A simple and efficient method for variable ranking according to their usefulness for learning

José R. Quevedo, Antonio Bahamonde, Oscar Luaces [1]

Artificial Intelligence Center, University of Oviedo at Gijón
E-33204 Gijón (Spain)

Abstract

The selection of a subset of input variables is often based on the previous construction of a ranking to order the variables according to a given criterion of relevancy. The objective is then to linearize the search, estimating the quality of subsets containing the topmost ranked variables. An algorithm devised to rank input variables according to their usefulness in the context of a learning task is presented. This algorithm is the result of a combination of simple and classical techniques, like correlation and orthogonalization, which allow the construction of a fast algorithm that also deals explicitly with redundancy. Additionally, the proposed ranker is endowed with a simple polynomial expansion of the input variables to cope with nonlinear problems. The comparison with some state-of-the-art rankers showed that these combination of simple components is able to yield high-quality rankings of input variables. The experimental validation is made on a wide range of artificial data sets and the quality of the rankings is assessed using a ROC-inspired setting, to avoid biased estimations due to any particular learning algorithm.

1 Introduction

It is acknowledged that variable selection plays a very important role in most (if not all) learning tasks to improve the accuracy and readability of learned models. Moreover, in practical applications, the reduction of dimensionality is directly related to ease the acquisition of data, what may mean important

[1] Corresponding author: Oscar Luaces. Artificial Intelligence Center. Campus de Viesques. E−33204 Gijón (Spain). Tel.: +34 985 18 2028. Fax: +34 985 18 2125. E-mail: oluaces@aic.uniovi.es

economical savings. Other times, variable selection benefits may be more intangible, for instance, associated with obtaining measurements, this selection can even avoid physical risks ran by people or animals, as in the case described in (Bahamonde et al., 2004).

A relaxed version of selection is variable ranking with respect to their relevance as prediction tools. In some applications, selection is probably too rigid, and what we really need is just a ranking of variables. Other times, ranking variables is used as a first step towards selection, as in (Guyon and Elisseeff, 2003; del Coz et al., 2005; Díez et al., 2004; Luaces et al., 2004).

Most of the algorithms devised to rank input variables fall in one of these categories: wrappers or filters. Wrappers rely on a learning algorithm to estimate the quality of the selected subset of variables. The success of kernel methods like SVM (Boser et al., 1992) in the field of machine learning gave rise to the construction of SVM-based variable ranking methods, both for linear (Guyon et al., 2002) and nonlinear models (Rakotomamonjy, 2003; Degroeve et al., 2002). These methods employ an iterative procedure where each step rules out one input variable; after one SVM training, the least useful variable is determined by a mechanism that is, in the nonlinear case, at least, quadratic in the number of support vectors. Therefore, unfortunately, their complexity becomes too high to be utilized with medium-size data sets.

On the other hand, filters are independent of the learning algorithm to be used after the ranking/selection. Ranking filters are commonly based on computing a score for each input variable relating it with the target class, like the correlation coefficient, information theoretic measures, discrimination abilities, etc.; then, variables are sorted according to such score. Although it is assumed that rankings obtained via filters can be suboptimal with respect to a given learning algorithm, they may be preferable to wrappers for computational reasons.

In this paper we present a new algorithm called `SPE-`*ranker*. It is a two-stage filter that first constructs a correlation-based ranking using a simplified polynomial expansion of the original input variables; the expansion is a heuristic approach used to extend the low complexity and good performance of linear rankers to general nonlinear problems. The second stage is an iterative process based on Gram-Schmidt orthogonalization whose aim is to detect and remove redundant variables. In fact, this procedure can be considered a kind of stepwise regression, and has been used to rank variables in (Chen et al., 1989; Stoppiglia et al., 2003; Guyon et al., 2003).

The consideration of redundant entries is important since in real world applications the number of these inputs may be high; in addition to economical reasons, it is known that redundant entries can affect the speed and accuracy of learning algorithms (Yu and Liu, 2004).

We have conducted an exhaustive experimentation on artificial data sets to test how good a simple approach like SPE-*ranker* can perform when compared with state-of-the-art rankers. The results obtained, which are shown in a quite large section at the end of the paper, reveal that the set of features formed by successive powers of each single variable (following (Guyon and Elisseeff, 2003) we call *variables* the raw input variables and *features* those descriptors constructed from the original input variables) are a good approximation to obtain good rankings in nonlinear problems, where a full polynomial expansion would yield intractable data sets. We will show that SPE-*ranker* is a filter whose achievements are equal or better than those reached by more complex state-of-the-art methods.

The paper is organized as follows: the next section briefly describes several state-of-the-art ranking methods, including wrappers and filters; the third section is devoted to describe our filter approach in detail; next, we report a comparison between our approach and other ranking methods based on the experimental results obtained on a large variety of artificially generated data sets; and finally we discuss some experimental results obtained on data sets designed *ad hoc* to reveal some limitations of our approach.

2 Some state-of-the-art rankers

In the following we briefly describe some state-of-the-art methods for variable ranking that belong to one of this categories: *wrappers* or *filters*. Let us notice that most of them are included as part of more complex selection methods but in our comparative study we will only focus on the rankings they produce. The only selection that we will consider is the ability of some algorithms, including SPE-*ranker*, to eliminate redundant variables.

2.1 The wrapper approach

In first place we are going to review some *wrappers* that use a support vector machine (SVM) as the learning algorithm. For ease of reference, let us consider that a given learning task starts with a training set

$$\mathcal{S} = \{(\boldsymbol{x}_{i,\cdot}, y_i) : \ i = 1, \ldots, n\},$$

where $\boldsymbol{x}_{i,\cdot} \in \mathbb{R}^m$ denotes the i-th example described by m input variables (in particular, $x_{i,j}$ is the value of the j-th input variable in the i-th example), and $y_i \in \{+1, -1\}$ is its class. Then, let us recall that a SVM is able to induce a

function of the form

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x}_{i,\cdot}, \boldsymbol{x}) + b, \tag{1}$$

where the predicted output is given by $\text{sign}(f(\boldsymbol{x}))$, K is known as the *kernel function* and it is usually defined as the scalar product between the images of two inputs in a Hilbert space; in symbols, $K(\boldsymbol{v}, \boldsymbol{v'}) = \langle \phi(\boldsymbol{v}), \phi(\boldsymbol{v'}) \rangle$. Therefore,

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle + b, \tag{2}$$

where $\boldsymbol{w} = \sum_{i=1}^{n} \alpha_i y_i \phi(\boldsymbol{x}_{i,\cdot})$ is the vector of weights of the input variables describing the example. In particular, when $\phi(\boldsymbol{v}) = \boldsymbol{v}$, K is named *linear kernel* and $f(\boldsymbol{x})$ is a linear separator.

The first input variable ranker that we are going to describe briefly was proposed by Rakotomamonjy (2003); for the sake of simplicity we will name it in the rest of the paper as `RM` (for **R**akotomamonjy's **M**ethod). Its ranking criterion orders the list of variables according to the influence of the variations of their weights; in fact, it is an extension to the nonlinear case of the widely used RFE (Guyon et al., 2002). This method removes in each iteration the variable with the lowest ranking value, computed as:

$$R_{\texttt{RM}}(i) = |\nabla_i \|\boldsymbol{w}\|^2| = \left| \sum_{k,j} \alpha_k \alpha_j y_k y_j \frac{\partial K(\boldsymbol{s} \cdot \boldsymbol{x}_{k,\cdot}, \boldsymbol{s} \cdot \boldsymbol{x}_{j,\cdot})}{\partial s_i} \right|, \quad i = 1, \ldots, m, \tag{3}$$

where $\boldsymbol{s}$ is a scaling factor used to simplify the computation of partial derivatives of the kernel. Once a variable is removed the ranking is recomputed applying the same method, that is, training again an SVM with the remaining input variables. The order in which variables are removed constitutes the final ranking given by the algorithm.

Degroeve et al. (2002) developed another SVM wrapper that was successfully used for splice site prediction of DNA sequences. We will name their method as `DM`. This method uses a ranking criterion such that variables are ordered with respect to the loss in predictive performance when they are removed. Moreover, the authors approximate the generalization performance when removing the i-th variable by the accuracy on the training set while setting the value of that variable, in every instance, to its mean value. The ranking criterion for this method can be expresed as

$$R_{\texttt{DM}}(i) = \sum_{k} y_k \cdot \sum_{j} \alpha_j y_j K(\boldsymbol{x}^{(i)}_{j,\cdot}, \boldsymbol{x}^{(i)}_{k,\cdot}), \quad i = 1, \ldots, m, \tag{4}$$

where $\boldsymbol{x}^{(i)}_{l,\cdot}$ denotes the vector $\boldsymbol{x}_{l,\cdot}$ where the value for the i-th component was replaced by its mean value computed in the training set. Notice that a higher value of $R_{\texttt{DM}}(i)$, that is, a higher accuracy on the training set when removing

the i-th variable, means a lower relevance of that variable. Therefore, we will remove the variable yielding the highest ranking value, as opposite to the `RM` ranking method described previously.

Another SVM-based ranking method was proposed by Weston et al. (2001). This method obtains the ranking by scaling the input variables by a real valued vector $\boldsymbol{\sigma}$, where larger values of σ_i indicate more useful features. The idea is to find $\boldsymbol{\sigma}$ taking into account that the kernel can be expressed as:

$$K_{\boldsymbol{\sigma}}(\boldsymbol{x}, \boldsymbol{x'}) = K(\boldsymbol{x} * \boldsymbol{\sigma}, \boldsymbol{x'} * \boldsymbol{\sigma}). \tag{5}$$

This can be achieved by minimizing the bound of the expectation of the error probability, which is

$$EP_{err} \leq \frac{1}{n} E\left\{\frac{R^2}{M^2}\right\} = \frac{1}{2} E\{R^2 W^2(\alpha^0)\} \tag{6}$$

if the training data of size n belong to a sphere of size R and are separable with margin M. The authors propose to find $\boldsymbol{\sigma}$ by minimizing bound (6) using gradient descent. In the rest of the paper we will refer to this method as `R2W2`.

2.2 The filter approach

The rest of the methods that we are going to summarize in this section are *filters*. For the sake of simplicity, in the rest of the paper we will denote column vectors with just one subindex, that is, $\boldsymbol{x}_j$ will mean $\boldsymbol{x}_{\cdot,j}$. Additionally, $\boldsymbol{y}$ stands for the target output; that is, the vector whose component y_i is the output of the i-th example.

The first filter that we are going to mention is used in (Stoppiglia et al., 2003) together with a method for selecting subsets of variables based on the ranking position of a random feature. For the purpose of our comparative study we are only going to focus on the ranking mechanism which, in fact, heavily inspired the approach presented in this paper. The ranking procedure proposed by Stoppiglia et al. is based on an embedded redundancy elimination mechanism as follows:

- Select the variable that best explains the target output $\boldsymbol{y}$. Such variable will be the one whose corresponding column vector, $\boldsymbol{x}_i$, in the training data set forms the smallest angle with $\boldsymbol{y}$; therefore, the ranking criterion can be expressed in terms of the square cosine of such angle as

$$R_{\texttt{SM}}(i) = \cos^2(\boldsymbol{x}_i, \boldsymbol{y}) = \frac{\langle \boldsymbol{x}_i, \boldsymbol{y} \rangle^2}{\|\boldsymbol{x}_i\|^2 \|\boldsymbol{y}\|^2}, \quad i = 1, \ldots, m. \tag{7}$$

Therefore, the variable selected will be the one that maximizes $R_{\texttt{SM}}$.

- Using the Gram-Schmidt orthogonalization, project the column vectors of the remaining input variables and the target output onto the space spanned by the column vectors selected up to this step.
- With the projected data repeat the process starting at the first step until an stopping criterion is met. Orthogonalization is used to discard the part of the target concept already explained by previously selected variables, so the variable selected in the next iteration will be the most relevant with respect to what is not yet explained.

The use of Gram-Schmidt orthogonalization can also be found in (Guyon et al., 2003), where the authors propose and compare several feature selection methods. One of the most promising, which we will refer to as $\texttt{GSRelief}_{\texttt{K}}$, coincides with SM, but instead of equation 7, it uses a variant of the ranking criterion of the popular Relief algorithm (Kira and Rendell, 1992) with Gram-Schmidt orthogonalization.

The original Relief criterion proposed by Kira and Rendell is to compute a score for each input variable, depending on how well such variable separates neighboring examples. The idea is to find for each example of the training set the nearest example of the same class (a *hit* example) and the nearest example of the opposite class (a *miss* example); the score assigned to each input variable $\boldsymbol{x}_i$ is then computed as the ratio between the average over all examples of the distance to the nearest miss and the average distance to the nearest hit, projected on $\boldsymbol{x}_i$. However, this idea is extended in (Guyon et al., 2003) using the averages of the distances to the k nearest hits and misses. That is what we denote in this paper as $\texttt{Relief}_{\texttt{K}}$.

An interesting approach that also deals with redundancy explicitly is found in (Yu and Liu, 2004), where the authors present FCBF, a filter consisting in two separate stages: the first one is a relevance analysis, aimed at ordering the input variables depending on a relevance score, which is computed as the *symmetric uncertainty* with respect to the target output. This ranking criterion can be expressed as

$$SU(\boldsymbol{x}_i, \boldsymbol{y}) = 2\left[\frac{IG(\boldsymbol{x}_i|\boldsymbol{y})}{H(\boldsymbol{x}_i) + H(\boldsymbol{y})}\right], \tag{8}$$

where $H(\boldsymbol{x}_i)$ is the entropy of the input variable $\boldsymbol{x}_i$ and $IG(\boldsymbol{x}_i|\boldsymbol{y})$ is the information gain used for instance in (Quinlan, 1993). Notice that the use of the entropy assumes that the values of $\boldsymbol{x}_i$ must be discrete. This stage is also used to discard irrelevant variables, which are those whose ranking score is below a predefined threshold.

The second stage is a redundancy analysis, aimed at selecting predominant features from the relevant set obtained in the first stage. This selection is

an iterative process that removes those variables for which $\boldsymbol{x}_i$ forms an *approximate Markov blanket*. As defined in (Yu and Liu, 2004), a variable $\boldsymbol{x}_i$ forms an approximate Markov blanket for any other variable $\boldsymbol{x}_j$ if and only if $SU(x_j, y) \geq SU(x_i, y)$ and $SU(x_i, x_j) \geq SU(x_i, y)$.

3 A simple ranker

In this paper we propose a new filter method for variable ranking based in Gram-Schmidt orthogonalization, like those used in the works of Stoppiglia et al. (2003) and Guyon et al. (2003). However, in our approach the orthogonalization does not have any influence on the ordering of the input variables; instead, it is used after the construction of a correlation-based ranking as a process devised for *redundancy* detection and elimination. This approach resembles the framework proposed in (Yu and Liu, 2004) in which relevancy and redundancy analyses are explicitly separated.

3.1 Nonlinear correlation-based ranking

In our approach we first construct a complete ranking using a correlation-based criterion. This criterion relies on the existence of a linear relationship between the input variables and the target output, circumstance which prevents its direct application on problems where such relation is not linear. However, we can transform the input space into a new space where a linear-in-its-parameters model can be tested. This is the same idea found in kernel-based learning methods which transform the original input space $\mathcal{X}$ into a feature space, $\Phi(\mathcal{X})$.

A polynomial model is acknowledged as a good approximation to nonlinear functions. However, a full polynomial expansion would require to augment the original input space with all possible monomials up to a given degree d. This approach is clearly unfeasible when objects are described by a large number of variables in the input space. For example, a full polynomial expansion up to degree 3 of a 150-dimensional input space would yield a feature space with 585,275 dimensions; this number increases up to 1,373,700 dimensions for 200 input variables.

To overcome these enormous feature spaces, we propose a *simplified polynomial expansion* (SPE) to transform the input space $\mathcal{X}$ into a new feature space $\varphi(\mathcal{X})$, which hopefully will have a quasi-linear relationship with the class. The feature space $\varphi(\mathcal{X})$ is constructed with the successive powers up to a given degree d of each input variable. For a m-dimensional input space, the

Function SPE-*ranker* $(x, y, d) : \{O, R\}$
/* SPE: Simplified Polynomial Expansion */
$\hat{x} \leftarrow (\varphi(x) - \textsc{Mean}\,(\varphi(x))) = [x_i^j - \textsc{Mean}\left(x_i^j\right)]$;
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i=1,\ldots,m; j=1,\ldots,d$
$\hat{y} \leftarrow [y - \textsc{Mean}\,(y)]$;
$\hat{x} \leftarrow \frac{\hat{x}}{\|\hat{x}\|}$; $\hat{y} \leftarrow \frac{\hat{y}}{\|\hat{y}\|}$; /* Normalize $\hat{x}$ and $\hat{y}$ */
/* COMPUTING THE RANKING... */
$O \leftarrow \emptyset$;
repeat
  $i_\star \leftarrow \underset{\substack{i=1,\ldots,m \\ i \notin O}}{\arg\max} (\langle \hat{x}_i^j, \hat{y} \rangle^2)$;
  $\qquad\qquad\qquad\qquad\quad j=1,\ldots,d$
  $O \leftarrow [O,\ i_\star]$; /* Append index $i_\star$ to the ranking */
until $(|O| = m)$; /* all variables are ranked */
/* REDUNDANCY DETECTION */
$\hat{x} \leftarrow [x_i - \underset{i=1,\ldots,m}{\textsc{Mean}\,(x_i)}]$; /* Using only original input variables */
$N \leftarrow \underset{i=1,\ldots,m}{[\|\hat{x}_i\|]}$; /* N is the vector of norms of $\hat{x}$, that is, $N_i = \|\hat{x}_i\|$ */
$q \leftarrow \emptyset$; $R \leftarrow \emptyset$; $i \leftarrow 0$;
repeat
  $i \leftarrow i + 1$;
  if $(i \notin R)$ **then**
    $k = O_i$; /* k is the index of the column vector to be appended to q */
    $q \leftarrow [q,\ \frac{\hat{x}_k}{\|\hat{x}_k\|}]$;
    $\hat{x} \leftarrow \hat{x}$ after removing $\hat{x}_k$;
    $N \leftarrow N$ after removing N_k;
    $\hat{x} \leftarrow \text{GS-}\textsc{project}\,(\hat{x}, q)$;
    $\hat{r} \leftarrow \{j : \frac{\|\hat{x}_j\|}{N_j} < \delta; j = 1, \ldots, |\hat{x}|\}$;
    /* $\hat{r}$ contains the indexes of the redundant variables detected in the current iteration, those whose $\|\cdot\| < \delta$ */
    $R \leftarrow [R,\ \hat{r}]$;
  end if
  $Stop \leftarrow (\#\textsc{cols}\,(q) \geq \#\textsc{rows}\,(q) \cdot \xi) \vee (i = m)$;
until $(Stop = \textsc{True})$
$O \leftarrow O$ after removing R; /* Redundant vars. are removed from the ranking */
return $(\{O, R\})$;

Algorithm 1. Pseudocode of SPE-*ranker*. Its input is the training set $\{\boldsymbol{x}, \boldsymbol{y}\}$, where $\boldsymbol{x}_i : i = 1, \ldots, m$ are the columns of input matrix $\boldsymbol{x}$ that collect the values of the i-th variable; $\boldsymbol{y}$ is the column vector of target values (class). It also needs the degree, d, to be used by the polynomial expansion. The output of SPE-*ranker* is the ranking of input variables ordered (O) by their relevancy and a set of redundant (R) variables.

dimension of $\varphi(\mathcal{X})$ will therefore be $m \cdot d$.

$$\varphi(x_1, \ldots, x_m) = (x_1, \ldots, x_m, x_1^2, \ldots, x_m^2, \ldots, x_1^d, \ldots, x_m^d). \tag{9}$$

By means of the SPE the ranking obtained in this first stage of our algorithm can take into account nonlinear correlation with the target output. Notice that we explicitly refuse to construct features as products among input variables to avoid a combinatorial explosion in the feature space. For this reason, our ranker will get into trouble trying to detect relevancy in XOR-like problems, where the target class only depends on products among input variables. But, as we will show in Section 5.1, this is a common difficulty for any ranker. However, when the relation between input variables and the target output is not exclusively based on these products, SPE-*ranker* exhibits a good performance.

Given that our aim is to obtain a ranking of variables in the input space (and not in the feature space), the transformation of the input space $\mathcal{X}$ into $\varphi(\mathcal{X})$ requires some changes in the original ranking criterion. Thus, the ranking score of a variable $\boldsymbol{x}_i$ will be given by the best score among those obtained by the powers of $\boldsymbol{x}_i$. In symbols, our ranking criterion can be expressed as

$$R_{\texttt{SPE}}(i) = \max_{j=1,\dots,d} \rho^2(\boldsymbol{x}_i^j, \boldsymbol{y}), \quad i = 1, \dots, m. \tag{10}$$

To compute the correlation coefficient efficiently we first standardize and normalize the feature vectors, as indicated in Algorithm 1. For efficiency reasons and given that we normalize the feature vectors after their standardization, we can skip the division by the standard deviation without affecting the results; so we are actually normalizing mean-centered vectors, as reflected in the pseudo-code. Standardization is a preprocessing step usually employed to avoid the bias caused by very different ranges of values of the input variables. It is easy to prove that the correlation coefficient between two vectors coincides with the cosine of the angle formed by those vectors standardized. Additionally, if those vectors are also normalized, then the cosine is simply their scalar product, so we can rewrite the ranking criterion of equation (10) as

$$\begin{aligned} R_{\texttt{SPE}}(i) &= \max_{j=1,\dots,d} (\rho^2(\boldsymbol{x}_i^j, \boldsymbol{y})) = \max_{j=1,\dots,d} (\cos^2(\hat{\boldsymbol{x}}_i^j, \hat{\boldsymbol{y}})) = \\ &= \max_{j=1,\dots,d} \left(\frac{\langle \hat{\boldsymbol{x}}_i^j, \hat{\boldsymbol{y}} \rangle^2}{\|\hat{\boldsymbol{x}}_i^j\|^2 \|\hat{\boldsymbol{y}}\|^2} \right) = \max_{j=1,\dots,d} (\langle \hat{\boldsymbol{x}}_i^j, \hat{\boldsymbol{y}} \rangle^2), \quad i = 1, \dots, m, \end{aligned} \tag{11}$$

where $\hat{\boldsymbol{c}}$ represents the standardized and then normalized version of a given vector $\boldsymbol{c}$.

3.2 Redundancy detection

The second stage of our algorithm deals with redundancy detection. For this purpose, and starting with the ranking obtained in the first stage, we rely on the Gram-Schmidt orthogonalization as follows:

(1) First we select the column vector $\boldsymbol{x}_i$ corresponding to the leading position of the ranking. Then a column matrix $\boldsymbol{q}$ is built with $\hat{\boldsymbol{x}}_i$, that is, the standardized and then normalized version of $\boldsymbol{x}_i$.
(2) The rest of the input variables (column vectors) are standardized and projected onto the null subspace of $\boldsymbol{q}$ using the modified Gram-Schmidt algorithm (Björk, 1967) for numerical stability reasons.
(3) The norms of the resulting projections are compared with the norms of the original standardized vectors. Those column vectors (i.e. those input variables) whose norm decrease more than a given threshold δ when projected onto the null subspace of $\boldsymbol{q}$ are considered redundant with respect to the input variables already included in $\boldsymbol{q}$, so they are removed.
(4) We normalize and include in $\boldsymbol{q}$ the projection corresponding to the next input variable following the order given by the ranking. If a stopping criterion is not met, then we repeat this process starting at step 2.

At the end of the process we obtain a list of redundant variables which are removed from the ranking obtained in the first stage of the algorithm SPE-*ranker*. Notice that, in contrast with SM and $\texttt{GSRelief}_{\texttt{K}}$, the orthogonalization process is only used to decide which variables should be removed.

The stopping criterion is raised when all the input variables have been processed or when the number of vectors appended to $\boldsymbol{q}$ is a fraction ξ of the vectors needed to become a base of the subspace. This condition is imposed to avoid the detection of redundancy due to pure algebraic reasons that arise when the number of examples is smaller than the number of input variables. For example, let us suppose that we have a data set with 4 examples and 100 input variables; once we have appended 4 vectors to $\boldsymbol{q}$, the rest of the column vectors can be expressed as a linear combination of those in $\boldsymbol{q}$, so the algorithm would label the remaining 96 variables as redundant. The condition in the stopping criterion lets the algorithm detect redundancy only when this process is reliable and not as a result of an algebraic limitation.

4 Experimental results

This section is devoted to show the empirical results obtained by our ranker and to compare them with the results obtained by some state-of-the-art rankers. In all cases, we are assuming that the learning task to be performed is a binary classification, although most of the rankers could deal with multi-class or even regression problems. To carry out the comparisons we have used a ROC-based measure to estimate the quality of the rankings obtained on a bunch of artificial data sets. In the rest of this section we detail how we will estimate the performance of the ranker, how we have devised the artificial data sets, and finally we show and discuss the results obtained, focusing on the behavior

of the rankers when varying redundancy, number of irrelevant variables and complexity of the model to be learned.

The MATLAB® source code of the algorithm and the data sets generator can be downloaded from `http://www.aic.uniovi.es/speranker`.

4.1 Performance estimation

Usually, the comparison among different subsets of input variables for a given data set is established in terms of accuracy yielded by different learners when applied to the projection of the original data set on such subsets of variables. This is a very reasonable approach if we do not know anything about the problem. However, given that we used artificially created data sets in our experiments, we know which are the relevant, redundant and irrelevant variables, so we can use a ROC-inspired setting to evaluate the quality of rankings, as was made in (Jong et al., 2004). Moreover, notice that if we want to test the performance of a ranker, we must avoid the intervention of additional processes, like learning, since they can only hide the role played by the ranker.

Thus, for a given ranking of m input variables we can draw the so-called ROC-FR curve (FR stands for Feature Ranking) with the points

$$\{(FPR(i), TPR(i)), i = 1, \ldots, m\},$$

where $TPR(i)$ (respectively $FPR(i)$) stands for True (False) Positive Rate and it is calculated as the fraction of true (false) relevant variables whose position in the ranking is higher than i. We will use the area under the ROC-FR curve, or AUC-FR for short, as an indicator of the quality of a ranking. A detailed explanation of the statistical meaning of the area under a ROC curve can be found in (Hanley and McNeil, 1982; Fawcett, 2003).

There is a slight difference in the way we calculate AUC-FR with respect to the work of Jong et al. when data sets have redundancy. In their case it is enough to have any combination of r variables from the $2r$ relevant ones (r originally relevant plus r redundant linear combinations of the originally relevant ones) in the top of a ranking to have a maximum AUC-FR. However, since we use one-to-one redundancy, that is, each redundant variable is a copy of just one relevant variable modified by a scale factor, we only consider as relevant either an original relevant variable or one of its redundant copies (if there is more than one), whatever occurs first starting from the top of the ranking. We argue the reason for using this kind of redundancy in Section 4.4.1

4.2 Artificial data sets construction

In order to compare the performance of some of the algorithms mentioned in Section 2, with the method proposed in this paper, we have devised a benchmark setting inspired in the statistical validation environment described in (Jong et al., 2004). Thus, we define a problem space where each data set is randomly generated according to the following six parameters:

- *Number of input variables* (m).
- *Number of examples* (n).
- *Number of relevant input variables* (r): This is the number of input variables involved in the computation of the target concept.
- *Degree* (d): This parameter sets the degree of the polynomial used to obtain the target value (class) as explained below.
- *Redundancy* (η): To take into account the effect of redundancy in the ranking algorithms we have made data sets where $\eta \cdot r$ irrelevant variables were replaced by η groups of redundant variables.
- *Noise* (σ): Input values of some data sets were perturbed by adding Gaussian noise with variance σ. Additionally, a $\sigma \times 100$ percentage of the examples were labeled with the incorrect class. Noise was added after redundancy.

The combination of different values for the given parameters yields different points in the problem space; for each of these points we estimated the performance of the algorithms as the median of the AUC-FR on 30 different data sets; we define this as a *single experiment*. In turn, for a group of single experiments we estimate its performance as the average performance on each single experiment.

The data sets are generated as follows: for a given point $(m, n, r, d, \eta, \sigma)$ in the problem space, each data set was constructed as a $n \times m$ matrix x whose values follow a uniform distribution in $[-1, 1]$. To assign a class label to each example, first we draw a random $r \times d$ matrix $(c_{i,j})$ with coefficients in $[-2, -1] \cup [+1, +2]$. We used this range to avoid that randomness could eventually produce coefficients close to 0, which would falsify the subset of relevant input variables given by parameter r. Then we define the following polynomial of degree d

$$P(v_1, \ldots, v_m) = \prod_{j=1}^{d} \left(\sum_{i=1}^{r} (c_{i,j} v_i) + b_j \right), \tag{12}$$

where b_j is a random independent term included to ensure that there will be monomials of all degrees up to d; also notice that there will be monomials formed by products of different variables. Then, the class label for each training example $\boldsymbol{x}_{i,\cdot}$ (i-th row of matrix $\boldsymbol{x}$) was determined by this polynomial as

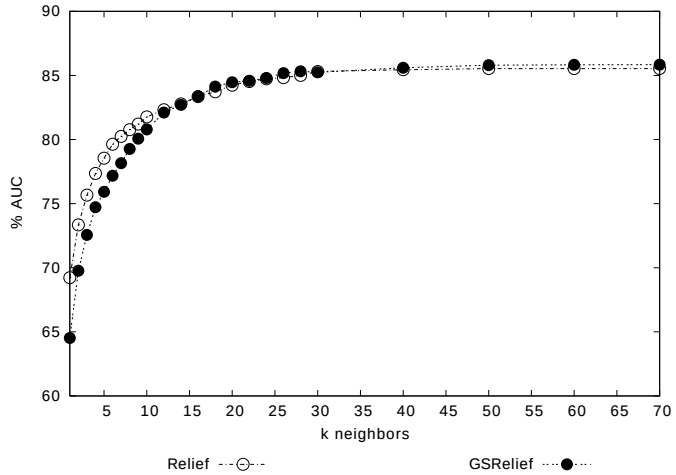


Figure 1. Selection of parameter k in the kNN based algorithms, that is, $\texttt{Relief}_{\texttt{K}}$ and $\texttt{GSRelief}_{\texttt{K}}$. For values of $k > 30$ the improvement is almost unappreciable.

follows:

$$y_i = \begin{cases} +1, & \text{if } P(\boldsymbol{x}_{i,\cdot}) > \text{median}(P(\boldsymbol{x}_{k,\cdot}) : k = 1, \ldots, n). \\ -1, & \text{otherwise.} \end{cases} \tag{13}$$

Finally, we only have to add the corresponding redundancy (η) and noise (σ).

4.3 Parameter setting for the algorithms

Some of the algorithms used in the experiments can be very sensitive to different values of its parameters. For instance, the election of the kernel or the degree of overfit allowed (regulated by a parameter named C) can have an impact on the results given by SVM-based algorithms, while scores of Relief-based rankers are influenced by the number of neighbors used (k).

We have made several trials in order to fine-tune these parameters. Thus, for the election of parameter C in `RM` and `DM` rankers we tried values of $C \in \{10^{-2}, 1, 10^2, 10^5\}$, choosing in each experiment the value that yielded the best results. This parameter has a default value in the implementation of $\texttt{R}^2\texttt{W}^2$ used in the experiments (the one included in the Spider toolbox (Weston et al., 2005)), so we used such fixed value for this ranker. Also, for the three SVM-based rankers (`RM`, `DM`, $\texttt{R}^2\texttt{W}^2$) we have always used a polynomial kernel of the same degree of the polynomial relation between the relevant variables and the target output.

In the same manner, for `SM` and `SPE-`*ranker* we used on each experiment a polynomial expansion up to the corresponding degree. In `SPE-`*ranker* we also

set $\xi = \frac{2}{3}$ (see the stopping criterion of Algorithm 1). The redundancy threshold, δ, is not fixed to any value; instead, it is recalculated at each iteration of the redundancy stage as a relative amount with respect to the decrease of the norms of the vectors projected. In symbols,

$$\delta = \frac{1}{2}\left(\frac{1 + \sum_{j=1}^{\#\mathrm{COLS}(\boldsymbol{x}')} \frac{\|\boldsymbol{x}'_j\|}{N_j}}{1 + \#\mathrm{COLS}(\boldsymbol{x}')}\right), \tag{14}$$

where $\boldsymbol{x}'$ is the matrix with the projections, $\boldsymbol{x}'_j$, of column vectors (see Algorithm 1) and N_j is the norm of the corresponding vector before the redundancy stage started.

Finally, for the Relief-based rankers we have constructed several artificial data sets, varying the parameters indicated in Section 4.2 to obtain a sample of problems from the problem space; we have applied both `Relief`$_{\mathtt{K}}$ and `GSRelief`$_{\mathtt{K}}$ on this sample, using different values for k, the number of neighbors. Figure 1 depicts the average results depending on such parameter. Since there was little or no improvement when using more than 30 neighbors we have fixed $k = 30$ for all the experiments described in the rest of this section.

The rest of parameters not explicitly mentioned here were set to their default values.

4.4 Summary of results

Following the procedure described above to construct the artificial data sets, we have conducted some experimental comparisons focusing on different characteristics of the problem space to analyze how they affect the performance (in terms of AUC-FR and computation time) of the algorithms being compared.

In the comparison we have not included the `FCBF` algorithm given that its AUC-FR scores were notably worse than those obtained by the rest of the algorithms (although it was the fastest ranker). We suspect that the reason for this poor performance is due to its necessity of using discrete input variables. The lack of a metric relation between the discretized variables and the target class puts `FCBF` in trouble when dealing with our artificial data sets, where the class is obtained as the sign of a polynomial involving the relevant input variables.

Notice also that the rankers `SM`, `GSRelief`$_{\mathtt{K}}$, and `SPE`-*ranker* share some basic components, although combined in different ways, so its inclusion in this comparative allows us to isolate and compare the benefits of each component as well as to identify the best combination. Table 1 summarizes the structural

Table 1
Different combinations of ranking criteria and redundancy detection give rise to different input variable rankers.

Ranker	Polynomial expansion	Ranking criterion	Redundancy detection (Gram-Schmidt)
`GSRelief`$_{\mathtt{K}}$	none	Relief	*during* ranking
`SM`	Simplified (SPE)	Correlation	*during* ranking
`SPE-`*ranker*	Simplified (SPE)	Correlation	*after* ranking

similarities among these rankers.

In the rest of this section we report the results obtained focusing on redundancy, number of irrelevant input variables and degree of the polynomial relation between the relevant variables and the class. Additionally, we also studied the performance on a specific kind of problems constructed following the indications in (Weston et al., 2001) to build nonlinear problems. The results obtained on these particular problems are shown and commented at the end of this section. We do not report comparisons of performance varying the number n of training examples; the reason is that increasing n yields a uniform improvement in all rankers.

4.4.1 Redundancy analysis

To analyze the effect of the presence of redundant variables we constructed artificial data sets varying the values of the parameters that define each point of the problem space; we used the following values: $m \in \{50, 100\}$, $n \in \{50, 100\}$, $r \in \{5, 10\}$, $d \in \{1 \text{ (linear)}, 2\}$, $\eta \in \{0 \text{ (no redundancy)}, 1, 2, 3, 4\}$, and $\sigma \in \{0, 0.05 \text{ (5\%)}\}$, yielding 160 points in the problem space. Recall that, for each point we estimate the performance as the median of 30 different data sets, so in this analysis we used $160 \times 30 = 4800$ randomly generated data sets.

Worth of mention is the kind of redundancy used in these experiments (when $\eta \neq 0$). Usually, redundancy is attained as a (non)linear combination of all or part of the relevant variables. However, in our experiments we used a one-to-one redundancy, that is, each redundant variable is a copy of just one relevant variable multiplied by a random scale factor. Notice that the presence of noise will differentiate slightly the values of redundant variables.

Considering the performance measure that we used (AUC-FR) to estimate the quality of a ranking, this kind of redundancy is harder to discover than the redundancy due to linear combinations of groups of relevant variables. Let's illustrate the rationale behind this affirmation with a simple example.

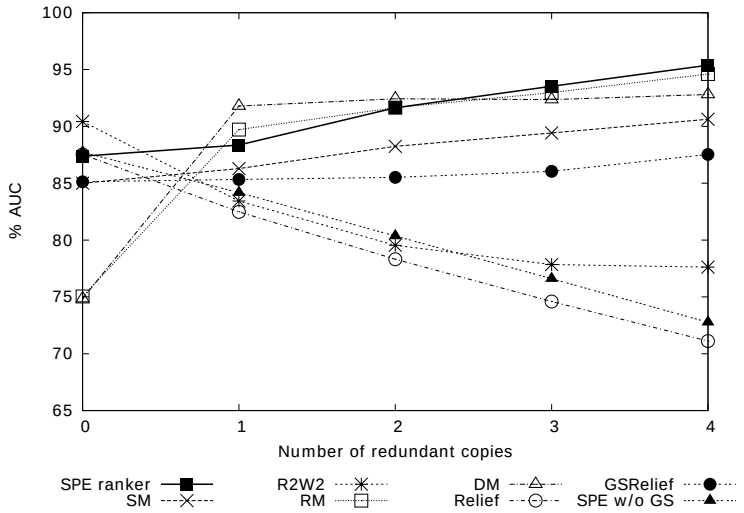


Figure 2. AUC-FR results obtained increasing the number of redundant copies of the relevant variables. In addition to the rankers used in the comparative study, we have also used a version of our algorithm with redundancy detection disabled, labeled as `SPE w/o GS`.

Suppose we have a data set D_1 with $\{a_1, a_2, a_3, a_4\}$ input variables, where the target concept depends only on a_1 and a_2 and the rest of input variables are redundant as follows: $a_3 = \alpha a_1$ and $a_4 = \beta a_2$. Among all the $4! = 24$ possible rankings, the best in terms of AUC-FR will be those whose two topmost variables are able to fully explain the target; for data set D_1 these should include either a_1 and a_2, a_1 and a_4, a_2 and a_3, or a_3 and a_4, yielding 16 different rankings with the highest AUC-FR score. Therefore, a randomly chosen ranking has a probability of $2/3$ to be one of the best possible rankings with this kind of redundancy.

Now let's consider a different data set D_2, almost identical to D_1 except for the redundant variables, which are defined as follows:

$$\begin{array}{l} a_3 = \alpha_1 a_1 + \alpha_2 a_2; \\ a_4 = \beta_1 a_1 + \beta_2 a_2; \end{array} \quad \text{and} \quad \begin{vmatrix} \alpha_1 & \alpha_2 \\ \beta_1 & \beta_2 \end{vmatrix} \neq 0.$$

In this case, the two topmost variables of any of the 24 possible rankings contain the information needed to explain the target, so the probability to choose one of the best rankings is 1. Clearly, the problem of finding the best ranking is easier in D_2 than in D_1.

Figure 2 depicts a summary of the results obtained on these artificial data sets, representing the average AUC-FR achieved on the single experiments. To evidence the usefulness of the redundancy detection part of our approach we have made a simple ablation analysis; thus, we have included in this comparative the results of a simplified version of our algorithm, labeled as `SPE`

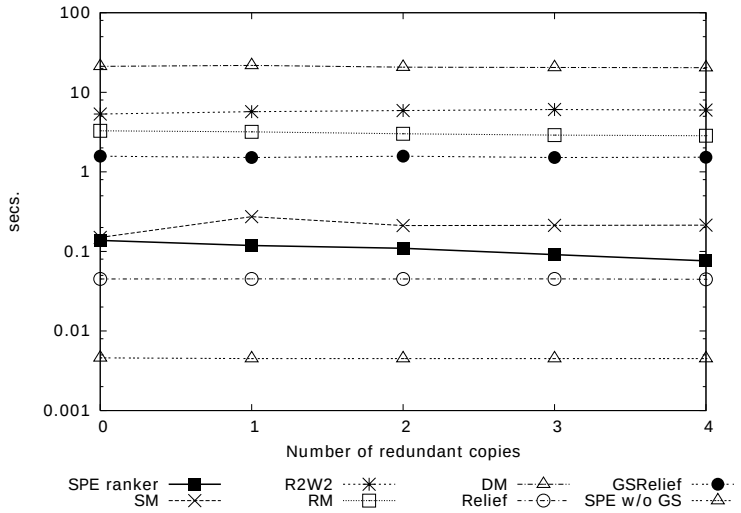


Figure 3. This graphic shows the effect of increasing redundancy on the average time in seconds needed to complete a single experiment (training on 30 different data sets).

`w/o GS`, where redundancy detection was disabled.

The trend observed indicates that these algorithms yield better performance as redundancy increases except $\texttt{Relief}_{\texttt{K}}$, $\texttt{R}^2\texttt{W}^2$ and, as expected, `SPE w/o GS`. We can also observe that `RM` and `DM` exhibit inferior performance than the rest of the algorithms when there is no redundancy at all ($\eta = 0$), but they are clearly benefited when redundancy appears; in fact, these two methods together with `SPE-`*ranker* achieve the highest scores in these block of experiments.

We have also analyzed the cost, in terms of computation time, of these algorithms. Figure 3 shows the average time in seconds needed by each algorithm to perform a single experiment (30 data sets). In general, the trend is that increasing the number of redundant variables does not affect the running time, with the exceptions of `SM`, which is faster when there is no redundancy, and `SPE-`*ranker*, whose running time decreases as the number of redundant copies increases. The reason is that our ranker skips redundant variables as soon as they are discovered by the Gram-Schmidt orthogonalization. Also, notice the log scale of the graph: the algorithms with similar performance to `SPE-`*ranker* are between ten and one hundred times slower.

4.4.2 Number of irrelevant input variables

This block of experiments is aimed at observing the effect of irrelevant input variables on the performance of the rankers. For this purpose we have constructed 4800 artificial data sets, as for the previous block of experiments described above, but focusing on the effect provoked by variations in the number of irrelevant variables. Although we have not explicitly defined a parameter

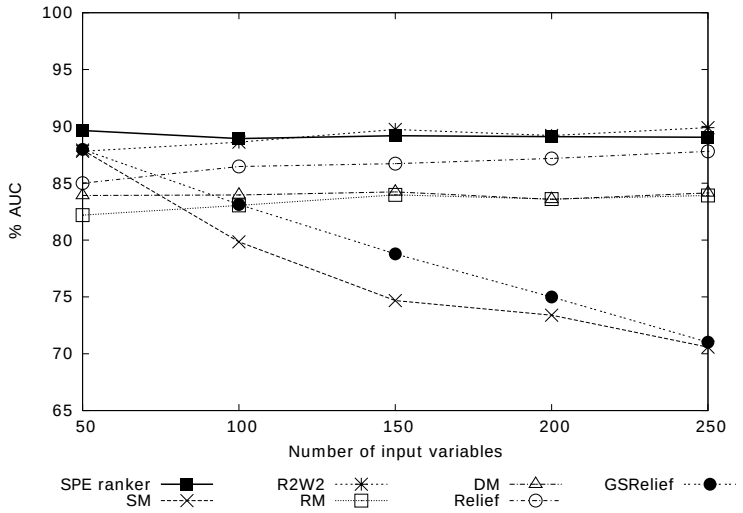


Figure 4. AUC-FR scores obtained increasing the number of input variables by adding irrelevant ones.

to determine the number of irrelevant input variables, this is implicitly defined as $(m - r)$ so we have experimented with several values for m, while r took only two different values. More precisely, we used the following parameters: $m \in \{50, 100, 150, 200, 250\}$, $n \in \{50, 100\}$, $r \in \{5, 10\}$, $d \in \{1, 2\}$, $\eta \in \{0, 1\}$, and $\sigma \in \{0, 0.05\}$.

A graphical summary of the AUC-FR obtained in this block of experiments is shown in Figure 4. The best results were achieved by SPE-*ranker* and $\mathtt{R^2W^2}$, which exhibited a similar and almost constant performance, with slight variations with respect to the variations in the number of input variables. The most affected algorithms were SM and $\mathtt{GSRelief_K}$, whose performance degraded notably as the number of input variables increased. This is due to the fact that both algorithms embed the orthogonalization process in the construction of the ranking; as long as these algorithms project the variables in successive iterations, they are "discounting" the information already included in the variables previously projected, so the more the algorithms advance, the more similar are the projections obtained and thus it is easier to choose a wrong variable to be added to the ranking.

Figure 5 shows the average computation time on this block of experiments. The graph shows that running time for all algorithms increases as the number of input variables increases, as expected. However, this effect is more prominent for RM and DM, which indicates that these algorithms do not scale well with respect to the number of input variables. There is also an significant difference in speed between the two algorithms that yielded the best rankings, SPE-*ranker* and $\mathtt{R^2W^2}$, in favor of the former which was approximately ten times faster than the later.

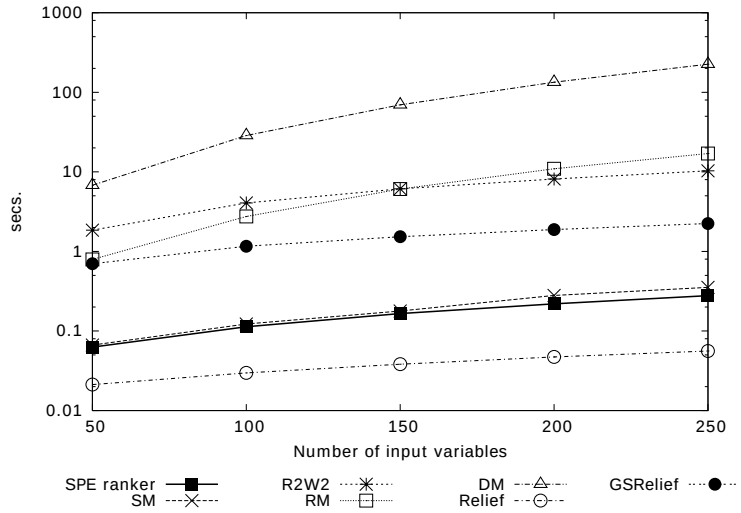


Figure 5. Effect on the running time of the variations in the number of input variables.

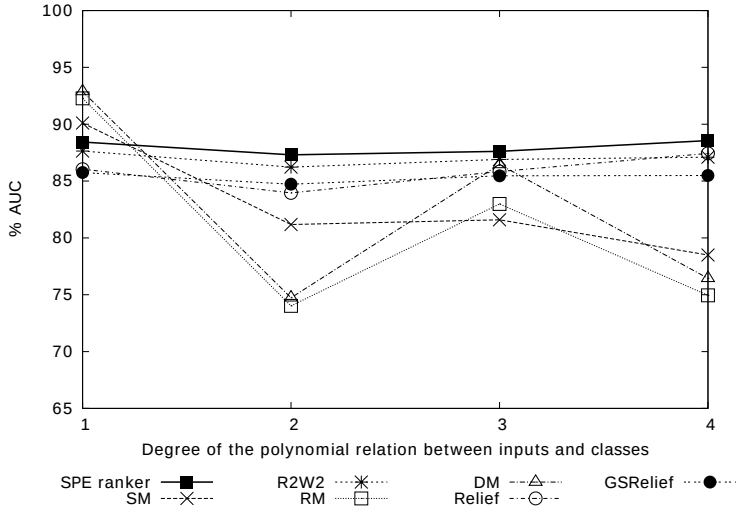


Figure 6. This graphic shows the AUC-FR scores obtained by the rankers depending on the complexity of the relation between the relevant variables and the target output.

4.4.3 Degree of the polynomial relation with the class

We made a third block of experiments to analyze the performance with respect to the complexity of the relation between the relevant input variables and the target output. Thus, in this case we have focused on the results obtained varying the degree of the polynomial used to determine the class of each case. The values of the parameters used to construct the data sets ranged as follows: $m \in \{50, 100\}$, $n \in \{50, 100\}$, $r \in \{5, 10\}$, $d \in \{1, 2, 3, 4\}$, $\eta \in \{0, 1\}$, and $\sigma \in \{0, 0.05\}$.

In this block of experiments we have observed that there is a group of algo-

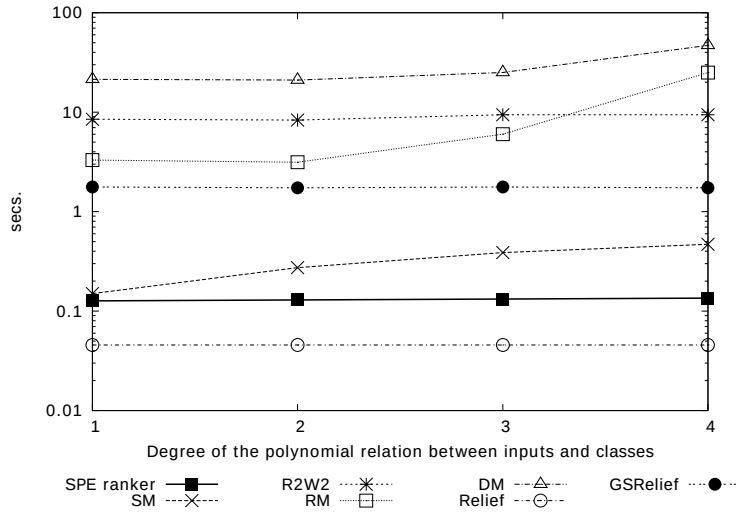


Figure 7. Running time and complexity.

rithms yielding an almost constant AUC-FR performance; this group is, in descending order of performance, SPE-*ranker*, $\texttt{R}^2\texttt{W}^2$, $\texttt{Relief}_\texttt{K}$ and $\texttt{GSRelief}_\texttt{K}$. On the other hand, we have found that for the linear case (degree 1) the algorithms that performed better were DM, SM and RM, which in this case is, in fact, RFE (Guyon et al., 2002). However, the performance of these algorithms clearly decreases as the degree increases. Notice also that, curiously, both RM and DM exhibited worse behavior for polynomials of even than odd degree.

The computation time of RM, DM and SM was also affected by the increase of complexity, as it is reflected in Figure 7; RM and DM together with $\texttt{R}^2\texttt{W}^2$ show the highest average running time; however, $\texttt{R}^2\texttt{W}^2$ exhibits an almost constant trend while RM and DM's trend is unpleasantly increasing with the complexity of the problem.

4.4.4 Weston's et al. nonlinear problems

The results obtained in the previous experiments revealed that the best AUC-FR performance on average was obtained by SPE-*ranker*, followed by $\texttt{R}^2\texttt{W}^2$, so we decided to analyze the behavior of our method on some of the artificial problems that were used in the empirical validation of $\texttt{R}^2\texttt{W}^2$. For this purpose we constructed several nonlinear data sets following the indications of the authors found in (Weston et al., 2001).

Figure 8(A) depicts the scores obtained by each of the compared algorithms, where each point is the median of 30 different data sets. Although for some algorithms there is a noticeable difference in performance between data sets with 20 examples and bigger data sets, the trend of the graphs indicates that the best results are achieved by RM and $\texttt{GSRelief}_\texttt{K}$, followed by $\texttt{Relief}_\texttt{K}$, $\texttt{R}^2\texttt{W}^2$, SPE-*ranker* and DM. Clearly, SM performs poorly on these data sets.

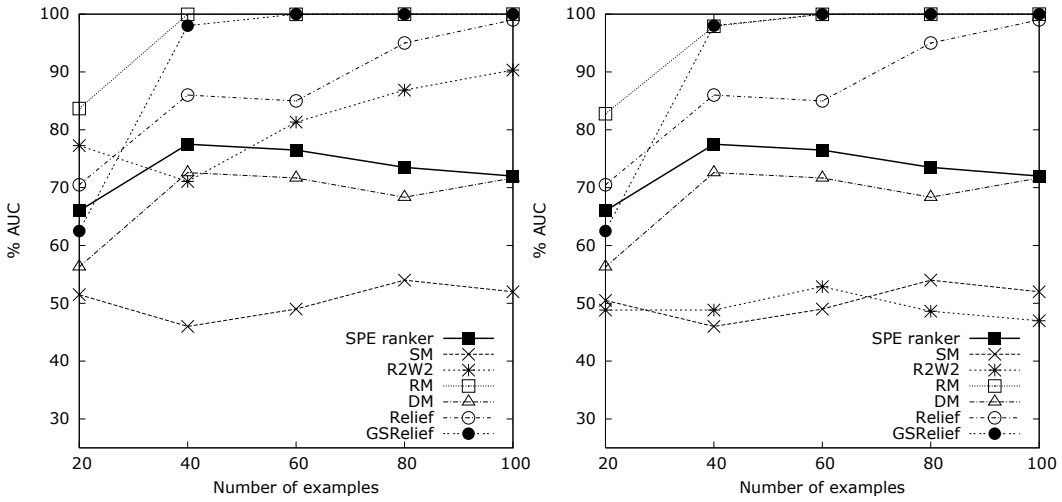


Figure 8. AUC-FR for Weston's et al. nonlinear problems. In (A) we used the original definition of the problems while in (B) we standardized input variables before applying the algorithms.

We have made an additional experiment with these data sets, whose results are drawn in Figure 8(B). In this case we have standardized the input variables before applying the rankers, a preprocessing stage commonly used to avoid the bias induced by differences between the ranges of input values. The results, depicted in Figure 8(B), show that now the standardized version is harder for $\mathtt{R^2W^2}$ than the original version. However, the rest of the algorithms are practically unaffected. In particular, SPE-*ranker* is not affected given that input standardization is part of our algorithm.

5 Some limitations of SPE-*ranker*

In this section we will show how our ranker deals with two particular difficulties related to the ranking and redundancy detection stages, respectively.

The first difficulty arises in problems where there are input variables which are useless by themselves but useful when they are considered together. This is an adverse situation for those algorithms which rank the variables according to their individual predictive power. In particular, SPE-*ranker* uses correlation for this purpose so it is expected that it will be affected under these circumstances.

In turn, correlation between variables can sometimes be the cause of a second difficulty to arise, in this case regarding the mechanism for redundancy detection. The idea is that a very high correlation or anti-correlation between input variables does not mean absence of variable complementarity, as pointed out by Guyon and Elisseeff (2003). The effect of having two complementary but

highly anti-correlated variables could make a redundancy detection algorithm to identify one of such variables as redundant; discarding that variable can be harmful for a posterior learning process that could eventually take advantage of both variables to accomplish its task.

We have constructed *ad hoc* data sets to test the behavior of our ranker under these particular circumstances. The results are presented in the rest of this section.

5.1 Apparently useless variables

A typical example of apparently useless variables can be found in problems where the target output has an XOR relation with some input variables. Here we present an experimental comparison between the rankers used in Section 4 when applied to XOR-like problems. The data sets for this experimental test were constructed with $n = 100$ examples described by $m = 100$ input variables whose values followed a uniform distribution in the interval $[-1, +1]$. We tried with different number of relevant input variables, $r = \{2, 3, 4, 5\}$, labeling each example with the target output calculated as

$$y_i = \text{sign}\left(\prod_{j=1}^{r} x_{i,j}\right). \tag{15}$$

According to these specifications we constructed 30 different data sets for each value of r, obtaining with each algorithm 30 values of AUC-FR whose median is graphically represented in Figure 9.

For those rankers that have a parameter to indicate the degree of the relation (SPE-*ranker*, SM, R^2W^2, RM, and DM) we used the value of the corresponding r, hoping to approximate the true relation of each data set as much as possible. However, the poor results (around 50% AUC-FR) show that none of the algorithms was able to correctly identify the relevant variables. Noticeable exceptions are Relief_K and RM which performed very good, but only in the two-dimensional version of the problem. However, when there were more than 2 input variables involved in the computation of the class, their performance decreased down to scores similar to those obtained by the rest of the algorithms.

A workaround to solve this problem would be to use a full polynomial expansion, allowing the ranker to detect the product of relevant input variables as a relevant feature. In fact, this is the original proposal of Stoppiglia et al. (2003). However, this approach would make many problems intractable due to the combinatorial explosion of new features, as explained in Section 3.1.

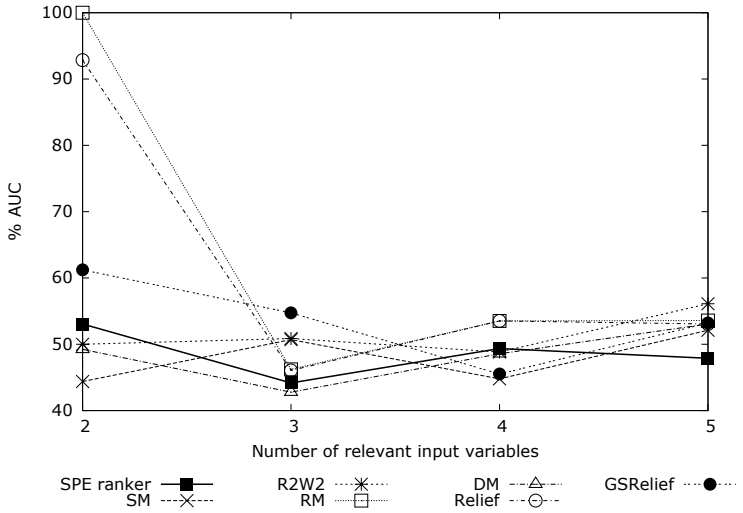


Figure 9. AUC-FR obtained on XOR-like problems. We have experimented with several degrees of complexity, from the classical bi-dimensional XOR up to problems with 5 relevant input variables.

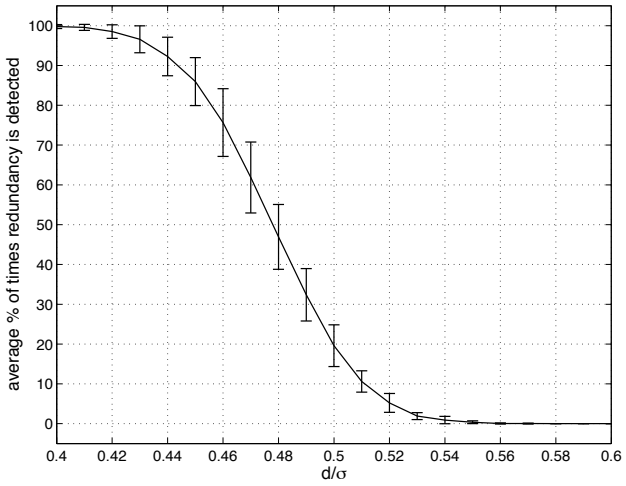
5.2 Highly anti-correlated complementary variables

To illustrate this situation, let us consider (Guyon and Elisseeff, 2003) a binary classification problem were examples are described by two highly anti-correlated variables, namely x_1 and x_2, that follow a normal distribution with a given standard deviation σ. Figure 11 depicts two problems of this kind, where examples of each class are distributed along two parallel lines and the centers of the classes are separated a distance d and aligned perpendicularly with respect to these lines. Clearly, none of the input variables alone are useful to separate the classes. However, when x_1 and x_2 are considered together, classes are easily separated.

To analyze the behavior of SPE-*ranker* in the presence of anti-correlated variables we have constructed artificial data sets to simulate this kind of problems, varying σ from 1 to 3 in steps of 0.1 and varying d as a fraction of σ, from 0.4σ to 0.6σ in steps of 0.01. For each combination of σ and d we randomly generated 200 different data sets with 100 examples of each class. Figure 10 depicts the average (and standard deviation) percentage of times SPE-*ranker* failed; that is, when SPE-*ranker* detected redundancy for each value of d/σ.

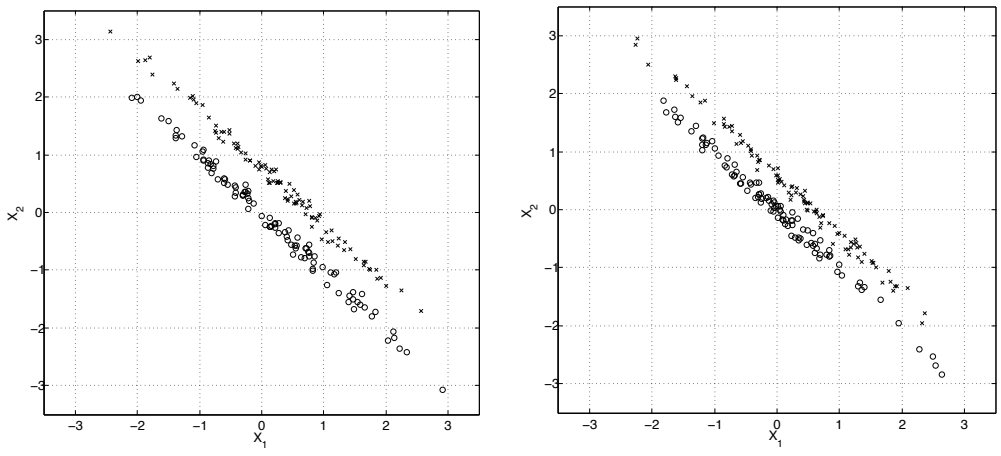
These results show that the classes must be closer than 0.54σ (Figure 11(A)) so that SPE-*ranker* begins to exhibit an undesirable behavior detecting redundancy. As the distance between classes decreases, the frequency of redundancy detection increases; moreover, when classes are closer than 0.42σ (Figure 11(B)) our ranker always discard one of the input variables.

We have computed the average correlation between x_1 and x_2 for each value of



d/σ	avg. corr(x_1, x_2)
0.40	-0.9580
0.42	-0.9539
0.44	-0.9498
0.46	-0.9456
0.48	-0.9410
0.50	-0.9366
0.52	-0.9320
0.54	-0.9266
0.56	-0.9215
0.58	-0.9163
0.60	-0.9107

Figure 10. Percentage of cases where complementary anti-correlated variables are unpleasantly considered redundant by SPE-*ranker*. Notice that when classes are separated more than 0.54σ our ranker behaves satisfactorily since it never detects redundancy between the complementary variables. The table at the right shows the average correlation coefficients between x_1 and x_2 in the data sets of the experiments.



(A) (B)

Figure 11. When classes are separated at least as in (A) SPE-*ranker* behaves correctly, considering that there is no redundancy in the input variables. When classes are closer than in (B) our ranker always discard one of the variables.

d and σ and we have observed a linear relation between such correlation and the ratio d/σ, which is shown in the table of Figure 10. Therefore, the results of the previous paragraph can be expressed in terms of correlation coefficients to conclude that (anti) correlation between variables should be higher than 0.95 to make our ranker have an incorrect behavior; however, for correlation values lower than 0.92 we can expect a correct behavior of SPE-*ranker*.

6 Conclusions

In this paper we have presented a filter approach to obtain a ranking of variables according to their usefulness for classification; moreover, our filter deals explicitly with redundancy. The filter was constructed as a combination of a *simplified polynomial expansion* (SPE) of the original input variables together with two classical techniques, *correlation* and *orthogonalization*; this simplicity confers the algorithm, named `SPE`-*ranker*, a good performance regarding computation time, while the quality of the rankings obtained is, in general, better than that of the rankings obtained by more complex state-of-the-art algorithms. In particular, the simplified polynomial expansion has proved to be a good heuristic approach to general nonlinear models.

The experimental validation, thoroughly described in the paper, was made on artificially generated data sets including a variety of conditions of noise, nonlinearity, redundancy, etc. We used a specialized version of the area under the ROC curve (AUC-FR) to estimate the quality of the rankings obtained. The aim of this measure is to assess the ranking abilities without the intervention of any learner.

7 Acknowledgments

The authors want to thank the thorough work of the reviewers whose interesting suggestions substantially improved the original version of this paper. The research described in this paper was partially supported by grant TIN2005-08288 of the Spanish Ministerio de Educación y Ciencia.

References

Bahamonde, A., Bayón, G. F., Díez, J., Quevedo, J. R., Luaces, O., del Coz, J. J., Alonso, J., Goyache, F., July 2004. Feature subset selection for learning preferences: A case study. In: Greiner, R., Schuurmans, D. (Eds.), Proceedings of the International Conference on Machine Learning (ICML '04). Banff, Alberta (Canada), pp. 49–56.

Björk, A., 1967. Solving linear least squares problems by Gram-Schmidt orthogonalization. Nordisk Tidshrift for Informationsbehadlung 7, 1–21.

Boser, B. E., Guyon, I., Vapnik, V., 1992. A training algorithm for optimal margin classifiers. In: Computational Learing Theory. pp. 144–152.

Chen, S., Billings, S. A., Luo, W., 1989. Orthogonal least squares methods and their application to nonlinear system identification. International Journal of Control 50 (5), 1873–1896.

Degroeve, S., De Baets, B., Van de Peer, Y., Rouzé, P., 2002. Feature subset selection for splice site prediction. Bioinformatics 18 (2), 75–83.

del Coz, J. J., Bayón, G. F., Díez, J., Luaces, O., Bahamonde, A., Sañudo, C., 2005. Trait selection for assessing beef meat quality using non-linear SVM. In: Saul, L. K., Weiss, Y., Bottou, L. (Eds.), Advances in Neural Information Processing Systems 17 (NIPS '04). MIT Press, Cambridge, MA, pp. 321–328.

Díez, J., Bayón, G. F., Quevedo, J. R., del Coz, J. J., Luaces, O., Alonso, J., Bahamonde, A., 2004. Discovering relevancies in very difficult regression problems: applications to sensory data analysis. In: Proceedings of the European Conference on Artificial Intelligence (ECAI '04). Valencia, Spain.

Fawcett, T., 2003. ROC graphs: Notes and practical considerations for researchers. Tech. Rep. HPL-2003-4, HP Laboratories.

Guyon, I., Bitter, H.-M., Ahmed, Z., Brown, M., Heller, J., December 2003. Multivariate non-linear feature selection with kernel multiplicative updates and Gram-Schmidt relief. In: Proceedings of the BISC FLINT-CIBI workshop. Berkeley.

Guyon, I., Elisseeff, A., 2003. An introduction to variable and feature selection. Journal of Machine Learning Research 3, 1157–1182.

Guyon, I., Weston, J., Barnhill, S., Vapnik, V., 2002. Gene selection for cancer classification using support vector machines. Machine Learning 46 (1–3), 389–422.

Hanley, J., McNeil, B., 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology 143 (1), 29–36.

Jong, K., Mary, J., Cornuéjols, A., Marchiori, E., Sebag, M., 2004. Ensemble feature ranking. In: Boulicaut, J. F., Esposito, F., Giannotti, F., Pedreschi, D. (Eds.), Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD '04). Pisa, Italy, pp. 267–278.

Kira, K., Rendell, L. A., 1992. A practical approach to feature selection. In: Proceedings of the Ninth International Conference on Machine Learning. Morgan Kaufmann, pp. 249–256.

Luaces, O., Bayón, G. F., Quevedo, J. R., Díez, J., del Coz, J. J., Bahamonde, A., 2004. Analyzing sensory data using non-linear preference learning with feature subset selection. In: Boulicaut, J. F., Esposito, F., Giannotti, F., Pedreschi, D. (Eds.), Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD '04). Pisa, Italy, pp. 286–297.

Quinlan, J., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, California.

Rakotomamonjy, A., 2003. Variable selection using SVM-based criteria. Journal of Machine Learning Research 3, 1357–1370.

Stoppiglia, H., Dreyfus, G., Dubois, R., Ousar, Y., March 2003. Ranking a random feature for variable and feature selection for SVMs. Journal of Machine Learning Research 3, 1399–1414.

Weston, J., Elisseeff, A., BakIr, G., Sinz, F., 2005. Spider: object-orientated machine learning library.
URL http://www.kyb.tuebingen.mpg.de/bs/people/spider

Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., Vapnik, V., 2001. Feature selection for SVMs. In: Leen, T. K., Dietterich, T. G., Tresp, V. (Eds.), Advances in Neural Information Processing Systems 13. MIT Press, pp. 668–674.

Yu, L., Liu, H., 2004. Efficient feature selection via analysis of relevance and redundancy. Journal of Machine Learning Research 5, 1205–1224.