alpha_d}{2}\Big)^2, \tag{2.37}$$

where $\mathcal{N}(\cdot)$ denotes the normal cumulative distribution function, α_d is the significance level of the test, and

$$S = \bar{a}_{..}\bar{b}_{..} \tag{2.38}$$

This test can be employed fruitfully in FS, to retain only features for which there is enough statistical evidence to reject the independence hypothesis.

2.3.10 Validation Methods

In order to evaluate the algorithm's performance and its true generalization capabilities, the model should be tested against some unseen data. For datasets that do not provide a separated test set, several validation tech-

niques have been suggested in the literature. Some well-known methods are introduced in this part.

Hold-Out Cross Validation (HOCV) is a commonly used validation approach, specifically in datasets which are not originally separated into training and test sets. The hold out method reserves some samples as test data and only the training part is used for the learning process. The performance of the classifier is then computed based on the errors in the test part. Some problems might arise in this approach. Indeed the error rate critically depends on the data division. For example, the samples in the training or test data might not be truly representative due uneven distribution of features and/or labels. Of course it is possible to control the distribution of class labels in the partitions, but the error is still biased due to the division. On top of that, the classifier learning is done based on fewer samples which might be insufficient. This approach can be improved by the *repeated holdout method*, which consists in repeating the same process for different randomly selected subsamples. It is more reliable since the final error rate is the average of all iterations, but, still problematic. The divisions might overlap or some samples might not be present in the test or training set at all.

K-Fold Cross Validation (KFCV) is another technique to evaluate the performance of a classifier. In this method, the data are split into K disjoint parts. Then, one part is reserved for testing the classifier and the rest for training. This is repeated for K times, each time a different section serving as the test data. Then the average error rate among the K iterations is computed. This approach guarantees that all samples appear in the test data once, but still the classifier is trained with fewer samples.

Leave One Out Cross Validation (LOOCV) is a special case of K-fold cross validation in which K equals the number of samples. At each iteration, only one sample is saved for testing and the rest are used to train the classifier. The iterations are equal to the number of samples and the final result is the average of all steps. This method not only removes the bias towards divisions that existed in the hold out method, but also resolves the problem of limitations in small size data, at a cost of an increased computational complexity. Employing this method, learning is performed using larger sets of data, so there is a higher chance of constructing a more accurate classifier. It also offers more reliability because the evaluation at each iteration is based on results on unseen data. Besides, the expected value of the error is closer to the true value due to the increased number of iterations.

Chapter 3

Distributed multivariate filter-based probabilistic framework

In this chapter, the distributed combinatorial optimization approach and the probabilistic framework at the basis of the proposed FS algorithm are discussed. We introduced distributed optimization scheme and the D^2CORFS algorithm in the first section. Then in the next section the DCORFS algorithm is explained.

3.1 Distributed optimization scheme

Feature selection is inherently a combinatorial problem and its complexity grows rapidly, as the number of features increases, and in case of large-sized data, it may become prohibitive. More importantly, in addition to the growth in the FS method's computational complexity, the ability of the FS algorithms to reach the optimal feature subset diminishes with the increase in the number of features, due to the corresponding exponential growth of the search space. This is the case for all FS methods, although with different levels. The number of possible feature subsets is equal to $2^{N_f} - 1$ and the search strategy adopted to explore the feature subset space is crucial.

To increase the efficiency of the search process on the subset space, a *distributed* combinatorial optimization approach has been introduced, which

exploits vertical partitioning of features and the exchange of information among these partitions. The main idea of the proposed scheme is to break down the complexity of the FS task into smaller tasks, which are not as dimensionally unbalanced as the original problem. Indeed, FS turns out to be more complicated when there are significantly more features than samples. In this scheme, however, FS task can be separately run on each division with much fewer features instead of the whole feature space. Besides, it is important to note that this distributed optimization scheme is generic and applicable with any FS method, meaning that the different feature selection methods can be performed locally on partitions and the obtaining results can be shared globally with other groups through an information exchange stage. This distributed FS architecture has been first employed in [10], and is also used in this thesis, to develop an applicable algorithm for distributed feature selection named D^{2}CORFS. To achieve this, different processors are responsible for performing the FS tasks on smaller subset of features separately and independently. Then the local results obtained in each partition are shared and combined with other optimization processors. Consequently, the original FS task can be improved by selecting and combining the features obtained from local searches that were the most promising [10].

D^{2}CORFS [8] is a novel multivariate filter method that exploits the distance correlation (dCor) index introduced in section 2.3.9 as a model evaluation metric to compare different feature subsets in each partition. Compared to the situation where the whole feature space is explored, the suggested scheme provides a more favorable framework for the application the dCor due to more balance between the number of features and the number of samples obtained from the vertical division. A sketch of the proposed scheme is depicted in Fig. 3.1. In the first step, the data are vertically partitioned, *i.e.* the full feature set $\mathcal{F} = \{f_1, ..., f_{N_f}\}$ is (randomly) divided into a number of non-overlapping subsets (denoted feature bins in the sequel) $\mathcal{F}_b^{(0)}$, $b = 1, \ldots, N_b$ of approximately the same size. The number of features in bins is a critical parameter to set in the algorithm, especially regarding the robustness with respect to the overfitting issue. The tuning of this parameter depends on many factors such as the adopted FS algorithm (FS$(\cdot)$), the number of features (N_F) and the sample size (N).Hence, some level of trial and error is required to tune it. In this work, following [6], we set $N_b = 2N_f/N$, which results in $N/2$ features for each bin.

An independent FS task is carried out on each feature bin $\mathcal{F}_b = \mathcal{F}_b^{(0)}$, according to the selection strategy of choice. The obtained solutions are compared and the one with the best performance (denoted $\mathcal{S}^*$) is shared

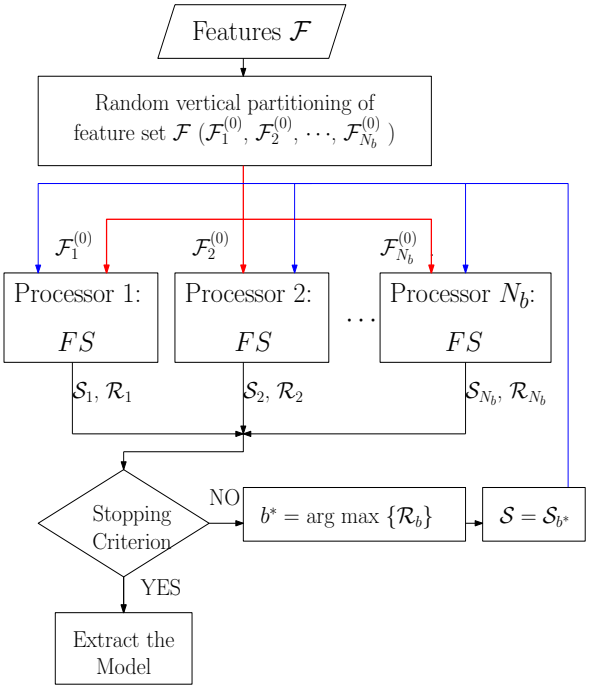

Figure 3.1: Flowchart of the proposed distributed scheme.

among all feature bins through feed back in next iterations. In other words, $\mathcal{F}_b = \mathcal{F}_b^{(0)} \cup \mathcal{S}^*$, *i.e.* each feature bin is reset to its initial state $\mathcal{F}_b^{(0)}$ and then augmented with the features corresponding to the current local best solution. The procedure is then repeated iteratively, alternating the execution of the independent FS tasks with the information exchange phase until convergence.

The information exchange phase guarantees that each feature bin contains the features of the best solution found so far, so that the new solution can only improve over the previous best (at least in principle). At the same time the dimension of the feature bins is kept low during all the selection process, so that the individual FS problems have appropriate feature-sample balancing. The procedure terminates when it is not possible to find a better solution than the previous best in any feature bin and all problems yield the same solution (alternative termination conditions can be applied, as explained later on).

A pseudocode of the proposed scheme is given in Algorithm 1. The local FS method applied at each processor is denoted DCORFS algorithm (see Section 3.2). The input parameters N_p, N_i, $\mu^{(0)}$, $\bar{\mu}$, and ϵ are actually input arguments of the DCORFS function and will be explained later. The other

inputs are the set of input/output observation pairs $\mathcal{D} = \{(\boldsymbol{u}_s^{(k)}, c^{(k)}), k = 1, \ldots, N\}$, the full set of features $\mathcal{R}$, the number of feature bins N_u, the maximum number iterations N_i, and the maximum number of allowed rounds of algorithm N_F. The algorithm returns the selected feature subset $\mathcal{E}^\star \subseteq \mathcal{R}$, along with its dCor value $\mathcal{R}_N^\star$.

The main loop goes from line 3 to 22. The vertical partitioning in N_b bins is carried out at line 1, *i.e.*, the original feature set $\mathcal{F} = f_1, ..., f_{N_f}$ is divided into N_b disjoint subsets (denoted feature bins in the sequel $\mathcal{F}_b^{(0)}$, $b = 1, \ldots, N_b$), which are selected randomly, and their sizes are approximately the same. The number of feature bins N_b is a critical parameter, especially regarding the robustness with respect to the overfitting issue. In this work, following [6], we set $N_b = 2N_f/N$, which results in $N/2$ features for each bin.

In lines 5 to 12 an independent local feature selection is carried out on each feature bin $\mathcal{F}_b = \mathcal{F}_b^{(0)}$, according to the selection strategy of choice. The obtained solutions are compared and the one with the best performance (denoted $\mathcal{S}^*$) is shared among all feature bins. In other words, $\mathcal{F}_b = \mathcal{F}_b^{(0)} \cup \mathcal{S}^*$, *i.e.* each feature bin is reset to its initial state $\mathcal{F}_b^{(0)}$ and then augmented with the features corresponding to the current local best solution. The procedure is then repeated iteratively, alternating the execution of the independent FS tasks with the information exchange phase until convergence.

The information exchange phase guarantees that each feature bin contains the features of the best solution found so far, so that the new solution can only improve over the previous best (at least in principle). At the same time the dimension of the feature bins is kept low during all the selection process, so that the individual FS problems have appropriate feature-sample balancing.

Finally, the termination conditions are given from line 13 to 21 which are explained in section 3.1.1. To avoid the overloading of irrelevant terms, the algorithm provides the option to share only the top ranked local best models at each iteration among the processors in the subsequent iteration. At the end, the algorithm returns the selected feature subset $\mathcal{S}^\star$ along with the corresponding classification performance $\mathcal{R}^\star$.

Algorithm 1 D^{2}CORFS

Input: $\mathcal{D}$, $\mathcal{F}$, N_b, N_r, N_i, N_p, $\mu^{(0)}$, $\bar{\mu}$, ϵ.
Output: $\mathcal{S}^\star$, $\mathcal{R}^\star$.

1: $\mathcal{F} = \mathcal{F}_1^{(0)} \cup \cdots \cup \mathcal{F}_{N_b}^{(0)}$
2: $\mathcal{S} = \emptyset$, $\mathcal{S}^\star = \emptyset$, $\mathcal{R}^\star = 0$
3: **for** $r = 1$ **to** N_r **do**
4: $\quad \mathcal{S} \leftarrow \mathcal{S}^\star$
5: $\quad$ **for** $b = 1$ **to** N_b **do**
6: $\quad\quad \mathcal{F}_b = \mathcal{F}_b^{(0)} \cup \mathcal{S}$
7: $\quad\quad (\mathcal{S}_b, \mathcal{R}_b) = \text{DCORFS}(D, \mathcal{F}_b, N_i, N_p, \mu^{(0)}, \bar{\mu}, \epsilon)$
8: $\quad\quad$ **if** $\mathcal{R}_b > \mathcal{R}^\star$ **then**
9: $\quad\quad\quad \mathcal{S}^\star \leftarrow \mathcal{S}_b$, $\mathcal{R}^\star \leftarrow \mathcal{R}_b$
10: $\quad\quad\quad$ **if** $\mathcal{R}^\star = 1$ **then return end if**
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: $\quad$ **if** $\cup_b \mathcal{S}_b = \cap_b \mathcal{S}_b$ **then return end if**
14: $\quad \mathcal{R}_{vec}^\star(r) = \mathcal{R}^\star$
15: $\quad$ **if** $r = N_r$ **then**
16: $\quad\quad$ **return**
17: $\quad$ **else if** $r \geq 3$ **then**
18: $\quad\quad$ **if** $(\mathcal{R}_{vec}^\star(r-1) = \mathcal{R}_{vec}^\star(r-2) = \mathcal{R}^\star)$ **then**
19: $\quad\quad\quad$ **return**
20: $\quad\quad$ **end if**
21: $\quad$ **end if**
22: **end for**

3.1.1 Termination Conditions

The termination conditions for the procedure are stated in line 10, 13, 16 and 19.

The first condition (line 10) states that if a model with maximum possible dCor value (*i.e*, it achieves perfect classification) is obtained at any local processor, the algorithm will be stopped because that solution is also a global optimizer.

An alternative condition stated in line 13 checks if all the local processors converge on the same solution.

Another condition that terminates the algorithm prematurely is the exceeding of the maximum number of rounds (r_{max}) which is a user defined parameter as an input and is shown in line 15.

The last enforced condition is when the algorithm fails to improve the current best solution over a number (*e.g.*, 3) of subsequent rounds (line 18).

3.2 The DCORFS algorithm

Multivariate filter methods based on sequential search strategies are probably the most popular approaches in dealing with large datasets because they are capable of removing redundant terms, at least partially. As already mentioned, however, sequential strategies have significant drawbacks, essentially originating from the fact that model variations are enforced based on a local assessment of the features (*e.g.*, a new term is added because it significantly improves the *current* model). For this reason, in this thesis, we introduce a novel multivariate filter FS algorithm based on the unbiased dCor criterion which employs a different search strategy, in that it implements model variations based on a *global* assessment of the features. In other words, unlike other (multivariate) filter methods, the importance of the features is evaluated not just individually or in pairs, but based on populations of extracted models.

The FS problem amounts to solving an optimization problem, whose objective is to find the subset of features $\mathcal{S}^* \subseteq \mathcal{F}$ that maximizes the dCor index $\mathcal{R}_N(\boldsymbol{f}_\mathcal{S}, c)$. A convenient way to tackle the problem above exploits the probabilistic reformulation of [22] (used to develop a wrapper FS method in [9]) obtained by associating a discrete random variable ϕ to the feature subsets $\mathcal{S}$ according to a probability distribution $\mathcal{P}_\phi$, which expresses the probability of each feature subset $\mathcal{S}$ to coincide with the target one. Accordingly, the dCor index becomes a function of $\boldsymbol{f}_\phi$ and is therefore a random variable with expectation given by

$$\mathbb{E}[\mathcal{R}_N(\boldsymbol{f}_\phi, c)] = \sum_{\mathcal{S} \subseteq \mathcal{F}} \mathcal{R}_N(\boldsymbol{f}_\mathcal{S}, c) \mathcal{P}_\phi(\mathcal{S}). \tag{3.1}$$

The expected value (3.1) is maximal if the mass of distribution $\mathcal{P}_\phi$ is all concentrated on the feature subset with highest dCor $\mathcal{S}^\star$. Therefore, the problem of finding $\mathcal{S}^\star$ can be reformulated as that of finding the target limit distribution

$$\mathcal{P}_\phi^\star = \arg\max_{\mathcal{P}_\phi} \mathbb{E}[\mathcal{R}_N(\boldsymbol{f}_\phi, c)], \tag{3.2}$$

such that $\mathcal{P}_\phi^\star(\mathcal{S}^\star) = 1$.

To model the probability that a feature $f_j \in \mathcal{S}^\star$, we parameterize $\mathcal{P}_\phi$ by associating a Bernoulli random variable ρ_j to each feature f_j:

$$\rho_j \sim \text{Be}(\mu_j), \quad \mu_j \in [0,1]$$

$j = 1, \ldots, N_f$, where μ_j denotes the feature inclusion probability (FIP) of the jth feature. The FIP represents how likely it is that a given feature is included in the solution. Initially, the FIPs are set to values which may reflect the prior knowledge on the most promising features or simply assign an equal probability to all of them. Then, the distribution is iteratively refined by taking into account the information gathered by sampling it. More in detail, at every iteration a population of feature subsets is extracted using the current Bernoullian distributions. The greater ρ_j, the more it is probable that f_j will be selected to be in the solution. Then each obtained feature subset is evaluated with the dCor criterion. After that, all features are assessed individually using (a sampled version of) the index $\mathcal{I}_j$ given by

$$\mathcal{I}_j = \mathbb{E}[\mathcal{R}_N(\boldsymbol{f}_\phi, c)|f_j \in \phi] - \mathbb{E}[\mathcal{R}_N(\boldsymbol{f}_\phi, c)|f_j \notin \phi], \tag{3.3}$$

for $j = 1, \ldots, N_f$. Index $\mathcal{I}_j$ compares the dCor criterion of the features subsets that include f_j with that of the remaining ones and thus can be interpreted as a global measure of the feature's importance. The greater the dCor value for the population of models containing a feature with respect to the remaining ones, the greater is the importance of that feature. Finally, the probability distribution is updated according to the update rule given by

$$\mu_j(i+1) = \text{sat}(\mu_j(i) + \gamma \mathcal{I}_j) \tag{3.4}$$

where i is the current iteration and $\text{sat}(\cdot)$ is a saturating function that ensures that μ_j remains within the $[0,1]$ interval. Parameter γ in (3.4) is an adaptive step-size defined as:

$$\gamma = \frac{1}{\lambda(\mathcal{R}_{\max} - \bar{\mathcal{R}}) + 0.1}, \tag{3.5}$$

where λ is a design coefficient that inversely affects the step size of the FIP update rule. $\mathcal{R}_{\max}$ and $\bar{\mathcal{R}}$ are the maximum and the average of the dCor values of the extracted feature subsets. If $\mathcal{R}_{\max}$ is much larger than $\bar{\mathcal{R}}$, the extracted model population does not convey sufficient information to evaluate the feature importance, so γ should be smaller. In other words,

the rationale behind γ is that it should be larger if the averaged index $\mathcal{I}_j$ is reliable (small variance of the dCor values), and smaller otherwise.

The iterative procedure terminates upon convergence of the probability distribution (or if the maximum number of iterations is exceeded). The selected feature subset is given by $\mathcal{S}^\star = \{f_j | \mu_j \geq \bar{\mu}\}$, where $\bar{\mu}$ is the prescribed acceptance threshold. A pseudocode of the proposed DCORFS algorithm is given below (see Algorithm 2).

Algorithm 2 DCORFS

Input: $\mathcal{D}$, $\mathcal{F}_b$, N_i, N_p, $\mu^{(0)}$, $\bar{\mu}$, ϵ
Output: $\mathcal{S}^\star$, $\mathcal{R}^\star$

for $j = 1$ **to** $|\mathcal{F}_b|$ **do**
  $\mu_j \leftarrow \mu^{(0)}$ FIP initialization
end for
for $i = 1$ **to** N_i **do**
  for $p = 1$ **to** N_p **do**
    $\phi_p \sim \mathcal{P}_\phi$ Extract sample feature subset
    $\mathcal{R}^p \leftarrow \mathcal{R}_N(\boldsymbol{f}_{\phi_p}, c)$ Compute dCor with (2.36)
  end for
  $\mathcal{R}_{\max} \leftarrow \max(\mathcal{R}^1, \dots \mathcal{R}^{N_p})$
  $\bar{\mathcal{R}} = \frac{1}{N_p} \sum_{p=1}^{N_p} \mathcal{R}^p$
  $\gamma = \frac{1}{\lambda(\mathcal{R}_{\max} - \bar{\mathcal{R}}) + 0.1}$
  for $j = 1$ **to** $|\mathcal{F}_b|$ **do**
    $\mathcal{I}_j \leftarrow \frac{\sum_{p|f_j \in \phi_p} \mathcal{R}^p}{\sum_{p|f_j \in \phi_p} 1} - \frac{\sum_{p|f_j \notin \phi_p} \mathcal{R}^p}{\sum_{p|f_j \notin \phi_p} 1}$
    $\mu_j \leftarrow \mathrm{sat}(\mu_j + \gamma \mathcal{I}_j)$ FIP update
  end for
  if $\max_{j=1,\dots,|\mathcal{F}_b|} |\mu_j(i) - \mu_j(i-1)| \leq \epsilon$ **then**
    break
  end if
end for
$\mathcal{S}^\star \leftarrow \emptyset$
for $j = 1$ **to** $|\mathcal{F}_b|$ **do**
  if $\mu_j \geq \bar{\mu}$ **then** $\mathcal{S}^\star \leftarrow \mathcal{S}^\star \cup \{f_j\}$ **end if**
end for
$\mathcal{R}^\star = \mathcal{R}_N(\boldsymbol{f}_{\mathcal{S}^\star}, c)$

The required inputs are the observations $\mathcal{D}$, the set of features $\mathcal{F}_b$ on which to perform the search, the maximum number of iterations N_i, the number of feature subsets to be extracted from the current distribution at each iteration N_p, the initial value of the FIPs $\mu^{(0)}$, the acceptance threshold $\bar{\mu}$,

and a convergence threshold ϵ. The algorithm returns the selected feature subset $\mathcal{S}^\star$, along with its dCor value $\mathcal{R}^\star$.

Chapter 4

Experimental Results

In this chapter we report the results of a series of experiments carried out to assess the performance of the proposed algorithm on eight well known microarray benchmarks: Breast, CNS (Central Nervous System), Colon, DLBCL (Diffuse Large B-Cell Lymphoma), Leukemia, Lung, Ovarian and Prostate cancers. The Breast dataset provides gene information retrieved from tumor material belonging to breast cancer patients, distinguishing those that developed metastases within 5 years from the other ones. The CNS dataset documents both failed and succeeded treatment cases of embryonal CNS tumors. The Colon dataset contains expression levels of 2000 genes for several colon tissue samples including both normal and cancerous ones. Gene analysis of diagnostic tumor specimens from DLBCL patients having received a specific chemotherapy treatment is reported in the DLBCL dataset, distinguishing between cured and fatal or refractory disease cases. The Leukemia dataset contains gene information extracted from bone marrow and peripheral blood samples of several leukemia patients, corresponding either to Acute Lymphoblast Leukemia (ALL) or Acute Myeloid Leukemia (AML). The Lung dataset provides genetic information regarding both Malignant Pleural Mesothelioma (MPM) and lung ADenoCArcinoma (ADCA) cases. The Ovarian dataset aims to identify ovarian cancer from proteomic patterns in serum. Finally, the Prostate dataset contains the expression level of 12600 genes for more than 100 tissue samples, a part of which are taken from prostate tumors. The main characteristics of the considered microarray datasets are summarized in Table 4.1 (see [7] and section 2.1.6 for a comprehensive review of these and other microarray datasets and specific references).

All eight datasets are biclass problems. The number of original features ranges from a few thousands to almost 25000. To reduce the feature search space, a dCor-based feature screening (with $\alpha_d \geq 0.9$ [1]) was applied as a preprocessing step to all the datasets except Colon and DLBCL, that already have a sufficiently small feature set. The number of samples is generally relatively small, with the exception of the Ovarian cancer dataset. Table 4.1 also reports the distribution of the samples over the classes both for the training test and the test set, when applicable (NP and NN are the total number of samples belonging to class 1 and 2, respectively). The class imbalance, measured as the skew ratio $\sigma = \frac{NP}{NN}$, is also given for both the training (σ_{tr}) and the test (σ_{te}) data, respectively. This information is important, since it is related to the achievable accuracy and reliability of classification algorithms across classes [32].

Half of the datasets (Breast, Leukemia, Lung and Prostate) are provided with a given training/test data subdivision, while the CNS, Colon, DLBCL and Ovarian datasets are not. For this reason, we analyzed first the former group of datasets with a Hold-Out Cross Validation (HOCV) method, using the training data to learn the model and the test data for its evaluation. Though the HOCV method is in principle applicable also to the other datasets, the results would be impossible to compare with the literature, in the absence of a nominal training-test data subdivision (subsection 4.4 discusses the sensitivity of the identification results to variations of the data subdivision). Therefore, a second analysis is performed, this time evaluating all datasets with a Leave-One-Out Cross Validation (LOOCV) approach, which is a particular case of k-Folds Cross Validation (k-FCV), with $k = N$. Briefly, the dataset is split into k equal (or, at least, balanced in size) and non-overlapping subsets (folds), possibly uniformly representative of all classes. Then, $k - 1$ folds are used for training and the remaining ones for testing, the procedure being repeated k times so that all folds are left once for testing. The algorithm performance is finally computed as the average over the k independent runs.

[1] Different values of α_d were used in the feature screening process depending on the adopted validation method, since the latter influences the distribution of the samples.

Table 4.1: Main characteristics of the considered microarray datasets.

Dataset	# features	after screening		class labels	Training set			Test set			σ_{tr}	σ_{te}
	total	HOCV	LOOCV		total	NP	NN	total	NP	NN		
Breast	24481	4990	2132	Relapse/non-Relapse	78	34	44	19	12	7	1.29	0.58
CNS	7129	–	1415	Class0/Class1	60	21	39	–	–	–	1.86	
Colon	2000	–	–	*n.a.*	62	22	40	–	–	–	1.82	
DLBCL	4026	–	–	Cured/Fatal	77	58	19	–	–	–	0.33	
Leukemia	7129	2688	2812	ALL/AML	38	13	25	34	10	24	1.92	2.40
Lung	12533	1872	2585	Mesothelioma/ADCA	32	16	16	149	15	134	1.00	8.93
Ovarian	15154	–	3368	Cancer/Normal	253	162	91	–	–	–	0.57	
Prostate	12600	2053	2472	Relapse/non-Relapse	102	52	50	34	25	9	0.96	0.36

Note. ALL = Acute Lymphoblastic Leukemia, AML = Acute Myeloid Leukemia.

The original features have been normalized in the $[0, 1]$ range according to:

$$\bar{f}_j^{(k)} = \frac{f_j^{(k)} - f_{p_{min}}}{f_{j_{max}} - f_{j_{min}}}, \tag{4.1}$$

for $k = 1, \ldots, N$, $j = 1, \ldots, N_f$, where $\bar{f}_j^{(k)}$ is the normalized numeric value of the kth observation of the jth feature in a given dataset, and $f_{j,\max}$ and $f_{j,\min}$ denote the maximum and minimum values of the same feature in the dataset, respectively.

To evaluate the performance of the proposed FS method we trained different classifiers on the selected features, namely two support vector machines (SVMs) with linear and nonlinear decision boundaries, a k-nearest neighbor (kNN, with $k = 5$), a logistic regression, an LDA, a naïve Bayes (NB) and a tree based classifier.

We employed various evaluation criteria especially designed to account for class imbalanced data. The sensitivity of the classifier is measured by the true positive rate $TPR = \frac{TP}{TP+FN}$, *i.e.* the ratio of the correctly classified positive samples over the total number of positive samples. Conversely, the specificity is captured by the true negative rate $TNR = \frac{TN}{TN+FP}$, *i.e.* the ratio of the correctly classified negative samples over the total number of negative samples. The *Gmean* $G = \sqrt{TPR \cdot TNR}$ and *Fscore* $F = 2\frac{TPR \cdot TNR}{TPR+TNR}$ indices combine both criteria (see Section 2.3.8 for more details).

The initial parameter setup for the D^{2}CORFS in the experiments is as follows: a maximum of $N_r = 5$ rounds is allowed for the distributed search scheme and the size of the feature bins is set as close as possible to $N/2$, so as to have ideally balanced datasets in the local FS problems. As for the DCORFS algorithm operating on each feature bin, the number of iterations is limited to $N_i = 100$, the number of feature subset extractions at each iteration is set to $N_p = 100$, the initial FIPs are set to $\mu_0 = 1/|\mathcal{F}_b|$, $\epsilon = 0.001$, and the acceptance threshold is $\bar{\mu} = 0.98$. The proposed algorithm was implemented in Matlab (version 2016a) and executed on an Intel(R) Core i7-3630QM machine, with 2.4GHz CPU, 8GB of RAM, and a 64-bit Operating System.

4.1 Performance analysis of D^{2}CORFS with HOCV

We first analyze the four datasets with explicit training-test division which can be addressed with the HOCV approach using the native training and test sets. Table 4.2 reports the best subset of features selected for each case, as well as the performances obtained with mentioned classifiers. A brief description of the selected genes is given in appendix A. The results are assessed in terms of the classification accuracy on the training (J_{tr}) and the test data (J_{te}), as well as TPR, TNR, G, and F. Apparently, the FS procedure selected very compact models in all cases, with 4 features at most, and a high classification accuracy was obtained (the results compare quite favorably with the literature, as shown later in Table 4.6).

Interestingly enough, though the Leukemia data are imbalanced in favor of negative samples, the obtained classifiers score better on the TPR index, than on the TNR. The computational time is sufficiently low, the lowest computational cost being observed for the Lung dataset. Indeed, the Lung dataset has the smallest number of samples, which in turn causes the size of the feature bins to be particularly small resulting in a very high computational efficiency.

Dataset	Best model ($\mathcal{S}^\star$)	Time [s]	Classifier	J_{tr}	J_{te}	TNR_{te}	TPR_{te}	G_{te}	F_{te}
Breast	$\{f_{512}, f_{3224}, f_{5377}, f_{10889}\}$	386.29	SVM (linear)	0.8077	0.8947	0.8571	0.9167	0.8864	0.8859
			SVM (nonlinear)	0.8974	0.5790	0.8571	0.4167	0.5976	0.5608
			KNN	0.8462	0.8421	0.8571	0.8333	0.8452	0.8451
			Logistic Regression	0.8077	0.7895	1.0000	0.6667	0.8165	0.8000
			LDA	0.8205	0.8947	0.8571	0.9167	0.8864	0.8859
			NB	0.8590	0.8947	0.8571	0.9167	0.8864	0.8859
			Tree	0.8974	0.7368	0.5714	0.8333	0.6901	0.6780
Leukemia	$\{f_{1924}, f_{3252}, f_{4847}, f_{5039}\}$	154.01	SVM (linear)	1.0000	0.9118	0.8750	1.0000	0.9354	0.9333
			SVM (nonlinear)	1.0000	0.8235	0.8750	0.7000	0.7826	0.7778
			KNN	1.0000	0.9118	0.8750	1.0000	0.9354	0.9333
			Logistic Regression	0.8421	0.3235	0.0417	1.0000	0.2041	0.0800
			LDA	1.0000	0.9118	0.8750	1.0000	0.9354	0.9333
			NB	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
			Tree	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Lung	$\{f_{3334}, f_{6571}, f_{11841}\}$	51.72	SVM (linear)	0.9688	0.9933	1.0000	0.9333	0.9661	0.9655
			SVM (nonlinear)	0.9688	1.0000	1.0000	1.0000	1.0000	1.0000
			KNN	0.9688	1.0000	1.0000	1.0000	1.0000	1.0000
			Logistic Regression	0.5000	0.1007	0.0000	1.0000	0.0000	0.0000
			LDA	0.9688	0.9799	1.0000	0.8000	0.8944	0.8889
			NB	1.0000	0.9128	0.9030	1.0000	0.9503	0.9490
			Tree	1.0000	0.8993	0.9030	0.8667	0.8847	0.8844
Prostate	$\{f_{4282}, f_{6185}, f_{8965}, f_{10494}\}$	295.78	SVM (linear)	0.9216	0.9706	1.0000	0.9600	0.9798	0.9796
			SVM (nonlinear)	0.9706	0.4412	0.7778	0.3200	0.4989	0.4534
			KNN	0.9510	0.9118	0.8889	0.9200	0.9043	0.9042
			Logistic Regression	0.9608	0.9412	1.0000	0.9200	0.9592	0.9583
			LDA	0.9314	0.9412	1.0000	0.9200	0.9592	0.9583
			NB	0.9314	0.8530	1.0000	0.8000	0.8944	0.8889
			Tree	0.9510	0.4412	1.0000	0.2400	0.4899	0.3871

Table 4.2: Performance of the best models obtained with D^2CORFS and HOCV, using various types of classifiers

4.2 Distribution of the feature values

A detailed analysis of the obtained models, with focus on the selected features, reveals several interesting aspects. Fig. 4.1 shows the values of the selected features for all samples, divided by class and training/test subset.

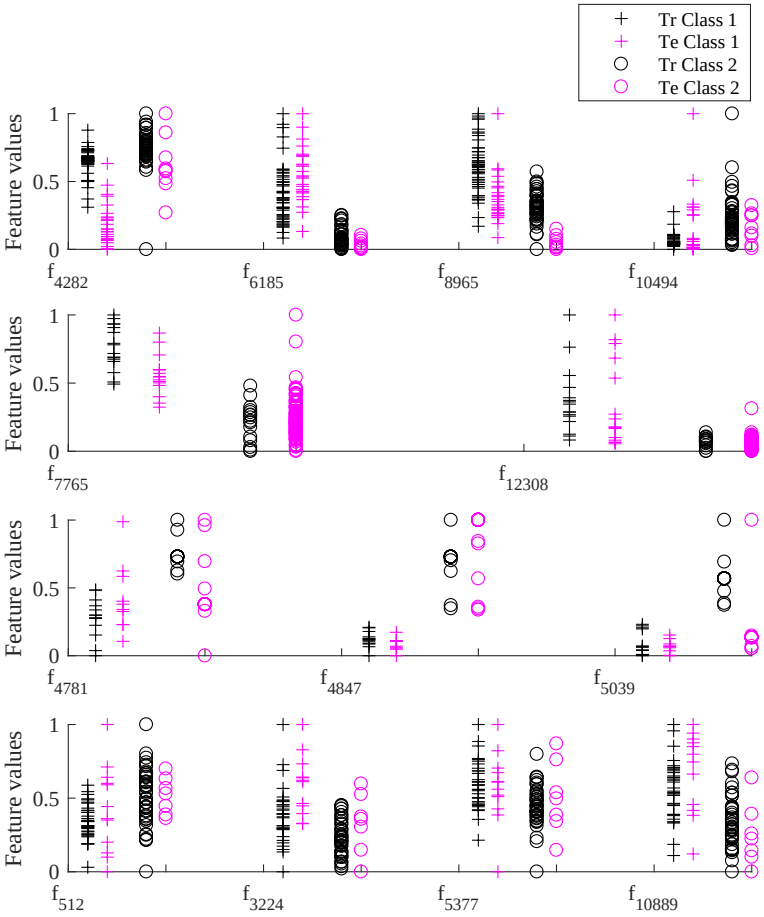

Figure 4.1: Distribution of the feature values of the models presented in Table 4.2. From top to bottom: Prostate, Lung, Leukemia, Breast

Models characterized by perfect performance on the training set have features with little or no overlap between different classes (see, *e.g.*, Lung and Leukemia datasets). This indicates that a perfectly legitimate model selection was operated based on the available information (training set). Unfortunately, the feature value distributions over classes turn out to be different on the test set, typically resulting in some classification errors. Such imprecision could not have been avoided based on the information gathered

Dataset	Classifier	Feature subset	$\mathcal{R}_N$	J_{tr}	J_{te}
Breast	NB	$\mathcal{S}^\star$	0.6126	0.8590	0.8947
		$\mathcal{S}^\star \setminus \{f_{512}\}$	0.6014	0.8333	0.7895
		$\mathcal{S}^\star \setminus \{f_{3224}\}$	0.5858	0.7821	0.6842
		$\mathcal{S}^\star \setminus \{f_{5377}\}$	0.5958	0.8205	0.8421
		$\mathcal{S}^\star \setminus \{f_{10889}\}$	0.5450	0.7949	0.7895
Leukemia	NB	$\mathcal{S}^\star$	0.9782	1.0000	1.0000
		$\mathcal{S}^\star \setminus \{f_{1924}\}$	0.9749	1.0000	0.9118
		$\mathcal{S}^\star \setminus \{f_{3252}\}$	0.9734	1.0000	0.8824
		$\mathcal{S}^\star \setminus \{f_{4847}\}$	0.9770	1.0000	0.9412
		$\mathcal{S}^\star \setminus \{f_{5039}\}$	0.9707	1.0000	1.0000
Lung	KNN	$\mathcal{S}^\star$	0.9150	0.9688	1.0000
		$\mathcal{S}^\star \setminus \{f_{3334}\}$	0.8833	0.9688	0.9732
		$\mathcal{S}^\star \setminus \{f_{6571}\}$	0.8990	0.9688	0.9933
		$\mathcal{S}^\star \setminus \{f_{11841}\}$	0.8755	0.9688	1.0000
Prostate	SVM	$\mathcal{S}^\star$	0.8216	0.9216	0.9706
		$\mathcal{S}^\star \setminus \{f_{4282}\}$	0.8108	0.9216	0.8529
		$\mathcal{S}^\star \setminus \{f_{6185}\}$	0.7848	0.9020	0.9118
		$\mathcal{S}^\star \setminus \{f_{8965}\}$	0.8008	0.9412	0.9706
		$\mathcal{S}^\star \setminus \{f_{10494}\}$	0.8145	0.9020	0.9412

Table 4.3: Redundancy analysis on the best models (see Table 4.2) for data with given training and test set.

from the training set, if not by luck. In other words, if a subset of features provides good class discrimination on the training set, it will provide good generalization only if the feature value distributions on the training and the test subsets are similar. It may well happen that better generalization is achieved through a model which is not optimal on the training set.

4.3 Redundancy analysis of obtained models

We performed an *a posteriori* analysis on the obtained models, both in terms of the dCor measure and the classifier performance, to investigate the presence of redundant biological information. The test is performed by removing one gene at a time from $\mathcal{S}^\star$, and re-evaluating the reduced feature subset. Table 4.3 reports the obtained results.

By inspecting Table 4.3, it is apparent that the dCor index indicates the absence of redundant terms in all selected models (the full model has the highest dCor). If the performance index on the training set J_{tr} were used

as a selection criterion (as would happen, *e.g.*, with a wrapper method), a smaller model would have been selected in the Prostate case, but without any improvement on the test data. Observe that in all four cases the model with the highest dCor achieves the best test performance. In general, the dCor-based filter method provides a good guess of the optimal model both in terms of size and performance, although additional accuracy improvements could occasionally be obtained complementing it with a wrapper method that optimizes directly on the classifier performance.

4.4 Model sensitivity on the data subdivision

To analyze the model sensitivity on the data subdivision in training and test data, we took the best model (see Table 4.2) obtained using the nominal training-test subdivision of the Leukemia dataset, and evaluated its performance with a Monte Carlo test over 1000 random training-test data subdivisions (generated so as to preserve the distribution among classes). On each run, the selected features are the same but the classifier is re-estimated on the corresponding training subset and evaluated on the test subset. The results are presented in Fig. 4.2, and show a non-neglectable sensitivity to the training-test data subdivisions. Indeed, the same performance of the nominal case is re-obtained less than 50% of the times, and on almost 10% of the runs a performance as low as $J_{te} = 0.91$ is achieved (corresponding to 3 errors over the 34 test samples). One possible explanation of this phenomenon is that there are some isolated samples which cannot be learnt by the model if they fall in the test portion of the data.

4.5 Performance analysis of D^{2}CORFS with LOOCV

The previous analysis confirms that the data subdivision is extremely critical and can greatly affect the quality of the results and ultimately the assessment of an algorithm. Applying the HOCV approach in the absence of a nominal training-test data subdivision would lead to results of questionable objectivity, and difficult to compare with the existing literature. For this reason, we carried out a different analysis on the available datasets using LOOCV, which is a Cross Validation method that does not depend on a specific data subdivision (as HOCV). Table 4.4 presents the results obtained following the LOOCV approach on all the datasets (the results are averaged over 10 repetitions). For each dataset, the best selected model structure is reported,

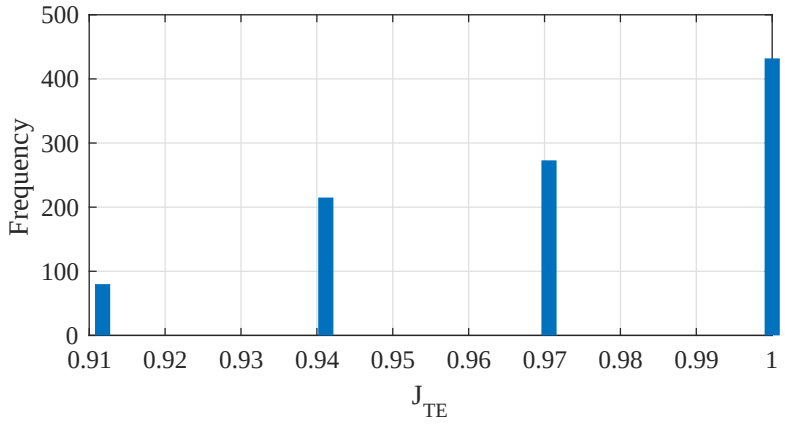

Figure 4.2: Accuracy of the Leukemia model obtained with the nominal training-test data splitting over 1000 alternative data-splittings.

together with the average computation time. Both the performance of the best model overall and the average performance of the best models over 10 runs are given, for the three types of classifiers considered.

It is interesting to note that where both the HOCV and LOOCV methods have been applied (*i.e.*, for Breast, Leukemia, Lung, and Prostate), the obtained models have a scarce overlap in terms of model structure. More specifically, the models obtained for the Breast dataset have 2 mutual features (but extremely different size), and the Lung and Prostate models have just one feature in common, while a totally different model structure was obtained in the remaining case. This is yet another indication of the impact that data subdivision can have on the results.

4.6 Diversity analysis of high performance models

Due to the discrete nature of the classification problem, multiple optimal (*i.e.*, with the same maximal accuracy) models can be obtained, which can make the model interpretation awkward. As an example, we explore this phenomenon with reference to the Leukemia dataset, where a model with 0 classification errors was previously obtained with LOOCV and an NB classifier (see Table 4.4). We repeated the selection process multiple times, each time forcing the exclusion of one of the regressors belonging to one of the previously selected models. This procedure enforces that at each repetition of the algorithm a different best model will be obtained. Table 4.5 shows

several instances of models with different structure but equal accuracy that are obtained in this way. These models contain combinations of two or three regressors taken from a restricted set of seven. Notice that some of the models have no common regressor. This suggests that there are different groups of the genes which contain the same amount of useful information to distinguish among the classes. This phenomenon could have several explanations, ranging from the high correlation of features, to the insufficient information carried by the training set.

4.7 Complexity analysis

We here analyze the computational complexity of the proposed algorithm as a function of the problem size (*i.e.*, the number of features N_f and samples N), and of some crucial design parameters (*e.g.*, the number of rounds N_r of the D^{2}CORFS algorithm, the number of iterations N_i of the DCORFS algorithm, and the number of the feature bins N_b). Let $\mathcal{F}$ be the full set of N_f features. Clearly, the model space to be explored grows exponentially with the number of features (the number of possible non-empty subsets of $\mathcal{F}$ is $2^{N_f} - 1$. At every iteration, each processor executes the DCORF algorithm on its feature bin, which performs three tasks: feature subset extraction and evaluation, regressor evaluation, and FIP update. The first task requires $N_p N'_f$ operations, where N_p is the number of feature subsets to be evaluated, and $N'_f \simeq N_f/N_b$ is the number of features in each feature bin. The evaluation of a feature subset by means of Equation (2.36) is of order $O(N^2 N'_f)$, whereas the calculation of all the indices $\mathcal{I}_j$, $j = 1, \ldots, N'_f$ requires an order of $N_p N'_f$ operations. Finally, the FIP update is linear in the number of features in the bin, *i.e.* $O(N'_f)$. The complexity of the DCORFS is then $O(N_i N'_f (N^2 + N_p))$, which is typically dominated by the first term. The complexity of the overall distributed scheme D^{2}CORFS is dominated by its main cycle which repeats up to N_r times the DCORFS on N_b feature bins, for an overall complexity of $O(N_r N_b N_i N'_f (N^2 + N_p)) \simeq O(N_r N_f N_i N^2))$.

An experimental characterization of the algorithm time complexity has also been carried out, the results of which are shown in Fig. 4.3. More precisely, Fig. 4.3 (top) reports the elapsed time for the DCORF algorithm, averaged over ten runs (the features are reshuffled at random in each run), as a function of the number of features in the bin and the number of iterations. The curves are characterized by an initial increase of the computational time with the growth of the feature space, followed by a saturation associated

with the reaching of the maximum allowed number of iterations. Indeed, as N_i increases, the saturation point shifts to the right. These curves can be used to properly set N_i with respect to the number of features in the bin, in order to obtain convergence prior to the saturation point. Fig. 4.3 (bottom) analyzes the elapsed time of the overall D^{2}CORFS algorithm, as a function of the problem sizes N_f and the number of bins N_b. As can be seen from the figure, it is quite apparent that the execution time decreases rapidly as the number of bins increases, but at a certain point it starts increasing again, though at a slower rate. This result emphasizes the importance of the N_b design parameter. If the number of bins is chosen too sparingly, the bin size will be too large, thus leading to insufficient search space reduction. This ultimately defies the very purpose of the distributed scheme, *i.e.* to break the problem complexity, and thus slows down the convergence of the algorithm. Conversely, if one employs too many small bins, most of these will initially not contain any useful feature and will presumably return inaccurate results, whereas only the processors associated to bins that contain features of the true model will typically produce meaningful results. As a consequence of this, the algorithm will require more rounds and thus more time to converge.

4.8 Comparative analysis with results in the literature

As already commented, a meaningful comparison can be obtained only if the same training-test data distribution is employed. For this reason we divided this comparative analysis into two parts depending on the cross validation method employed. First, we consider the Breast, Leukemia, Lung, and Prostate datasets using a HOCV approach. A reliable comparison is possible, since these datasets are provided with a nominal training-test distribution. Then we consider all eight databases of Table 4.1 using a LOOCV approach, which employs the data for training and testing in a unique and consistent way.

Table 4.6 reports a comparison with the results documented in [6], [46] and [25], which consider the Breast, Leukemia, Lung, and Prostate datasets using a HOCV approach based on the nominal training-test distribution of the data. To account for the randomized nature of the D^{2}CORFS algorithm (due to the random distribution of the features in the bins and to the nature of the DCORFS algorithm employed on each local FS sub-problem), we present

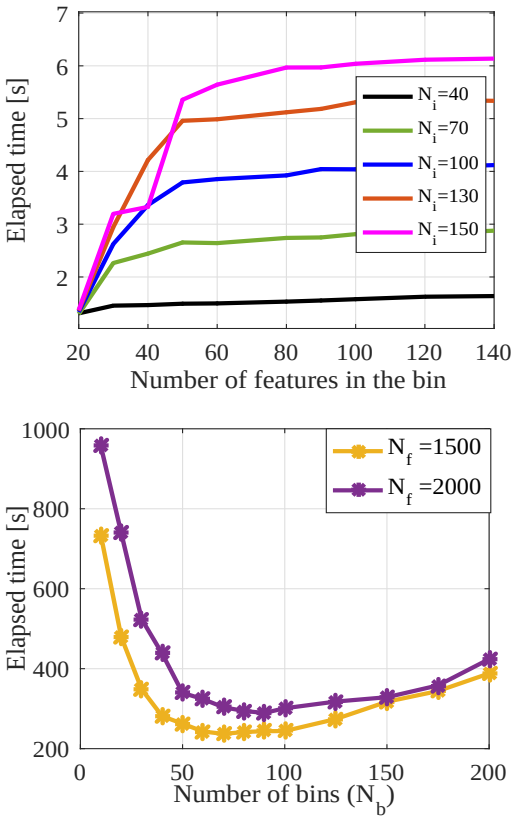

Figure 4.3: Complexity analysis: Dependency of the DCORFS execution time on the number of features in the bin and the number of iterations (top), dependency of the D^{2}CORFS execution time on the problem size and the number of bins (bottom).

the averaged results of 5 independent runs besides the best ones. Both the classification accuracy on the test set and the model size are reported ($\bar{J}_{te}$ and $\overline{|\mathcal{S}|}$ denote the averages, and $J^{\star}_{te}$ and $|\mathcal{S}^{\star}|$ the values associated to the best models, respectively).

Apparently, the proposed method achieves comparable performance with respect to the best of the competitor methods. Moreover, the obtained models are extremely compact in terms of the number of selected features, which indicates the effectiveness of the FS approach in pointing out to the expert the really important features.

Table 4.7 provides a comparison with the methods documented in the literature that study all the eight datasets considered in this work, using a LOOCV approach. The proposed method systematically provides a promising performance, scoring better or equivalently to the competitor methods on five out of eight datasets, and generally ranking among the best methods. In the case for which more documented results can be found in the literature (Leukemia), it obtains perfect performance both on the training and the test set, using only 2 features. It is also confirmed that the method tends to provide a good trade-off between accuracy and compactness of the selected model, which is important both for the robustness of the classifier and for model interpretation purposes.

Dataset	Best model ($\mathcal{S}^\star$)	Time [s]	Classifier	$J^\star_{te}$	J_{te}
Breast	$\{f_{512}, f_{1205}, f_{1872}, f_{3232}, f_{3773}, f_{4382}, f_{5098}, f_{6859}, f_{7127}, f_{7997}, f_{8776}, f_{10827}, f_{10889}, f_{12275}, f_{12437}, f_{12572}, f_{13800}, f_{17881}, f_{19694}, f_{19906}, f_{20437}, f_{22422}, f_{23322}\}$	8137.83	SVM	0.8969	0.8598
			SVMn	0.8969	0.8567
			KNN	0.8660	0.8392
			LogReg	0.8969	0.8299
			LDA	0.8866	0.8557
			NB	0.8454	0.8083
			Tree	0.7732	0.7113
CNS	$\{f_{320}, f_{1054}, f_{2496}, f_{2513}, f_{3320}, f_{3731}, f_{4484}, f_{4509}\}$	123.89	SVM	0.9000	0.8583
			SVMn	0.8667	0.8433
			KNN	0.8833	0.8350
			LogReg	0.8333	0.7867
			LDA	0.8667	0.8517
			NB	0.8833	0.8450
			Tree	0.8500	0.6950
Colon	$\{f_{249}, f_{377}, f_{765}, f_{1482}, f_{1644}, f_{1772}\}$	584.08	SVM	0.9032	0.8823
			SVMn	0.9032	0.8871
			KNN	0.8710	0.8581
			LogReg	0.8548	0.8177
			LDA	0.9032	0.8855
			NB	0.9194	0.9065
			Tree	0.8548	0.8258
DLBCL	$\{f_{57}, f_{209}, f_{1807}, f_{2115}, f_{2208}\}$	588.29	SVM	0.9870	0.9597
			SVMn	0.9870	0.9675
			KNN	0.9740	0.9584
			LogReg	0.9481	0.8727
			LDA	0.9870	0.9533
			NB	0.9740	0.9468
			Tree	0.9221	0.8727
Leukemia	$\{f_{2288}, f_{6041}\}$	170.02	SVM	0.9722	0.9722
			SVMn	1.0000	1.0000
			KNN	0.9722	0.9722
			LogReg	0.6806	0.6806
			LDA	0.9722	0.9722
			NB	1.0000	1.0000
			Tree	1.0000	1.0000
Lung	$\{f_{3334}, f_{4336}, f_{7200}, f_{8370}\}$	2239.40	SVM	1.0000	0.9939
			SVMn	1.0000	0.9967
			KNN	0.9945	0.9923
			LogReg	0.9503	0.7536
			LDA	0.9945	0.9884
			NB	0.9779	0.9751
			Tree	0.9834	0.9613
Ovarian	$\{f_{182}, f_{1680}, f_{2236}\}$	3383.12	SVM	1.0000	1.0000
			SVMn	1.0000	1.0000
			KNN	1.0000	1.0000
			LogReg	1.0000	1.0000
			LDA	1.0000	0.9976
			NB	1.0000	1.0000
			Tree	0.9802	0.9648
Prostate	$\{f_{5314}, f_{6185}, f_{9850}, f_{11052}\}$	4042.18	SVM	0.8456	0.7728
			SVMn	0.8677	0.7890
			KNN	0.9338	0.8765
			LogReg	0.9485	0.8243
			LDA	0.8456	0.7971
			NB	0.8456	0.7559
			Tree	0.9191	0.8478

Table 4.4: Performance of the best models obtained with D^2CORFS and LOOCV.

Feature	$\mathcal{S}^*$	$\mathcal{S}_1^*$	$\mathcal{S}_2^*$	$\mathcal{S}_3^*$	$\mathcal{S}_4^*$	$\mathcal{S}_5^*$
f_{2288}	✓					✓
f_{4052}		✓	✓	✓	✓	
f_{4167}		✓			✓	
f_{4230}				✓		
f_{4328}			✓			
f_{4847}					✓	✓
f_{6041}	✓	✓	✓	✓		

Table 4.5: Diversity analysis of the models with maximum accuracy (0 classification errors) for the Leukemia dataset, obtained with LOOCV and a NB classifier.

Dataset	Method	$\bar{J}_{te}$	$J_{te}^{\star}$	$\overline{\lvert\mathcal{S}\rvert}$	$\lvert\mathcal{S}^{\star}\rvert$
Breast	D^{2}CORFS	0.67	0.89	6.8	4
	DRF+SVM [6]	–	0.84	–	97
	DRF+kNN [6]	–	0.79	–	52
	DRF+NB [6]	–	0.79	–	40
Leukemia	D^{2}CORFS	0.93	1.00	3.2	4
	DRF+SVM [6]	–	0.91	–	15
	DRF+kNN [6]	–	0.94	–	4
	DRF+NB [6]	–	0.94	–	6
	ABC+DANN [25]	0.88	0.94	3.0	3
	L1-norm SVM [46]	0.84	–	24.9	–
	Elastic Net [46]	0.84	–	36.7	–
	PAEN [46]	0.85	–	21.9	–
	DrSVM [46]	0.85	–	67.7	–
	WDRSVM [46]	0.86	–	19.9	–
Lung	D^{2}CORFS	0.97	1.00	2.9	3
	DRF+SVM [6]	–	0.96	–	2
	DRF+kNN [6]	–	0.98	–	4
	DRF+NB [6]	–	0.99	–	8
	L1-norm SVM [46]	0.84	–	29.1	–
	Elastic Net [46]	0.84	–	39.4	–
	PAEN [46]	0.85	–	26.3	–
	DrSVM [46]	0.86	–	54.4	–
	WDRSVM [46]	0.86	–	23.8	–
Prostate	D^{2}CORFS	0.90	0.97	3.8	4
	DRF+SVM [6]		0.97	–	30
	DRF+kNN [6]		0.62	–	35
	DRF+NB [6]		0.26	–	12

Table 4.6: Comparative analysis with the HOCV approach.

Dataset	Method	$\bar{J}_{te}$	$J^{\star}_{te}$	$\overline{\lvert\mathcal{S}\rvert}$	$\lvert\mathcal{S}^{\star}\rvert$
Breast	D^{2}CORFS	0.84	0.90	21.9	23
	Local-learning based [72]	–	0.78	–	4
	α DD [82]	0.69	0.88	–	–
CNS	D^{2}CORFS	0.84	0.90	6.7	8
	α DD [82]	0.72	0.90	–	–
Colon	D^{2}CORFS	0.88	0.92	8.0	6
	α DD [82]	0.83	0.92	–	–
	Filter mRMR+SVM [68]	–	0.89	–	4
	Filter mRMR+RVM [68]	–	0.94	–	7
	RMIFS+NB [75]	–	0.97	–	6
	RMIFS+ID3 [75]	–	0.95	–	6
	RMIFS+Logistic [75]	–	1.00	–	6
	ERGS [12]	–	0.84	–	100
DLBCL	D^{2}CORFS	0.95	0.99	7.1	5
	Local-learning based [72]	–	0.97	–	10
	α DD [82]	0.71	0.88	–	–
	MOBBBO [48]	1.00	1.00	5.7	5
	LSLS [49]	–	0.82	–	10
Leukemia	D^{2}CORFS	0.98	1.00	2.0	2
	α DD [82]	0.92	0.97	–	–
	Filter mRMR+SVM [68]	–	0.97	–	4
	Filter mRMR+RVM [68]	–	1.00	–	3
	RMIFS+NB [75]	–	1.00	–	4
	RMIFS+ID3 [75]	–	1.00	–	4
	RMIFS+Logistic [75]	–	1.00	–	4
	LSLS [49]	–	0.81	–	50
	ERGS [12]	–	1.00	–	80
	SL-RFE [83]	–	0.94	–	20
	FS-RFE [83]	–	0.94	–	40
	MRMR [18]	1.00	–	6.0	–
	MBF [84]	1.00	–	9.0	–
Lung	D^{2}CORFS	0.99	1.00	4.4	3
	α DD [82]	0.98	1.00	–	–
	LSLS [49]	–	0.99	–	30
	ERGS [12]	–	1.00	–	100
Ovarian	D^{2}CORFS	1.00	1.00	3.0	3
Prostate	D^{2}CORFS	0.80	0.93	3.3	4
	Local-learning based [72]	–	0.84	–	6
	α DD [82]	0.91	0.96	–	–
	MOBBBO [48]	0.98	1.00	11.9	12
	LSLS [49]	–	0.74	–	25
	ERGS [12]	–	0.94	–	10

Table 4.7: Comparative analysis with the LOOCV approach.

Chapter 5

Conclusion

In this thesis a novel FS method has been developed that is especially designed for large and dimensionally unbalanced classification problems, such as those that arise in connection with microarrays. Its strength resides on three pillars, namely an evaluation criterion for candidate feature subsets based on the distance correlation concept, a distributed optimization approach, and a randomized selection procedure. The dCor index appears to be a particularly robust criterion with respect to overfitting and redundancy issues, which are common with multivariate filter methods. The distributed combinatorial optimization scheme is used to handle the severe asymmetry of microarray datasets, by dividing the feature set into several feature bins and running independently the FS algorithm on each of them. The best solutions are retained and shared among the feature bins and the procedure is iterated until convergence. Thanks to this "divide et impera" approach, the FS algorithm is always employed on small and dimensionally balanced datasets, for better accuracy and reliability of the results, as well as a reduced computational complexity. The FS algorithm at the core of the method introduces another factor that improves the reliability of the method, in that it re-enforces the probability to select a feature based on an aggregate performance evaluation of a population of feature subsets, which allows for a more reliable assessment of the importance of that particular feature. The overall method has been tested on several microarray benchmark datasets, with quite promising results. Indeed, the resulting classifiers achieve high accuracy levels while using information only from an extremely small number of features.

Bibliography

[1] Enrique Alba, Jose Garcia-Nieto, Laetitia Jourdan, and El-Ghazali Talbi. Gene selection in cancer classification using pso/svm and ga/svm hybrid algorithms. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 284–290. IEEE, 2007.

[2] Ash A Alizadeh, Michael B Eisen, R Eric Davis, Chi Ma, Izidore S Lossos, Andreas Rosenwald, Jennifer C Boldrick, Hajeer Sabet, Truc Tran, Xin Yu, et al. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769):503–511, 2000.

[3] Uri Alon, Naama Barkai, Daniel A Notterman, Kurt Gish, Suzanne Ybarra, Daniel Mack, and Arnold J Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–6750, 1999.

[4] Jun Chin Ang, Andri Mirzal, Habibollah Haron, and Haza Nuzly Abdull Hamed. Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection. *IEEE/ACM transactions on computational biology and bioinformatics*, 13(5):971–989, 2016.

[5] David A Bell and Hui Wang. A formalism for relevance and its application in feature subset selection. *Machine learning*, 41(2):175–195, 2000.

[6] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. Distributed feature selection: An application to microarray data classification. *Applied soft computing*, 30:136–150, 2015.

[7] Verónica Bolón-Canedo, Noelia Sánchez-Marono, Amparo Alonso-Betanzos, José Manuel Benítez, and Francisco Herrera. A review of microarray datasets and applied feature selection methods. *Information Sciences*, 282:111–135, 2014.

[8] A. Brankovic, M. Hosseini, and L. Piroddi. A distributed feature selection algorithm based on distance correlation with an application to microarrays. *submitted paper, available on request*, 2017.

[9] Aida Brankovic, Alessandro Falsone, Maria Prandini, and Luigi Piroddi. A feature selection and classification algorithm based on randomized extraction of model populations. *IEEE transactions on cybernetics*, 2017.

[10] Aida Brankovic and Luigi Piroddi. A distributed feature selection scheme with partial information sharing. Technical report, Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 2017.

[11] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[12] B Chandra and Manish Gupta. An efficient statistical feature selection approach for classification of gene expression data. *Journal of biomedical informatics*, 44(4):529–535, 2011.

[13] Wikipedia contributors. Genome — wikipedia, the free encyclopedia, 2018. [Online; accessed 23-January-2018].

[14] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[15] Scott A Czepiel. Maximum likelihood estimation of logistic regression models: theory and implementation. *Available at czep. net/stat/mlelr. pdf*, 2002.

[16] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent data analysis*, 1(3):131–156, 1997.

[17] Anthony Mihirana De Silva and Philip HW Leong. *Grammar-based feature generation for time-series prediction*. Springer, 2015.

[18] Chris Ding and Hanchuan Peng. Minimum redundancy feature selection from microarray gene expression data. *Journal of bioinformatics and computational biology*, 3(02):185–205, 2005.

[19] Justin Doak. Cse-92-18-an evaluation of feature selection methodsand their application to computer security. 1992.

[20] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[21] Edward R Dougherty. Small sample issues for microarray-based classification. *Comparative and Functional Genomics*, 2(1):28–34, 2001.

[22] Alessandro Falsone, Luigi Piroddi, and Maria Prandini. A randomized algorithm for nonlinear model structure selection. *Automatica*, 60:227–238, 2015.

[23] F Ferri, P Pudil, M Hatef, and J Kittler. Comparative study of techniques for large-scale feature selection. *Pattern Recognition in Practice IV*, 1994:403–413, 1994.

[24] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.

[25] Beatriz A Garro, Katya Rodríguez, and Roberto A Vazquez. Designing artificial neural networks using differential evolution for classifying dna microarrays. In *Evolutionary Computation (CEC), 2017 IEEE Congress on*, pages 2767–2774. IEEE, 2017.

[26] Todd R Golub, Donna K Slonim, Pablo Tamayo, Christine Huard, Michelle Gaasenbeek, Jill P Mesirov, Hilary Coller, Mignon L Loh, James R Downing, Mark A Caligiuri, et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *science*, 286(5439):531–537, 1999.

[27] Gavin J Gordon, Roderick V Jensen, Li-Li Hsiao, Steven R Gullans, Joshua E Blumenstock, Sridhar Ramaswamy, William G Richards, David J Sugarbaker, and Raphael Bueno. Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. *Cancer research*, 62(17):4963–4967, 2002.

[28] Rajeshwar Govindarajan, Jeyapradha Duraiyan, Karunakaran Kaliyappan, and Murugesan Palanisamy. Microarray and its applications. *Journal of pharmacy & bioallied sciences*, 4(Suppl 2):S310, 2012.

[29] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[30] Mark A Hall and Lloyd A Smith. Practical feature subset selection for machine learning. 1998.

[31] Mark Andrew Hall. Correlation-based feature selection for machine learning. 1999.

[32] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[33] Ingrid Hedenfalk, David Duggan, Yidong Chen, Michael Radmacher, Michael Bittner, Richard Simon, Paul Meltzer, Barry Gusterson, Manel Esteller, Mark Raffeld, et al. Gene-expression profiles in hereditary breast cancer. *New England Journal of Medicine*, 344(8):539–548, 2001.

[34] Zena M Hira and Duncan F Gillies. A review of feature selection and feature extraction methods applied on microarray data. *Advances in bioinformatics*, 2015, 2015.

[35] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[36] Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

[37] Thanyaluk Jirapech-Umpai and Stuart Aitken. Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes. *BMC bioinformatics*, 6(1):148, 2005.

[38] Alan Jović, Karla Brkić, and Nikola Bogunović. A review of feature selection methods with applications. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on*, pages 1200–1205. IEEE, 2015.

[39] Kenji Kira and Larry A Rendell. The feature selection problem: Traditional methods and a new algorithm. In *Aaai*, volume 2, pages 129–134, 1992.

[40] Lev Klebanov and Andrei Yakovlev. How high is the level of technical noise in microarray data? *Biology Direct*, 2(1):9, 2007.

[41] Daphne Koller and Mehran Sahami. Toward optimal feature selection. Technical report, Stanford InfoLab, 1996.

[42] Sotiris Kotsiantis. Feature selection for machine learning classification problems: a recent overview. *Artificial Intelligence Review*, pages 1–20, 2011.

[43] Vipin Kumar and Sonajharia Minz. Feature selection. *SmartCR*, 4(3):211–229, 2014.

[44] Cosmin Lazar, Jonatan Taminau, Stijn Meganck, David Steenhoff, Alain Coletta, Colin Molter, Virginie de Schaetzen, Robin Duque, Hugues Bersini, and Ann Nowe. A survey on filter techniques for feature selection in gene expression microarray analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(4):1106–1119, 2012.

[45] Yukyee Leung and Yeungsam Hung. A multiple-filter-multiple-wrapper approach to gene selection and microarray data classification. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(1):108–117, 2010.

[46] J. Li, Y. Wang, Y. Cao, and C. Xu. Weighted doubly regularized support vector machine and its application to microarray classification with noise. *Neurocomputing*, 173:595–605, 2016.

[47] Runze Li, Wei Zhong, and Liping Zhu. Feature screening via distance correlation learning. *Journal of the American Statistical Association*, 107(499):1129–1139, 2012.

[48] Xiangtao Li and Minghao Yin. Multiobjective binary biogeography based optimization for feature selection using gene expression data. *IEEE Transactions on NanoBioscience*, 12(4):343–353, 2013.

[49] Bo Liao, Yan Jiang, Wei Liang, Wen Zhu, Lijun Cai, and Zhi Cao. Gene selection using locality sensitive laplacian score. *IEEE/ACM transactions on computational biology and bioinformatics*, 11(6):1146–1156, 2014.

[50] Huan Liu, Rudy Setiono, et al. A probabilistic approach to feature selection-a filter solution. In *ICML*, volume 96, pages 319–327. Citeseer, 1996.

[51] Sebastián Maldonado, Richard Weber, and Jayanta Basak. Simultaneous feature selection and classification using kernel-penalized support vector machines. *Information Sciences*, 181(1):115–128, 2011.

[52] Patrick E Meyer and Gianluca Bontempi. On the use of variable complementarity for feature selection in cancer classification. In *Workshops on Applications of Evolutionary Computation*, pages 91–102. Springer, 2006.

[53] Carrie Blanchette MikeWest, Holly Dressman, Erich Huang, Seiichi Ishida, Rainer Spang, Harry Zuzan, Jeffrey R Marks, and Joseph R Nevins. Predicting the clinical status of human breast cancer using gene expression profiles. 2001.

[54] Patrenahalli M. Narendra and Keinosuke Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on computers*, 9(C-26):917–922, 1977.

[55] Hanchuan Peng and Fuhui Long. An efficient max-dependency algorithm for gene selection. In *36th Symposium on the interface: Computational Biology and Bioinformatics*, volume 57, page 65, 2004.

[56] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[57] Emanuel F Petricoin, Ali M Ardekani, Ben A Hitt, Peter J Levine, Vincent A Fusaro, Seth M Steinberg, Gordon B Mills, Charles Simone, David A Fishman, Elise C Kohn, et al. Use of proteomic patterns in serum to identify ovarian cancer. *The lancet*, 359(9306):572–577, 2002.

[58] Scott L Pomeroy, Pablo Tamayo, Michelle Gaasenbeek, Lisa M Sturla, Michael Angelo, Margaret E McLaughlin, John YH Kim, Liliana C Goumnerova, Peter M Black, Ching Lau, et al. Prediction of central nervous system embryonal tumour outcome based on gene expression. *Nature*, 415(6870):436–442, 2002.

[59] Ronaldo C Prati, Gustavo EAPA Batista, and Maria Carolina Monard. Class imbalances versus class overlapping: an analysis of a learning system behavior. In *Mexican international conference on artificial intelligence*, pages 312–321. Springer, 2004.

[60] CRC PRESS. Computational methods of feature selection.

[61] Pavel Pudil, Jana Novovičová, and Josef Kittler. Floating search methods in feature selection. *Pattern recognition letters*, 15(11):1119–1125, 1994.

[62] Sebastian Raschka. Naive bayes and text classification i-introduction and theory. *arXiv preprint arXiv:1410.5329*, 2014.

[63] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM, 2001.

[64] Yvan Saeys, Iñaki Inza, and Pedro Larrañaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

[65] Dari Shalon, Stephen J Smith, and Patrick O Brown. A dna microarray system for analyzing complex dna samples using two-color fluorescent probe hybridization. *Genome research*, 6(7):639–645, 1996.

[66] Alok Sharma, Seiya Imoto, and Satoru Miyano. A top-r feature selection algorithm for microarray gene expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 9(3):754–764, 2012.

[67] Dinesh Singh, Phillip G Febbo, Kenneth Ross, Donald G Jackson, Judith Manola, Christine Ladd, Pablo Tamayo, Andrew A Renshaw, Anthony V D'Amico, Jerome P Richie, et al. Gene expression correlates of clinical prostate cancer behavior. *Cancer cell*, 1(2):203–209, 2002.

[68] Mehran Soltani, Mohammad Hasan Shammakhi, Saeed Khorram, and Hamid Sheikhzadeh. Combined mrmr filter and sparse bayesian classifier for analysis of gene expression data. In *Signal Processing and Intelligent Systems (ICSPIS), International Conference of*, pages 1–5. IEEE, 2016.

[69] Diane M Strong, Yang W Lee, and Richard Y Wang. Data quality in context. *Communications of the ACM*, 40(5):103–110, 1997.

[70] Yanmin Sun, Andrew KC Wong, and Mohamed S Kamel. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):687–719, 2009.

[71] Yijun Sun and Jian Li. Iterative relief for feature weighting. In *Proceedings of the 23rd international conference on Machine learning*, pages 913–920. ACM, 2006.

[72] Yijun Sun, Sinisa Todorovic, and Steve Goodison. Local-learning-based feature selection for high-dimensional data analysis. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1610–1626, 2010.

[73] Gábor J Székely and Maria L Rizzo. The distance correlation t-test of independence in high dimension. *Journal of Multivariate Analysis*, 117:193–213, 2013.

[74] Gábor J Székely, Maria L Rizzo, Nail K Bakirov, et al. Measuring and testing dependence by correlation of distances. *The annals of statistics*, 35(6):2769–2794, 2007.

[75] Jian Tang and Shuigeng Zhou. A new approach for feature selection from microarray data based on mutual information. *IEEE/ACM transactions on computational biology and bioinformatics*, 13(6):1004–1015, 2016.

[76] Saravanan Thirumuruganathan. A detailed introduction to k-nearest neighbor (knn) algorithm. *Retrieved on July*, 21:2015, 2010.

[77] Y Tu, G Stolovitzky, and U Klein. Quantitative noise analysis for gene expression microarray experiments. *Proceedings of the National Academy of Sciences*, 99(22):14031–14036, 2002.

[78] Laura J Van't Veer, Hongyue Dai, Marc J Van De Vijver, Yudong D He, Augustinus AM Hart, Mao Mao, Hans L Peterse, Karin Van Der Kooy, Matthew J Marton, Anke T Witteveen, et al. Gene expression profiling predicts clinical outcome of breast cancer. *nature*, 415(6871):530–536, 2002.

[79] Maria Fernanda B Wanderley, Vincent Gardeux, René Natowicz, and Antônio de Pádua Braga. Ga-kde-bayes: an evolutionary wrapper method based on non-parametric density estimation applied to bioinformatics problems. In *ESANN*, 2013.

[80] Guangtao Wang, Qinbao Song, Baowen Xu, and Yuming Zhou. Selecting feature subset for high dimensional data via the propositional foil rules. *Pattern Recognition*, 46(1):199–214, 2013.

[81] Hui Wang, David Bell, and Fionn Murtagh. Relevance approach to feature subset selection. In *Feature Extraction, Construction and Selection*, pages 85–99. Springer, 1998.

[82] Xiaosheng Wang and Osamu Gotoh. A robust gene selection method for microarray-based cancer classification. *Cancer informatics*, 9:CIN–S3794, 2010.

[83] Dan Wei, Shutao Li, and Mingkui Tan. Graph embedding based feature selection. *Neurocomputing*, 93:115–125, 2012.

[84] Eric P Xing, Michael I Jordan, Richard M Karp, et al. Feature selection for high-dimensional genomic microarray data. In *ICML*, volume 1, pages 601–608. Citeseer, 2001.

[85] Pengyi Yang, Bing B Zhou, Zili Zhang, and Albert Y Zomaya. A multi-filter enhanced genetic ensemble system for gene selection and sample classification of microarray data. *BMC bioinformatics*, 11(1):S5, 2010.

[86] C Deniz Yenigün and Maria L Rizzo. Variable selection in regression using maximal correlation and distance correlation. *Journal of Statistical Computation and Simulation*, 85(8):1692–1705, 2015.

[87] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of machine learning research*, 5(Oct):1205–1224, 2004.

[88] Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study. *Artificial intelligence review*, 22(3):177–210, 2004.

Appendix A

List of gene accession numbers and descriptions

Feature	Gene accession number	Gene Definition
f_{512}	$D42044$	HS KIAA0090 mRNA, partial cds
f_{1205}	NM_003164	HS syntaxin 5 (STX5), transcript variant 1, mRNA
f_{1872}	$Contig53223$	Lymphoblastoid cell lines from CEPH/Utah families
f_{3224}	NM_020120	HS UDP-glucose glycoprotein glucosyltransferase 1 (UGGT1), transcript variant 1, mRNA
f_{3232}	NM_020123	HS transmembrane 9 superfamily member 3 (TM9SF3) ,mRNA
f_{3773}	$Contig57586\_RC$	Lymphoblastoid cell lines from CEPH/Utah families
f_{4382}	$NM\_004273^{\dagger}$	Homo sapiens carbohydrate sulfotransferase 3 (CHST3), mRNA
f_{5098}	NM_020346	Homo sapiens solute carrier family 17 member 6 (SLC17A6), mRNA
f_{5377}	$Contig54742\_RC$	Lymphoblastoid cell lines from CEPH/Utah families
f_{6859}	NM_021204	Homo sapiens enolase-phosphatase 1 (ENOPH1), transcript variant 1, mRNA
f_{7127}	NM_013262	Homo sapiens myosin regulatory light chain interacting protein (MYLIP), mRNA
f_{7997}	$NM\_020638^{\dagger}$	Homo sapiens fibroblast growth factor 23 (FGF23), mRNA
f_{8776}	NM_006115	Homo sapiens preferentially expressed antigen in melanoma (PRAME), transcript variant 1, mRNA
f_{10827}	$NM\_004994^{\dagger}$	Homo sapiens matrix metallopeptidase 9 (MMP9), mRNA
f_{10889}	$AL080059$	HS mRNA; cDNA DKFZp564H142 (from clone DKFZp564H142)
f_{12275}	NM_014567	Homo sapiens BCAR1, Cas family scaffold protein (BCAR1), transcript variant 6, mRNA
f_{12437}	NM_006584	Homo sapiens chaperonin containing TCP1 subunit 6B (CCT6B), transcript variant 1, mRNA
f_{12572}	$AF055033$	Homo sapiens clone 24645 insulin-like growth factor binding protein 5 (IGFBP5) mRNA, complete cds
f_{13800}	$Contig47544\_RC$	NA
f_{17881}	$D13540^{\dagger}$	HS SH-PTP3 mRNA for protein-tyrosine phosphatase, complete cds
f_{19694}	NM_018354	Homo sapiens transmembrane protein 74B (TMEM74B), transcript variant 1, mRNA
f_{19906}	NM_016946	Homo sapiens F11 receptor (F11R), transcript variant 1, mRNA
f_{20437}	$Contig2399\_RC$	Egl nine homolog 1 (C. elegans)
f_{22422}	$Contig35256$	Lymphoblastoid cell lines from CEPH/Utah families
f_{23322}	$Contig32002\_RC$	Lymphoblastoid cell lines from CEPH/Utah families

Table A.1: Description of the selected genes for Breast cancer dataset.

Note. HS = Homo Sapiens, NA = Not Available. Symbol † stands for genes found in KEGG disease pathway.

Feature	Gene accession number	Gene Definition
f_{320}	$D26561\_cds3\_at$	Human papillomavirus type 5b proviral genes for L1, E6, E7, E1 proteins
f_{1054}	$J02611\_at$	Human apolipoprotein D mRNA
f_{2496}	$S76475\_at^{\dagger}$	trkC [human, brain, mRNA, 2715 nt]
f_{2513}	$S78296\_at$	neurofilament-66 [human, fetal brain, mRNA, 3197 nt]
f_{3320}	$U50136\_rna1\_at$	Human leukotriene C4 synthase (LTC4S) gene
f_{3731}	$U78180\_at$	Human sodium channel 2 (hBNaC2) mRNA, alternatively spliced
f_{4484}	$X69398\_at$	HS mRNA for OA3 antigenic surface determinant
f_{4509}	$X71348\_at^{\dagger}$	HS vHNF1-C mRNA

Table A.2: Description of the selected genes for CNS cancer dataset.

Feature	Gene accession number	Gene Definition
f_{249}	$M63391^{\dagger}$	Homo sapiens desmin gene, complete cds
f_{377}	$Z50753$	HS mRNA for GCAP-II/uroguanylin precursor
f_{765}	$M76378^{\dagger}$	HS cysteine-rich protein (CRP) gene, exons 5 and 6
f_{1482}	$T64012$	acetylcholine receptor protein, delta chain precursor (Xenopus laevis)
f_{1644}	$R80427$	147223 C4-DICARBOXYLATE TRANSPORT SENSOR PROTEIN DCTB (Rhizobium leguminosarum)
f_{1772}	$H08393^{\dagger}$	collagen alpha 2(XI) chain (HS)

Table A.3: Description of the selected genes for Colon cancer dataset.

Feature	Gene accession number	Gene Definition
f_{57}	$GENE3991X$	dynein light intermediate chain 2, cytosolic (PFI0315c)
f_{209}	$GENE3282X$	not yet annotated molecular function unknown
f_{1807}	$GENE2789X$	DBL containing protein, unknown function (PFA0665w)
f_{2115}	$GENE1343X$	SNO glutamine amidotransferase family protein (PF11_0169)
f_{2208}	$GENE1215X$	Plasmodium exported protein,unknown function (PF10_0375)

Table A.4: Description of the selected genes for DLBCL cancer dataset.

Feature	Gene accession number	Gene Definition
f_{1924}	$M31158\_at$	protein kinase, cAMP-dependent, regulatory, type II, beta PRKAR2B
f_{2288}	$M84526\_at^{\dagger}$	DF D component of complement (adipsin)
f_{3252}	$U46499\_at$	microsomal glutathione S-transferase 1 MGST1
f_{4052}	$X04085\_rna1\_at^{\dagger}$	catalase(CAT)
f_{4167}	$X15414\_at$	aldo-keto reductase family 1, member B1(aldose reductase) AKR1B1
f_{4230}	$X52142\_at$	CTP synthase (CTPS)
f_{4328}	$X59417\_at$	proteasome (prosome, macropain) subunit, alpha type, 6 (PSMA6)
f_{4847}	$X95735\_at$	zyxin ZYX
f_{5039}	$Y12670\_at$	leptin receptor overlapping transcript LEPROT
f_{6041}	$L09209\_s\_at$	APLP2 Amyloid beta (A4) precursor-like protein 2

Table A.5: Description of the selected genes for Leukemia cancer dataset.

Feature	Gene accession number	Gene Definition
f_{3334}	33328_at	heart development protein with EGF-like domains 1 (HEG1)
f_{4336}	$34320\_at^{\dagger}$	polymerase I and transcript release factor (PTRF)
f_{6571}	36533_at	prostaglandin I2 (prostacyclin) synthase (PTGIS)
f_{7200}	37157_at	calbindin 2 (CALB2)
f_{8370}	38315_at	aldehyde dehydrogenase 1 family, member A2 (ALDH1A2)
f_{11841}	41755_at	COBL-like 1 (COBLL1)

Table A.6: Description of the selected genes for Lung cancer dataset.

Feature	Gene accession number	Gene Definition
f_{182}	$MZ2.8234234$	NA
f_{1680}	$MZ245.24466$	NA
f_{2236}	$MZ434.68588$	NA

Table A.7: Description of the selected genes for Ovarian cancer dataset.

Feature	Gene accession number	Gene Definition
f_{4282}	41385_*at*	erythrocyte membrane protein band 4.1-like 3 (EPB41L3)
f_{5314}	34730_*g_at*	trophinin (TRO)
f_{6185}	37639_*at*	hepsin (HPN)
f_{8965}	37720_*at*†	heat shock 60kDa protein 1 (chaperonin, HSPD1)
f_{9850}	40282_*s_at*†	complement factor D (adipsin, CFD)
f_{10494}	32598_*at*	neural EGFL like 2 (NELL2)
f_{11052}	1664_*at*†	insulin-like growth factor 2 (INS-IGF2)

Table A.8: Description of the selected genes for Prostate cancer dataset.