On the scalability of feature selection methods on high-dimensional data

V. Bolón-Canedo · D. Rego-Fernández · D. Peteiro-Barral · A. Alonso-Betanzos · B. Guijarro-Berdiñas · N. Sánchez-Maroño

Received: date / Accepted: date

Abstract Lately, derived from the explosion of high-dimensionality, researchers in Machine Learning became interested not only in accuracy, but also in scalability. Although scalability of learning methods is a trending issue, scalability of feature selection methods has not received the same amount of attention. This research analyzes the scalability of state-of-the-art feature selection methods, belonging to filter, embedded and wrapper approaches. For this purpose, several new measures are presented, based not only on accuracy but also on execution time and stability. The results on seven classical artificial datasets are presented and discussed, as well as two cases study analyzing the particularities of microarray data and the effect of redundancy. Trying to check if the results can be generalized, we included some experiments with two real datasets. As expected, filters are the most scalable feature selection approach, being INTERACT, ReliefF and mRMR the most accurate methods.

1 Introduction

In the last years, the dimensionality of datasets involved in data mining applications has increased steadily, as can be seen in [52]. This advent of large-scale data carries new opportunities and challenges to computer scientists, giving the opportunity for discovering subtle population patterns and heterogeneities that were not possible with small-scale data. However, the massive sample size and high dimensionality of data introduce new computational challenges, including among others the appropriateness of the measurements for error rate and scalability. Large-scale data are encountered in

Laboratory for Research and Development in Artificial Intelligence (LIDIA)
Computer Science Dept.
University of A Coruña, 15071
A Coruña, Spain
E-mail: vbolon@udc.es

various areas, as Internet search, social networks, finance, business sectors, meteorology, genomics, complex physics simulations, biological, and environmental research. The characteristics of volume, velocity, variety and veracity bring challenges to current machine learning techniques, making it not only desirable but essential to scale up machine learning techniques for modeling and analyzing big data from various domains. Several conferences and challenges have been celebrated in the last years, such as PASCAL Large Scale Learning Challenge in the 25th International Conference on Machine Learning (ICML 2008), the NIPS (Neural Information Processing Systems) 2011 Workshop on Big Learning, in which, among several other points, there was one devoted to methods for dealing with huge numbers of features or the 2013 IEEE International Conference on Big Data. During these last years, various approaches trying to deal with the problem of scalability in machine learning [1,6,48, 39] and even some studies comparing the scalability properties of several algorithms have been published [36,37].

A dataset is said to be of high dimensionality when it presents some of the following characteristics: the number of learning samples is very high; or the number of input features is very high; or the number of groups or classes to be classified is very high. A special case of high dimensionality is the one that occurs when having a much higher number of features than samples, so many of the features can be irrelevant or redundant. These features can have negative effect on learning models, and decrease their performance significantly [52]. As a typical example, DNA microarray data could contain more than 30000 features with a sample size of usually less than 100. With datasets of this type, most techniques can become unreliable.

Theoretically, having more data should give more discriminating power. However, the nature of high dimensionality of data can cause the so-called problem of *curse of dimensionality* or *Hughes effect* [20]. This phenomenon occurs when the model has to be learned from a finite number of data samples in a high-dimensional feature space with each feature having a number of possible values (either discrete of continuous), and so an enormous amount of training data are required to ensure that there are several samples with each combination of values. The Hughes effect is therefore known as the situation in which with a fixed number of training samples, the predictive power of the learner reduces as the feature dimensionality increases.

Dimensionality reduction techniques are usually applied to deal with this problem, so that the set of features needed for describing the problem can be reduced. *Feature selection* is one of those techniques and consists of detecting the relevant features and discarding the irrelevant ones in order to reduce the input dimensionality and, most of the time, to achieve an improvement in performance [14]. The benefits of feature selection have been proven by researchers in diverse high-dimensional fields such as bioinformatics [3] or intrusion detection [4]. There exist two major approaches in feature selection: *individual evaluation* and *subset evaluation*. Individual evaluation is also known as feature ranking [15] and assesses individual features by assigning them weights according to their degrees of relevance. On the other hand, subset evaluation produces candidate feature subsets based on a certain search strategy. Besides this classification, feature selection methods can also be divided into three models: *filter*, *wrapper* and *embedded* methods. While wrapper models use a specific prediction method as a black box to score subsets of features as part of the

selection process, filter models rely on the general characteristics of the training data to select features with independence of any predictor. Halfway these two models one can find the embedded methods, which perform feature selection as part of the training process of the prediction model. By having some interaction with the classifier, wrapper and embedded methods tend to give better performance results than filters.

There exists a vast body of feature selection methods in the literature, based on distinct metrics (e.g. entropy, probability distributions, information theory or the predictor's accuracy). The proliferation of feature selection algorithms, however, has not brought about a general methodology that allows for intelligent selection from existing algorithms. In order to make a correct choice, a user not only needs to know the domain well, but also is expected to understand technical details of available algorithms [29]. On top of this, most algorithms were developed when dataset sizes were much smaller, but nowadays distinct compromises are required for the case of small-scale and large-scale (big data) learning problems. Small-scale learning problems are subject to the usual approximation-estimation trade-off. In the case of large-scale learning problems, the trade-off is more complex because it involves not only the accuracy of the selection but also other aspects. Stability, that is the sensitivity of the results to training set variations, is one of such factors, with a few studies published regarding the behavior of filters in the case in which training set is small, but the number of features can be high [8,11,13]. The other important aspect, scalability, that is the behavior of the algorithms in the case in which the training set is increasingly high is still more scarce in the scientific literature [35], and the studies are mainly concentrated in obtaining scalability in a particular application [30], modifying certain previously existing approaches [43], or adopting on-line [19] or parallel [51] approaches. In general, one can say that most of the classical feature selection approaches that are univariate -that is each feature is considered separately- have an important advantage in scalability, but at the cost of ignoring feature dependencies, and thus perhaps leading to lower performances than other feature selection techniques. To improve performance, multivariate techniques are proposed, but at the cost of reducing scalability [2]. In this situation, the scalability of a feature selection method becomes extremely important.

In this research, the scalability of state-of-the-art feature selection methods is studied, checking their performance in an artificial controlled experimental scenario, contrasting the ability of the algorithms to select the relevant features and to discard the irrelevant ones when the dimensionality increases and without permitting noise or redundancy to obstruct this process. For analyzing scalability, new evaluation measures are proposed, which need to be based not only on the accuracy of the selection, but also on other aspects such as the execution time or the stability of the returned features. Finally, real experiments are presented in order to check if the conclusions extracted from this theoretical study can be extrapolated to real scenarios.

The aim of this work is to provide the interested user with some insights about the use of feature selection methods and their abilities when it comes to scalability issues. Although it is well known, for instance, that univariate methods are computationally less expensive than multivariate ones, it is also interesting to know, among each group, which methods are the most affordable. Moreover, there are situations in which an user is willing to sacrifice accuracy of the selection in favor of reducing

the execution time, and the analysis presented here can help. Finally, through some of the metrics proposed to measure distinct aspects of scalability, we are able to give the user an idea about, for example, the minimum amount of data that is necessary to achieve an acceptable accuracy for an specific feature selection method.

The rest of the paper is organized as follows: Section 2 describes the feature selection methods included in this research; Section 3 presents the artificial datasets employed; Section 4 introduces the new measures for assessing the scalability. Sections 5 and 6 report the experimental results on artificial and real data, respectively. Finally, Section 7 provides the discussion and conclusion.

2 Feature selection

Feature selection consists of detecting the relevant features and discarding the irrelevant ones, with the goal of obtaining a subset of features that describes properly the given problem with a minimum degradation of performance. It presents several advantages [14], such as:

- Improving the performance of the machine learning algorithms.
- Data understanding, gaining knowledge about the process and perhaps helping to visualize it.
- Data reduction, limiting storage requirements and perhaps helping in reducing costs.
- Simplicity, possibility of using simpler models and gaining speed.

2.1 Feature relevance and redundancy

Intuitively, a feature can be seen as relevant if it contains some information about the target. In a more formal way, John & Kohavi [21] classified features into three disjoint categories: strongly relevant, weakly relevant, and irrelevant features. According to them, a feature X is considered strongly relevant when the removal of X results in a deterioration of the prediction accuracy of an ideal Bayes classifier. A feature X is said to be weakly relevant if it is not strongly relevant and there exists a subset of features S, such that the performance of the ideal Bayes classifier on S is worse than the performance on $S \cup \{X\}$. A feature is defined as irrelevant if it is neither strongly nor weakly relevant.

Furthermore, a feature can also be considered as redundant, usually in terms of feature correlation [46]. It is widely accepted that two features are redundant to each other if their values are completely correlated, but it might not be so easy to determine feature redundancy when a feature is correlated with a set of features. According to Yu & Liu [46], a feature is redundant and hence should be removed if it is weakly relevant and has a Markov blanket [24] within the current set of features. Since irrelevant features should be removed anyway, they are excluded from this definition of redundant features. Therefore, the optimal subset would contain all the strongly relevant features and those weakly relevant but non-redundant.

2.2 Feature selection methods

This work will test the scalability of the three types of feature selection methods: filters, embedded and wrappers. Some of the methods (Chi-Squared, ReliefF, Information Gain, mRMR, FS-P and SVM-RFE) follow the ranking approach, which consists of assessing individual features by assigning them weights according to their relevance. The remaining algorithms (CFS, FCBF, INTERACT, consistency-based and the wrapper considered) follow the subset evaluation approach, which consists of producing candidate feature subsets based on a certain search strategy. The following subsections describe the methods chosen for the experimental study.

2.3 Filters

2.3.1 ChiSquared

This univariate filter is based on the χ^2 statistic [27] and evaluates each feature independently with respect to the classes. The higher chi-squared, the more relevant is the feature with respect to class.

2.3.2 ReliefF

This multivariate method [25] is an extension of the original Relief algorithm [23]. The original Relief works by randomly sampling an instance from the data and then locating its nearest neighbor from the same and opposite class. The values of the features of the nearest neighbors are compared to the sampled instance and used to update relevance scores for each feature. The rationale is that a useful feature should differentiate between instances from different classes and have the same value for instances from the same class. ReliefF adds the ability of dealing with multiclass problems and is also more robust and capable of dealing with incomplete and noisy data. This method may be applied in all situations, has low bias, includes interaction among features and may capture local dependencies which other methods miss.

2.3.3 Information Gain

The Information Gain filter [38] is one of the most common univariate methods of evaluating features. It is a univariate filter that evaluates the features according to their information gain and considers a single feature at a time. It provides an orderly classification of all the features, and then a threshold is required to select a certain number of them according to the order obtained.

2.3.4 mRMR (minimum Redundancy Maximum Relevance)

This multivariate algorithm [34] selects features that have the highest relevance with the target class and are also minimally redundant, i.e., selects features that are maximally dissimilar to each other. Both optimization criteria (Maximum-Relevance and Minimum-Redundancy) are based on mutual information.

2.3.5 CFS (Correlation-based Feature Selection)

This is a simple multivariate filter algorithm that ranks feature subsets according to a correlation based heuristic evaluation function [18].The bias of the evaluation function is toward subsets that contain features that are highly correlated with the class and uncorrelated with each other. Irrelevant features should be ignored because they will have low correlation with the class. Redundant features should be screened out as they will be highly correlated with one or more of the remaining features. The acceptance of a feature will depend on the extent to which it predicts classes in areas of the instance space not already predicted by other features.

2.3.6 FCBF (Fast Correlation-Based Filter)

The fast correlated-based filter method [47] is a multivariate algorithm that measure feature-class and feature-feature correlation. FCBF starts by selecting a set of features that is highly correlated to the class based on symmetrical uncertainty (SU), which is defined as the ratio between the information gain and the entropy of two features. Then, it applies three heuristics that remove the redundant features and keep the feature that is more relevant to the class. FCBF was designed for high-dimensionality data and has been shown to be effective in removing both irrelevant and redundant features. However, it fails to take into consideration the interaction between features.

2.3.7 INTERACT

The INTERACT algorithm [50] is a subset multivariate filter which uses the same goodness measure as FCBF filter (SU), but it also includes the consistency contribution, which is an indicator about how significantly the elimination of a feature will affect consistency. The algorithm consists of two major parts. In the first part, the features are ranked in descending order based on their SU values. In the second part, features are evaluated one by one starting from the end of the ranked feature list. If the consistency contribution of a feature is less than an established threshold, the feature is removed, otherwise it is selected.

2.3.8 Consistency-based

The filter based on consistency [9,28] evaluates the worth of a subset of features by the level of consistency in the class values when the training instances are projected onto the subset of features. From the space of features, the algorithm generates a random subset S in each iteration. If S contains fewer features than the current best subset, the inconsistency index of the data described by S is compared with the index of inconsistency in the best subset. If S is as consistent or more than the best subset, S becomes the best subset. The criterion of inconsistency, which is the key to success of this algorithm, specify how large can be the reduction of dimension in the data. If the rate of consistency of the data described by selected characteristics is smaller than a set threshold, it means that the reduction in size is acceptable. Notice that this method is multivariate.

2.4 Embedded methods

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

This embedded method was introduced by Guyon in [16]. Its novelty relies in the fact that the importance of a feature is indicated by the weights in a SVM solution. Therefore, feature selection is done backward by iteratively training a SVM classifier and removing each time the least important feature according to the SVM's weights.

2.4.2 Feature Selection - Perceptron, FS-P

FS-P [31] is an embedded method based on a perceptron. A perceptron is a type of artificial neural network that can be seen as the simplest kind of feedforward neural network: a linear classifier. The basic idea of this method consists on training a perceptron in the context of supervised learning. The interconnection weights are used as indicators of which features could be the most relevant and provide a ranking.

2.5 Wrappers

The idea of the wrapper approach is to select a feature subset using a learning algorithm as part of the evaluation function. Instead of using subset sufficiency, entropy or another explicitly defined evaluation function, a kind of "black box" function is used to guide the search. The evaluation function for each candidate feature subset returns an estimate of the quality of the model that is induced by the learning algorithm. This can be rather time consuming, since, for each candidate feature subset evaluated during the search, the target learning algorithm is usually applied several times (e.g. in the case of 10-fold cross validation being used to estimate model quality). By default, in the implementation provided in Weka [17], the algorithm starts with the empty set of features and searches forward –although it also allows backward search–, adding features until performance does not improve further. The learning algorithm to evaluate the goodness of a subset of features is a free parameter, so it allows the user to choose any desired classifier.

2.6 Computational complexity

Table 1 shows a summary of the methods used in this paper together with their classification and theoretical complexity (where n is the number of samples and m is the number of features). Wrappers are excluded from this list because they are formed by combining a search strategy with an induction algorithm, so there are as many as combinations of both techniques and their complexity depend on the complexities of the techniques chosen.

Table 1 Summary of feature selection methods used in this work. Notice that n is the number of samples and m is the number of features.

	Uni / Multivariate	Functional view	Structural view	Complexity
Chi-Squared	Univariate	Ranker	Filter	nm
ReliefF	Multivariate	Ranker	Filter	n^2m
Information Gain	Univariate	Ranker	Filter	nm
mRMR	Multivariate	Ranker	Filter	nm^2
CFS	Multivariate	Subset	Filter	nm^2
FCBF	Multivariate	Subset	Filter	$nmlogm$
INTERACT	Multivariate	Subset	Filter	nm^2
Consistency	Multivariate	Subset	Filter	nm^2
SVM-RFE	Multivariate	Ranker	Embedded	$max(n,m)m^2$
FS-P	Multivariate	Ranker	Embedded	nm

3 Artificial datasets

A common problem when testing the effectiveness of a feature selection method on real data is that the relevant features are usually not known in advance. In these cases, the performance of the feature selection methods clearly rely on the performance of the learning method used afterwards and it can vary notably from one method to another. The main objective of this research is to study the scalability of feature selection methods with independence of the learning method (i.e. classifier, cluster method, etc.). For this reason, the authors have chosen to use artificial datasets to perform this task. The main advantage of these artificial scenarios is the knowledge of the set of optimal features that must be selected, thus the degree of closeness to any of these solutions can be assessed in a confident way.

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

3.1 Corral

The Corral dataset [21] has six binary features (i.e. f_1, f_2, f_3, f_4, f_5, f_6), and its class value is $(f_1 \wedge f_2) \vee (f_3 \wedge f_4)$. Feature f_6 is irrelevant and f_6 is correlated to the class label by 75%, which means that is has the same value than the class label in 75% of the samples. Then, the correct behavior for a given feature selection method is to select the four relevant features (f_1, f_2, f_3, f_4) and to discard the irrelevant ones.

3.2 The Led problem

The LED problem [7] is a simple classification task that consists of, given the active leds on a seven segments display, identifying the digit that the display is representing. Thus, the classification task to be solved is described by seven binary features (f_1 - f_7) and ten possible classes available ($C = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$). A 1 in a feature indicates that the led is active, and a 0 indicates that it is not active.

3.3 Monk1, Monk2 and Monk3

The MONK's problems [44] rely on an artificial robot domain, in which robots are described by six different features $(x_1, \ldots, x_6)$. The learning task is a binary classification task. The logical description of the class of the third problems (Monk1, Monk2 and Monk3) are:

Monk1: $(x_1 = x_2) \vee (x_5 = 1)$

Monk2: $(x_n = 1)$ exactly two $n \in 1, 2, 3, 4, 5, 6$, i.e. two (and only two) of the six features must be 1 and the others must be 0

Monk3: $(x_5 = 3 \wedge x_4 = 1) \vee (x_5 \neq 4 \wedge x_2 \neq 3)$

It is necessary to bear in mind that in Monk3, 5% of the data points are misclassifications, i.e. noise in the target. This means that the correct class of 5% of the samples was shifted.

3.4 XOR

In this problem, the class attribute takes binary values. Features f_1 and f_2 are correlated with the class value with XOR operation (i.e., class equals $f_1 \oplus f_2$). This is a hard dataset for the sake of feature selection because of its non-linearity (unlike Corral dataset, which is a multi-variate dataset).

3.5 Parity

The parity problem is a classic problem where the output is $f(x_1, \ldots, x_n) = 1$ if the number of $x_i = 1$ is odd and $f(x_1, \ldots, x_n) = 0$ otherwise. In this work we have set $n = 3$.

3.6 SD1, SD2 and SD3

These three synthetic datasets (SD1, SD2 and SD3) [53] are challenging problems because of their high number of features (around 4000 in the original datasets) and the small number of samples (75), besides of a high number of irrelevant features. These characteristics reflect the problematic of microarray data, and it is necessary to introduce some new definitions of multiclass relevancy features: full class relevant (FCR) and partial class relevant (PCR) features. Specifically, FCR denotes genes (features)

that serve as candidate biomarkers for discriminating all cancer types. However, PCR are genes (features) that distinguish subsets of cancer types.

SD1, SD2 and SD3 are three-class datasets with 75 samples (each class containing 25 samples) generated based on the approach described in [10]. Each synthetic dataset consists of both relevant and irrelevant features. The relevant features in each dataset are generated from a multivariate normal distribution using mean and covariance matrices [53].

SD1 is designed to contain only 20 FCR. Two groups of relevant genes are generated from a multivariate normal distribution, with 10 genes in each group. Genes in the same group are redundant with each other and the optimal gene subset for distinguishing the three classes consists of any two relevant genes from different groups.

SD2 is designed to contain 10 FCR and 30 PCR. Four groups of relevant, i.e., FCR and PCR, genes are generated from a multivariate normal distribution, with 10 genes in each group. Genes in each group are redundant to each other and in this dataset, only genes in the first group are FCR genes while genes in the three last groups are PCR genes. The optimal gene subset to distinguish all the three classes consists of four genes, one FCR gene from the first group and three PCR genes each from one of the three remaining groups.

SD3 has been designed to contain only 60 PCR. Six groups of relevant genes are generated from a multivariate normal distribution, with 10 genes in each group. Genes in the same group are designed to be redundant to each other and the optimal gene subset to distinguish all the three classes thus consists of six genes with one from each group.

It has to be noted that the easiest dataset in order to detect relevant features is SD1, since it contains only FCR features and the hardest one is SD3, due to the fact that it contains only PCR genes, which are more difficult to detect.

3.7 Configuration of datasets

For assessing the scalability of the methods, different configurations of these datasets were used. In particular, the number of features ranges from 2^3 to 2^7 while the number of samples ranges from 2^3 to 2^{14} (all pairwise combinations). In the case of the SD datasets, the number of features ranges from 2^6 to 2^{12} while the number of samples ranges from 3^2 to 3^5. Notice that the number of relevant features is fixed and it is the number of irrelevant features the one that varies, randomly generated. When the number of samples increases, the new instances are also generated from a random distribution. For example, if we are using the Corral dataset with 2^4 features, this means that we kept fixed the original features of the problem (four relevant features, one correlated 75% to the class and one irrelevant) and then we added 10 extra irrelevant features, to complicate the feature selection task.

Table 2 shows a summary of the different artificial datasets employed in this research, depicting the problems covered by them, as well as the number of features and samples. The column "Relevant features" indicates the relevant features that should be selected for a good feature selector, according to the definitions given in this section. For example, since we have seen that the class label for Monk1 dataset is given

Table 2 Summary of the synthetic datasets used.

Dataset	No. of features	No. of samples	Relevant features	False correlation	Noise	Non linear	No. feat >> No. samples
Corral	2^3 - 2^7	2^3 - 2^{14}	1-4	✓			
Led	2^3 - 2^7	2^3 - 2^{14}	1-7		✓		
Monk1	2^3 - 2^7	2^3 - 2^{14}	1,2,5				
Monk2	2^3 - 2^7	2^3 - 2^{14}	1-6			✓	
Monk3	2^3 - 2^7	2^3 - 2^{14}	2,4,5		✓		
XOR	2^3 - 2^7	2^3 - 2^{14}	1,2			✓	
Parity	2^3 - 2^7	2^3 - 2^{14}	1-3			✓	
SD1*	2^6 - 2^{12}	3^2 - 3^5	G_1, G_2				✓
SD2*	2^6 - 2^{12}	3^2 - 3^5	$G_1 - G_4$				✓
SD3*	2^6 - 2^{12}	3^2 - 3^5	$G_1 - G_6$				✓

* G_i means that the feature selection method must select only one feature within the i-th group of features.

by the equation $(x_1 = x_2) \vee (x_5 = 1)$, this means that the relevant features that should be selected are features 1, 2 and 5.

The first seven datasets in the table are classical datasets, having more samples than features, so they will be studied together. SD1, SD2 and SD3 are datasets which represent the characteristics of microarray data, with more features than samples, and will be analyzed independently.

Notice that with the set of datasets chosen for this study, we cover different types of relevance. For example, in the case of XOR dataset, the first two features are both needed (and they only are relevant in the presence of the other). As for SD datasets, only one feature per group of relevant features is necessary, so once one of the features in this groups is selected, the remaining nine become irrelevant, etc. When we evaluate the ability of the feature selection methods to select the correct features, we are taking all these types of relevance into account.

4 Evaluation metrics

The goal of this research is to assess the scalability of several feature selection methods. For this purpose, some evaluation measures need to be defined, covering different aspects that must be addressed: accuracy, stability and computational time. First, we propose some metrics which take into account the accuracy of the selected features, motivated by the measures proposed in [40,45,49]. It is also important to have measures which evaluate the stability of feature selection, i.e. the insensitivity of the result of a feature selection algorithm to variations in the training set. For this reason, new metrics for assessing this issue will also be proposed, based on [22,26,42]. Last but not least, the training time (which will be reported in seconds) is also a measure of success for the scalability of a feature selection method.

Regarding the accuracy and stability, the evaluation measures used in this work are also divided in two types: the ones devoted to subset methods and those devoted to ranking methods. For the sake of clarity, we have decided that all these measures are desirable to be minimized. Therefore, the measures related to how *accurate* the selection is are focused on the *error*, while the metrics related to the *stability* are now considered as related to the *distance* between rankings or subsets of features. All these measures but the training time are bounded between 0 and 1.

In the next subsections, the measures for accuracy and stability evaluation are described in which:

- $feat\_sel$ stands for the subset of selected features, in the case of subset methods, or for the ranking of features returned, in the case of rankers.
- $feats$ is the total set of features,
- $feat\_rel$ is the subset of relevant features, and $feat\_irr$ represents the subset of irrelevant features (both of them known *a priori*).

4.1 Measures for ranker methods

This section describes the evaluation measures applied to the feature selection methods which return an ordered ranking of the features. In order to evaluate the accuracy of the rankers, the following measures are proposed:

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

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

- The *average_error* (E) evaluates the mean of E_i, in which $i \in feats\_sel$ and E_i is the average fraction of relevant features ranked above a particular feature i.

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

In relation with the distance, these are the proposed measures:

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

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

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

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

where

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

4.2 Measures for subset methods

Analogously to the previous section, this subsection presents the measures used when using subset feature selection methods. The measures to evaluate the error in selecting relevant features are:

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

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

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

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

Finally, two additional measures are proposed to evaluate distance between subsets:

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

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

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

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

4.3 Summary of measures

Using the measures described above, related with the error and the distance, and motivated by the methodology proposed in [41], we define three figures from which eight scalar measures are extracted that cover jointly the aspects of *error*, *distance* and *computational time*.

- Error surface: *Feature size* vs *Sample size* vs *Error*. It is obtained by displaying the evolution of the error measure across the feature-sample space. The following scalar measures are computed:
  1. *MinEr*: minimum error.
  2. *Er5%*: the minimum amount of data (features x samples) for which the error drops below a threshold (5% of error).
  3. *VuEr*: volume under the error surface.
- Distance surface: *Feature size* vs *Sample size* vs *Distance*. It is obtained by displaying the evolution of the distance measure across the feature-sample space.
  4. *MinDi*: minimum distance.
  5. *Di5%*: the minimum amount of data (features x samples) for which the distance drops below a threshold (5% of distance).
  6. *VuDi*: volume under the distance surface.
- Training time surface: *Feature size* vs *Sample size* vs *Traning time*. It is obtained by displaying the evolution of the training time across the feature-sample space.
  7. *MaxTt*: training time in seconds for the maximum amount of data tested.
  8. *VuTt*: volume under the training time surface.

To obtain these graphs, it is necessary to have a single measure for the error and the distance. Therefore, since all the measures are bounded between 0 and 1 and are desired to be minimized, the arithmetic mean of the measures of the same group is computed, that is,

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

In Figure 1 we can see an example of how the three measures related to error are calculated. Notice that this is an example in 2D and that is why we are referring to *AuEr* instead of *VuEr*.

5 Experimental results and scalability analysis

This section shows the scalability results according to the measures explained above. The experiments are performed such that, for each configuration of a determined

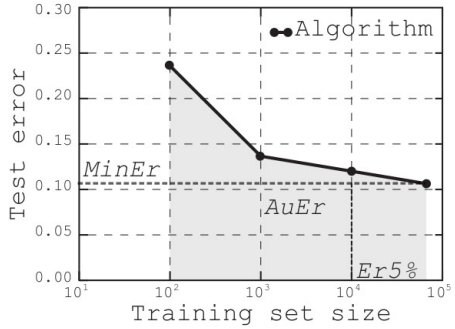


Fig. 1 Example of scalar measures related to test error vs. training set size

number of features and samples, 10 repetitions are executed, in which the data is extracted randomly from the complete available dataset. For example, if we are testing a configuration with 8 features and 16 samples, the samples are selected randomly from the whole set of samples. The process to extract the features is, however, different. If we are dealing with a dataset which has 4 relevant features (as Corral), these are fixed and the remained 4 are extracted randomly among the set of irrelevant features. The reason to repeat this procedure 10 times is to be able to measure the stability of the methods; i.e. the sensitivity of the algorithms to changes in the training set. Note, moreover, that a "good" algorithm should be able to select the 4 relevant features no matter what the 4 remaining features are. Section 5.1 is devoted to show the scalability of filter methods, while Sections 5.2 and 5.3 are dedicated to wrapper and embedded methods, respectively. Notice that mRMR and FS-P are implemented in Matlab, while the remaining feature selection methods are available in the popular Weka tool [17]. All experiments were carried out on an Intel(R) Xeon (R) CPU W3550 @ 3.07 QUAD-CORE with 12 GB RAM.

5.1 Scalability of filters

This subsection studies the scalability properties shown by the eight filter methods considered (FCBF, CFS, consistency-based, INTERACT, InfoGain, ReliefF, Chi-Squared and mRMR). As an example, Figures 2 and 3 plot an scalar metric of scalability per row (error, distance and time) of ranker and subset methods, respectively, for Corral dataset.

In this color-based graphs, the x-axis refers to the sample size, while the y-axis relates to the feature size. Notice that the cold colors represent low values of the evaluation measures whereas the warm ones stand for high values. If the color scale changes more in the sample size axis, it means that this metric is more affected by the sample size than by the feature size. However, if colors are stable in some axis, it means that this metric is not affected by increments in sample or feature size in each case.

In general, the error is more affected by the number of samples than of features. This means that these methods are not highly affected by irrelevant features (remember that, when we increase the number of features to test, what we do is to add irrelevant features) as long as they are provided with enough samples to learn the classification task. Among the ranker methods (Figure 2), ReliefF seems to be the method that requires the smallest number of features to achieve a low error, although at the cost of needing the highest running time (up to 500 seconds). This distinction in behavior is not so notable in subset methods (Figure 3).

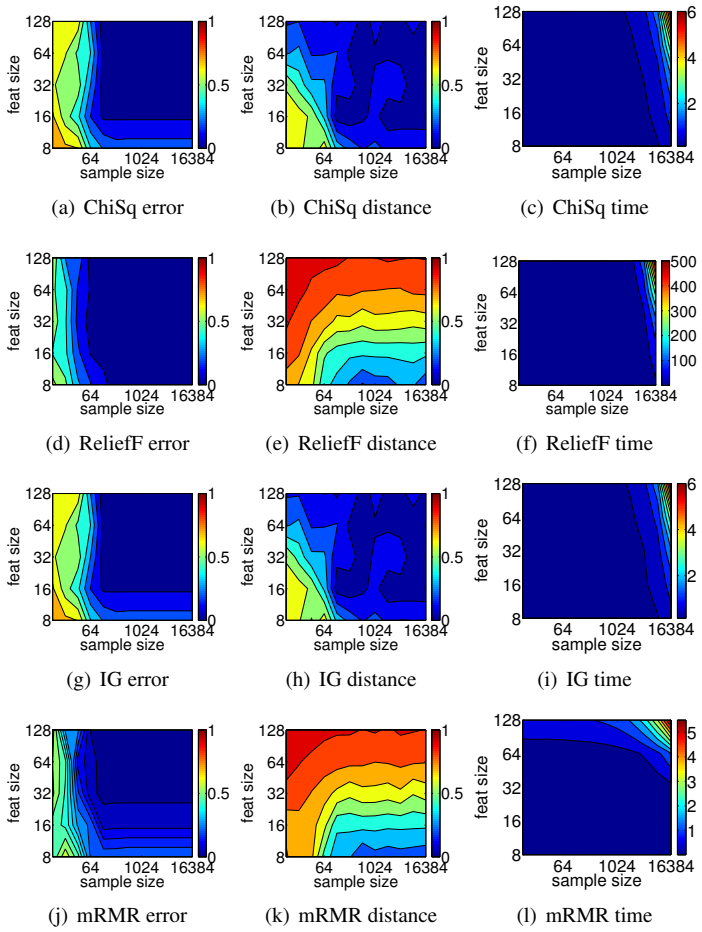
In terms of distance, which is a manner of measuring the stability of the features selected by the algorithms, ranker methods (Figure 2) present two different behaviors based on if they are univariate or multivariate. Chi-Square and InfoGain are univariate, i.e. they only consider features independently, therefore when they are trained with enough samples and features, their results are very stable, specially when the number of features is fixed. On the other hand, ReliefF and mRMR are multivariate, i.e. they consider interaction between features. In this case, as expected, these methods are mainly affected by the number of features, producing more dissimilar results as the number of features increases. Regarding subset methods (Figure 3), they seem to be more affected by the sample size. In the case of the Consistency-based filter, after a determined number of samples, it is not affected by the increase in the number of irrelevant features. This is happening because it tries to find the minimum number of features such that any function maps from the values of the features to the class labels so this method is very effective in selecting a small subset of relevant features.

Regarding the training time, it has to be noted that these plots are not bounded between 0 and 1, since the time has not been normalized. For this reason, the reader has to bear in mind that in some cases, such as ReliefF (Figure 2(f)) the time is up to 500 seconds, since this is the only method with a complexity quadratic in the sample size. In the remaining cases, the times are between 1 second (FCBF) and 12 seconds (Consistency and INTERACT). As expected, the training times raise significantly with large amounts of both samples and features. On the contrary, mRMR is more influenced by the number of features, as its theoretical complexity is quadratic in the number of features.

For the sake of brevity, it is not possible to plot all the measures for all the datasets considered in this study, although the supplementary material can be accessed online[1]. Table 3 depicts the eight scalar measures extracted from the graphs as explained in Section 4.3 that were calculated for every ranker filter over the first seven datasets of Table 2. Similarly, results are shown in Table 4 for the subset filters. Notice that the lower the value, the better the performance of the feature selection method.

It is remarkable the behavior of ReliefF in terms of error, since it is able to achieve the minimum for all datasets considered. As pointed out in a previous study of state-of-the-art feature selection methods [5], ReliefF is good choice independently of the particularities of the data, since it has proven to be effective in detecting redundant features, even in complex scenarios such as XOR, although this comes at the cost of requiring more time than other ranker methods. Notice that XOR and Parity are

[1] http://www.lidiagroup.org/index.php/en/materials-en.html



(a) ChiSq error (b) ChiSq distance (c) ChiSq time

(d) ReliefF error (e) ReliefF distance (f) ReliefF time

(g) IG error (h) IG distance (i) IG time

(j) mRMR error (k) mRMR distance (l) mRMR time

Fig. 2 Measures of scalability of filter ranker selection methods in the Corral dataset

datasets in which the classification task is non-linear, so the remaining methods do not perform well in this type of scenarios (some minimum errors are up to 77%).

Regarding the distance, it is easy to note that subset methods tend to obtain a minimum of 0 in more cases than ranker methods. This can be explained because of the differences between approaches. If we are dealing with a dataset as Corral, in which only the first four features are relevant and the remaining ones are irrelevant, even when a ranker method performs correctly and ranks the four relevant features on the top of the ranking, it needs to deal with the order of the remaining features. If they are equally irrelevant, the order might be random. This is correct from the point of view of error measures, but it might lead to high distance values. However, since subset methods select a subset of the features, if a method performs correctly then it only selects the four relevant features and discards the rest so it will achieve satisfactory results both in error and distance measures.

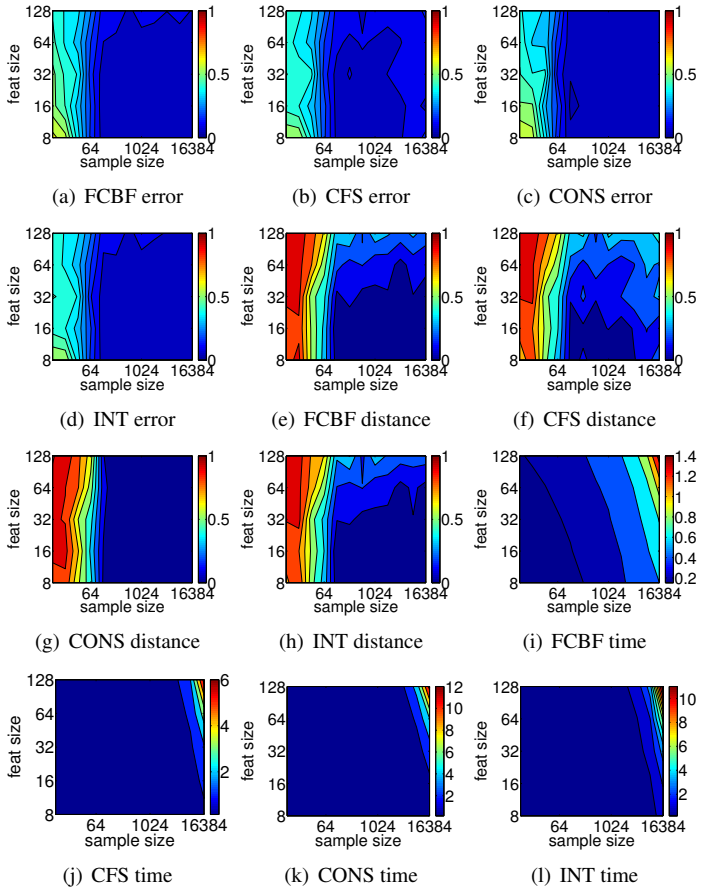


Fig. 3 Measures of scalability of subset methods in the Corral dataset

In order to determine which feature selection method performs better in terms of the three groups of measures –error, distance and time–, some statistical tests will be applied. In particular, the Friedman test [12] is a non-parametric equivalent of the repeated-measures ANOVA. It ranks the algorithms for each dataset separately, the best performing algorithm ranked first, the second best ranked second, etc. If the null-hypothesis is rejected, we can proceed with a post-hoc test. The Nemenyi test [32] is similar to the Tukey test for ANOVA and is used when all algorithms are compared in pairs. The performance of two algorithms is significantly different if the corresponding average ranks differ by at least a critical difference.

Figure 4 plots the results of the statistical tests carried out to compare the ranker filters according to their performance on all datasets. The blue bar represents the best result, the grey bars represent results that are not significantly worse than the best result, and red bars represent results that are significantly worse than the best result (marked in blue). Notice that grey bars represent methods that are not statistically

Table 3 Precision, stability and time measures for ranker filters on classical datasets

	Dataset	MinEr	Er5%	VuEr	MinDi	Di5%	VuDi	MaxTt	VuTt
Chi-Squared	Corral	0.0079	32768	9.1029	0.0167	262144	8.4638	6.0592	18.6429
	Led	0.0000	128	2.1988	0.0000	256	3.7704	6.0282	18.4839
	Monk1	0.6369	256	30.6699	0.0000	1024	6.0577	6.2042	18.7184
	Monk2	0.0117	262144	24.1209	0.0536	2097152	11.8554	6.2060	18.8257
	Monk3	0.5958	1024	28.9323	0.0000	524288	8.8972	6.1988	18.6721
	Xor	0.7790	128	36.9307	0.0000	512	5.0799	6.1139	18.7577
	Parity	0.7232	64	34.1360	0.0000	4096	6.2034	6.1754	18.8630
ReliefF	Corral	0.0000	4096	4.1150	0.1881	4096	29.0698	515.9786	575.9859
	Led	0.0000	256	2.1872	0.0000	512	22.8144	514.8453	581.2695
	Monk1	0.0000	32768	25.5442	0.0508	131072	33.0739	595.3573	658.5203
	Monk2	0.0000	2048	11.7908	0.2389	65536	31.4784	595.5261	658.0240
	Monk3	0.0000	32768	25.3734	0.1841	131072	35.7227	595.0200	658.4013
	Xor	0.0000	512	3.2454	0.0254	65536	33.9882	517.1896	578.3513
	Parity	0.0000	1024	8.2046	0.0836	65536	32.7301	516.7137	579.7248
Info Gain	Corral	0.0080	32768	9.1012	0.0167	262144	8.4473	6.1201	18.6495
	Led	0.0000	128	2.2039	0.0000	128	2.7849	6.0985	18.4220
	Monk1	0.6369	256	30.6701	0.0000	1024	6.0498	6.2304	18.6275
	Monk2	0.0117	262144	24.1223	0.0536	2097152	11.7702	6.1584	18.6140
	Monk3	0.5958	1024	28.9316	0.0042	524288	8.9692	6.1515	18.6950
	Xor	0.7790	128	36.9307	0.0000	512	5.0780	6.1464	18.7278
	Parity	0.7232	64	34.1360	0.0000	4096	6.2021	6.2804	18.8549
mRMR	corral	0.0080	32768	6.2805	0.1952	4096	28.9932	5.8203	16.6733
	Led	0.1464	262144	22.1064	0.0000	1024	24.3721	5.6627	16.3052
	Monk1	0.4292	2048	24.5068	0.6638	128	37.3092	5.2267	16.0599
	Monk2	0.0000	131072	17.4071	0.3630	65536	35.5783	5.1817	15.9455
	Monk3	0.2592	262144	19.0268	0.6653	256	36.6742	5.2082	16.1028
	Xor	0.4500	8192	25.8610	0.6751	128	37.8049	5.8208	16.7248
	Parity	0.5077	128	27.8600	0.6939	128	37.9318	5.8902	16.7714

worse than the best one but this does not imply that these methods are significantly better than those marked with red bars. There might be also cases in which methods with grey bars are significantly better than methods with red bars, as will be commented throughout the text. For example, in Fig. 4(d) it is easy to see that the InfoGain bar does not overlap with those bars from ReliefF and mRMR, so we can say that InfoGain is significantly better than ReliefF and mRMR.

In terms of error, ReliefF clearly outperforms the other methods according to the minimum error achieved. In fact, and as commented before, in Table 3 one can see that the minimum error obtained by this algorithm is zero for all datasets. As expected, this is reflected also in the volume under the curve, where ReliefF is better than ChiSquared and InfoGain with significant differences. Focusing on the minimum amount of data for which the error drops below the 5% of its minimum value (Er5%), InfoGain is the only one which improves significantly its performance with respect to mRMR, while among the other methods there are no significant differences.

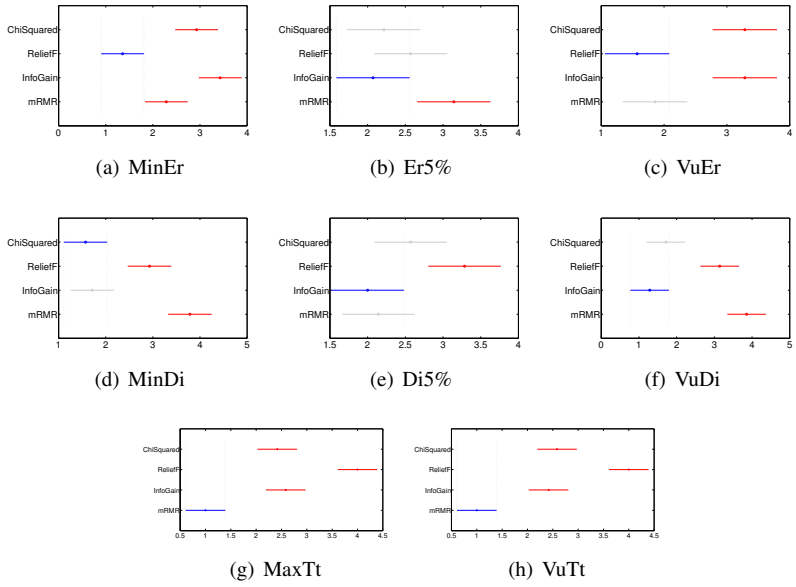
However, when examining the results related with the distance measure, InfoGain and ChiSquared are significantly better than ReliefF and mRMR in two metrics (MinDi and VuDi). This was expected since the former are univariate methods and the latter are multivariate and, therefore, more sensitive to changes in the training set. However, InfoGain and mRMR are significantly better than ReliefF in terms of

Table 4 Precision, stability and time measures for subset filters on classical datasets

	Dataset	MinEr	Er5%	VuEr	MinDi	Di5%	VuDi	MaxTt	VuTt
FCBF	Corral	0.0556	2048	6.7892	0.0000	2048	12.5822	1.4894	15.8205
	Led	0.0424	8192	4.4189	0.0000	128	2.5076	1.4831	15.7518
	Monk1	0.3625	131072	21.7390	0.0000	1024	16.0508	1.5214	15.6129
	Monk2	0.0153	262144	19.8610	0.0593	262144	28.2990	1.5487	15.8478
	Monk3	0.1983	524288	15.3464	0.0000	4096	14.0688	1.4964	15.8101
	Xor	0.4941	1024	25.1410	0.0000	512	23.1093	1.4875	15.7131
	Parity	0.4910	1024	25.8070	0.0000	4096	24.4750	1.5799	15.7908
CFS	Corral	0.0556	8192	7.3681	0.0000	8192	14.6420	6.3441	19.8427
	Led	0.0424	8192	4.3379	0.0000	128	2.7192	6.4241	19.9178
	Monk1	0.3625	131072	21.5999	0.0000	1024	16.0058	6.4388	19.5991
	Monk2	0.0696	2097152	18.2050	0.2000	1024	28.6166	6.4575	19.7130
	Monk3	0.2001	524288	15.2719	0.0000	4096	14.3732	6.3198	19.5398
	Xor	0.1979	16384	14.3294	0.0000	512	23.4340	6.3857	19.6698
	Parity	0.3094	2048	17.7179	0.0000	2048	24.3923	6.4405	19.7047
Consistency	Corral	0.0389	2048	5.9283	0.0000	4096	9.6545	12.7385	29.3157
	Led	0.0911	4096	6.2874	0.0000	128	1.2504	12.1598	28.0777
	Monk1	0.3767	8192	21.5879	0.0000	1024	8.4597	9.4863	24.3099
	Monk2	0.4924	1024	28.0370	0.9444	128	59.7000	8.9761	23.4063
	Monk3	0.1784	524288	15.1243	0.0000	2048	8.9236	10.9881	26.3761
	Xor	0.4948	1024	24.2377	0.8667	1024	52.7780	9.2464	23.9472
	Parity	0.4902	512	24.9375	0.9111	128	51.9956	9.3657	24.0343
INTERACT	Corral	0.0556	2048	6.5871	0.0000	2048	12.1686	11.9269	29.0494
	Led	0.0424	8192	4.3853	0.0000	128	2.8674	11.7323	28.5087
	Monk1	0.3625	131072	21.6233	0.0000	1024	16.0588	12.0135	28.7705
	Monk2	0.0153	262144	17.6704	0.0593	262144	27.4113	12.0833	28.7912
	Monk3	0.1942	524288	15.1910	0.0000	4096	14.1113	11.9160	28.8462
	Xor	0.1979	16384	14.3258	0.0000	512	23.4367	11.9244	28.7487
	Parity	0.3094	2048	17.7327	0.0000	2048	24.4046	11.9783	28.8000

Di5%, which means that although mRMR is not the method which achieves the minimum distance, it stabilizes rapidly. Finally, with regard to the training time, mRMR clearly beats the other methods, even when its theoretical complexity is lower than that of InfoGain and ChiSquared. Remark that ReliefF obtains very good results in terms of error (0%) even when for XOR and Parity the minimum error obtained by other methods ranges from 40.87% to 76.50%. Nevertheless, this is accomplished at the expense of a much longer training time than the other rankers (see Table 3, around 100 times longer).

Figure 5, in turn, displays the results of the statistical tests executed to compare the different subset filters based on their scalability properties on all datasets. With regard to the error, the results obtained by INTERACT outperform significantly those achieved by FCBF and Consistency-based in terms of both MinEr and VuEr. Notice that this is an interesting result, since INTERACT and FCBF share the first part of their algorithms, which consists of calculating the feature-class correlation by means of symmetrical uncertainty. Then, FCBF removes the redundant features based on the concept of predominant features whereas INTERACT removes features that are not consistent. In light of these results, it seems that the INTERACT's strategy is more adequate for the suite of problems tested in this work. With regard to Er5%, the



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

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

(g) MaxTt (h) VuTt

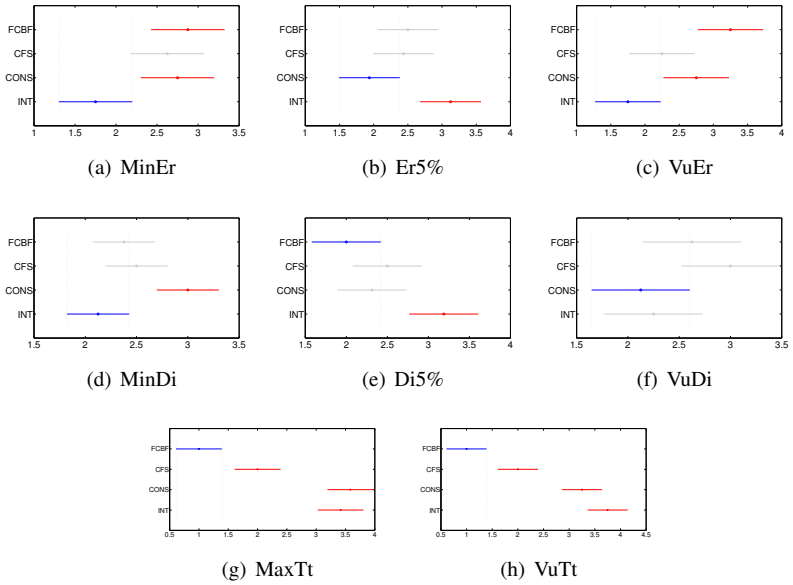
Fig. 4 Comparison of scalability measures for ranker filters (ChiSquared, InfoGain, ReliefF and mRMR)

performance of the filter Consistency-based is significantly better than INTERACT, which means that the former requires a significantly smaller amount of data than the latter to achieve its lowest error.

In terms of distance, FCBF achieves a low value in a significantly smaller amount of data than INTERACT (see Figure 5(e)), as well as outperforms the filter Consistency-based significantly regarding the minimum distance. As for the training time, FCBF is the option which requires the lowest training times. Actually, from the subset methods, it is the only one with a theoretical complexity smaller than quadratic in the number of features. But also notice that for the other three methods, CFS obtains significantly better results than Consistency-based and INTERACT, even when they all have the same theoretical complexity ($\mathcal{O}(nm^2)$, see Table 1).

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

In light of those results, Table 5 shows an overview of the behavior of the different FS methods in the three considered metrics (error, distance and training time), where the larger the number of dots, the better the behavior. The number of dots is calculated according to the following. For each metric, we recall the worst value for all methods and datasets. Each specific result is rated with one bullet if it is below the 20% of the worst value; two dots if it is between 20% and 40%, and so on. Then, we compute the mean for each FS method across all datasets. Notice that, since the error and distance



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

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

(g) MaxTt (h) VuTt

Fig. 5 Comparison of scalability measures for subset filters (FCBF, CFS, Cons and INTERACT)

measures are not the same for ranking and subset methods, we separate accordingly the methods to calculate the number of dots, for the sake of fairness.

According to the summary presented in Table 5, it remains clear the predominance of ReliefF in terms of accuracy of the selection, although InfoGain shows better performance according to stability. The methods that require a smaller training time in this set of experiments are mRMR and FCBF.

Table 5 Overview of the behavior regarding scalability of classical datasets (notice that the larger the number of dots, the better the behavior).

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

5.1.1 Case study: SD datasets

These synthetic datasets have a small ratio between number of samples and features, which makes difficult the task of feature selection. This is the problematic present

in microarray data, a hard challenge for machine learning researchers. Besides this particularity of the data, there is a high number of irrelevant features for the task of gene classification and, moreover, the presence of redundant variables is a critical issue.

Tables 6 and 7 report the eight scalar measures defined in section 4.3 for the ranker and subset methods, respectively, along with all the pairwise combinations between filter and dataset for the SD datasets (three last rows of Table 2). Focusing on the ranker methods, one can see that in terms of error (MinEr, Er5% and VuEr), mRMR seems to be the best alternative, obtaining 0% of minimum error with a small amount of data. In turn, with regard to the distance, Chi-Squared and InfoGain –univariate methods– return stable rankings (low distance) but they need a significant amount of data to do it, while ReliefF and mRMR –multivariate methods– show high distances (in the case of mRMR, over 75%). This might seem surprising, since one can expect that a method which obtains the minimum error (0%) would also be highly stable. However, SD datasets have the particularity of consisting of groups of features equally relevant, but redundant among each other so when one of the features in the group is selected (or ranked top) the remaining ones in the group are deemed as irrelevant. Therefore, if we have two groups of relevant features (10 in each group), as it is the case with SD1 (see Section 3.6), mRMR can be selecting on the top of the ranking features 1 and 11 in the first repetition, 2 and 12 in the second repetition and so on. In this case, the error will be 0, but the method will achieve a high distance since the rankings are very dissimilar. Moreover, when performing several repetitions of a particular experiment, it is common that low minimum errors are obtained together with high variance.

Finally, the training time required by mRMR is in the order of thousands of seconds while the remaining methods require in the order of seconds. The only of these methods which theoretical complexity is quadratic to the number of features is mRMR, and thus its time raises significantly as the number of features increases (in these experiments, up to 4096 features). Notice that ReliefF in this case does not require high times as in the previous set of experiments, because it is quadratic in the number of samples and linear in the number of features, and SD datasets have a small number of samples. To sum up, in this scenario the best ranker in terms of error seems to be mRMR, but at the cost or requiring large amounts of time and not being stable. In turn, InfoGain and Chi-Square obtain also acceptable errors, they become stable although require certain amount of data, and the computational cost is acceptable (low training time).

Table 7 displays the results for the subset filters. First of all, it is worth mentioning that the poor distance results are due to the fact that the SD datasets have an extremely high number of features (up to 4096), so the more features, the more difficult is to select stable subsets of features. Having said that, FCBF is clearly the best option since it obtained the best results in terms of error and training time, although at the cost of being the least stable. To sum up, Table 8 provides some guidelines for the specific scalability aspects considered for the SD datasets. Note that the larger the number of bullets, the better the behavior.

Table 6 Precision, stability and time measures for ranker filters on SD datasets

	Dataset	MinEr	Er5%	VuEr	MinDi	Di5%	VuDi	MaxTt	VuTt
Chi-Sq	SD1	0.0013	995328	2.9749	0.0046	124416	1.9844	1.5427	8.9223
	SD2	0.0292	497664	4.5912	0.0110	995328	2.8399	1.5328	8.8875
	SD3	0.0506	497664	5.3777	0.0144	995328	3.8343	1.5704	8.9160
ReliefF	SD1	0.0018	995328	2.5605	0.4659	15552	15.6611	6.2052	12.2663
	SD2	0.0458	995328	5.4852	0.3294	15552	15.2985	6.2139	12.2390
	SD3	0.0348	995328	6.8454	0.5138	31104	15.7350	6.2747	12.2809
IG	SD1	0.0012	995328	2.9873	0.0046	124416	1.9709	1.5720	8.8356
	SD2	0.0286	497664	4.5834	0.0112	995328	2.8593	1.5265	8.8789
	SD3	0.0516	497664	5.3965	0.0145	995328	3.8127	1.5387	8.8891
mRMR	SD1	0.0000	15552	2.2081	0.8482	5184	17.1517	1173.0168	3824.0483
	SD2	0.0000	15552	3.9693	0.7537	15552	16.9643	1173.5756	3832.1888
	SD3	0.0000	15552	4.3885	0.8606	576	17.0893	1170.2292	3822.7756

Table 7 Precision, stability and time measures for subset filters on SD datasets

	Dataset	MinEr	Er5%	VuEr	MinDi	Di5%	VuDi	MaxTt	VuTt
FCBF	SD1	0.0000	15552	5.4619	0.9096	1728	17.5728	1.4802	9.2476
	SD2	0.0059	31104	5.5995	0.9167	576	17.5259	1.5305	9.4102
	SD3	0.0127	31104	5.6006	0.8556	576	17.6263	1.5731	9.4523
CFS	SD1	0.2558	5184	6.6367	0.7612	124416	16.7912	33.9655	63.9189
	SD2	0.2118	15552	6.5132	0.8336	5184	16.8861	33.3939	63.1207
	SD3	0.2121	20736	6.4843	0.6889	576	17.0439	39.2505	69.8266
Cons	SD1	0.2059	5184	6.6976	0.6713	15552	16.3032	8.5832	17.5273
	SD2	0.2186	5184	6.6849	0.7581	15552	16.6353	10.9020	19.2501
	SD3	0.2342	10368	6.6529	0.7953	15552	16.8165	12.1414	20.6664
INT	SD1	0.1948	5184	6.7782	0.6784	15552	16.5033	8.1722	20.2094
	SD2	0.2233	5184	6.6980	0.7819	15552	16.8550	13.5587	22.1351
	SD3	0.2236	10368	6.5545	0.7222	576	17.1614	15.6778	23.6967

Table 8 Overview of the behavior regarding scalability of SD datasets (notice that the large the number of dots, the better the behavior).

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

5.1.2 Case study: varying the redundancy

Redundancy is an important problem for feature selection methods, since selecting redundant features is not desirable and it can hinder the subsequent process of classification. Therefore, in this case study we will analyze the stability of the filter methods

when the level of redundancy varies. For this, we chose a dataset used in a recent research work [33]. It is a synthetic dataset which represents a binary classification problem where only the first 50 features are relevant to the target class (the total number of samples and features can be configured by the user). Instances of the positive class are i.i.d. drawn from a normal distribution with mean $\mu_+ = (1, ..., 1, 0, ..., 0)$ —i.e. the first 50 values are ones and the remaining ones are zeros— and covariance matrix:

$$\Sigma = \begin{bmatrix} \Sigma^*_{50\times 50} & \mathbf{0}_{50\times 50} \\ \mathbf{0}_{50\times 50} & \mathbf{I}_{50\times 50} \end{bmatrix}$$

where $\Sigma^*_{50\times 50}$ is the matrix with ones on the diagonal. The mean for the negative class is taken equal to $\mu_- = (-1, ..., -1, 0, ..., 0)$. A parameter ρ controls the degree of redundancy everywhere else. The larger the value of ρ, the more the 50 relevant features will be correlated to each other.

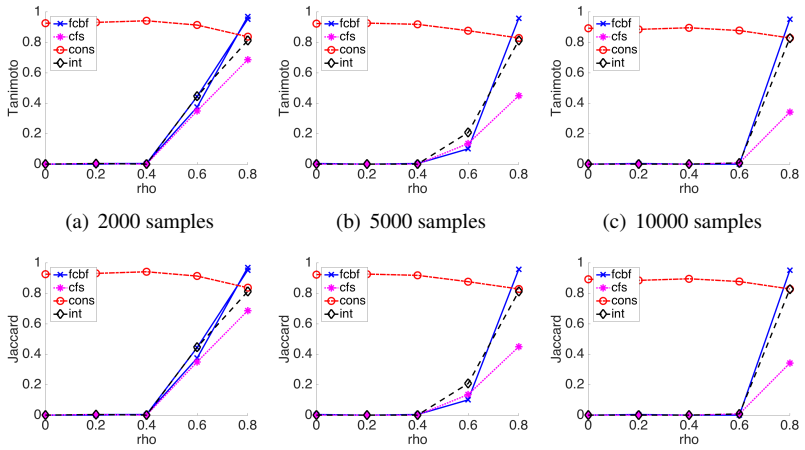
In these experiments we only analyze the variations in stability under different values of ρ. This is because, in this scenario, it makes no sense to employ the measures defined for the error. We know that the 50 first features are the relevant features, but when a high level of redundancy is induced, it might be possible that not all the 50 first features are necessary for a correct identification of the problem. However, the error measures defined herein do not take this aspect into account, since the set of relevant features remains fixed. In fact, what happens is that univariate methods such as Information Gain or Chi-Square (which do not take interactions between features into account) seem to obtain better error performance than multivariate methods such as ReliefF or mRMR, which do not select some of the relevant features when the degree of redundancy increases.

As for the training time, we think that it is not interesting to measure it in this case study, since increasing the degree of redundancy does not have an effect on the training time. For all these reasons, we focus the analysis of this case study on the stability of the feature selection methods.

Our experimental procedure is as follows. We set the number of features as 100, in which the first 50 features are the relevant ones. We try different numbers of samples $(2000, 5000, 10\,000)$ and several degrees of redundancy $\rho = [0, 0.2, 0.4, 0.6, 0.8]$, aiming at checking how this affects to the stability of the selection.

Figure 6 shows the Tanimoto and Jaccard stability metrics for the subset filter methods (FCBF, CFS, Consistency-based and INTERACT). The behavior of Consistency-based is very unstable. Notice that this method is based on a consistency measure that attempts to find a minimum number of features that can separate the classes as consistently as the full set of features—given that an inconsistency is defined as two instances having the same features but different class labels. The problem with this method is that it tends to select too small subsets of features, making it very unstable and even obtaining bad subsequent classification results [5]. If we focus on the remaining methods, it can be seen that, on the one hand, increasing the degree of redundancy makes the methods more unstable. This is because this kind of methods are able to detect redundancy so they do not select some of the 50 relevant features. But, as the level of redundancy increases, it becomes more difficult to select the same

subset of features across the different repetitions of the procedure. And, on the other hand, increasing the number of samples contributes to obtain more stable selections, since the feature selection methods have more data to decide which features to select. In light of these experiments, the subset method that seems to be more stable with respect to increasing redundancy is CFS, which also showed outstanding results in terms of distance when applied to classical datasets (see Table 5). Actually, CFS is the only method that is able to improve stability when the number of samples increases. Notice that the remaining methods, when $\rho = 0.8$ do not become more stable as the number of samples is larger.



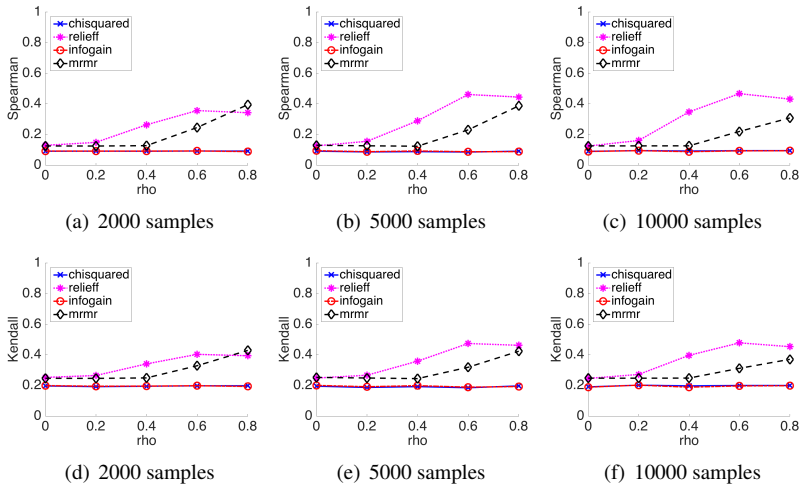
(a) 2000 samples (b) 5000 samples (c) 10000 samples

(d) 2000 samples (e) 5000 samples (f) 10000 samples

Fig. 6 Tanimoto (top) and Jaccard (bottom) stability results when varying the number of samples and the degree of redundancy (ρ)

Figure 7 depicts the Spearman and Kendall stability metrics for the ranker filter methods (Chi-Squared, ReliefF, Information Gain and mRMR). We can see two different behaviors depending on if the methods are univariate (Chi-Squared and Information Gain) or multivariate (ReliefF and mRMR). Univariate methods do not take into account interaction between features, so they are not capable of detecting redundancy. This is the reason why their stability is not affected by the increasing degree of redundancy. They are, in fact, selecting always the 50 fist features over the irrelevant features, but in different order over the repetitions of the experiment. On the contrary, multivariate methods can detect redundancy and discard redundant features. Therefore, they do not rank all the 50 relevant features over the irrelevant ones. The problem is that they chose as "relevant" different subsets of features at each repetition of the experiment, and so they become unstable as the degree of redundancy increases. The mRMR method becomes more stable when increasing the sample size, because the more samples it has to estimate the mutual information, the more reliable and stable it will be. However, increasing the sample size does not have the same effect on the stability of ReliefF. ReliefF decides the relevance score for each feature

based on aggregating the scores over all instances in a training set, so it is surprising that having more data would produce less stable results.



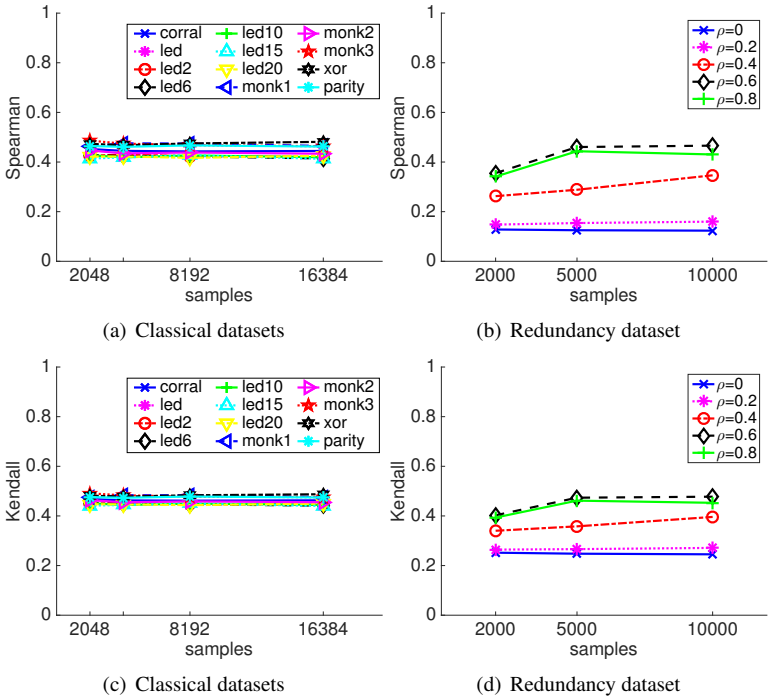
(a) 2000 samples (b) 5000 samples (c) 10000 samples

(d) 2000 samples (e) 5000 samples (f) 10000 samples

Fig. 7 Spearman (top) and Kendall (bottom) stability results when varying the number of samples and the degree of redundancy (ρ)

Trying to shed light on this issue, we compare the behavior of ReliefF on this synthetic dataset with different levels of overlap with the behavior of ReliefF on the classical artificial datasets used in the rest of the paper. For the sake of comparison, in the case of the classical datasets, we are only showing the results when the number of samples ranges from 2048 to 16384, and the number of features is fixed to 128. For an easier visualization, we are also displaying how stability changes for ReliefF when trying different levels of redundancy in the dataset studied in this case study. Figure 8 presents this comparison, in which the top graphs are related to Spearman measure and bottom graphs are related to Kendall measure.

As can be seen, on the classical artificial datasets, increasing the level of samples does not have a big impact on the stability of ReliefF, however, if we focus on the right part of the figure, we can see a different behavior. When using the redundancy dataset, we can easily see that for low levels of redundancy, the stability is sort of maintained. However, when the level of redundancy increases, ReliefF becomes more unstable as the number of samples increases. This finding suggest that when ReliefF tries to find the relevant features in an scenario with redundancy, having more samples makes it more difficult to give the same relevance to the features. This might be caused by the fact that ReliefF works by selecting randomly a sample and finding the closest same-class instances and the closets different-class instances. If the level of redundancy among the relevant features is high enough, it might be possible that more ties between closest instances would happen, which can increase when having more instances to check. In this scenario, ReliefF would become more unstable as the



(a) Classical datasets (b) Redundancy dataset

(c) Classical datasets (d) Redundancy dataset

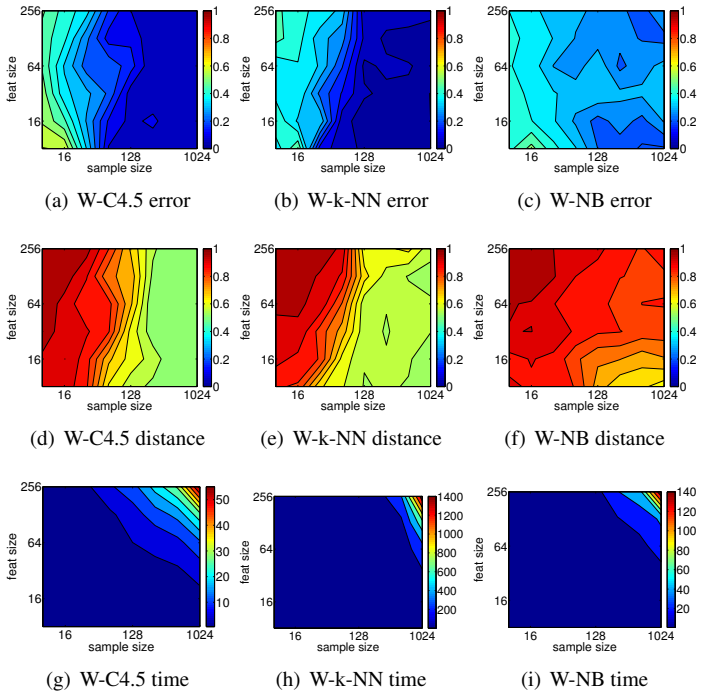
Fig. 8 Spearman (top) and Kendall (bottom) stability results for ReliefF with different datasets when increasing the number of samples

number of samples increases. An interesting line of future research might be to study in depth the behavior shown by ReliefF in the presence of redundant features.

5.2 Scalability of wrappers

This section reveals the scalability of wrapper methods. For this sake, we have chosen three representative classifiers (C4.5, k-NN and naive Bayes) which will be used to assess the relative usefulness of the subsets of variables. Notice that the search strategy is *best first*, starting with the empty set of features and searching forward (which tends to select larger subsets of features than the backward search strategy). Figure 9 displays the results achieved by the wrapper combined with these three classifiers applied on the Corral dataset. It can be seen that, in terms of error, C4.5 and k-NN reported an acceptable behavior, although it seems that they are more affected by the sample dimension and the minimum error is achieved with around 128 samples. The minimum error obtained by the wrapper using naive-Bayes is slightly higher, but this method appears to be more stable. A similar behavior was also shown on the remaining datasets (see Table 9). The wrapper combined with NB shows the poorest results in terms of error due to two reasons: (a) it cannot solve non-linear problems, such as Parity or XOR; and (b) since this classifier is robust to irrelevant features, the wrap-

per search strategy does not need to remove all the irrelevant features because NB can obtain acceptable classification accuracies even so, but the measures that we are using in this work get penalized if irrelevant features are included.



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

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

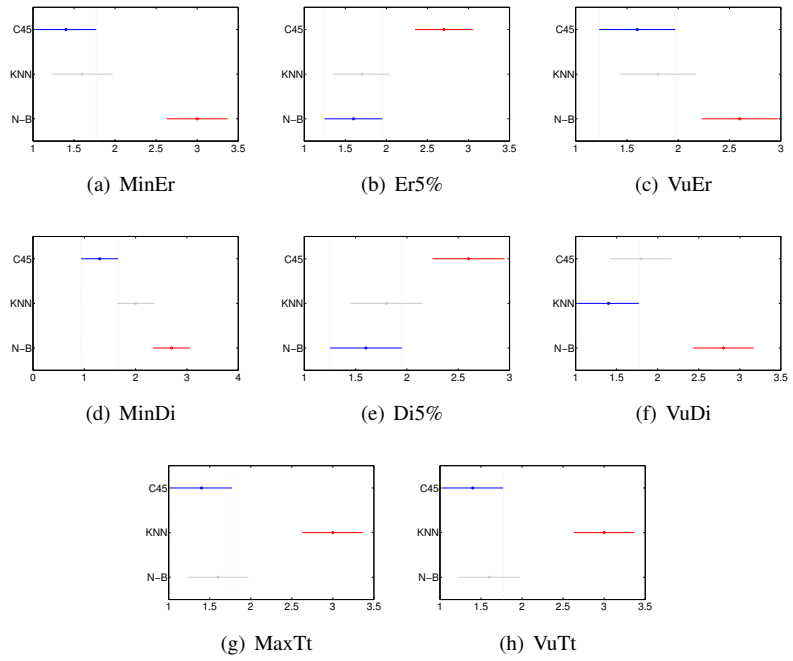
(g) W-C4.5 time (h) W-k-NN time (i) W-NB time

Fig. 9 Measures of scalability of wrappers in the Corral dataset, showing feature size vs. sample size

Table 9 Precision, stability and time measures for wrappers on classical datasets

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

In terms of distance, a similar behavior to that of the error is reported in Figure 9. The wrapper combined with C4.5 and k-NN obtained the minimum distance using around 128 samples, whereas naive Bayes, although being more stable, achieved a higher minimum distance. Focusing on Table 9, it is easy to note that this circumstance also happened with the rest of the datasets. The most different results were found on the training time. According to Table 9, the wrapper using C4.5 employed the lowest time on datasets Corral, Led and Monk3. When combining the wrapper with naive Bayes, the lowest values were achieved on Parity and XOR, whereas the k-NN classifier caused the wrapper to consume more training time (around 139 times higher when comparing the MaxTt value on Parity dataset with those of the k-NN and naive Bayes learning methods).



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

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

(g) MaxTt (h) VuTt

Fig. 10 Comparison of scalability measures for wrappers (combined with C4.5, k-NN and naive Bayes)

Figure 10 shows the results of the statistical tests executed to compare the different wrapper methods based on their scalability properties on all datasets. The use of C4.5 and kNN as learning algorithms obtained significantly better results in terms of error than the use of naive Bayes. With regard to the measures related to the distance, using kNN seems a good option since it achieves the best results or it is not significantly different from the best result for the three scalar measures (MinDi, Di5% and VuDi). It is interesting to see, from graphs 10(g) and 10(e), that using C4.5 allows the wrapper to achieve the lowest distance but it requires a significantly larger amount of

data than the other two methods to do it. Finally, in terms of training time, C4.5 and naive Bayes required significantly less time than k-NN.

In light of the above, the best learning algorithm to be combined with the wrapper seems to be C4.5. Although k-NN shows a good behavior in terms of error and distance, the training time is quite high, probably because of the time burden required to sort the samples in order to find the nearest neighbors. C4.5 achieves a lower error and distance in a shorter time. Notice that studying the scalability of wrapper methods is not as straightforward as with filters, since the former involve learning algorithms, which add more complexity to the task. Moreover, wrappers select features based on the classification performance, so it might happen that some result that we consider poor according to the measures proposed in Section 4.3, turn out to obtain outstanding classification results. Having said that, Table 10 provides some guidelines for the specific scalability aspects considered for wrapper methods based only on the experiments carried out here. Notice that this comparison is only among the wrapper methods, not including the other types of methods tested in these experiments. Again, the larger the number of bullets, the better the behavior.

Table 10 Overview of the behavior regarding scalability of wrappers (notice that the large the number of dots, the better the behavior).

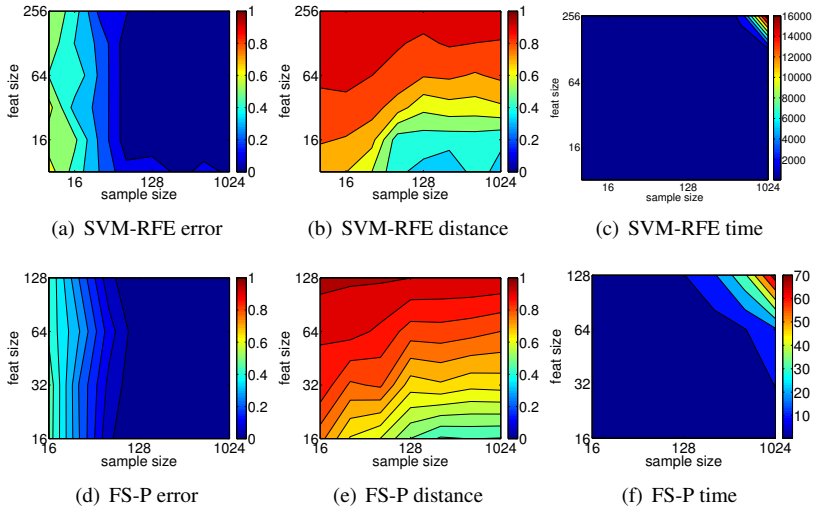
Method	Error	Distance	Training time
W-C45	●●●●	●●	●●●●●
W-k-NN	●●●●	●●●	●●●
W-NB	●●●	●●	●●●●●

5.3 Scalability of embedded methods

Finally, this section studies the scalability of two embedded methods: FS-P and SVM-RFE. Figure 11 shows the results obtained with Corral dataset. As can be seen, the maximum training time required by SVM-RFE is almost 18000 seconds. This high time is due to the recursive nature of the method thus preventing its application to the remaining datasets, that are more complex than Corral.

In terms of error (Figure 11 and Table 11), the embedded methods seem to be more affected by the number of samples than of features. Both of them achieved the minimum error with around 128 samples, but FS-P performed better. Regarding the distance, their performance is comparable to those of the multivariate ranker filters ReliefF and mRMR, being more affected by the number of features than of samples, due to the possible combinations of features.

Focusing on the training time, SVM-RFE requires a much longer time than FS-P. For this reason, FS-P seems to be a better option since in the rest of the measures the performance is similar. However, only one dataset is not enough to draw strong conclusions. For this reason, FS-P was applied to the remaining datasets. Tables 12 and 13 report the eight scalar measures defined in section 4.3 after applying the FS-P



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

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

Fig. 11 Measures of scalability of embedded methods in the Corral dataset, showing feature size vs. sample size

Table 11 Precision, stability and time measures for embedded methods on Corral dataset

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

embedded method to classical and SD datasets, respectively. In terms of error, FS-P obtains the best results with Corral and Led datasets while it shows a poor performance with non-linear datasets, such as XOR or the Monk problems. This result may be caused because FS-P uses a linear perceptron so it can only solve linear problems. Regarding stability, FS-P obtains poor results, except for the case of Led dataset, but at the expense of needing a larger amount of data. Finally, the highest training times were required by the SD datasets, since they have the largest number of features. Among the classical datasets (Table 12), it is surprising the maximum training time obtained with Led dataset, which is almost four times that of the remaining datasets. It can be due to the fact that Led dataset is the only multiclass dataset, which complicates the learning process.

Table 12 Precision, stability and time measures for embedded method FS-P on classical datasets

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

Table 13 Precision, stability and time measures for embedded method FS-P on SD datasets

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

In this case, we have not provide a summary table with dots, since the comparative would not be complete, as SVM-RFE was only applied to Corral dataset.

6 Real datasets

In order to check if the behavior shown by the different feature selection methods can be extrapolated to the real world, two real datasets were chosen. These datasets present different properties for the analysis to be performed, whereas one has a large number of features, the other one has a large number of samples. The first one, Colon Cancer[2], is a microarray binary dataset with 2000 features and 62 samples, very similar to the SD family of datasets introduced in Section 3, which consists of detecting if a patient has colon cancer or not. The second dataset, called KDD Cup 99[3], is a benchmark dataset in the intrusion detection field. It contains 494 021 samples represented by employing 41 features, in which the task is to distinguish between an attack and a normal connection.

When it comes to real datasets, the desired output is not known a priori, so instead of using the error measures defined in Section 4, we had to use the classification error. For this sake, we have opted for calculating this error with a SVM classifier and a 5-fold cross validation.

For KDD Cup 99 dataset, we have performed experiments in which the samples range from 2^3 to 2^{14}, and the number of features between 2^3 and 2^4, using also the available 41 of the original dataset. Analogously, for Colon dataset, the number of samples tested is 3^2, 3^3 and the maximum (62) while the number of features ranges from 2^6 to 2^{10} and includes the maximum (2000).

6.1 Scalability of filters

Figures 12 and 13 show the level graphs for ranker and subset methods, respectively. In general, the errors are quite low, especially when having a sufficient number of features and samples. Notice, however, that this might be also a merit of the classifier used to compute the error, since in this case we are not aware of the relevant features a priori.

In terms of distance, it might seem surprising to see the plots concerning distance for ReliefF and mRMR. In previous experiments (Section 5.1), these plots showed a

[2] Colon Cancer dataset is available on `http://datam.i2r.a-star.edu.sg/datasets/krbd`

[3] KDD Cup 99 dataset is available on `http://kdd.ics.uci.edu/kddcup99/kddcup99.html`

"horizontal" pattern, i.e. they seem to degrade when the number of features increased, as a result of evaluating more combinations of features. However, in this case the pattern is "vertical" and the results improve as the number of samples increases. Notice that in this case the number of samples is significantly smaller than in the previous experiment, so it is expected that when having such a small set of samples for such a large number of features to evaluate in the presence of others, random results occur, leading to extremely different rankings. Moreover, in these real experiments we cannot ensure that the relevant features are always present in the different configurations of samples and features. This erratic behavior is also shown by subset filters, as can be seen in Figure 13. Since ChiSquare and Information Gain do not take interaction between features into account, more stable results are obtained.

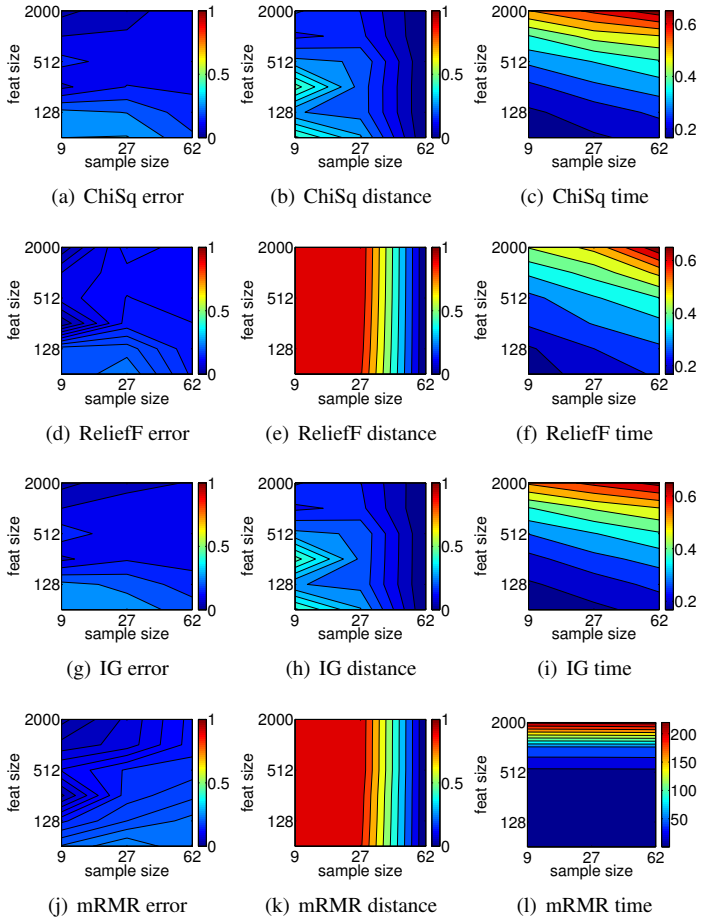
With regard to the training time, among the ranker methods mRMR was the one which required the highest one, as it is quadratic to the number of features. As for the subset methods, and as happened in the previous experiments, FCBF is the approach that requires the lowest running time.

Figures 14 and 15 are dedicated to the scalability analysis of ranker and subset methods, respectively, on the KDD Cup 99 dataset. Like it happened with Colon dataset, the results in terms of error are quite satisfactory, although it has to be reminded that this might be also a merit of the SVM classifier. With regard to the distance, for both ranker and subset methods the results are very irregular. This is caused by the fact that with real datasets, the relevant features are not known a priori so we cannot ensure that they are included in the set of features and samples to test. Notice that, for each execution of n samples and m features, these are selected randomly from the whole dataset, and this is repeated 10 times to evaluate the stability, so the feature selection methods have to deal with very different sets of data. However, when dealing with artificial datasets, we fixed the relevant features so they are included in every execution, and they are the irrelevant features those that are samples randomly. In this latter case, it is easier for a feature selection method to obtain stable selections.

Regarding the training time, it is noticeable that the highest one was achieved by ReliefF, as expected, since it degrades quadratically when the number of samples increases, and the KDD Cup 99 dataset has more samples than features. Tables 14 and 15 depict the numerical results from these experiments, supporting the analysis performed above.

Table 14 Precision, stability and time measures for ranker filters on real datasets

filter-dataset	MinEr	Er5%	VuEr	MinDi	Di5%	VuDi	MaxTt	VuTt
chisquared-colon	0.0333	18000	1.5942	0.0000	7936	1.5940	0.6810	3.4860
chisquared-kddcup99	0.0104	131072	1.2498	0.0000	131072	12.9097	2.3232	11.3446
relieff-colon	0.0444	2304	1.4508	0.0000	7936	6.7189	0.6926	3.4151
relieff-kddcup99	0.0084	262144	1.0907	0.1071	8192	21.6267	172.3317	252.5261
infogain-colon	0.0333	18000	1.5815	0.0000	7936	1.6029	0.6678	3.4680
infogain-kddcup99	0.0104	131072	1.2126	0.0000	16384	13.0735	2.3172	11.3569
mrmr-colon	0.0333	18000	1.5090	0.0000	7936	6.7274	229.9029	450.1347
mrmr-kddcup99	0.0308	1312	1.5807	0.1500	65536	17.8095	0.4451	2.0204



(a) ChiSq error (b) ChiSq distance (c) ChiSq time

(d) ReliefF error (e) ReliefF distance (f) ReliefF time

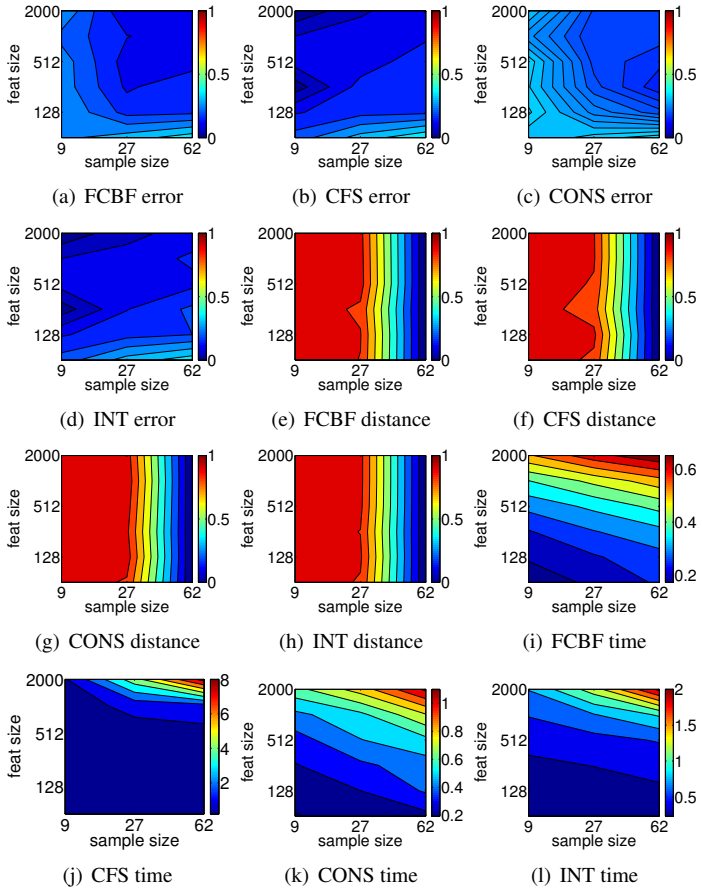
(g) IG error (h) IG distance (i) IG time

(j) mRMR error (k) mRMR distance (l) mRMR time

Fig. 12 Measures of scalability of ranker selection methods in the Colon dataset

Table 15 Precision, stability and time measures for subset filters on real datasets

filter-dataset	MinEr	Er5%	VuEr	MinDi	Di5%	VuDi	MaxTt	VuTt
fcbf-colon	0.1000	54000	2.0955	0.0000	3968	6.6089	0.6915	3.6158
fcbf-kddcup99	0.0449	32768	2.9742	0.1333	32768	15.3748	1.0354	10.8304
cfs-colon	0.0111	18000	1.6019	0.0000	3968	6.4831	8.4432	9.5253
cfs-kddcup99	0.0250	512	2.4682	0.0000	262144	13.8616	2.3600	11.8342
cons-colon	0.1452	15872	2.4580	0.0000	3968	6.7341	1.1232	4.6749
cons-kddcup99	0.0232	262144	2.4855	0.0400	131072	17.0028	4.3950	15.6335
int-colon	0.0111	18000	1.5792	0.0000	3968	6.5561	2.0580	5.8275
int-kddcup99	0.0250	512	1.8250	0.0933	131072	12.9825	4.2328	16.6535



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

(d) INT error (e) FCBF distance (f) CFS distance

(g) CONS distance (h) INT distance (i) FCBF time

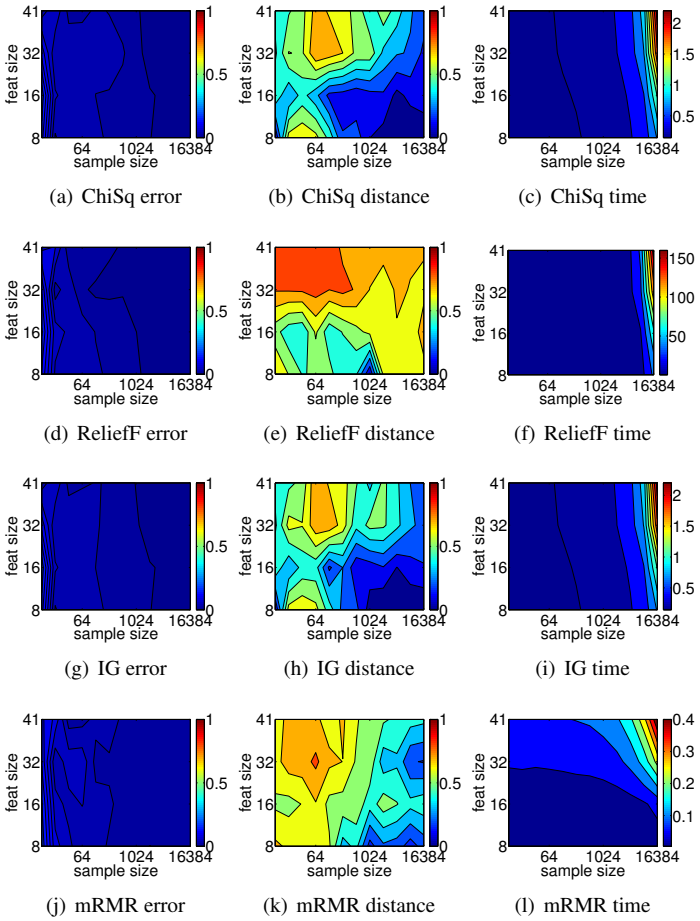
(j) CFS time (k) CONS time (l) INT time

Fig. 13 Measures of scalability of subset methods in the Colon dataset

6.2 Scalability of wrappers

In this subsection we present experiments with wrapper methods. Since the characteristics of microarray data prevent the use of wrappers because they tend to overfit due to the extremely high number of features versus a extremely small number of samples, the Colon dataset was not included. Even so, for KDD Cup 99 the maximum number of samples to test was downgraded to 1024. Figure 16 plots the level graphs whereas Table 16 reports the numerical details. As happened with filters, the errors are also low, which might raise the question of if it is worth using wrappers –given their computational burden– instead of filters –which are a less expensive approach–.

Regarding the distances obtained by the wrappers, it seems that they are highly unstable, for the same reason explained in the previous subsection. Finally, the results in terms of training time confirm the trend seen with artificial datasets, with k-NN requiring the highest time. Notice that, for the training time, the results are not bounded



(a) ChiSq error (b) ChiSq distance (c) ChiSq time

(d) ReliefF error (e) ReliefF distance (f) ReliefF time

(g) IG error (h) IG distance (i) IG time

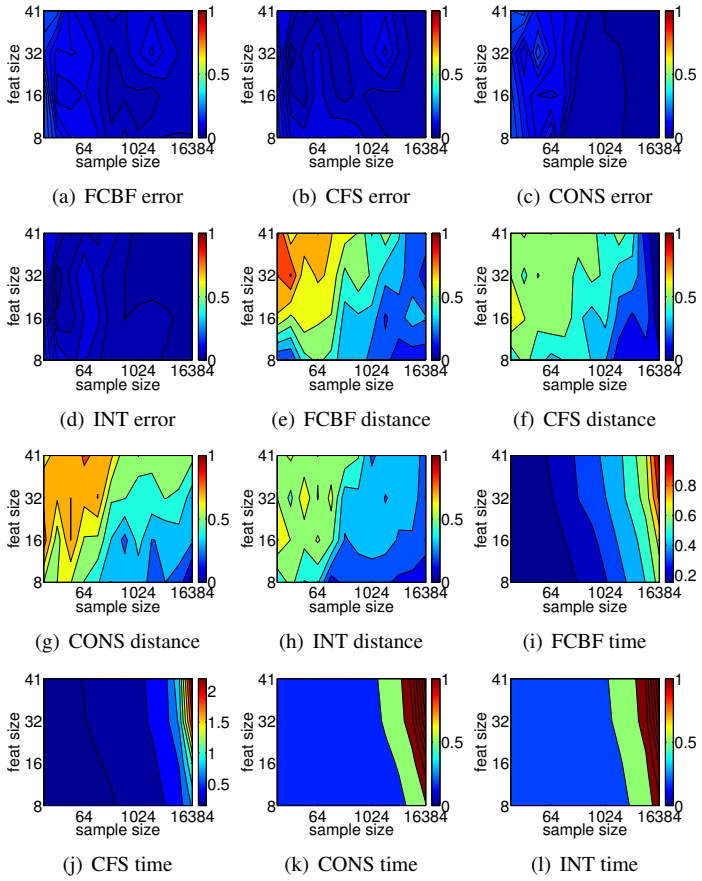
(j) mRMR error (k) mRMR distance (l) mRMR time

Fig. 14 Measures of scalability of ranker selection methods in the KDD Cup 99 dataset

between 0 and 1, since the time has not been normalized, so the maximum values of the axes are different from one method to another.

Table 16 Precision, stability and time measures for wrapper methods on KDD Cup 99 dataset

method-dataset	MinEr	Er5%	VuEr	MinDi	Di5%	VuDi	MaxTt	VuTt
c45-kddcup99	0.0153	41984	1.0715	0.7261	8192	10.1141	16.0380	31.5529
knn-kddcup99	0.0336	32768	0.9415	0.7113	16384	10.3375	185.0736	229.5728
n-b-kddcup99	0.0153	41984	0.5895	0.6943	512	9.9817	14.9804	27.9467



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

(d) INT error (e) FCBF distance (f) CFS distance

(g) CONS distance (h) INT distance (i) FCBF time

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

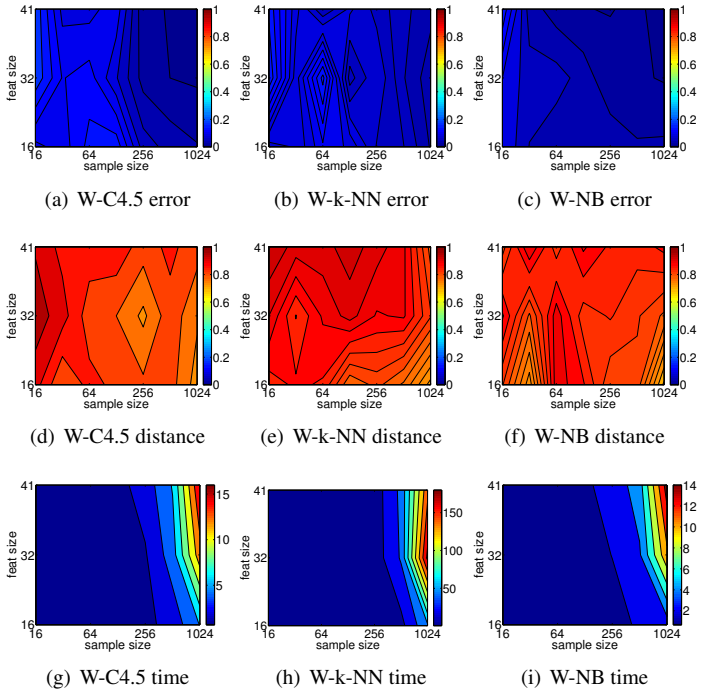
Fig. 15 Measures of scalability of subset methods in the KDD Cup 99 dataset

6.3 Scalability of embedded methods

Figure 17 and Table 17 show the scalability results for KDD Cup 99 since, as happened with wrappers, the computational burden of embedded methods prevent their use on Colon dataset, which has 2000 features. Both methods (SVM-RFE and FS-P) achieve very low errors (under 1%) and are quite unstable, although it has to be noted that FS-P is more affected by the number of features. The running time is surprisingly low, although it has to be reminded that the maximum number of features is 41.

Table 17 Precision, stability and time measures for embedded methods on KDD Cup 99 dataset

method-dataset	MinEr	Er5%	VuEr	MinDi	Di5%	VuDi	MaxTt	VuTt
fs-kddcup99	0.0094	1312	0.3426	0.4394	256	8.7551	12.5350	29.9188
svm-kddcup99	0.0063	512	0.2650	0.3307	256	8.2131	1.4487	8.7761

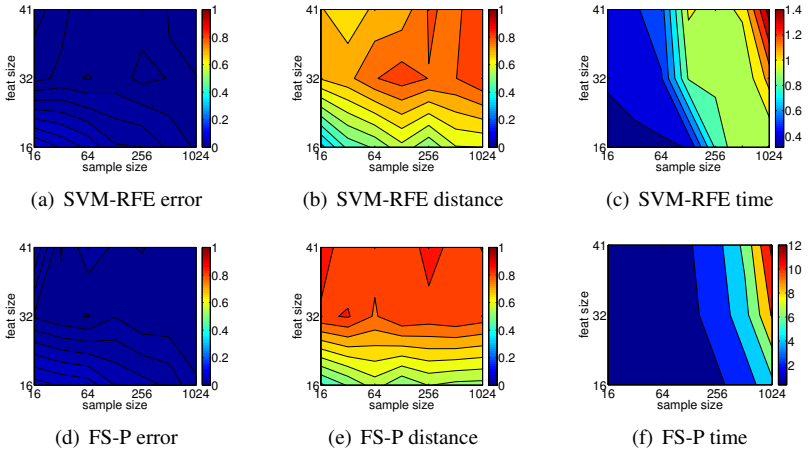


(a) W-C4.5 error (b) W-k-NN error (c) W-NB error

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

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

Fig. 16 Measures of scalability of wrappers in the KDD Cup 99 dataset, showing feature size vs. sample size



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

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

Fig. 17 Measures of scalability of embedded methods in the KDD Cup 99 dataset, showing feature size vs. sample size

6.4 Analysis of results and another approach

In general, it is difficult to know if the conclusions drawn with artificial datasets can be extrapolated to real datasets. The first obstacle that we found is that in real datasets

it is impossible to know the relevant features a priori, therefore the measures proposed in this work to evaluate the efficiency of the feature selection methods in that sense are useless. With real datasets, we have opted for using the classification error, although in this way the goodness of the specific classifier also plays an important role. The minimum error for KDD Cup 99 was achieved by the embedded method SVM-RFE, although ReliefF also obtained a remarkable result. As for Colon dataset, both CFS and INTERACT obtained the lowest error rates. Concerning distance, the comparative with the artificial experiments is not fair either, since in that scenario the relevant features were always present so it was easier to produce stable selections. Lastly, in terms of time, especially when using filters, the different methods showed the same trends as with artificial datasets. ReliefF and mRMR were again the two filters than required the highest training times; the former when the number of instances is high (KDD Cup 99) and the latter when the number of features is high (Colon).

An alternative to using the classification error would be to apply each feature selection method on the whole dataset and deem the selected features as relevant. In this way, it is possible to employ the same error measures used in the rest of the paper, which were defined in Section 4. As an example, in Figure 18 we show the scalability results or ranker methods in the Colon dataset. If we compare these results with those obtained when using the classification error (see Figure 12), the main differences are related to the error, as expected. When using the classification error (Figure 12), it possible that a given classifier can disregard the presence of irrelevant features, leading to low errors. However, when using the measures defined in Section 4 (Figure 18), the inclusion of an irrelevant feature on the top of the ranking is highly penalized.

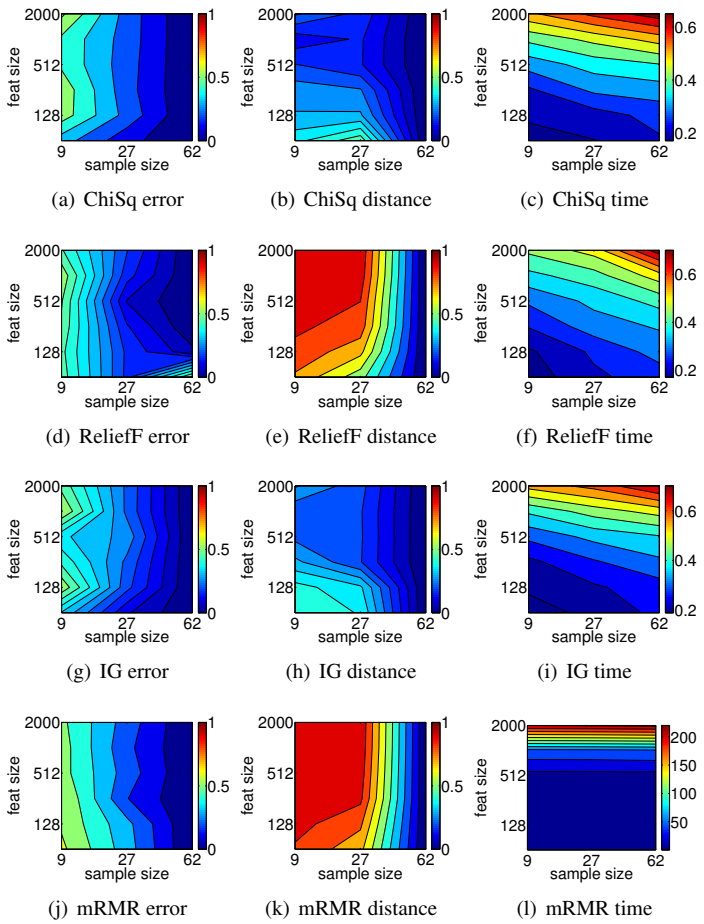
In terms of distance, the results with this new approach (Figure 18) are slightly better than the previous one (Figure 12). This is happening because with the new approach, and as happened with the artificial datasets, the relevant features are fixed and present in all subsets of features and samples, making easier to achieve stabler selections of features. Regarding the time, the results are practically identical. For the sake of brevity, we are not including the rest of the graphs in this paper, but they can be accessed online[4].

7 Discussion and conclusions

With the advent of high dimensional datasets in machine learning, feature selection has received an important amount of attention from researchers and data-mining practitioners due to its capacity to improve both performance and computational efficiencies. In this scenario, scalability is becoming a very important trending issue. An algorithm is said to be scalable if it is suitable, efficient and practical when applied to large datasets. However, the current state is that the issue of scalability is far from being solved although it is present in a diverse set of problems.

In this paper, we present an analysis of the scalability of feature selection methods, which has not received much consideration in the literature. The scenario chosen

[4] http://www.lidiagroup.org/index.php/en/materials-en.html



(a) ChiSq error (b) ChiSq distance (c) ChiSq time

(d) ReliefF error (e) ReliefF distance (f) ReliefF time

(g) IG error (h) IG distance (i) IG time

(j) mRMR error (k) mRMR distance (l) mRMR time

Fig. 18 Measures of scalability of ranker selection methods in the Colon dataset with an alternative approach

to check the scalability of classical feature selection methods was a set of 11 artificial datasets, so as to be able to assess the degree of closeness to the optimal solution in a confident way, independently of the classifier. In our opinion, this type of analysis has to be done on controlled scenarios, with a manageable size, to be able to give the reader an idea about how scalable the classical methods are before applying them to an extremely high-dimensional problem. Therefore, if a given method is not scalable even on this controlled scenario, it would not be worthy applying it to a big real problem. On the contrary, if the performance of such method is promising, it would be a good candidate to be applied in a Big Data scenario.

A total of eight well-known filter-based feature selection algorithms were evaluated, covering both ranking and subset methods. Moreover, some representatives of embedded and wrapper methods were also considered in this study. For determining the scalability of the methods, several new measures were proposed, based not

only on the accuracy but also on the execution time and stability, and their adequacy was demonstrated. In light of the experimental results, some guidelines have been proposed:

- Among the subset filters, INTERACT obtained good scalability results, especially in terms of minimum error and distance. However, if we are interested in really low training times, FCBF showed an acceptable accurate results in a short training time.
- As for the ranker methods, ReliefF turned out to be very precise in selecting the relevant features, although this comes at the prize of large training times when the number of samples is high. On the other hand, the ranker method mRMR also achieved good results in terms of error, but the training time raises significantly when it deals with large amounts of features.
- With regard to the stability of the filters evaluated, i.e. the sensitivity of the methods to variations in the training set, univariate ranker methods (such as Chi-Squared or Information Gain) are more stable than multivariate methods (such as ReliefF or mRMR), since the latter have to deal with interactions between features. It is worth mentioning that subset methods, although being also multivariate, are much more stable than their counterparts within the ranker approach. This is happening because ranker methods have to order all the features, even the irrelevant ones, while subset methods only select the relevant features so if they are behaving correctly, it is easier to select consistent subsets of features.
- As expected, the scalability of the wrapper methods depends on the classifier chosen, being C4.5 a good option in terms of scalability.
- When using embedded methods, the well-known SVM-RFE algorithm achieves promising results in terms of error at the cost of requiring high training times.
- In general, the authors suggest the use of filters, since they carry out the feature selection process with independence of the induction algorithm and are faster than embedded and wrapper methods, scaling better to Big Data problems.

Finally, the feature selection methods were also tested over two real datasets, trying to check if the conclusions drawn on artificial data could be extrapolated. The obtained results were difficult to compare, since the set of relevant features was not known a priori. In any case, some behaviors were again shown by the tested methods. For example, univariate rankers (Information Gain and Chi-Squared) were again more stable than multivariate rankers (ReliefF and mRMR) and, in terms of time, ReliefF required the highest training time when dealing with a dataset with a large number of features while mRMR did the same when dealing with large amounts of features.

In light of these results, it becomes clear that state-of-the-art feature selection methods will have scalability problems when approaching Big Data. In general, one can say that most of the classical feature selection approaches that are univariate -that is each feature is considered separately- have an important advantage in scalability, but at the cost of ignoring feature dependencies, and thus leading to lower performances than other feature selection techniques. To improve performance, multivariate filter techniques are proposed, but at the cost of requiring high training times. These multivariate methods are based on computing pairwise correlations in their al-

gorithms designs. Imagine the implications of this when dealing with a million of features; it would be necessary to cope with a trillion of correlations. Note that this poses an enormous challenge for machine learning researchers that has not been addressed yet. The need for scalable yet efficient methods is obvious, since as demonstrated in this work, existing feature selection methods will be inadequate to cope with this unprecedented number of features, either in terms of training time or efficiency in selecting the relevant features.

8 Acknowledgments

This research has been economically supported in part by the Ministerio de Economía y Competitividad of the Spanish Government (research project TIN2015-65069-C2-1-R), by European Union FEDER funds and by the Consellería de Industria of the Xunta de Galicia (research project GRC2014/035). Financial support from the Xunta de Galicia (Centro singular de investigación de Galicia accreditation 2016-2019) and the European Union (European Regional Development Fund - ERDF), is gratefully acknowledged (research project ED431G/01).

References

1. A. Ahmed and E. P. Xing. Scalable dynamic nonparametric bayesian models of contents and users. In *In International Joint Conference on Artificial Intelligence, IJCAI*, pages 3111–3116, 2013.
2. A. Alonso-Betanzos, V. Bolón-Canedo, D. Fernández-Francos, I. Porto-Díaz, and N. Sánchez-Maroño. *Efficiency and Scalability methods for computational intellect*, chapter Up-to-Date feature selection methods for scalable and efficient machine learning, pages 1–26. IGI Global, 2013.
3. V. Bolon-Canedo, N. Sanchez-Marono, and A. Alonso-Betanzos. On the effectiveness of discretization on gene selection of microarray data. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 3167–3174. IEEE, 2010.
4. V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos. Feature selection and classification in multiple class datasets: An application to kdd cup 99 dataset. *Expert Systems with Applications*, 38(5):5947–5957, 2011.
5. V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos. A review of feature selection methods on synthetic data. *Knowledge and information systems*, 34(3):483–519, 2013.
6. L. Bottou and O. Bousquet. The tradeoffs of large-scale learning. *Optimization for Machine Learning*, page 351, 2011.
7. L. Breinman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and regression trees. *Wadsworth and Brooks-Cole Advanced Books and Software, Pacific Grove, California, USA*, 1984.
8. G. Brown, A. Pocock, M.-J. Zhao, and M. Luján. Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. *The Journal of Machine Learning Research*, 13:27–66, 2012.
9. M. Dash and H. Liu. Consistency-based search in feature selection. *Artificial Intelligence*, 151(1-2):155–176, 2003.
10. R. Díaz-Uriarte and S. A. De Andres. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):3, 2006.
11. A. Fahad, Z. Tari, I. Khalil, I. Habib, and H. Alnuweiri. Toward an efficient and scalable feature selection approach for internet traffic classification. *Computer Networks*, 57:2040–2057, 2013.
12. M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.
13. G. Gulgezen, Z. Cataltepe, and L. Yu. Stable and accurate feature selection. In *Machine Learning and Knowledge Discovery in Databases*, pages 455–468. Springer, 2009.
14. I. Guyon. *Feature extraction: foundations and applications*, volume 207. Springer, 2006.

15. I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.
16. I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.
17. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
18. M. A. Hall and L. A. Smith. Practical feature subset selection for machine learning. 1998.
19. S. C. Hoi, J. Wang, P. Zhao, and R. Jin. Online feature selection for mining big data. In *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, pages 93–100. ACM, 2012.
20. G. Hughes. On the mean accuracy of statistical pattern recognizers. *Information Theory, IEEE Transactions on*, 14(1):55–63, 1968.
21. G. H. John, R. Kohavi, K. Pfleger, et al. Irrelevant features and the subset selection problem. In *ICML*, volume 94, pages 121–129, 1994.
22. M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.
23. K. Kira and L. A. Rendell. A practical approach to feature selection. In *Proceedings of the ninth international workshop on Machine learning*, pages 249–256. Morgan Kaufmann Publishers Inc., 1992.
24. D. Koller and M. Sahami. Toward optimal feature selection. In *In 13th International Conference on Machine Learning*, pages 284–292, 1995.
25. I. Kononenko. Estimating attributes: analysis and extensions of relief. In *Machine Learning: ECML-94*, pages 171–182. Springer, 1994.
26. R. Kumar and S. Vassilvitskii. Generalized distances between rankings. In *Proceedings of the 19th international conference on World wide web*, pages 571–580. ACM, 2010.
27. H. Liu and R. Setiono. Chi2: Feature selection and discretization of numeric attributes. In *Tools with Artificial Intelligence, 1995. Proceedings., Seventh International Conference on*, pages 388–391. IEEE, 1995.
28. H. Liu and R. Setiono. A probabilistic approach to feature selection - a filter solution. In *In Proceedings of the 13th International Conference on Machine Learning*, pages 319–327. Morgan Kaufmann, 1996.
29. H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):491–502, 2005.
30. D. Luo, F. Wang, J. Sun, M. Markatou, J. Hu, and S. Ebadollahi. Sor: Scalable orthogonal regression for non-redundant feature selection and its healthcare applications. In *SIAM Data Mining Conference*, pages 576–587, 2012.
31. M. Mejía-Lavalle, E. Sucar, and G. Arroyo. Feature selection with a perceptron neural net. In *Proceedings of the international workshop on feature selection for data mining*, pages 131–135, 2006.
32. P. Nemenyi. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, 1963.
33. S. Nogueira and G. Brown. Measuring the stability of feature selection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 442–457. Springer, 2016.
34. H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238, 2005.
35. D. Peteiro-Barral, V. Bolon-Canedo, A. Alonso-Betanzos, B. Guijarro-Berdinas, and N. Sanchez-Marono. Scalability analysis of filter-based methods for feature selection. *Advances in Smart Systems Research*, 2(1):21–26, 2012.
36. D. Peteiro-Barral, V. Bolón-Canedo, A. Alonso-Betanzos, B. Guijarro-Berdiñas, and N. Sánchez-Maroño. Toward the scalability of neural networks through feature selection. *Expert Systems with Applications*, 4(8):2807–2816, 2012.
37. D. Peteiro-Barral and B. Guijarro-Berdiñas. A study on the scalability of artificial neural networks training algorithms using multiple-criteria decision-making methods. In *Artificial Intelligence and Soft Computing*, volume 7894 of *Lecture Notes in Computer Science*, pages 162–173. Springer, 2013.
38. J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
39. V. C. Raykar, R. Duraiswami, and L. H. Zhao. Fast computation of kernel estimators. *Journal of Computational and Graphical Statistics*, 19(1):205–220, 2010.
40. L. Rokach, A. Schclar, and E. Itach. Ensemble methods for multi-label classification. *arXiv preprint arXiv:1307.1769*, 2013.

41. S. Sonnenburg, V. Franc, E. Yom-Tov, and M. Sebag. Pascal large scale learning challenge. In *25th International Conference on Machine Learning (ICML2008) Workshop. J. Mach. Learn. Res*, volume 10, pages 1937–1953, 2008.
42. C. Spearman. The proof and measurement of association between two things. *The American journal of psychology*, 15(1):72–101, 1904.
43. Y. Sun, S. Todorovic, and S. Goodison. A feature selection algorithm capable of handling extremely large data dimensionality. In *In Proceedings of the 2008 SIAM Internactional Conference in Data Mining*, pages 530–540, 2008.
44. S. B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Dzeroski, S. E. Fahlman, D. Fisher, et al. The monk's problems a performance comparison of different learning algorithms. 1991.
45. G. Tsoumakas, I. Katakis, and I. Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2010.
46. L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *The Journal of Machine Learning Research*, 5:1205–1224, 2004.
47. L. Yu and H. Liu. Efficient feature selection via analysis of relevance and redundancy. *J. Mach. Learn. Res.*, 5:1205–1224, 2004.
48. M. YUI and I. KOJIMA. A database-hadoop hybrid approach to scalable machine learning. 2013.
49. M.-L. Zhang, J. M. Peña, and V. Robles. Feature selection for multi-label naive bayes classification. *Information Sciences*, 179(19):3218–3229, 2009.
50. Z. Zhao and H. Liu. Searching for interacting features. In *IJCAI*, volume 7, pages 1156–1161, 2007.
51. Z. Zhao, R. Zhang, J. Cox, D. Duling, and W. Sarle. Massively parallel feature selection: an approach based on variance preservation. *Machine Learning*, 92:195–220, 2013.
52. Z. A. Zhao and H. Liu. *Spectral feature selection for data mining*. Chapman & Hall/CRC, 2011.
53. Z. Zhu, Y.-S. Ong, and J. M. Zurada. Identification of full and partial class relevant genes. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 7(2):263–277, 2010.